紧密衔接工序组联动的综合调度算法

谢志强 [1, 2] 滕宇峥 [2] 杨 静 [1]

摘 要 由于现有的工序间存在紧密衔接条件的复杂产品综合调度问题, 采用的移动交换算法不易于软件实现且没有考虑移动工序后产生的连锁反应引起较高算法复杂度的问题, 提出将具有紧密衔接约束条件的工序组进行统一联动的综合调度算法. 该算法利用将具有紧密衔接约束条件的工序分组的扩展加工工艺树模型, 按路径上属于工序组的工序个数多少确定所在路径工序组调度的次序, 通过降低对工序组的限制要求降低算法复杂度; 对于被调度工序组中各工序的前序工序, 按工序组中工序的加工顺序确定调度次序, 对某个工序的前序工序采用复杂度较低的拟关键路径法确定工序的调度次序; 调度完所有紧密衔接工序组后, 剩余的标准工序按拟关键路径法确定调度顺序; 采取工序首次适应调度算法调度标准工序和工序组, 由于工序组中工序采取按序紧密衔接的联动调度方式确定工序组的开始时间, 避免了二次调整, 进一步降低了算法复杂度. 分析和实例表明, 所提出的综合算法比以往算法复杂度更低, 调度结果更优且更易于实现.

关键词 综合调度, 紧密衔接, 工序组, 标准工序, 首次适应调度算法

DOI 10.3724/SP.J.1004.2011.00371

Integrated Scheduling Algorithm with No-wait Constraint Operation Group

XIE Zhi-Qiang [1, 2] TENG Yu-Zheng [2] YANG Jing[1]

Abstract The movement and exchange algorithm can hardly resolve the existing complex products scheduling problem with no-wait constraint between operations. An integrated scheduling algorithm is proposed which makes no-wait constraint operation group (NWCOG) into linkage. The algorithm uses the expansion processing tree to make the operations with no-wait constraint into operation group, and determines the scheduling order of the operation group according to the number of operations which belong to NWCOG on path. Then lower complexity is obtained by reducing the restrictions of the NWCOG. For the preorder operations of each operation scheduled in the group, the scheduling order is determined by the processing order of operations in the group. For preorder operation of some operation, the scheduling order is determined by the allied critical path method (ACPM) which has lower complexity. After each of NWCOG is finished scheduling, the remained standard operations are scheduled by ACPM. Then, the first fit scheduling method (FFSM) is used to schedule the standard operations and the NWCOG. As the start time of the operation group is determined by the linkage-scheduling method, the secondary adjustments can be avoided and the complexity can be reduced. Analysis and examples show that the proposed algorithm has a lower complexity, better scheduling results and can be easier to implement.

Key words Integrated scheduling, no-wait constraint, operation group, standard operation, first fit scheduling method (FFSM)

现代工业中的生产调度主要有两种形式: 1) 对于大批量相同产品, 学者们提出一些针对加工或装配流水线的调度算法, 如张长胜等的混合算法[1] 和胡蓉等的差分进化算法[2] 等; 2) 对于多品种小批量产品, 学者们提出有关车间调度的算法, 如闫利军等的混合优化算法[3] 和 Wei 等的基于强化学习的作业车间动态调度方法[4] 等. 这些算法的主要特点是先在加工生产线上加工产品的各工件, 然后再在装配线上组装产品.

