DOI：10.3901/JME.2013.14.182

基于改进滚动时域优化策略的动态调度方法*

刘国宝[1] 张 洁[1, 2]
(1. 上海交通大学机械与动力工程学院 上海 200240；
2. 上海交通大学机械系统与振动国家重点实验室 上海 200240)

摘要：通过考虑带有并行机的 Job Shop 调度过程中设备故障对生产的影响，以最大完工时间和任务提前/拖期时间加权最小为目标建立数学模型。基于滚动时域优化基本框架，设计改进的事件和周期混合驱动的重调度机制。改进的事件驱动机制能过滤掉不必要的重调度，周期驱动机制能跟踪系统的变化，二者结合充分发挥两类驱动机制各自的优势。针对基本蚁群算法(Ant colony system algorithm，ACS)搜索时间长、易陷入局部最优解的缺点，引入精英蚂蚁策略和最大最小蚂蚁机制，设计改进蚁群系统算法(Improved ant colony system algorithm, IACS)。通过与 ACS、遗传算法(Genetic algorithm，GA)的对比试验验证了 IACS 算法的优越性。该方法作为离散制造车间生产调度系统的组成部分在上海某军工企业生产车间实施，用实际生产数据验证动态调度方法的有效性。

关键词：并行机 混合驱动 重调度 蚁群算法

中图分类号：TH165

Dynamic Schedule Method Based on Improved Rolling Time Domain Optimization Strategy

LIU Guobao[1] ZHANG Jie[1, 2]
(1. School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai 200240;
2. State Key Laboratory of Mechanical System and Vibration, Shanghai Jiao Tong University, Shanghai 200240)

Abstract：Analyzed the impact of machine breakdowns in job shop schedule problem with parallel machines, a mathematical model with minimizing the weighted value of makespan and early due date as optimization goal is established. Based on the basic framework of the rolling time domain optimization, a hybrid driven schedule mechanism combining events and cycle is designed. Improved event driven mechanism filters out unnecessary scheduling, meanwhile, cycle driving mechanism can track system changes. The combination can give full play to their strengths. Due to the long searching time and easy falling into local optimum solution of Ant colony algorithm(ACS), an improved ant colony system algorithm(IACS) mixing elite ant strategy and max-min ant mechanism is designed. The contrast experiment with ACS and genetic algorithm(GA) verified the superiority of IACS. The dynamic scheduling method is implemented as a constituent part of the scheduling system in discrete manufacturing workshop of a military enterprise in Shanghai, and the validity is verified by the production data.

Key words：Parallel machines Hybrid driven Dynamic schedule Ant colony algorithm

0 前言

实际生产过程中工件处理时间、设备状况、订单等参数是不确定或者未知的，按照确定环境下建立的模型进行优化计算得到的方案在实施的时候，可能不再是最优，甚至不再可行。因此如何处理动态事件影响下生产调度问题在理论界和工程界得到了越来越多的关注。当前针对动态事件影响下的调度问题通常采用的措施是重调度，重调度策略主要分为完全反应式和预测反应式两种[1]。

预测反应式调度是离散型制造车间进行重调度的常用策略[1]，它包含两个基本步骤：① 不考虑车间层未来的动态事件生成一个预调度方案；② 由

* 国家自然科学基金(60934008)和国家高技术研究发展计划(863 计划，2012AA040907)资助项目。20120829 收到初稿，20130116 收到修改稿

一定的驱动机制触发，对预调度方案进行更新，重调度可以保持调度方案的可行性或者改善调度性能。滚动调度是一种基于预测的调度方法，方剑等[2]针对 Job Shop 调度问题提出了基于工件的滚动调度的方法，主要的研究内容在于怎样确定滚动窗口，以及怎样进行再调度以保证生产的连续性。预测反应式调度方案的关键在于设计触发机制并采用合理的方法进行重调度。YAMAMOTO 等[3]提出了三阶段的重调度策略，首先根据车间当前信息在新调度开始之前生成预调度方案，对车间内现有资源进行预调度；然后每当有新工序开始加工或者工序加工完毕时，对比实际调度与预调度方案之间的偏差；最后根据系统与预调度方案的偏差对原方案进行修订，对未加工工件进行重调度。滚动调度策略作为一种处理大规模调度问题的计算复杂性和动态不确定性的有效方法，正受到越来越多的关注，而滚动调度策略中触发调度窗口滚动的滚动机制正成为近年来动态生产调度研究的热点。

ABUMIZAR 等[4]提出一种重调度算法，当扰动发生时，不是全部重新也不是等扰动解除后再继续原调度，而是仅仅重新调度那些直接或间接受扰动影响的工件，减少了由于扰动引起的生产周期的增加，同时也减少了与初始调度的偏差，效率和稳定性都较好。LIU 等[5]做了大量仿真试验，从仿真数据中产生出训练样本用于训练神经网络，并将训练后的神经网络用于动态调度。JONES 等[6]提出一种解决实时排序和调度问题混合方法的框架，综合运用了神经网络、遗传算法和实时仿真等方法。CHEN 等[7]采用一种具有固定时间间隔的周期性策略来进行调度，并结合遗传算法，采用基于任意键编码的方法寻找一个调度使得生产闲余时间和延迟提前惩罚最小，取得了较好的效果。启发式算法通常仅对一个目标提供可行解，并且缺乏对整体性能的有效把握和预见能力；人工智能方法能够在系统当前状态和给定优化目标的基础上，对自身的知识库进行搜索，选择最优的调度策略，但开发周期长且成本高；仿真方法对于复杂制造系统的描述较为理想，但仿真结果的可信度严重依赖仿真模型、仿真方法及仿真试验输入数据，而且很难从特定的试验中提炼出一般的规律性，通用性差。近年来，群集智能方法[8-9]由于具有普遍适用性和较低的经验复杂性等优点，得到了广泛的重视，它不需要进行大量的概率计算，具有很强的并行性。

