文章编号: 1001-0920(2009)04-0481-07

# 动态零等待流水线调度问题的滚动策略及优化算法

钱 斌[1,2], 王 凌[1], 黄德先[1], 江永亨[1], 王 雄[1]
(1. 清华大学 a. 自动化系 b. 清华信息科学与技术国家实验室, 北京 100084; 2. 昆明理工大学 自动化系, 昆明 650051)

**摘 要**: 针对工件动态到达的零等待流水线调度问题,提出一种基于工件的滚动策略. 证明了在该策略下全局调度性能随着局部调度的逐步滚动可得到不断改善. 将该策略与基于差分进化的混合算法有机结合,能有效处理动态零等待流水线调度问题. 最后通过实验验证了所提出策略和算法的有效性.

**关键词**: 动态调度; 滚动策略; 差分进化; 全局罚函数
**中图分类号**: F273 **文献标识码**: A

# Rolling strategy and optimization algorithm for dynamic no-wait flow shop scheduling problem

*QIAN Bin*[1,2], *WANG Ling*[1], *HUANG De-xian*[1], *JIANG Yong-heng*[1], *WANG Xiong*[1]
(1a. Department of Automation, 1b. Tsinghua National Laboratory for Information Science and Technology, Tsinghua University, Beijing 100084, China; 2. Department of Automation, Kunming University of Science and Technology, Kunming 650051, China. Correspondent: QIANG Bin, E-mail: qianb04@mails.tsinghua.edu.cn)

**Abstract:** In this paper, a job-based rolling strategy is proposed for no-wait flow shop scheduling problem with jobs dynamically arriving. It is proved theoretically that this strategy guarantees the improvements of the global scheduling performances over the initial schedule at each step. Then, the proposed strategy is reasonably combined with a hybrid differential eolution (HDE) algorithm to effectively deal with dynamical no-wait flow shop. Finally, experiment results demonstrate the effectiveness of the proposed strategy and HDE.

**Key words:** Dynamic scheduling; Rolling strategy; Differential evolution; Global penalty

## 1 引 言

零等待流水线(Flow shop)调度问题是一类具有很强工程背景的 NP-hard 问题[1]. 典型的 Flow shop 问题假定工件可在缓冲区内等待,但实际中由于生产工艺和储存设备的限制,要求工件从开始加工到完成之间不能出现中断,直到完成所有操作为止. 这类调度称为零等待 Flow shop 调度问题(NWFSP),它广泛存在于化工、炼钢和制药等工业领域. Rock[1]已证明机器数量大于 2 的 NWFSP 是 NP-hard 问题.

考虑调度环境中存在的各种动态不确定因素的车间调度称为动态调度. 近 10 年来,动态调度引起了许多研究者的兴趣,已成为国际上调度研究的一个前沿方向[2],国内也出现了一些相关研究[3]. 动态调度处理不确定因素的模式可归为两类:主动模式和反应模式. 若调度环境中某些不确定因素可进行一定程度的预测和描述,即可在动态调度中主动考虑不确定因素,则这类动态调度称为主动式调度[4-7]. 若调度环境中不确定因素无法预料,只有发生后才能知道其有关属性,则这类动态调度称为反应式调度或重调度[8-13].

本文针对加工工件动态到达的 NWFSP (DNWFSP),研究相应的反应式调度策略和算法. 对于反应式调度,Wu 等[9]利用遗传算法来求解单机双目标的反应式重调度问题;Alagoz 等[12]采用分枝定界算法来求解并行机重调度问题;Szelke 等[14]提出了基于知识的智能优化方法进行重调度;王冰等[13]提出了滚动部分重调度策略来处理动态单机调度问题,因采用灵活的驱动机制而能及时反馈调度环境中发生的动态扰动.

**收稿日期**: 2008-02-23; **修回日期**: 2008-04-15.
**基金项目**: 国家自然科学基金项目(60834004,70871065); 国家 863 计划项目(2007AA04Z155,2007AA04Z193).
**作者简介**: 钱斌(1976—),男,云南曲靖人,讲师,博士生,从事生产调度理论与方法的研究; 王凌(1972—),男,江苏武进人,副教授,博士,从事优化理论与方法等研究.

滚动调度策略的思想源于预测控制[15]. 王冰等[16]针对确定性单机调度问题,提出一种不断改善全局初始调度性能的滚动重调度策略,并把该策略推广到动态单机调度[13]. 目前,对于滚动策略的研究主要集中于单机问题,在更为复杂的 NWFSP 上尚无研究. 对于 DNWFSP,本文采用滚动调度策略对问题进行分解优化和滚动预测,从而降低了问题的复杂度,提高了对动态不确定性的适应程度. 对于分解后的子问题,根据其规模的大小,采用枚举所有排列或运用高效的混合差分进化算法(HDE)[17]进行求解.

## 2 动态零等待 Flow shop 调度问题描述

### 2.1 确定性零等待 Flow shop 调度问题描述

NWFSP 研究 $m$ 台机器上 $n$ 个工件的流水加工过程,每个工件在各机器上的加工顺序相同,每台机器加工各工件的顺序相同. 已知工件 $\pi_j$ 在机器 $k$ 上的加工时间 $p(\pi_j,k)$,并约定每个工件在每台机器上只加工一次,每台机器在某一时刻一次只能加工一个工件. 为了满足零等待约束,同一工件的加工必须连续完成,即要求各工件在给定机器上的完成时间必须等于它在下一台机器上的开始加工时间.

需要指出的是,实际生产中零等待不可能是绝对的. 比如化工生产中原料在各阶段加工前,通常需要对当前反应罐进行清洗;再如工件在机器间的运输时间. 对于实际问题,把机器准备时间、工件运输时间等因素加入到工件的加工时间中,这样可使问题满足零等待约束.