当要求连续完成的紧前工序的结束时间与紧后工序的开始时间相同时, 属于工序间存在紧密衔接. 实际生产中最常见的例子如化学工业生产中的混合配料, 当某一配料加工完成之后, 必须马上进入下一道工序, 工序之间不能存在时间空隙, 以防化学配料的变质, 影响产品质量; 用模具加工铁制品, 必须趁钢水炽热时浇铸等. 由于工序间增加了限制条件, 不仅需要考虑工序间的前后顺序约束关系, 还要考虑

收稿日期 2010-03-09 录用日期 2010-12-27
Manuscript received March 9, 2010; accepted December 27, 2010
国家自然科学基金 (60873019, 61073043), 黑龙江省自然科学基金 (F200901), 中国博士后科学基金 (20090460880), 黑龙江博士后科学基金 (LBH-Z09214), 哈尔滨市优秀学科带头人项目 (2010RFXXG054) 资助
Supported by National Natural Science Foundation of China (60873019, 61073043), Natural Science Foundation of Heilongjiang Province (F200901), China Postdoctoral Science Foundation (20090460880), Heilongjiang Province Postdoctoral Science Foundation (LBH-Z09214), and Harbin Outstanding Academic Leader Foundation of Heilongjiang Province of China (2010RFXXG054)
1. 哈尔滨工程大学计算机科学与技术学院 哈尔滨 150001 2. 哈尔滨理工大学计算机科学与技术学院 哈尔滨 150080
1. School of Computer Science and Technology, Harbin Engineering University, Harbin 150001 2. School of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150080

工序间前后无空闲时间的约束关系, 使工序间存在紧密衔接的调度问题的复杂度进一步加大[5].

对于以上调度形式中有少量工序紧密衔接的情况, 主要解决方法有混合整数规划方法[6]、拉格朗日松弛法[7]、局部探求记忆法[8]、滚动策略[9] 和免疫调度算法[10] 等.

随着社会对产品个性化需求的增加, 多品种小批量产品, 特别是单件产品的订单将会越来越多. 如果对单件产品进行先加工后装配的方式, 必然割裂产品加工和装配内在的并行关系, 影响产品的制造效率[11], 因此, 加工和装配一同处理的综合调度逐渐引起人们的关注.

综合调度属于更复杂的 NP-hard[12−13] 问题. 谢志强等提出了一系列树状结构复杂产品综合调度问题的解决方法, 主要有通过提出标准工序、延迟工序等概念, 使非紧密衔接工序变为一般工序, 解决动态非紧密衔接工序调度问题[14]; 提出最少加工时间的设备选择方法, 解决柔性调度问题[15]; 利用层优先法解决复杂产品动态问题[16] 等. 虽然这些方法较好地解决有关综合调度问题, 但工序间都不涉及紧密衔接的约束条件.

由于求解复杂产品紧密衔接的调度问题更加复杂, 相应的研究成果很少. 如熊禾根等提出了考虑工件间约束关系的调度算法[17], 但仅在传统车间调度中考虑了部分工件间的约束关系; Schuster 采用的禁忌搜索算法[18] 求解部分工件间有约束关系且工序间存在紧密衔接的调度问题. 由于这些方法只能解决部分工件间存在约束关系的调度问题, 很难用它们来求解所有工件间存在约束关系的复杂产品工序紧密衔接时的加工和装配综合调度问题.

为了解决复杂产品紧密衔接条件下的调度问题, 文献 [19] 提出首先对工序进行无紧密衔接条件的调度加工, 然后利用移动交换算法对于已经调度的工序进行重新排列, 来满足工序间紧密衔接的条件. 但在调度时仅考虑移动交换算法本身的时间复杂度, 并没有考虑其后续工序由于连锁反应而产生的较高的时间复杂度, 而且后续工序的约束条件可能导致移动交换算法并不适用, 因此不能解决一般情况下的紧密衔接调度问题.

本文在已有研究成果的基础上, 利用已有的扩展加工工艺树[19], 提出解决复杂产品工序间紧密衔接问题的新方法. 该方法通过提出紧密衔接工序组, 优先调度紧密衔接工序组中的工序及工序组的紧前工序, 将具有紧密衔接约束条件的工序组使用工序组首次适应调度算法进行统一联动, 标准工序按拟关键路径法调度. 实现工序间存在紧密衔接条件的复杂产品综合调度优化. 最后用实例验证了算法的优化效果.