制造系统中动态事件可分两大类。① 资源相关：设备故障、工人旷工、设备负荷限制、物料缺陷等；② 任务相关[9]：紧急插单、订单取消、交货期变更、优先级变更、加工时间变化等。动态事件的出现使得制造系统的调度是一个动态的过程，需要相应的方法来解决该类问题。本文研究带有并行机的 Job Shop 调度过程中考虑设备故障的情况，设定原材料供应满足任务需求，任务加工时间、交货期不变，并行机效率各不相等。建立响应故障扰动的动态生产调度模型，在滚动时域优化框架下，设计改进的动态调度策略及时响应状态的变化，并将蚁群系统用于车间调度，设计高效的动态调度算法，实现生产过程稳定和调度结果有效。

1　动态调度数学模型

离散加工车间按设备功能分为 M 个加工区，每个加工区内有 K_m 台非等效并行机；车间可同时加工 V 种不同类型的零件且零件具有各自不同的加工工序 J_v，每道工序对应一个设备组并且工序预先确定；待排产任务数为 I，每个任务包含 Q_{iv} 个相同类型的零件，每个任务具有独立交货期 d_i、提前/拖期惩罚系数 α_i/β_i；加工过程中，每个任务不重复访问同一设备组。

本文将各调度目标进行合理简化，从任务完工时间、交货期满意度、生产成本和设备利用率等方面的要求建立重调度窗口内的优化目标。多目标的优化设计目标为：确定每一个任务的每一道工序在某一具体时间开始设备上加工，保证产品准时交货水平下使得生产的成本最低，并满足以下假设条件：每个零件在固定时刻只能在一台设备上加工，任何零件没有强占加工的特权；需要重调度时，正在加工的工序不受影响，继续加工直到该工序完成。

目标函数

$$f=\omega_1\min\sum_{i=1}^{I}F_i+\omega_2\min\sum_{i=1}^{I}(\alpha_iE_i+\beta_iD_i) \tag{1}$$

$$\omega_1+\omega_2=1$$

$$E_i=\max\{0,d_i-F_i\}$$

$$D_i=\max\{0,F_i-d_i\}$$

式中，E_i 为任务提前惩罚；D_i 为任务拖期惩罚。

约束条件如下所述。

完工时间约束

$$F_i=\max\sum_{m=1}^{M}\sum_{k=1}^{K_m}\sum_{j=1}^{J_i}(t_{ijmk}^{E}) \tag{2}$$

式中，t_{ijmk}^{E} 为任务 i 的工序 j 在设备组 m 内设备 k 的结束加工时间；J_i 为重调度时预调度方案中任务

i 尚未完工的工序数。

动态扰动约束

$$\delta = \min \sum_{i=1}^{I} \sum_{j=1}^{J_i} \left(\left| t'_{ij} - t_{ij} \right| \right) \tag{3}$$

式中，t'_{ij} 为重调度方案中任务 i 的第 j 道工序的开工时间；t_{ij} 为预调度方案中任务 i 的第 j 道工序的开工时间。

设备平衡约束

$$\varphi = \min \sum_{m=1}^{M} \sum_{k=1}^{K_m} \left[\sum_{i=1}^{I} \sum_{j=1}^{J_i} (T_{vjmk}^{P}) - T_{\text{avg}} \right] \tag{4}$$

$$X_{ijmk} + X_{jimk} = 1 \tag{5}$$

式中，T_{vjmk}^{P} 为零件 v 的工序 j 在设备组 m 内设备 k 的加工时间；T_{avg} 为所有设备的平均加工时间；X_{ijmk} 为 0-1 变量，1 表示任务 i 在设备组 m 中的设备 k 上先于任务 j 加工。

设备产能约束

$$\sum_{j=1}^{\mathrm{J}} \sum_{i=1}^{I} (t_{ijmk}^{E} - t_{ijmk}^{D}) \le C_{mk}^{p} \tag{6}$$

式中，C_{mk}^{p} 为设备组 m 内设备 k 在调度周期内最大负荷。

加工时间约束

$$t_{ijmk}^{E} \le t_{ijmk}^{S} + Q_{iv} T_{vjmk}^{P} \tag{7}$$

$$t_{i_1,jmk}^{E} \le t_{i_2,jmk}^{S} - T_{vjmk}^{S} \quad \forall j,m,k,v_2 \in A(v_1) \tag{8}$$

式中，t_{ijmk}^{S} 为任务 i 的工序 j 在设备组 m 内设备 k 的开始加工时间。T_{vjmk}^{S} 为表示零件 v 的工序 j 在设备组 m 内设备 k 的加工准备时间。

2 基于混合驱动的滚动调度策略

滚动时域优化是预测控制中滚动优化原理的一种具体表现形式。预测控制与传统的最优控制不同在于：它不是遍及全过程的优化，也不是只进行一次优化，而是一种有限时段的滚动优化。在每一个采样时刻，优化性能指标只涉及从该时刻起未来有限时间，而到下一个采样时刻，这一优化时刻同时向前推移，如此反复滚动进行。在动态不确定环境下，考虑远时段内的优化是没有意义的，而且由于环境变化的不可预知性，优化需要反复进行，因此滚动优化思想对于动态环境具有特殊的针对性。