令 $\pi=\{\pi_1,\pi_2,\cdots,\pi_n\}$ 为所有工件的一个排序,$C(\pi_j)$ 为工件 $\pi_j$ 在机器 $m$ 上的加工完成时间,$T_C(\pi)$ 为所有工件在机器 $m$ 上的加工完成时间之和. 记 $L(\pi_{j-1},\pi_j)$ 为由于连续生产要求而引起的相邻两工件 $\pi_{j-1}$ 和 $\pi_j$ 的开始加工时间之差,如图 1 所示. 则有

$$L(\pi_{j-1},\pi_j)=p(\pi_{j-1},1)+\max\Big[0,\max_{2\le k\le m}\Big\{\sum_{h=2}^{k}p(\pi_{j-1},h)-\sum_{h=1}^{k-1}p(\pi_j,h)\Big\}\Big];\tag{1}$$

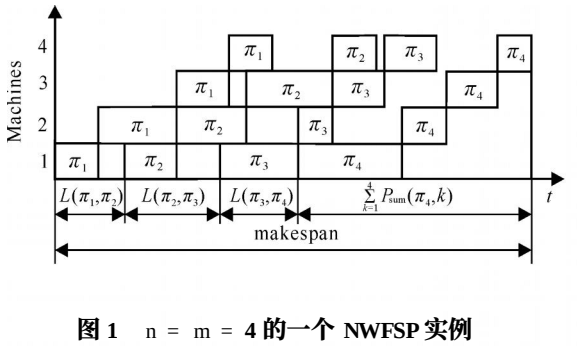


**图 1 n = m = 4 的一个 NWFSP 实例**

$$C(\pi_1)=\sum_{k=1}^{m}p(\pi_1,k);\tag{2}$$

$$C(\pi_j)=\sum_{i=2}^{j}L(\pi_{i-1},\pi_i)+\sum_{k=1}^{m}p(\pi_j,k);\tag{3}$$

$$T_C(\pi)=\sum_{\pi_j\in\pi}C(\pi_j),\ j=2,3,\cdots,n.\tag{4}$$

### 2.2 动态零等待 Flow shop 调度问题描述

对于确定性零等待 Flow shop 问题,假设所有工件在开始调度时已到达加工机器,即所有工件的信息均已知. 在实际生产中,工件往往是先后随机到达的. 在 2.1 节模型的基础上,假设调度初始时刻将有 $n$ 个工件先后到达,各工件到达时间服从一定范围的均匀分布. 令 $r(\pi_j)$ 为工件 $\pi_j$ 的到达时间,$S_t(\pi_j)$ 和 $D_C(\pi_j)$ 分别为 $\pi_j$ 在动态环境中的开始加工时间和加工完成时间,$D_{TC}(\pi)$ 为动态环境中所有工件的加工完成时间之和. 本文采用最小化 $D_{TC}(\pi)$ 作为全局调度指标,于是有

$$S_t(\pi_1)=r(\pi_1);\tag{5}$$

$$S_t(\pi_j)=\max\{S_t(\pi_{j-1})+L(\pi_{j-1},\pi_j),r(\pi_j)\};\tag{6}$$

$$D_C(\pi_j)=S_t(\pi_j)+\sum_{k=1}^{m}p(\pi_j,k);\tag{7}$$

$$D_{TC}(\pi)=\sum_{\pi_j\in\pi}D_C(\pi_j),\ j=1,2,\cdots,n.\tag{8}$$

## 3 滚动调度策略与优化算法

在现实生产中,虽然不可能预知所有工件的信息,但往往可预知近期到达工件的信息. 本文采用的 Job-based 滚动调度策略正是基于这一考虑. 对于滚动策略分解得到的局部调度子问题,采用先进的智能调度算法进行求解,对算法既要关注求解质量,又要考虑实时性.

### 3.1 具有全局罚函数的滚动调度策略

为使局部子调度能兼顾全局性能,从而保证整个调度过程的稳定性,本文提出一种具有全局罚函数的滚动调度策略. 令滚动窗口长度(即滚动窗口中包含的工件数)为 $W$,执行步长(即在本次滚动调度中实际执行加工的工件数)为 $E$.

#### 3.1.1 初始调度

对于工件动态到达的调度问题,假设工件的初始调度 $\pi_{ni}$ 为工件到达先后的排序,即先到达先加工. 在初始时刻,各工件的信息(即 $r(\pi_j)$ 和 $p(\pi_j,k)$)并不知道,因此 $\pi_{ni}$ 为一虚拟调度.

#### 3.1.2 Job-based 滚动调度

令工件按先后顺序从滚动窗口最右端进入,随着新工件的到达不断左移. 设第 $k$ 次滚动时滚动窗口中的原始工件排列为 $\pi_{noS\_sub}(k)$,求出子调度后为 $\pi_{sub}(k)$(即由枚举或优化算法得到的局部调度子问

题的解). 在刚开始执行Job-based滚动调度时,当滚动窗口中的工件数首次达到 $W$ 时,有 $\pi_{\text{noS\_sub}}(1)$. 对这一局部调度子问题(即工件数为 $W$,机器数为 $m$ 的 NWFSP) 进行求解,得到子调度 $\pi_{\text{sub}}(1)$. 将 $\pi_{\text{sub}}(1)$ 中工件均左移 $E(E < W)$ 步,这时滚动窗口中剩余的工件数为 $W - E$. 随着工件的不断到达,当窗口中的工件数再次达到 $W$ 时,有 $\pi_{\text{noS\_sub}}(2)$. 可求出 $\pi_{\text{sub}}(2)$,并将其中工件均左移 $E$ 步. 这一过程不断重复,直到 $n$ 个工件的调度全部求出. 每当求得 $\pi_{\text{sub}}(k)$ 时,立刻把 $\pi_{\text{sub}}(k)$ 中工件均左移 $E$ 步,而不必等到前 $E$ 步都执行完. 即调度求解与实际执行可分开进行,移出的空位等待新到工件添入.