1 问题描述

在实际生产中, 复杂产品的约束有工件内工序的加工次序和工件间的装配次序. 解决复杂产品具有紧密衔接条件的综合调度问题需要考虑工序间前后顺序的约束关系和部分工序的紧密衔接约束关系. 为了简化调度分析, 将加工设备和装配设备统一定义为资源设备, 将加工和装配统一定义为加工. 于是, 问题的数学描述为:

设产品有多个工件, 工件由多个工序组成, 共有 n 个工序, 在 m 个设备上加工, 一道工序在某一时刻只能被一台设备加工, 一台设备在某一时刻只能加工一道工序; 设备一旦加工某道工序, 则直到该工序加工完毕后, 这台设备才能加工其他工序; 每道工序都必须在其所有前序工序加工完后, 方可开始加工; 部分工序的结束时间必须为其后续工序开始时间. 每道工序的加工时间已知, 且与加工顺序无关; 允许设备在工序达到之前闲置. C_i 为工序 i 在其设备的结束时间, S_{ik}, T_{ik}, E_{ik} 分别为工序 i 在设备 k 上的开始时间、加工时间和完工时间. 所以, 数学模型为

$$\begin{aligned}&\min\left(\max C_i\right),\quad i=1,2,\cdots,n\\&\text{s. t.}\\&\quad\min(S_{ik}) &(1)\\&\quad S_{ik}-E_{xy}\geq 0 &(2)\\&\quad S_{ik}-E_{(i-1)k}\geq 0 &(3)\\&\quad S_{ak}-E_{bk'}=0 &(4)\end{aligned}$$

式中, $i=1,2,\cdots,n$; k, $k'=1,2,\cdots,m$; 在设备 y 上调度加工的工序 x 是在设备 k 上调度加工工序 i 的工艺约束前序工序; 式 (1) 表示工序在尽可能早的开始时间加工; 式 (2) 表示任一在 k 设备上加工的工序 i 必须在其全部前序工序加工结束后开始加工; 式 (3) 表示同一设备 k 上第 i 道工序必须在第 $i-1$ 道工序完成后开始加工; 式 (4) 表示工序 a 和工序 b 是紧密衔接关系.

为了简化工序间存在紧密衔接约束的复杂产品的调度问题, 通过对工序间存在紧密衔接约束的复杂产品生产工艺进行约束分析, 定义如下:

定义 1 (标准工序). 不具有紧密衔接约束条件的工序.

定义 2 (工序组). 由多个工序组成的一组工序, 这些工序的结束时间或开始时间必须为其紧后或紧前工序的开始或结束时间.

定义 3 (工序组中工序联动调度). 在满足工序组的约束条件下, 按工序组中工序的约束关系依次连续调度, 确定该工序组的开始时间.

定义 4 (相关工序). 某个工序组中各工序的前序工序的总和.

2　复杂产品紧密衔接调度及加工算法设计

为了充分利用树状结构具有末端分支繁衍, 可通过简单的循环迭代确定各节点的特点[20], 采用将紧密衔接工序用虚线框标出的扩展加工工艺树[19]表示存在紧密衔接工序的产品.

为了缩短具有紧密衔接约束条件的复杂产品加工时间, 降低调度算法的时间复杂度, 采取从两个方面优化的策略, 第一个是确定工序的调度顺序, 第二个是确定工序在设备上的开始加工时间, 相应地提出了工序调度顺序的算法和确定工序开始加工时间的算法.

2.1　确定复杂产品工序的调度顺序的算法

由于具有紧密衔接条件的复杂产品中包含工序的结束时间为其后续工序开始时间的工序组, 调度工序组中的工序时, 需要连续考虑两个以上工序的开始加工时间, 所以工序组中工序对于加工设备的要求较高. 为了让工序组少受干扰, 减少加工设备由于过早地被占用而增加的额外限制, 考虑优先调度工序组中的工序.