滚动窗口技术是滚动时域优化的核心。在应用滚动窗口技术进行动态调度时，首先定义完工窗口、调度加工窗口及等待窗口三个任务窗口，如图 1 所示。等待窗口存放所有的等待进行调度的任务，完工窗口存放所有进行加工完毕的任务，调度加工窗口指滚动优化原理中的优化时域窗口，从等待窗口中选取一定数量的任务放入调度加工窗口，进行调度之后这些被调度的任务按照调度结果进行加工。调度加工窗口包括了正在加工任务集及未加工任务集，正在加工任务集是经过调度的且已经处于加工过程中的任务集合，未加工任务集是经过调度的但还未开始加工的任务集合。重调度时可利用在加工过程中所获取的局部信息选取一定数量的任务进行调度优化，构建重调度优化集。滚动时域优化方法是一种基于模型和优化的控制，它能跟踪系统的变化，在时间上不断推动该优化集的滚动，应用调度优化算法求解调度子问题，通过不断优化局部最优来实现全局优化。

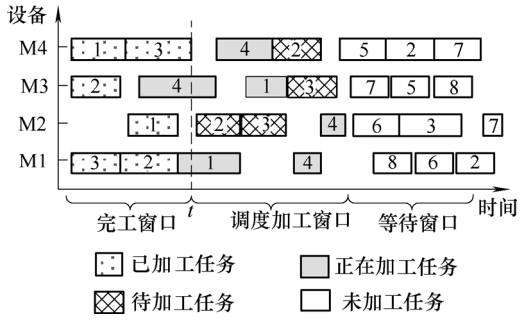


图 1 任务窗口与任务集关系

滚动调度策略应用的关键是选择什么样的滚动机制来确定信息预测窗口和滚动优化窗口向前滚动的方式，也即如何确定下一个决策点以及在此决策点所考虑的任务集合。在周期性滚动机制中，间隔期选则过大则求解问题的规模增大同时无法快速响应系统中发生的干扰，过小则调度过于频繁影响了系统的稳定性。如果将满足条件的动态因素定义为关键事件，当关键事件发生时，采用事件驱动滚动机制，没有关键事件发生时，采用周期性滚动机制，充分发挥两类滚动机制各自的优势，本文考虑设备故障作为关键事件。

传统的事件驱动机制容易忽略加工过程可缓冲部分扰动的情况，产生不必要的重调度。本文通过建立损益模型对事件驱动机制进行改进，并对模型定量分析以确定设备故障时是否启动重调度，过滤不必要的重调度。损益模型在优化目标的基础上衡量新方案可行性。当有扰动出现，新的调度结果或许比原结果更优，但是调度方案调整的消耗有可能无法补偿扰动发生后仍执行原调度方案带来的损失。由于优化目标函数越小越优，所以优化目标函

数值越大则损失越大，建立模型

$$F_{pl} = f_a(t) - f_b(t) - \Delta f(t) \tag{9}$$

式中，$f_a(t)$为原方案执行结果的目标函数值，$f_b(t)$为新方案执行结果的目标函数值，$\Delta f(t)$为重调度实施损失函数，任务搬运、机器调整等带来的损失；定义 $\Delta f(t)=T_i^{f,k}+T_i^g+T_i^m$，$T_i^g$ 为装卸任务所用时间，T_i^m 为机器调整所用时间，$T_i^{f,k}$ 为任务 i 由机器 f 到机器 k 的搬运时间。

当 $F_{pl}>0$ 时，针对扰动的发生，新的调度结果与调度实施带来的损失的和仍然优于原调度结果，则按新调度方案进行。

当 $F_{pl}\le 0$ 时，针对扰动的发生，新的调度结果也许优于原调度结果，但是在机器调整以及搬运任务等调整过程中所造成的损失难以补偿调度新方案带来的收益，则保持原调度方案。

调度加工窗口内的任务数量和待加工任务的选取规则会影响调度结果和调度的整体效率。由于本文采用事件和周期混合滚动机制，因此滚动窗口采用基于时间的窗口。在当前 t 时刻按照局部调度结果进行加工，经过一定时间 ΔT 之后，将所有已经完工的任务移出窗口，再从等待加工的任务中选入若干任务进入窗口，在 $t+\Delta T$ 开始进行新的调度决策。在图 2 中，滚动窗口时间步长为 ΔT，预测窗口步长为 ΔT_p，$W(l)$为第 l 个预测窗口，$P_S(l)$表示完工窗口，$F_S(l)$表示预测窗口之外等待窗口。

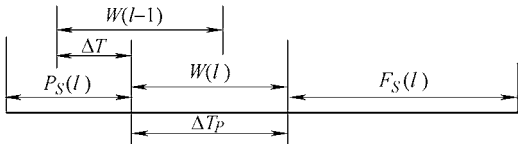


图 2　基于时间的窗口滚动示意图

在周期性滚动机制下，滚动调度的次数和系统对动态因素的适应能力由 ΔT 决定；ΔT 过小，计算量是减少了，但由对于全局信息考虑较少，就可能导致全局性能的下降；ΔT 过大，要获得最优调度则存在计算上的困难，而采用启发式方法求子问题的次优解，又可能降低最终全局解的性能。由于 ΔT 即优化集的最大有效优化容量选取没有一个固定的标准，一般根据问题特性和对算法效率的要求来指定其大小，与调度算法的优化规模紧密相关。在给定 ΔT 的情况下，判断那些工序进入到滚动窗口是很重要的。下面设计动态调度优化算法，按规则确定滚动窗口中的任务排序。

3　改进的蚁群算法

目前蚁群算法在动态调度研究中应用还不多，但由于其具有原理简单、通用性强、受限制条件的约束较小等特点，在动态调度问题中的研究前景被广泛看好。针对基本蚁群算法搜索时间长、易陷入局部最优解[11-13]的缺点，本文提出改进蚁群算法，分别采用精英策略和最大最小蚂蚁机制对基本蚁群算法进行改进。