设 $\pi_{\text{before\_sub}}(k)$ 为求解 $\pi_{\text{sub}}(k)$ 时当前滚动窗口之前已求出的调度,$\pi_{\text{after\_sub}}(k)$ 为当前滚动窗口之后所有工件按先后到达顺序的排列. 显然,$\pi_{\text{after\_sub}}(k)$ 为一虚拟排列. 由于调度指标 $D_{TC}(\pi)$ 可按工件进行分解,在求解第 $k$ 次子调度之前,全局调度 $\pi_{g\_\text{noS}}(k)$ 的调度指标 $D_{TC}^{\text{noS}}(\pi_{g\_\text{noS}}(k))$ 可由如下 3 部分构成:

$$D_{TC}^{\text{noS}}(\pi_{g\_\text{noS}}(k)) = D_{TC}^{\text{noS}}(\pi_{\text{before\_sub}}(k)) + D_{TC}^{\text{noS}}(\pi_{\text{noS\_sub}}(k)) + D_{TC}^{\text{noS}}(\pi_{\text{after\_sub}}(k)). \quad (9)$$

求出第 $k$ 次子调度之后,全局调度 $\pi_g(k)$ 的调度指标为

$$D_{TC}(\pi_g(k)) = D_{TC}(\pi_{\text{before\_sub}}(k)) + D_{TC}(\pi_{\text{sub}}(k)) + D_{TC}(\pi_{\text{after\_sub}}(k)). \quad (10)$$

虽然 $D_{TC}^{\text{noS}}(\pi_{\text{before\_sub}}(k)) = D_{TC}(\pi_{\text{before\_sub}}(k))$,但因 $\pi_{\text{noS\_sub}}(k)$ 有可能与 $\pi_{\text{sub}}(k)$ 不同,所以 $D_{TC}^{\text{noS}}(\pi_{\text{after\_sub}}(k))$ 与 $D_{TC}(\pi_{\text{after\_sub}}(k))$ 可能不等.

### 3.1.3　局部调度子问题的指标

为反映局部目标对全局的影响,只对滚动窗口中前 $W-1$ 个工件进行调度问题(即工件数为 $W-1$,机器数为 $m$ 的 NWFSP) 求解,而第 $W$ 个工件 $\pi^W(k)$ 的位置不变,并把 $\pi^W(k)$ 被推迟加工的时间加入到指标中.

令 $\pi_{\text{noS\_sub}}(k) = \{\pi_{\text{noS\_sub}}^{W-1}(k), \pi^W(k)\}$. 其中:$\pi_{\text{noS\_sub}}^{W-1}(k)$ 为 $\pi_{\text{noS\_sub}}(k)$ 中前 $W-1$ 个工件;$\pi^W(k)$ 为第 $W$ 个工件,且 $\pi^W(k)$ 在求出子调度前后不变. 这时有

$$D_{TC}^{\text{noS}}(\pi_{\text{noS\_sub}}(k)) = D_{TC}^{\text{noS}}(\pi_{\text{noS\_sub}}^{W-1}(k)) + D_C^{\text{noS}}(\pi^W(k)), \quad (11)$$

$$D_{TC}(\pi_{\text{sub}}(k)) = D_{TC}(\pi_{\text{sub}}^{W-1}(k)) + D_C(\pi^W(k)). \quad (12)$$

其中:$\pi_{\text{sub}}(k) = \{\pi_{\text{sub}}^{W-1}(k), \pi^W(k)\}$,$\pi_{\text{sub}}^{W-1}(k)$ 为 $\pi_{\text{sub}}(k)$ 中前 $W-1$ 个工件,$\pi^W(k)$ 为第 $W$ 个工件.

设第 $k$ 次求解子调度时滚动窗口中前 $W-1$ 个工件的任意排列为 $\pi_{W-1}(k)$. 本文在局部调度子问题中采用如下带全局罚函数的指标:

$$\min_{\pi_{W-1}(k)} [D_{TC}(\pi_{W-1}(k)) + (|\pi_{\text{after\_sub}}(k)| + 1)\Delta D_{TC}(k)]. \quad (13)$$

其中:$\Delta D_{TC}(k)$ 为 $\pi^W(k)$ 被推迟的开始加工时间,$|\pi_{\text{after\_sub}}(k)|$ 为第 $k$ 次求解子调度时滚动窗口之后的剩余工件数. 令求出第 $k$ 次子调度前 $\pi^W(k)$ 的开始加工时间为 $S_t^{\text{noS}}(\pi^W(k))$,求出第 $k$ 次子调度后 $\pi^W(k)$ 的开始加工时间为 $S_t(\pi^W(k))$. 则有

$$\Delta D_{TC}(k) = \max\{S_t(\pi^W(k)) - S_t^{\text{noS}}(\pi^W(k)), 0\}.$$

第 $k$ 次求解局部调度子问题后,$\pi_{\text{sub}}(k)$ 为滚动窗口中前 $W-1$ 个工件按式(13) 求得的最优调度解 $\pi_{\text{sub}}^{W-1}(k)$ 加上 $\pi^W(k)$ 所得的排列.

### 3.1.4　滚动时域调度的全局性能分析

设在动态调度中局部子调度共滚动求解 $H$ 次,即 $k = 1, 2, \cdots, H$. 如果在最后阶段滚动窗口中的工件数不足 $W$,则对所有剩下的工件进行调度求解. 至此,整个调度过程结束.