由于工序组的相关工序是工序组的约束条件, 为了保证工序组的联动调度, 在确定优先调度工序组后, 按拟关键路径法[20] 优先调度该工序组的前序工序.

调度完所有的工序组后, 剩余的工序作为复杂产品加工工艺树的一部分, 按拟关键路径法[20] 确定工序的调度顺序.

2.1.1　确定工序组及其相关工序的调度顺序

复杂产品中具有紧密衔接工序的工序组不一定唯一, 当扩展加工工艺树中有多个工序组时, 需要判断工序组调度的先后顺序.

由于同一路径上紧密衔接工序的个数较多时, 该路径上的工序对加工设备的要求相对较高, 因此采用优先调度紧密衔接工序个数多的路径的策略, 减少其他路径上的工序对设备的占用, 使工序组尽早开始.

当不同路径上的工序组中紧密衔接工序总个数相同时, 相关工序少的工序组受相关工序影响较小, 优先调度相关工序少的工序组, 可减少影响工序组调度的工序数量.

如果相关工序依然相等, 根据拟关键路径法, 优先调度加工时间长的路径, 可以使影响产品总加工时间的主要工序尽早加工, 从而缩短产品的完工时间[20], 所以优先调度加工时间长的路径上的紧密衔接工序组.

由于工序组内的工序按约束顺序调度, 工序组中越早开始调度工序的开始时间对工序组的结束时间影响越大, 所以为了让工序组中先调度工序尽早开始, 按工序组中工序调度次序确定各前序工序的调度次序.

2.1.2　使用拟关键路径法确定剩余标准工序的调度顺序

当确定工序组的相关工序和工序组的调度顺序后, 按确定相关工序调度顺序的方法, 对于剩余的标准工序按拟关键路径法确定调度顺序.

2.1.3　复杂产品工序调度顺序的算法步骤

步骤 1. 确定各路径上工序组中工序的个数.

步骤 2. 选择未调度路径上工序组中工序个数最多的路径, 当所选路径不唯一时, 工序组相关工序少的路径优先选择.

步骤 3. 在选择的路径上, 确定计划调度的工序组.

步骤 4. 对计划调度的工序组中的工序, 按调度次序, 循环执行步骤 2 ∼ 4, 确定各工序前序工序中计划调度的工序组.

步骤 5. 对于前序工序中的标准工序, 采用拟关键路径法确定调度顺序.

步骤 6. 若某一工序组中的前序工序调度完毕, 对该工序组中的工序按约束次序连续调度. 循环 3, 4, 5, 6, 直到全部工序组调度完毕.

步骤 7. 确定全部工序组中工序的调度顺序后, 使用拟关键路径法确定剩余工序的调度顺序.

步骤 8. 结束.

2.1.4　复杂产品工序调度顺序算法的实现说明

步骤 1. 用链表 List O 表示原始复杂产品工艺树, 链表中节点 L 结构为

$$\mathrm{L}: \quad A, S, T, M, Q, P, r, T_s, T_b, T_e$$

其中, A 为工艺树中某节点的工序名, S 为指向工序 A 紧后工序的指针, T 为工序 A 的加工时间, M 为工序 A 的加工设备, Q 为工序 A 是否为工序组中工序的标识 ($Q = 0, 1, 2, \cdots$. $Q = 0$ 时为非工序组中工序, 属于同一工序组的工序 Q 相等且大于 0), P 为指向工序 A 紧前工序的指针, 如不存在前序工序, 则 P 为空. r 为工序 A 是否确定顺序的标识 ($r = 0, 1$. $r = 0$ 为工序未确定加工顺序), T_s 为工序 A 最早可以开始的加工时间, T_b 为工序 A 实际开始加工时间, T_e 为工序 A 的结束时间.

建立工序调度顺序链表 List$X$, 其中节点结构为

$$\mathrm{L}: \quad A, head, next$$

其中, A 为工艺树中某节点的工序名, $head$ 为指向先工序 A 调度工序的指针, $next$ 为指向后工序 A 调度工序的指针.