任务选择加工设备构建解序列的过程中，蚂蚁采用伪随机比例的状态转移规则来选择设备组 m 内设备 k_m，规则由式(10)给出。α、β 两个参数，分别为信息素启发式因子和期望启发式因子，决定了信息素和启发式信息的相对影响力，即解的构建过程中所累积的信息与未来启发式信息在蚂蚁路径选择中的相对重要性。q 是均匀分布在[0,1]的一个随机变量。

$$k_m = \begin{cases} \arg\max\limits_{m\in K_m}\left\{\left(\tau_{j,k_m}^i\right)^\alpha\left(\eta_{j,k_m}^i\right)^\beta\right\} & q<q_0 \\ S & \text{其他} \end{cases} \tag{10}$$

蚂蚁选择当前可能的最优移动方式继续开发已有路径的概率为 q_0，同时蚂蚁以概率$(1-q_0)$有偏向性的探索新的路径，蚂蚁对任务 i 游历到工序 j 时选择设备组 m 内设备 k_m 的概率 S 由式(11)决定。φ 为可选设备集；τ_{j,k_m}^i 为任务 i 的工序 j 和设备 k_m 之间的信息素水平；$\eta_{j,k_m}^i=\left(C_{P,k_m}-C_{k_m}\right)\Big/T_{vjmk}^P$ 为任务与设备之间的启发式信息，设备初始设定一个较大的可承载负荷 C_{P,k_m}，随着各工序设备确定，C_{k_m} 动态减小，η_{j,k_m}^i 使已承载负荷少且加工时间短的设备被选中作为加工设备的概率大。

$$P_{j,k_m}^i=\frac{\left(\tau_{j,k_m}^i\right)^\alpha\left(\eta_{j,k_m}^i\right)^\beta}{\sum_{l\in\varphi}\left(\tau_{j,k_m}^i\right)^\alpha\left(\eta_{j,k_m}^i\right)^\beta} \tag{11}$$

蚂蚁已经选择的工序节点和不符合加工顺序约束的工序节点被放人禁忌表中，禁忌表中的工序节点不能作为蚂蚁下一步选择的工序节点；蚂蚁每走完一步，就应该对走过路径上的信息素进行挥发，避免其他蚂蚁走相同的路径以致陷入局部最优解，因此需要更新局部信息素。局部信息素更新方式如式(12)，局部信息素蒸发率 ξ 满足 $0<\xi<1$，$\tau_0=\tau_{\max}$ 是信息素初始值。

$$\tau_{j,k_m}^i \leftarrow (1-\xi)\tau_{j,k_m}^i+\xi\tau_0 \tag{12}$$

为加快收敛速度，而又不影响全局优化能力，在较短的时间内找到最优解，本文采用基于精英蚂蚁排序策略的全局信息素更新方法。该方法保留全局的最优解，在每次迭代完成后，只将求解结果中排名前几位的精英蚂蚁用于信息素更新，将精英蚂蚁所经路径按从小到大的顺序排列，即 $L^1(t)\le L^2(t)\le\cdots\le L^{\sigma}(t)$，并根据路径长度赋予不同的权重，路径长度越短权重越大，则全局信息素更新规则如式(13)。式中 $\Delta\tau^r_{ij}(t)=1/L^r(t)$，$\sigma$ 为精英蚂蚁个数，ω 为全局最优解的个数，$\omega-r$ 为第 r 个全局最优解的权重。

$$\tau^i_{j,k_m}(t+1)\leftarrow(1-\rho)\tau^i_{j,k_m}(t)+\sum_{r=1}^{\sigma}(\omega-r)\Delta\tau^r_{ij}(t)+\omega\Delta\tau^{\text{best}}_{ij}(t) \tag{13}$$

为避免算法过早收敛于并非全局最优的解，最大最小蚂蚁机制将各条路径可能的信息素浓度限制于[$\tau_{\min}$，$\tau_{\max}$]，超过这个范围的值按式(14)重设

$$\begin{cases}\tau^i_{j,k_m}(t)=\tau_{\max} & \tau^i_{j,k_m}(t)>\tau_{\max}\\ \tau^i_{j,k_m}(t)=\tau_{\min} & \tau^i_{j,k_m}(t)<\tau_{\min}\end{cases} \tag{14}$$

当出现新的全局最优解时，更新 $\tau_{\min}$ 和 $\tau_{\max}$，更新规则如式(15)所示，C 为常量，$0<\rho<1$ 为全局信息素挥发因子，L^{best} 为当前全局最优解长度，σ 为精英蚂蚁个数。

$$\begin{cases}\tau_{\max}(t)=(1+\sigma)/2(1-\rho)L^{\text{best}}\\ \tau_{\min}(t)=\tau_{\max}(t)/20\end{cases} \tag{15}$$

综上所述，对于 n 个任务 m 个设备组 m_k 台设备的带有非等效并行机的 Job Shop 调度，基于改进蚁群算法的动态调度算法详细流程如下，算法流程图如图 3 所示。

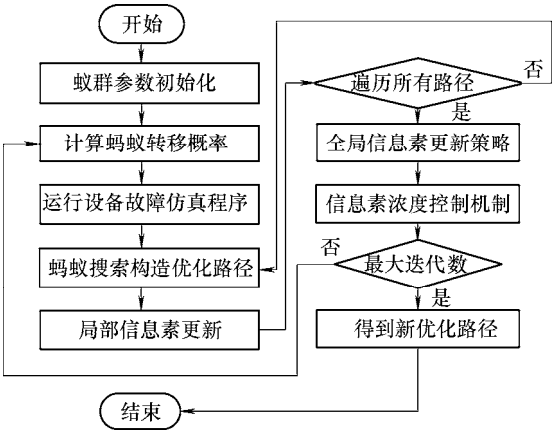


图 3 基于改进蚁群算法的动态调度算法流程

步骤 1：蚁群参数初始化参数。任务 i 与任意设备 k_m 间信息素初始值设定为 $\tau_{\max}$；N_{iter} 为迭代次数，N_{ants} 为蚂蚁数量，N 为任务数；令迭代项 I=1。