**定理 1**　在带全局罚函数的滚动调度下,有

$$D_C(\pi^W(k)) + D_{TC}(\pi_{\text{after\_sub}}(k)) \le D_C^{\text{noS}}(\pi^W(k)) + D_{TC}^{\text{noS}}(\pi_{\text{after\_sub}}(k)) + (|\pi_{\text{after\_sub}}(k)| + 1)\Delta D_{TC}(k). \quad (14)$$

**证明**　令 $\pi_{\text{after\_sub}}(k) = \{\pi_{af}^1(k), \cdots, \pi_{af}^{|\pi_{\text{after\_sub}}(k)|}(k)\}$,$D_C^{\text{noS}}(\pi_{af}^i(k))$ 为工件 $\pi_{af}^i(k)$ 在求出第 $k$ 次子调度之前的完工时间. 当求出子调度后,导致 $\pi^W(k)$ 和滚动窗口之后所有工件的开始加工时间均推迟 $\Delta D_{TC}(k)$. 这时为最差情况,最差指标为

$$D_C^{\text{worst}}(\pi^W(k)) + D_{TC}^{\text{worst}}(\pi_{\text{after\_sub}}(k)) = D_C^{\text{noS}}(\pi^W(k)) + \Delta D_{TC}(k) + \sum_{j}^{|\pi_{\text{after\_sub}}(k)|} (D_C^{\text{noS}}(\pi_{af}^i(k)) + \Delta D_{TC}(k)) = D_C^{\text{noS}}(\pi^W(k)) + \sum_{j}^{|\pi_{\text{after\_sub}}(k)|} D_C^{\text{noS}}(\pi_{af}^i(k)) + (|\pi_{\text{after\_sub}}(k)| + 1)\Delta D_{TC}(k) = D_C^{\text{noS}}(\pi^W(k)) + D_{TC}^{\text{noS}}(\pi_{\text{after\_sub}}(k)) + (|\pi_{\text{after\_sub}}(k)| + 1)\Delta D_{TC}(k). \quad (15)$$

因为

$$D_C(\pi^W(k)) + D_{TC}(\pi_{\text{after\_sub}}(k)) \le D_C^{\text{worst}}(\pi^W(k)) + D_{TC}^{\text{worst}}(\pi_{\text{after\_sub}}(k)),$$

所以定理得证. □

**定理 2**　在带全局罚函数的滚动调度下,有

$$D_{TC}(\pi_{\text{sub}}^{W-1}(k)) + (|\pi_{\text{after\_sub}}(k)| + 1)\Delta D_{TC}(k) \le D_{TC}^{\text{noS}}(\pi_{\text{noS\_sub}}^{W-1}(k)). \quad (16)$$

证明 由于 $\pi_{sub}^{W-1}(k)$ 为滚动窗口中前 $W-1$ 个工件按式(13)求得的最优调度解,则有

$$D_{TC}(\pi_{sub}^{W-1}(k)) + (|\pi_{after\_sub}(k)|+1)\Delta D_{TC}(k) \leq D_{TC}(\pi_{W-1}(k)) + (|\pi_{after\_sub}(k)|+1)\Delta D_{TC}(k). \quad (17)$$

在式(17)中,若取 $\pi_{W-1}(k) = \pi_{noS\_sub}^{W-1}$,则有

$$D_{TC}(\pi_{W-1}(k)) = D_{TC}^{noS}(\pi_{noS\_sub}^{W-1}(k)),$$

$\Delta D_{TC}(k) = 0$. 代入式(17),则定理得证. □

**定理 3** 在带全局罚函数的滚动调度下,有

$$D_{TC}(\pi_g(k)) \leq D_{TC}^{noS}(\pi_{g\_noS}(k)). \quad (18)$$

证明 将式(14)和(16)相加,再用式(11)和(12)进行合并,可得

$$D_{TC}(\pi_{sub}(k)) + D_{TC}(\pi_{after\_sub}(k)) \leq D_{TC}^{noS}(\pi_{sub}(k)) + D_{TC}^{noS}(\pi_{after\_sub}(k)). \quad (19)$$

由于 $D_{TC}(\pi_{before\_sub}(k)) = D_{TC}^{noS}(\pi_{before\_sub}(k))$,式(19)两边分别加上 $D_{TC}(\pi_{before\_sub}(k))$ 和 $D_{TC}^{noS}(\pi_{before\_sub}(k))$ 后,不改变原不等式关系,则有

$$D_{TC}(\pi_{before\_sub}(k)) + D_{TC}(\pi_{sub}(k)) + D_{TC}(\pi_{after\_sub}(k)) \leq D_{TC}^{noS}(\pi_{before\_sub}(k)) + D_{TC}^{noS}(\pi_{sub}(k)) + D_{TC}^{noS}(\pi_{after\_sub}(k)). \quad (20)$$

将式(9)和(10)代入(20),则定理得证. □

定理 3 在 $k=1$ 和 $k=H$ 时仍然成立. 当 $k=1$ 时,$D_{TC}^{noS}(\pi_{before\_sub}(1)) = D_{TC}(\pi_{before\_sub}(1)) = 0$;当 $k=H$ 时,$\Delta D_{TC}(H) = 0$.