步骤 2. 将包含工序组的链表 List O 按路径进行拆分, 一个路径是由叶节点按 S 指针直到根节点的所有工序. 当某一路径上的叶节点不是工序组中工序节点时, 寻找该路径上第一个属于工序组中工序的节点, 若该节点的前序工序中存在工序组中工序时, 该路径忽略不计. 对每个确定的路径, 除去路径中 $Q = 0$ 的节点, 分别为各路径的工序组中的工序按加工顺序建立链表, 链表名为 Lists 1, $\cdots$, Lists k (链表个数 $k < n$, n 为复杂产品工序总数). 其中链表 Lists i 中节点的结构为

$$\mathrm{L}: \quad A, S, Q, e, P, r$$

其中, A 为链表 Lists i 中某工序组中工序的名字, S 为指向工序 A 在链表 Lists i 中紧后工序的指针, Q 为工序组标识且不等于 0, 属于同一工序组的工序 Q 值相等, P 为指向工序 A 在链表 Lists i 中紧前工序的指针, e 为指向工序 A 前序非工序组工序的指针, r 为工序 A 是否被确定调度顺序的标识.

步骤 3. 按 Lists 1, $\cdots$, Lists k 中节点总个数由多至少使用冒泡排序重新排列链表的顺序, 结果为 List $1'$, $\cdots$, List k' ($k = k' < n$, n 为复杂产品工序总个数). 为了表示排序后的链表和链表中的工序以及链表间的关系, 用 Lists $R'.L_a$ 表示链表 Lists R' 中第 a 个节点, 变量 R, a 的初值均为 1. 当前调度工序用变量 G 表示, 当前调度工序组队列链表用 List R 表示, 链表 List R 的节点结构与 List X 节点结构相同. 对于包含 G 的其他链表用 B_c 表示, 其中 c 与当前母链表 Lists R' 对应, 即 $c = R$, 初始时 $B_c = \Phi$. 用 H_x 记录调度中断的路径, 用 l_x 记录路径中断点的位置, 用 G_x 记录该断点的内容 (其中 x 为链表标记, $1 \le x \le k$).

步骤 4. 从链表 Lists R' 中寻找第 a 个节点.

1) 如果该节点的属性 r 为 0, 即 Lists $R'.L_a.r = 0$, 则工序 Lists $R'.L_a$ 为当前调度工序, 此时 $G =$ Lists $R'.L_a$, 并将 G 插入链表 List R 中. 然后, Lists $R'.L_a.r = 1$, 转步骤 5.

2) 如果 Lists $R'.L_a.r = 1$, $G =$ Lists $R'.L_a$, 转步骤 6.

步骤 5. 如果 $R + 1 > k$, 转步骤 6; 否则从 Lists $(R+1)'$ 至 Lists k' 中寻找包含 G 的链表, 若不存在, 转步骤 6; 若存在, 转步骤 8.

步骤 6. 如果不存在除工序组外前序工序, 即 $G.e = \Phi$, 转步骤 7; 如果 $G.e \neq \Phi$, 按拟关键路径法确定 $G.e$ 及所有紧前工序中 $Q = 0$ 且 $r = 0$ 的节点调度顺序, 并将节点插入 List X 中, List O 中相应节点 $r = 1$, 转步骤 7.

步骤 7. 如果 $c = \Phi$, 即 c 为母链; 如果 $G.Q = [G.S].Q$, $a = a + 1$, 转步骤 4; 如果 $G.Q \neq [G.S].Q$, 转步骤 10; 否则:

1) 如果当前调度工序的紧后工序为母链表中的工序 G_c, 即 $G.S = G_c$; 又如果当前调度工序的紧后工序不在同一工序组, 即 $G.Q \neq [G.S].Q$, 消除该子链 Lists R', 将该子链对应的工序组队列链表 List R 加入 List X, 清空 List R, 转步骤 9; 如果 $G.Q = [G.S].Q$, 将当前的工序组队列链表 List R 加入其母链表 List c 前部, 清空 List R, 转步骤 9.