步骤 2：任取一只蚂蚁，另 s=1。

步骤 3：任务加工开始时间和加工完成时间分别构成开始、完成时间矩阵，蚂蚁 s 在进行搜索之初可选工序集合 D_s 为所有任务的第一个工序所组成的集合。

(1) 蚂蚁 s 按照状态转移规则选择 D_s 中任务 i 的工序 j 对应设备，蚂蚁每走完一步，走过路径上的信息素便进行蒸发。

(2) 将此工序安排在相应设备上，并更新开始时间矩阵和完成时间矩阵，将与该工序同属一个任务的紧后工序放入 D_s 中，替代掉该工序，置 i=i+1。

(3) 对于 D_s 中的所有工序，按照式(11)计算蚂蚁 s 选择设备的概率并安排在相应设备上，更新开始时间矩阵和完成时间矩阵。

(4) 若该工序 j 为所属任务的最后一个工序，则从 D_s 中将该工序 j 删去，否则将与该工向同属一个任务的紧后工序放入 D_s 中，替代掉该工序 j。D_s 中与工序 j 不在同一设备上的工序，在下一次为任务 i+1 选择工序时，不需重复计算。若 i=N，转步骤 4，否则，令 i=i+1，转步骤 3 中的(3)。

步骤 4：若 $s<N_{\text{ants}}$，令 s=s+1，转步骤 3 中(1)，否则，转步骤 5。

步骤 5：比较各蚂蚁搜搜路径，采用精英蚂蚁策略进行全局信息素更新。

步骤 6：若出现新的全局最优解，则更新信息素浓度阀值，同时更新 $\tau_{\min}$，$\tau_{\max}$，否则，转步骤 7。

步骤 7：若 $I<N_{\text{iter}}$，令 I=I+1，转步骤 2，否则，结束。

4 实例仿真与分析

4.1 试验数据

采用上海某军工研究所加工车间生产数据进行仿真试验，基于 Microsoft Visual Studio 2008 为开发平台，算法采用 C#语言编程实现，在 Core2 E7400 2.80 GHz,1.98 GB 的环境进行。该企业是典型的军工制造基础生产单元，以离散式生产为主、流程生产为辅、装配生产为重点，企业产品部件较多，批量小。提供测试数据的车间主要进行结构件的加工，车间以班组为单位组织生产，班组按设备功能划分，各班组有数量不等的非等效并行机。任务信息如表 1 所示，各零件的加工工序对应的设备组及加工时间分别见表 2、3。

表 1　任务基本信息表

任务编号 i	交货期 d_i	提前惩罚 α_i	拖期惩罚 β_i	零件编号 v	数量 q_{iv}
JP-A101	150	3	1	TF750VE-0A	10
JP-A102	150	0	1	VT0303-150	10
JP-A103	150	0	2	PJ26.00-900	10
JP-A104	150	3	1	SLCF-X15C	10
JP-A201	150	1	2	PJ26.00-900	5
JP-A202	150	0	3	SLCF-X15C	5

表 2　零件加工工序对应的设备组

零件编号	零件名称	设备组			
		工序 1	工序 2	工序 3	工序 4
TF750VE-0A	附加台面	镁铝加工	立铣	磨光	淬火
VT0303-150	温控接头	镁铝加工	磨光	混砂	淬火
PJ26.00-900	挡板本体	混砂	立铣	磨光	—
SLCF-X15C	横梁体	立铣	混砂	镁铝加工	—

表 3　零件各工序加工时间　　min

零件编号	镁铝加工	立铣	磨光	混砂	淬火
TF750VE-0A	5	3	4	—	2
VT0303-150	4	—	3	5	4
PJ26.00-900	—	7	6	6	—
SLCF-X15C	4	5	—	3	—

考虑到任务搬运时间，为了简化计算，这里忽略不同零件的重量，假设搬运工具承载不同任务都是速度都相同，设备组信息如表 4 所示，设备组间任务搬运时间、设备的调整时间如表 5 所示。以制造期和任务提前/拖期完工时间最短为优化目标，且试验中取 $\omega_1=\omega_2=0.5$。设调度周期内设备故障次数服从参数为 $\lambda=0.5$ 的泊松分布，设备维修时间服从参数为 $\lambda=0.5$ 的指数分布。改进蚁群算法的参数设置如下：蚂蚁数 $N_{ants}=50$，迭代次数 $N_{iter}=50$，信息素启发式因子 $\alpha=1$，期望启发式因子 $\beta=3$，整体信息素蒸发速率 $\rho=0.1$，局部信息素蒸发率 $\xi=0.1$。