**定理 4** 在带全局罚函数的滚动调度下,有

$$D_{TC}(\pi_{ini}) \geq D_{TC}(\pi_g(1)) \geq D_{TC}(\pi_g(2)) \geq \cdots \geq D_{TC}(\pi_g(k)) \geq \cdots \geq D_{TC}(\pi_g(H)). \quad (21)$$

证明 由于 $D_{TC}(\pi_{ini}) = D_{TC}^{noS}(\pi_{g\_noS}(1))$,$D_{TC}(\pi_g(k)) = D_{TC}^{noS}(\pi_{g\_noS}(k+1))$,由定理 3 得

$$D_{TC}(\pi_{ini}) = D_{TC}^{noS}(\pi_{g\_noS}(1)) \geq D_{TC}(\pi_g(1)) = D_{TC}^{noS}(\pi_{g\_noS}(2)) \geq \cdots \geq D_{TC}(\pi_g(k)) = D_{TC}^{noS}(\pi_{g\_noS}(k+1)) \geq \cdots \geq D_{TC}(\pi_g(H-1)) = D_{TC}^{noS}(\pi_{g\_noS}(H)) \geq D_{TC}(\pi_g(H)).$$

因此定理得证. □

由定理 4 知,每次求出局部子调度后,全局调度都不劣于求出局部子调度之前的全局调度. 随着滚动子调度问题的不断求解,全局调度的调度指标相对于初始调度能不断得到改善,保证了整个调度过程的稳定性. 初始调度的性能指标是整个滚动调度过程中性能指标的上界.

### 3.2 混合差分进化算法

差分进化算法(DE)[18,19]是一种自适应、自组织和自学习的迭代寻优过程,采用实数编码、基于差分的变异操作和一对一的竞争生存策略,避免了复杂的遗传操作. 它可根据当前种群的个体间差异动态调整其搜索策略,具有较好的全局寻优能力和内在的隐并行性. DE 算法是在实数空间进行搜索,对其研究和应用主要集中于连续优化领域,并已取得了大量的成果[20].

有的学者对 DE 进行改进,成功地用于求解组合优化问题[17,21-23]. 例如一种混合差分进化算法(HDE)[17]被成功地用于求解流水线调度问题,它根据调度问题解空间结构信息,利用 DE 的并行搜索能力进行全局搜索,并用紧凑的邻域结构进行局部搜索. 通过大量标准测试问题的测试,验证了 HDE 是一种高效和鲁棒的算法. 本文在求解局部调度子问题时,采用了 HDE 算法.

## 4 仿真与分析

本节通过数值仿真对所提出的滚动调度策略进行分析和验证. 仿真测试问题用类似于王冰等[16]的方式产生,工件加工时间 $p(\pi_j,k)$ 在 1 ~ 10 间一致均匀分布,工件到达时间 $r(\pi_j)$ 在 1 ~ $50.5n\alpha$ 间一致均匀分布. 其中:$n$ 为工件数;$\alpha$ 为表示工件到达快慢的参数,$\alpha$ 取值小则表示工件到达快. 本文中 $\alpha$ 取值为 0.2,0.4,0.6,0.8,1.0,1.5. 测试问题的规模 $n \times m$ 取 300 ×10,300 ×20,500 ×10,500 ×20 计 4 组.

为了验证滚动调度策略,考虑一类特殊问题,即对每个 $n \times m$ 问题,随机选取一台机器,认为这台机器上 20 % 的工件由于某种不确定性(如工艺不稳定或机器出现故障等)导致加工时间需要延长. 亦即在随机选择的某台机器上,需要延长加工时间的 20 %$n$ 个工件服从 1 ~ $n$ 之间的均匀分布. 对应的加工时间为 250 + rand[1,50],其中 rand[1,50] 表示 1 ~ 50 间均匀分布的随机数.

仿真硬件环境为:Pentium IV 3.0 GHz,RAM 1.0 G;软件平台为 Windows XP 操作系统,Delphi 7.0 编程环境. 针对不同的 $\alpha$,每一问题分别测试 10 次(即每次测试使用不同的随机生成的问题参数).

仿真比较时采用两种策略:一种是本文提出的使用全局罚函数的滚动调度策略,用 GPRS 表示;另一种是不使用全局罚函数的滚动调度策略,用 RS 表示. 就是每次求解局部调度子问题时,对滚动窗口中 $W$ 个工件使用 $\min\limits_{\pi_W(k)}[D_{TC}(\pi_W(k))]$ 为调度指标,其中 $\pi_W(k)$ 为滚动窗口中所有工件的任意排列.

考虑两组滚动参数 $\{W,E\} = \{7,3\}$ 和 $\{W,E\} = \{16,3\}$. 对于 $\{W,E\} = \{7,3\}$,由于规模较小,使用穷举所有排列来求调度解;对于 $\{W,E\} = \{16,3\}$,由于无法在合理时间内穷举,使用 HDE 运行 30

代来求解.

### 4.1　特殊问题的测试

考虑上述特殊问题,测试结果如表 1 和表 2 所示. 其中

$$ARI = [D_{TC}(\pi_{ini}) - avg\_D_{TC}(\pi_g(H))]/D_{TC}(\pi_{ini}) * 100\%,$$

表示平均相对提高百分比,$avg\_D_{TC}(\pi_g(H))$ 为 $D_{TC}(\pi_g(H))$ 的平均值.

表 1　特殊问题在滚动参数{7,3}下两种策略的对比

| {7,3} α | 300 ×10 | | 500 ×10 | |
|---|---|---|---|---|
| | RS ARI | GPRS ARI | RS ARI | GPRS ARI |
| 0.2 | **12.197** | 9.865 | **8.342** | 7.137 |
| 0.4 | 9.472 | **9.554** | 6.685 | **7.014** |
| 0.6 | 7.717 | **9.438** | 5.636 | **6.951** |
| 0.8 | 8.264 | **9.239** | 5.974 | **6.921** |
| 1.0 | 6.458 | **7.430** | 4.996 | **5.930** |
| 1.5 | 1.332 | **1.342** | 1.004 | **1.007** |
| | 300 ×20 | | 500 ×20 | |
| 0.2 | **12.983** | 10.627 | **8.861** | 7.801 |
| 0.4 | 10.891 | **10.344** | 7.623 | **7.664** |
| 0.6 | 9.939 | **10.267** | 7.071 | **7.624** |
| 0.8 | 9.121 | **10.197** | 6.581 | **7.580** |
| 1.0 | 7.514 | **8.859** | 5.671 | **6.870** |
| 1.5 | 1.204 | **1.271** | 1.270 | **1.429** |
| average | 8.091 | **8.203** | 5.810 | **6.161** |