2) $G.S \neq G_c$, $G.Q = [G.S].Q$, $a = a + 1$, 转步骤 4; 若 $G.Q \neq [G.S].Q$, 转步骤 10.

步骤 8. 对于包含 G 的链表, 用变量 B_c 记录相关链表序号, $B_c = \{b_1, b_2, \cdots, b_m\}$, 其中 c 与当前链表 Lists R' 对应, 为 B_c 中链表的标记, 即此时 $c = R$. 为了利用前面设计的调度方法并方便工序 G 的调度, $H_R = R$ (记录原始链表), $G_R = G$ (记录待调度工序), $l_R = a$ (表示 G_R 在链表中的位置), $R = \min\{B_c\}$, $a = 1$, 转步骤 4.

步骤 9. 如果 $\{B_c\} = \Phi$, $R = H_c$, $a = l_R$, 转步骤 4; 如果 $\{B_c\} \neq \Phi$, $R = \min\{B_c\}$, $a = 1$, 转步骤 4.

步骤 10. 将队列链表 List R 加入 List X 中, 清空 List R, 如果 Lists R' 调度结束 ($G.S = \Phi$), 转步骤 11; 否则 $a = a + 1$, 转步骤 4.

步骤 11. $R = \min\{$未调度链表序号$\}$, $a = 1$, 转步骤 4; 如果链表均调度, 转步骤 12.

步骤 12. 将剩余节点按拟关键路径法确定加工顺序.

2.2 确定工序在设备上的调度加工

确定工序的调度顺序以后, 在进行调度时, 由于设备资源较少, 可能会出现设备占用的情况, 对于具有紧密衔接条件的工序组也可能存在工序均可以在最早可以开始加工时间进行调度, 但不能满足紧密衔接约束条件的状况. 因此, 使用首次适应调度算法调度加工标准工序和工序组中工序.

2.2.1 使用首次适应调度算法加工标准工序

由于在调度工序时, 如果使用设备紧凑策略, 很可能会破坏已经调度的工序组的紧密衔接约束条件, 使调度工作更加复杂, 所以在进行调度加工时采用首次适应调度方法. 对于标准工序, 当设备在工序最早可以加工时间段内被占用时, 向后寻找最早可以加工的时间段进行调度, 如不存在, 则将工序放在已调度工序的末尾进行调度, 不对已经进行调度的工序进行移动.

2.2.2 使用首次适应调度算法加工工序组中工序

紧密衔接工序组在使用首次适应调度算法后, 需要判断多个工序在满足工序前后约束关系的基础上能否满足工序的结束时间为其紧后工序的开始时

间. 如果不满足, 需要判断工序能否进行后移, 如果可以, 将其紧前工序后移, 如果不可以, 则需要在设备的末尾进行调度加工, 从而实现工序组联动调度加工.

调度步骤如下:

步骤 1. 首先判断工序组第一个工序能否在最早可以开始时间段加工, 如可以, 调度该工序; 如不可以, 向后寻找可以加工的第一个时间段至设备末尾, 调度该工序, 转步骤 2.

步骤 2. 判断工序组的下一个工序能否在其最早可以开始加工时间段加工, 如可以, 转步骤 3; 如不可以, 向后寻找可以加工的最早时间段, 转步骤 3.

步骤 3. 如该工序与其紧前工序紧密衔接, 转步骤 7; 否则, 转步骤 4.

步骤 4. 判断工序组已经调度的工序能否向后移动, 如均可以, 将已经调度的紧前工序向后移动, 转步骤 7; 如存在不可以的工序, 且最后一个调度的工序在设备末尾调度加工, 转步骤 6; 否则, 转步骤 5.