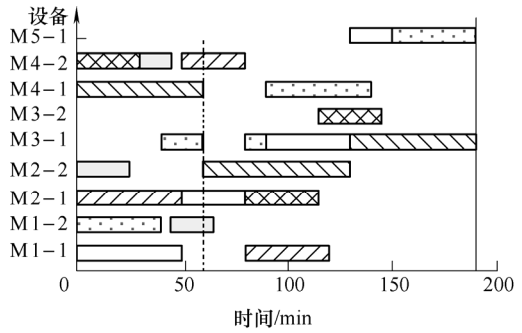
表 4　设备组信息表

设备组编号	M-1	M-2	M-3	M-4	M-5
设备组名称	镁铝加工	立式铣床	磨光机	混砂机	淬火炉
组内设备数	3	3	2	2	1

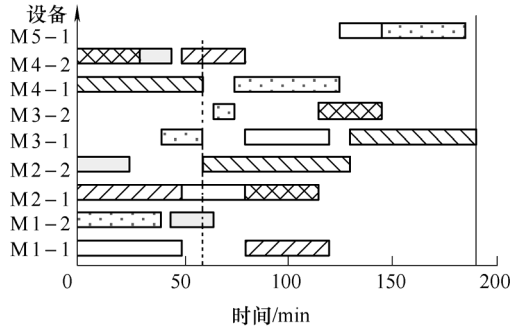
表 5　任务在各设备组间搬运时间　　min

设备组编号	调整时间	M-1	M-2	M-3	M-4	M-5
M-1	0.8	0	1	2	3	2
M-2	0.6	1	0	1	4	3
M-3	0.3	2	1	0	5	2
M-4	0.5	3	4	5	0	4
M-5	0.8	2	3	2	4	0

以下选取两次试验进行说明：在 t=60 min 时刻磨光机设备组内设备 M-1 出现故障，故障修复时间为 20 min。启动预调度方案，继续按原调度方案进行调度的结果 $f_a(t)$=405，重调度后的调度结果 $f_b(t)$=425，重调度实施损失函数 $\Delta f(t)$=3，由式(9)求得损益值 F_{pl}=-23<0，故障设备的维修时间被设备空闲时间缓冲掉，所以仍按原调度进行结果更佳，调度结果如图 4 所示。



(a) 按原调度执行结果



(b) 按重调度方案执行结果

图 4　t=60 min 时刻 M3-1 故障调度方案甘特图

□任务1 □任务2 ▨任务3 ▨任务4 ▨任务5 ▨任务6

在 t=110 min 时刻磨光机设备组内设备 M3-1 出现故障，故障修复时间为 20 min。启动预调度方案，继续按原调度方案进行调度的结果 $f_a(t)$=455，重调度后的调度结果 $f_b(t)$=410，重调度实施损失函数 $\Delta f(t)$=4，由式(9)求得损益值 F_{pl}=41>0，所以进行重调度，调度结果如图 5 所示。

根据本文的动态调度策略，满足损益函数条件的设备故障发生时，采用事件驱动机制，其余时间采用周期性滚动机制。在周期性滚动机制下，滚动

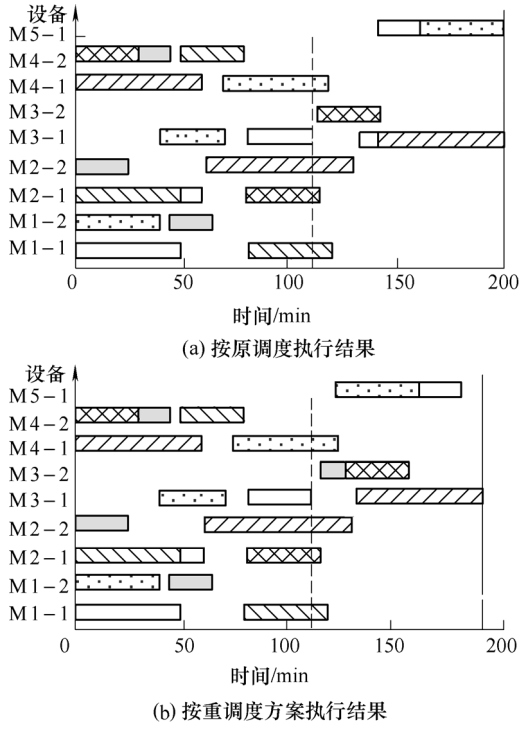


图 5 t=110 min 时刻 M3-1 故障调度方案甘特图

□任务1 □任务2 ▨任务3 ▨任务4 ▨任务5 ■任务6

调度的次数和系统对动态因素的适应能力由滚动窗口时间步长 ΔT 决定，设置不同步长分别进行试验，验证 ΔT 与算法的优化规模的相关性。为了使试验结果更具说服力，试验在保持零件种类、加工工序不变的情况下选取任务数量 N=100，对应的零件索引服从均匀分布 U[1,4]，任务交货期随机生成。假设在 t=10 min 时刻设备组 M3 设备 M3-1 出现故障，故障修复时间为 10 min，根据响应故障的损益模型得损益值 F_{pl}=-57<0，因此按原调度执行得到目标函数值为 3 742，制造期为 1 482 min。

由表 6 试验结果可知，当步长取 ΔT=300～500，滚动次数为 3～5 次时，动态调度的整体效果最好，计算整体耗时最少、目标函数较其他最优、设备利用率最高。分析试验结果可知：滚动调度中优化集滚动步长会影响调度结果和调度的整体效率。ΔT 过小，局部子调度问题的规模减小，优化集内计算量减少，但对全局信息考虑较少，全局性能降低，设备利用率降低；另外 ΔT 过小会使周期驱动的重调度过于频繁，反而整体耗时过大。ΔT 过大，子调度问题规模增大，降低了最终全局解的性能，设备利用率不高，且优化集内调度过程耗时过长，整体计算时间亦不甚理想。

4.2 动态调度策略可行性分析

通过大量仿真试验获得调度周期内不同故障时刻损益函数值，通过回归分析[14]确定设备故障影响下损益函数与故障时刻之间的关联，从定量分析的角度研究设备故障时刻对动态调度目标影响程度。由于对数据分析之前不知曲线类型，故以下采用多项式回归。假设多项式回归模型

$$y_t = \alpha + \beta_1 x_t + \beta_2 x_t^2 + \cdots + \beta_k x_t^k + \varepsilon_t \tag{16}$$