表 2　特殊问题在滚动参数{16,3}下两种策略的对比

| {16,3} α | 300 ×10 | | 500 ×10 | |
|---|---|---|---|---|
| | RS ARI | GPRS ARI | RS ARI | GPRS ARI |
| 0.2 | **29.472** | 28.522 | **19.903** | 19.382 |
| 0.4 | 27.240 | **27.737** | 18.706 | **19.171** |
| 0.6 | 25.478 | **26.067** | 18.014 | **18.600** |
| 0.8 | 22.072 | **22.891** | 16.458 | **17.434** |
| 1.0 | 15.710 | **16.251** | 13.847 | **14.553** |
| 1.5 | 2.051 | **2.111** | 1.184 | **1.218** |
| | 300 ×20 | | 500 ×20 | |
| 0.2 | **31.059** | 29.220 | **20.590** | 19.615 |
| 0.4 | 28.352 | **28.642** | 19.304 | **19.475** |
| 0.6 | 26.899 | **27.148** | 18.241 | **18.923** |
| 0.8 | 22.690 | **23.341** | 16.813 | **17.844** |
| 1.0 | 15.052 | **15.937** | 13.250 | **14.823** |
| 1.5 | 1.745 | **1.770** | 1.272 | **1.300** |
| average | 20.652 | **20.803** | 14.798 | **15.195** |

由表 1 可以看出,除 $\alpha = 0.2$ 外,GPRS 均好于 RS,这说明对于该类特殊问题 GPRS 更为有效.

图 2 为 3 种策略下对 300 ×10 问题在滚动参数取{7,3} 和 $\alpha = 1$ 时的测试结果. 其中:横坐标为滚动子调度问题求解的次数 $k$,纵坐标为滚动调度每次求解对应的全局调度指标 $D_{TC}(\pi_g(k))$; No_strategy 表示不采取任何策略,即按到达的先后顺序加工,这时对应的全局调度指标为 $D_{TC}(\pi_{ini})$.

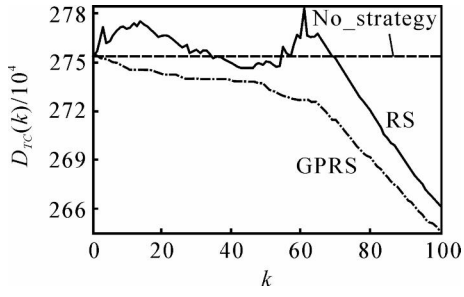


图 2　300 ×10 特殊问题在取{7,3} 和 α = 1 时的结果

由图 2 可见,GPRS 在局部调度中考虑了全局指标,可保证 $D_{TC}(\pi_g(k+1)) \geq D_{TC}(\pi_g(k))$ (定理 4),因此整个曲线呈下降趋势,整体性能最好;RS 只采用滚动策略,并未考虑局部对全局的影响,因此整个曲线出现振荡,有时所得全局指标甚至大于 $D_{TC}(\pi_{ini})$. 其他问题的测试结果与图 2 类似.

由表 2 可以看出,除 $\alpha = 0.2$ 外,GPRS 均好于 RS,这再次验证了 GPRS 的有效性. 由表 1 和表 2 可见,滚动参数取{16,3} 时,性能比取{7,3} 时明显要好,这说明可预知的工件数越多,越应增加滚动窗口的长度. 滚动策略在处理工件快速到达的问题时优势明显,但随着 $\alpha$ 的增大,工件的到达时间变慢,使得滚动策略的优势减弱. 另外,HDE 虽然不能象穷举所有排列那样来保证所得的调度解为最优,但作为一种高效和鲁棒的算法,它能保证得到满意解.

在滚动参数{7,3} 和{16,3} 下,分别使用穷举和 HDE 求解调度子问题的平均运行时间如表 3 所示. 可以看出,HDE 所增加的时间并不多,远小于{16,3} 下穷举的时间. 因此在求解动态调度问题时,需要针对具体问题,将滚动策略与高效算法有机结合才能有效、实时地解决问题. 另外,采用滚动机制使得问题得以分解,每个子问题的规模都较小,其求解时间一般在1 s以内. 在实际的化工、冶炼等行业,工件的加工时间和到达时间一般以小时或天来

表 3　滚动参数{7,3} 和{16,3} 下运行 HDE 30 代求解调度子问题的平均运行时间　s

| 滚动参数 | 300 ×10 和 500 ×10 | | 300 ×20 和 500 ×20 | |
|---|---|---|---|---|
| | RS | GPRS | RS | GPRS |
| {7,3} | 0.142 | 0.020 | 0.148 | 0.022 |
| {16,3} | 0.903 | 0.888 | 0.946 | 0.922 |

计量,故调度子问题的求解时间可以忽略.

采用 RS 时,评价指标 $D_{TC}(\pi_W(k))$ 的计算复杂度为 $O(Wm)$. 采用 GPRS 时,$D_{TC}(\pi_{W-1}(k))$ 的计算复杂度为 $O(Wm)$, $\Delta D_{TC}(k)$ 的计算复杂度为 $O(m)$,故评价指标 $D_{TC}(\pi_{W-1}(k)) + (|\pi_{after\_sub}|(k)+1)\Delta D_{TC}(k)$ 的计算复杂度仍为 $O(Wm)$.