步骤 5. 将工序组最后一个调度的工序向后寻找可以调度的时间段至末尾, 转步骤 4.

步骤 6. 将工序组的全部工序放在设备末尾调度, 转步骤 7.

步骤 7. 重复判断工序组是否存在其余紧密衔接工序, 如存在, 转步骤 2; 否则, 转步骤 8.

步骤 8. 结束.

注: 关于设备末尾: 紧前工序所使用设备的已经调度工序的结束时间大于等于后续工序使用设备的已经调度工序的结束时间, 紧前工序在最早可以开始时间调度; 反之, 小于后续工序使用设备的结束时间, 在调度紧前工序时, 使该工序的结束时间不早于后续工序使用设备已经调度工序的结束时间.

2.2.3 复杂产品工序在设备上加工的算法实现说明

用 List $X.L_a$ 代表 List X 中第 a 个节点, a 的初始值为 1. 对已经排序的链表 List X 的结构进行扩展, 使其结构变为

$$\mathrm{L}: \quad A, S, T, M, Q, P, T_s, T_b, T_e$$

其中, A 为工艺树中某节点的工序名, S 为指向工序 A 紧后工序的指针, T 为工序 A 的加工时间, M 为工序 A 的加工设备, Q 为工序 A 是否为工序组中工序的标识 ($Q = 0, 1, 2, \cdots$. $Q = 0$ 时为非工序组中工序, 属于同一工序组的工序 Q 相等且大于 0), P 指向工序 A 紧前工序的指针, 如不存在前序工序, 则 P 为空, T_s 为工序 A 最早可以开始的加工时间, T_b 为工序 A 实际开始加工时间, T_e 为工序 A 的结束时间.

用链表 Listed W 表示工序在第 W 台设备上的加工, 其结构为

$$\mathrm{L}: \quad A, S, T_b, T_e, P$$

其中, A 为工艺树中某节点的工序名, S 为指向工序 A 紧后工序的指针, T_b 为工序 A 实际开始加工时间, T_e 为工序 A 的结束时间, P 指向工序 A 紧前工序的指针, 如不存在前序工序, 则 P 为空.

用链表 Listing Z 存储正在调度加工的工序组中工序; c 为 Listing Z 中节点个数, c 初始值为 0, Listing Z 的初始值为空. D 表示当前加工节点, N 表示当前加工工序组中工序的最后一个工序, J 表示当前已加工工序组的前序工序. 节点结构均与 List X 相同. 用 t 记录工序组中工序向后移动的时间.

步骤 1. 如果 $a > n$ (总个数), 结束; 否则, 如果 $a < n$, 从 List X 中取出第 a 个节点作为当前加工节点, 即 $D =$ List $X.L_a$; 并寻找该节点的加工设备, 即 $W = D.M$.

步骤 2. 如果当前加工工序为标准工序, 即 $D.Q = 0$, 从 $D.T_s$ 开始向后按最早适应调度算法确定 D 在 Listed W 中位置, $a = a + 1$, 转步骤 1; 否则, 如果该工序为工序组中的工序, 即 $D.Q \neq 0$, 转步骤 3.

步骤 3. 将 D 放入链表 Listing Z 中, 从 $D.T_s$ 开始向后按最早适应调度算法确定 D 在 Listed W 中的位置, $c = c + 1$.

步骤 4. 如果 $c = 1$, $a = a + 1$, 转步骤 1; 否则, 转步骤 5.

步骤 5. 用变量 N 记录已经加工的工序组中最后一个工序, 即 $N =$ Listing $Z.L_c$; 如果该工序与其紧前工序满足紧密衔接的约束条件, 即 $N.T_b = [N.P].T_e$, 转步骤 8; 否则, $d = c - 1$, 转步骤 6.

步骤 6. 该工序与其紧前工序不能满足紧密衔接的约束条件, 即 $N.T_b \neq [N.P].T_e$; $t = N.T_b - [N.P].T_e$, 用 J 表示该节点的前序工序, 即 $J =$ Listing $Z.L_d$.