式中，x_t 表示发生故障的时刻，y_t 表示当前故障时刻下的损益函数值，且 β_1>0, β_2>0,…, β_k>0。令 $x_{t1}=x_t, x_{t2}=x_t^2, \cdots, x_{tk}=x_t^k$，式(16)变为多元一次的线性方程为

$$y_t = \alpha + \beta_1 x_{t1} + \beta_2 x_{t2} + \cdots + \beta_k x_{tk} + \varepsilon_t \tag{17}$$

式中，观测误差 $\varepsilon_t \sim N(0,\sigma^2)$，$y_t$ 的估计值 $\hat{y}_t = a + b_1 x_{t1} + b_2 x_{t2} + \cdots + b_k x_{tk}$。

根据最小二乘法原理，残差平方和 $\sum_{t=1}^{n}(y_t-\hat{y}_t)^2$ 达到最小的曲线为回归曲线，令 $S_{S_e}=\sum_{t=1}^{n}(y_t - a - b_1 x_{t1} - b_2 x_{t2} - \cdots - b_k x_{tk})^2$，要使 S_{S_e} 最小，应该有

$$\frac{\partial S_{S_e}}{\partial b_1}=0 \qquad \frac{\partial S_{S_e}}{\partial b_2}=0 \qquad \cdots \qquad \frac{\partial S_{S_e}}{\partial b_k}=0$$

求解整理后得

$$b_i = \frac{n\sum_{t=1}^{n} x_{ti}y_t - \sum_{t=1}^{n} x_{ti}\sum_{t=1}^{n} y_t}{n\sum_{t=1}^{n} x_{ti}^2 - (\sum_{t=1}^{n} x_{ti})^2} \tag{18}$$

$$a = \frac{1}{n}\left(\sum_{t=1}^{n} y_t - \sum_{i=1}^{k} b_i \sum_{t=1}^{n} x_{ti}\right) \tag{19}$$

50 次仿真试验后，利用散点作图得回归曲线如图 6 所示。根据最小二乘法由式(18)、(19)求得 α=-60.65, β_1=1.04, β_2=-0.0036，回归方程为 y_t=-60.65+1.04x_t-0.003 6x_t^2。

表 6 不同滚动窗口步长调度结果对比表

步长	滚动次数/次	计算时间/s	制造期/min	目标函数值	设备利用率(%)
100	16	409	1 552	4 214	52
200	8	133	1 505	3 920	57
300	5	92	1 310	3 328	77
400	4	73	1 292	3 396	81
500	3	65	1 259	3 529	82
600	3	67	1 286	3 540	78
700	2	73	1 379	3 632	73
800	2	82	1 495	3 776	65
1000	2	80	1 526	3 828	61
1500	1	86	1 482	3 742	58

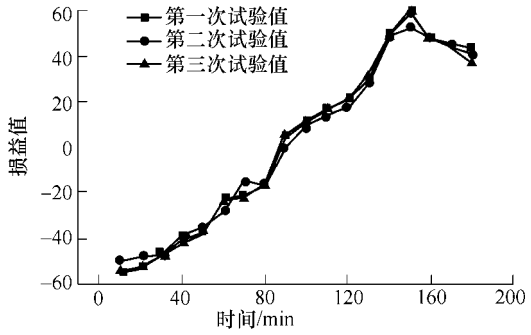


图 6　损益函数回归曲线

根据图 6 分析，调度周期内设备故障时刻与损益函数之间回归曲线是二次多项式，若在前期发生设备故障，一般会保持原调度方案，因为故障设备的维修时间一般会在设备空闲时间中缓冲掉，此时如果进行重调度是没有必要的，反而增加成本；若设备故障在后期发生，也多趋向于选择保持原调度方案，由于这时多数任务基本按原调度方案加工完成，此时设备调整以及搬运任务等过程所造成的损失难以补偿新方案带来的收益。

4.3　动态调度算法性能试验

为测试动态调度改进蚁群算法的性能，改进蚁群系统算法(Improved ant colony system algorithm, IACS)除与基本蚁群算法(Ant colony system algorithm, ACS)进行对比外，还采用李俊芳[15]求解多阶段非等效并行机调度问题的遗传算法(Genetic algorithm, GA)作为对比算法。GA 采用基于工序的编码方式，分两步进行解码。取交叉概率 P_c=0.60，变异概率 P_v=0.001，种群规模为 100，最大迭代次数设为 100。蚁群算法参数设置及试验数据参照第 4.1 节。分别设置设备组数 M=5，任务数量 N=10，随机生成 6 组算例，其中任务对应的零件编号服从均匀分布 U[1, 4]，以计算时间、调度目标值、首次收剑点为对比标准，计算结果如表 7 所示。

表 7　不同动态调度算法求解结果对比表

算例	计算时间/s			调度目标值/min			首次收敛点/次		
	IACS	ACS	GA	IACS	ACS	GA	IACS	ACS	GA
A	5.64	8.78	11.32	422	434	437	32	45	50
B	6.27	8.02	12.22	428	430	436	29	50	56
C	5.06	9.42	10.36	432	438	438	32	52	62
D	5.70	8.52	10.63	420	434	432	28	48	58
E	6.26	9.76	11.26	428	448	460	34	43	63
F	5.94	9.44	11.32	432	448	454	33	38	68