当滚动参数取$\{7,3\}$ 时,由于是用穷举求解调度子问题,则采用 RS 时评价次数为 $7!$,相应的总体评价复杂度为 $O(7!\times 7m)$;采用 GPRS 时评价次数为 $6!$,相应的总体评价复杂度为 $O(6!\times 7m)$. 算法的运行时间主要取决于总体评价复杂度,故采用 GPRS 求解调度子问题的平均运行时间约为采用 RS 时的 1/7. 这与表 3 中的数据相吻合.

当滚动参数取$\{16,3\}$ 时,由于是用 HDE 求解调度子问题,则采用 RS 和 GPRS 时的评价次数相同,两种策略下 HDE的总体评价复杂度均为 $O(K\times 16m)$,其中 $K$ 为 HDE 的总体评价次数. 虽然采用 RS 和采用 GPRS 时评价指标的计算复杂度相同,但采用 GPRS 求解子问题的规模相对较小,相应评价指标的实际计算量也略小. 因此,采用 GPRS 求解调度子问题的平均运行时间比采用 RS 时要小. 这与表 3 中的数据相吻合.

需要指出的是,采用 GPRS时,计算 $\Delta D_{TC}(k)$ 需要用到 $S_t^{noS}(\pi^W(k))$,而 $S_t^{noS}(\pi^W(k))$ 的计算复杂度为 $O(Wm)$. 在每次求解调度子问题时,$S_t^{noS}(\pi^W(k))$ 只在开始时执行一次,故对算法总体评价复杂度的影响可以忽略. 由上述分析可知,对于特殊问题,在调度指标中引入全局罚函数,不仅能改进求解质量,而且可减少子问题的求解时间.

### 4.2 一般问题的测试

考虑所有工件加工时间 $p(\pi_j,k)$ 在 1～10 间一致均匀分布,测试结果如表 4 和表 5 所示.

由表 4 和表 5 可见,RS 和 GPRS 性能相近,RS 略好些. 这说明对于一般问题,对滚动窗口中前 $W-1$ 个工件最小化 $D_{TC}(\pi_{W-1}(k))$,可能导致第 $W$ 个工件的开始加工时间推后,但由于工件的加工时间均匀分布,这种推后对滚动窗口后的工件影响并不明显. 与特殊问题相比,随着 $\alpha$ 的增大,滚动策略作用的减弱更加明显.

3 种策略对 300 ×10 一般问题在滚动参数取$\{7,3\}$ 和 $\alpha=1$ 时的测试结果如图 3 所示. 定理 4 保证了 GPRS 对应的曲线一定下降,RS 和 GPRS 对应的曲线几乎重合. 因此,对于工件加工时间均匀分布的问题,采用 RS 便可解决. 从表 4 和表 5 也可看出,采用参数{16,3} 下的滚动策略与 HDE 相结合,其性能更加优越. 这与 4.1 节的结论是一致的.

表 4 一般问题在滚动参数$\{7,3\}$ 下两种策略的对比

| $\{7,3\}$ $\alpha$ | 300 ×10 RS ARI | 300 ×10 GPRS ARI | 500 ×10 RS ARI | 500 ×10 GPRS ARI |
|---|---|---|---|---|
| 0.2 | **18.536** | 18.219 | **20.377** | 19.658 |
| 0.4 | **0.218** | 0.215 | **0.115** | 0.113 |
| 0.6 | **0.027** | 0.026 | **0.025** | 0.024 |
| 0.8 | **0.019** | **0.019** | **0.010** | **0.010** |
| 1.0 | **0.008** | **0.008** | **0.005** | **0.005** |
| 1.5 | **0.003** | **0.003** | **0.002** | **0.002** |
| | 300 ×20 | | 500 ×20 | |
| 0.2 | **23.944** | 21.920 | **24.222** | 22.305 |
| 0.4 | **1.457** | 1.439 | **0.688** | 0.681 |
| 0.6 | **0.073** | 0.071 | **0.073** | 0.072 |
| 0.8 | **0.036** | **0.036** | **0.022** | 0.021 |
| 1.0 | **0.017** | **0.017** | **0.010** | **0.010** |
| 1.5 | **0.008** | **0.008** | **0.005** | **0.005** |
| average | **3.695** | 3.498 | **3.796** | 3.576 |

表 5 一般问题在滚动参数$\{16,3\}$ 下两种策略的对比

| $\{16,3\}$ $\alpha$ | 300 ×10 RS ARI | 300 ×10 GPRS ARI | 500 ×10 RS ARI | 500 ×10 GPRS ARI |
|---|---|---|---|---|
| 0.2 | **20.820** | 20.781 | **21.459** | 21.464 |
| 0.4 | 0.199 | **0.200** | **0.151** | **0.151** |
| 0.6 | **0.027** | **0.027** | **0.018** | **0.018** |
| 0.8 | **0.020** | **0.021** | **0.013** | **0.013** |
| 1.0 | **0.011** | **0.011** | **0.007** | **0.007** |
| 1.5 | **0.005** | **0.005** | **0.002** | **0.002** |
| | 300 ×20 | | 500 ×20 | |
| 0.2 | **29.415** | 29.078 | **30.109** | 29.585 |
| 0.4 | 0.798 | **0.800** | 0.564 | **0.565** |
| 0.6 | **0.072** | **0.072** | 0.049 | **0.050** |
| 0.8 | **0.032** | **0.032** | **0.024** | **0.024** |
| 1.0 | **0.025** | **0.025** | **0.015** | **0.015** |
| 1.5 | **0.008** | **0.008** | **0.004** | **0.004** |
| average | **4.286** | 4.255 | **4.368** | 4.325 |

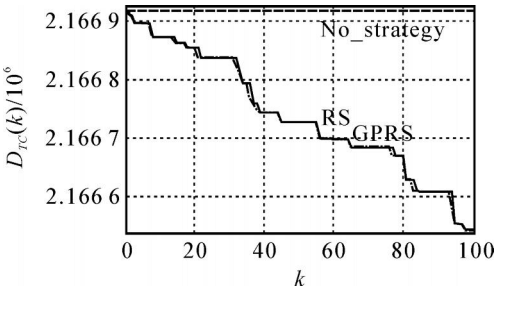


图 3 300 ×10 一般问题在$\{7,3\}$ 和 $\alpha=1$ 时的结果

## 5　结　　论

滚动调度策略边预测边调度的思想既适应工件信息逐渐获得的动态调度,又能控制局部调度子问题的规模.对于工件动态到达的 NWFSP,本文提出一种带全局罚函数的滚动调度策略,并从理论上证明了这一策略能逐渐改善全局调度性能.在动态车间调度问题中,问题模型、调度策略和求解算法是密切相关的.本文在研究中将这三者综合考虑,突出了基于问题模型的调度策略,采用高效和鲁棒的智能算法 HDE 来求解局部调度子问题.大量的仿真实验表明,所提出的策略和算法能有效处理动态零等待流水线调度问题.

## 参考文献( References)


[1] Rock H. The three-machine no-wait flow shop is NP-complete[J]. J of the ACM, 1984, 31(2): 336-345.

[2] Herroelen W, Leus R. Project scheduling under uncertainty: Survey and research potentials [J]. European J of Operational Research, 2005, 165(2): 289-306.

[3] 钱晓龙, 唐立新, 刘文新. 动态调度的研究方法综述[J]. 控制与决策, 2001, 16(2): 141-145.
(Qian X L, Tang L X, Liu W X. Dynamic scheduling: A survey of research methods[J]. Control and Decision, 2001, 16(2): 141-145.)

[4] Wang J. A fuzzy robust scheduling approach for product development projects [J]. European J of Operational Research, 2004, 152(1): 180-194.

[5] Daniels R L, Carrillo J E. Beta-robust scheduling for single-machine systems with uncertain processing times [J]. IIE Trans, 1997, 29(11): 977-985.

[6] Kouvelis P, Daniels R L, Vairaktarakis G. Robust scheduling of a two-machine flow shop with uncertain processing times[J]. IIE Trans, 2000, 32(5): 421-432.

[7] Jensen M T. Generating robust and flexible job shop schedules using genetic algorithms[J]. IEEE Trans on Evolutionary Computation, 2003, 7(3): 275-288.

[8] Sabuncuoglu I, Bayiz M. Analysis of reactive scheduling problems in a job shop environment[J]. European J of Operational Research, 2000, 126(3): 567-586.

[9] Wu S D, Storer R H, Chang P C. One-machine rescheduling heuristics with efficiency and stability as criteria[J]. Computers & Operations Research, 1993, 20(1): 1-14.

[10] Ruedee R R, William G, Ferrell J, et al. Dynamic rescheduling that simultaneously considers efficiency and stability[J]. Computers & Industrial Engineering, 2004, 46(1): 1-15.

[11] Guo B, Nonaka Y. Rescheduling and optimization of schedules considering machine failures [J]. Int J of Production Economics, 1999, (60/61): 503-513.

[12] Alagoz O, Azizoglu M. Rescheduling of identical parallel machines under machine eligibility constraints [J]. European J of Operational Research, 2003, 149(3): 523-532.

[13] Wang B, Lai X P, Xi L F. Single machine partial rescheduling with bi-criterion based on genetic algorithm [J]. Lecture Notes in Computer Science, 2005, 3644: 947-956.

[14] Szelke E, Kerr R M. Knowledge-based reactive scheduling [J]. Production Planning and Control, 1994, 5(2): 124-145.

[15] 席裕庚. 动态不确定环境下广义控制问题的预测控制[J]. 控制理论与应用, 2000, 17(5): 665-670.
(Xi Y G. Predictive control of general control problems under dynamic uncertain environment [J]. Control Theory & Applications, 2000, 17(5): 665-670.)

[16] Wang B, Xi Y G, Gu H Y. Terminal penalty rolling scheduling based on an initial schedule for single-machine scheduling problem [J]. Computers & Operations Research, 2005, 32(11): 3059-3072.

[17] Qian B, Wang L, Hu R, et al. A hybrid differential evolution for permutation flow-shop scheduling[J]. Int J of Advanced Manufacturing Technology, 2008, 38(7/8): 757-777.

[18] Storn R, Price K. Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces[J]. J of Global Optimization, 1997, 11(4): 341-359.

[19] Price K, Storn R, Lampinen J. Differential evolution: A practical approach to global optimization [M]. Berlin: Springer, 2005.

[20] 刘波, 王凌, 金以慧. 差分进化算法研究进展[J]. 控制与决策, 2007, 22(7): 721-729.
(Liu B, Wang L, Jin Y H. Advances in differential evolution[J]. Control and Decision, 2007, 22(7): 721-729.)

[21] Tasgetiren M F, Liang Y C, Sevkli M, et al. Differential evolution algorithm for permutation flowshop sequencing problem with makespan criterion [C]. Proc of 4th Int Symp on Intelligent Manufacturing Systems. Sakarya, 2004: 442-452.

[22] Onwubolu G, Davendra D. Scheduling flow-shops using differential evolution algorithm[J]. European J of Operational Research, 2006, 171(2): 674-692.

[23] Qian B, Wang L, Huang D X, et al. Scheduling multi-objective job shops using a memetic algorithm based on differential evolution [J]. Int J of Advanced Manufacturing Technology, 2008, 35(9/10): 1014-1027.