通过表 7 对比可以看出，IACS 和 ACS 计算时间明显优于 GA，其中以 IACS 最优，由于蚁群算法不需要进行大量的概率计算，用蚁群算法来构造车间动态调度可以明显地提高整个调度系统的计算性能。由于蚁群算法的正反馈机制，在调度目标值方面，IACS 和 ACS 均优于 GA，IACS 基于精英蚂蚁策略针对传统蚁群算法贪婪式搜索缺点做出改进后，求解质量明显提高。随着迭代的深入，算法收敛性趋于平稳， ACS 首次收敛点一般在第 50 次左右出现，IACS 通过引入最大最小蚂蚁机制和精英蚂蚁策略，首次收敛点提前到 30 左右；而 GA 首次收敛点大多集中在 60 左右，这也是蚁群算法在计算时间方面优于遗传算法的原因。

本文动态调度方法在上海某军工研究所生产车间试用过程中，车间生产过程中铸件中关重件工序间平均等待时间由原来的 25 min 减少到 20 min，减少 20%；同类产品(如大型铸件)交付周期由原来的 30 d 缩短到 25 d，缩短 16%，客户满意度提高 6%。在工业现场的实际应用进一步证明了动态调度策略的可行性和算法的有效性。

5　结论

(1) 针对带有非等效并行机的 Job Shop 动态调度问题，建立了考虑设备故障情况下实现最大完工时间和任务提前/拖期时间加权最小的动态调度多目标优化模型。

(2) 基于滚动时域优化基本框架，设计了改进事件和周期混合驱动的重调度机制，基于大量试验数据，采用回归分析方法从理论上验证了动态调度策略的可行性。

(3) 引入精英蚂蚁策略和最大最小蚂蚁机制，设计基于改进蚁群算法的动态调度算法 IACS，并通过与 ACS、GA 对比试验验证了算法的优越性。

(4) 企业应用表明，该方法用于复杂非等效并行机制造系统的动态调度问题是有效且可行的。

参 考 文 献

[1] HUANG Y G, KANAL L N, TRIPATHI S K. Reactive scheduling for a single machine：Problem definitiong analysis and heuristic solution[J]. International Journal of Computer Integrated Manufacturing，1990, 3(1)：6-12.

[2] 方剑, 席裕庚. 基于遗传算法的滚动调度策略[J]. 控制理论与应用, 1997, 14(4)：1-8.
FANG Jian, XI Yugeng. The genetic algorithm-based rolling horizonscheduling sreategy[J]. Control Theory & Applications, 1997, 14(4)：1-8.

[3] YAMAMOTO M, NOF S Y. Scheduling/rescheduling in the manufacturing operating system environment[J]. International Journal of Production Research，1985, 23(4)：705-722.

[4] ABUMAIZAR R J, SVESTKA J A. Rescheduling job shops under random disruptions[J]. International Journal of Production Research, 1997, 35(7)：2065-2082.

[5] LIU H J, DONG J. Dispatching rule selection using artificial neural networks for dynamic planninnand scheduling[J]. Journal of Intelligent Manufacturing, 1996, 7(2)：243-250.

[6] JONES A, RABELO K Y, YUEHWERN Y. A hybrid approach for real-time sequencing and scheduling[J]. International Journal of Materials and Product Technology, 1995, 8(2)：145-154.

[7] CHEN K J, JI P. A genetic algorithm for dynamic advanced planning and scheduling (DAPS) with afrozen interval[J]. Expert Systems with Applications, 2006, 33(4)：1004-1010.

[8] 鞠全勇, 朱剑英. 多目标批量生产柔性作业车间优化调度[J]. 机械工程学报，2007, 43(8)：148-154.
JU Quanyong, ZHU Jianying. Multi-objective optimization of batch production soft job shop scheduling[J]. Chinese Journal of Mechanical Engineering, 2007, 43(8)：148-154.

[9] CHRISTOFOROS C, KRZYSZTOF F, KHALIL S H. A hybrid searching method for the unrelated parallel machine scheduling problem[C]//6th IFIP WG 12.5 International Conference, AIAI 2010, Larnaca, Cyprus, October 6-7, 2010, 2010：230-237.

[10] 黄洪钟, 李丽. 并行工程中设计任务的动态调度[J]. 机械工程学报，2002, 38(增刊)：164-167.
HUANG Hongzhong, LI Li. Dynamic scheduling of design tasks in concurrent engineering[J]. Chienese Journal of Mechanical Engineering, 2002, 38(Suppl.)：164-167.

[11] CHANG P T, LIN K P, PAI P F. Ant colony optimization system for a multi-quantitative and qualitative objective job-shop parallel machine scheduling problem[J]. International Journal of Production Research, 2009, 46(20)：5719-5759.

[12] ARNAOUT J P, RABADI G, MUSA R. A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times[J]. Journal of Intelligent Manufacturing, 2010, 21：693-701.

[13] JOLAI F, AMALNICK M S, ALINAGHIAN M. A hybrid memetic algorithm for maximizing the weighted number of just-in-time jobs on unrelated parallel machines[J]. Journal of Intelligent Manufacturing, 2011, 22：247-261.

[14] 冯勤. 基于回归数据挖掘预测系统的分析与研究[D]. 天津：天津大学, 2005.
FENG Qin. Analysis and research on regression data mining forecast system [D]. Tianjin：Tianjin University, 2005.

[15] 李俊芳, 尹兆涛. 带有并行机的混合 Job Shop 调度问题[J]. 中国管理信息化, 2010, 13(14)：65-67.
LI Junfang. YIN Zhaotao. Analysis hybrid job shop scheduling problem with paralle machines[J]. China Management Informationization, 2010, 13(14)：65-67.

作者简介：刘国宝(通信作者)，男，1986 年出生。主要研究方向为制造系统的建模与调度优化。

E-mail：liuguobao.jlu@hotmail.com

张洁，女，1963 年出生，博士，教授，博士研究生导师。主要研究方向为计算机集成制造系统。

E-mail：jie.zhang@.sjtu.edu.cn