步骤 7. 判断 J 节点能否后移时间段 t, 使之满足 $J.T_e = [J.S].T_b$:

1) 如果可以, 将其向后移动, 同时 Listing Z, List O, List X 中 T_b, T_e 调整, $d = d - 1$; 如果 $d \leq 0$, $a = a + 1$, 转步骤 1, 否则, 转步骤 6.

2) 如果不可以且 $d = c - 1$, D 寻找下一个满足其调度时间的时间段, 转步骤 5; 当 $d \neq c - 1$, 将 Listing Z 中所有节点放在设备末尾进行调度.

步骤 8. 如果当前加工工序与其后续加工工序不属于同一工序组, 即 $N.Q \neq [N.S].Q$, 清空 Listing Z, $c = 0$, $a = a + 1$, 转步骤 1; 如果 $N.Q = [N.S].Q$, $a = a + 1$, 转步骤 1.

3 算法复杂度分析

设产品工序数为 n, 设备数为 m, 属于工序组中工序的个数为 k, 已经加工的工序组中工序的个数为 p, 当前工序组中已经加工的工序个数为 q. 算法的复杂度分析如下:

1) 将具有紧密衔接约束条件的复杂产品工艺图构造成扩展加工工艺树, 根据工艺树中代表工序的节点属性建立反映工序间约束关系的链表, 由于每个工序的紧前、紧后工序对这个工序是已知的, 所以建立 n 个节点的链表只需操作 n 次, 所以建立原始链表的时间复杂度为 $\mathrm{O}(n)$.

2) 根据工序属性, 建立仅存在工序组中工序的链表, 通过判断 n 个工序的属性 P 是否为 Φ, 判断叶节点工序, 所以确定叶节点工序的操作为 n 次; 根据工序的紧后工序属性, 建立以每个叶节点直到根节点并去除所有 $Q=0$ 的节点的路径链表, 对于路径中 $Q\neq 0$ 的节点按路径累加统计, 建立仅存在工序组中工序的链表并计算出各链表中节点个数最多需要 $2n$ 次操作; 建立仅存在工序组中工序的链表的复杂度为 $\mathrm{O}(n)$, 对已经建立的链表按节点个数由多到少排序命名, 由于路径数最多为工序组中元素个数 k, 所以使用冒泡排序法对路径进行排序的操作次数的复杂度为 $\mathrm{O}(k^2)$; 建立仅存在工序组中工序的链表的时间复杂度为 $\max\{\mathrm{O}(n),\mathrm{O}(k^2)\}$.

3) 确定工序组中工序及标准工序的调度顺序. 由于标准工序采用拟关键路径法确定调度顺序, 标准工序的总个数为 $n-k$, 所以根据文献 [20] 确定 $n-k$ 个标准工序的时间复杂度为 $\mathrm{O}((n-k)^2)$. 确定工序组中工序的调度顺序时, 需要寻找可能存在的前序工序组中工序, 需要对其余未调度的、仅包含工序组中工序的链表进行遍历查询, 当有 k 个工序组中的工序时, 每个工序需要查询的操作次数为 k, 全部 k 个工序需要查询操作的总次数为 k^2, 确定工序组中工序的调度顺序的时间复杂度为 $\mathrm{O}(k^2)$, 所以所有工序的调度顺序的时间复杂度为 $\mathrm{O}(n^2)$.

4) 确定工序组中工序及标准工序在设备上的加工.

工序在设备上加工时, 标准工序采用首次适应调度算法确定工序的开始加工时间, 由于有 m 台加工设备, 所以在每个设备上进行加工时, 需要比较的次数平均为已经调度工序数/m, 对于标准工序, 最坏情况为全部工序组中工序均加工完毕, 然后加工全部标准工序, 所以需要比较的次数最多为 $(k+1)/m$, $(k+2)/m$, $\cdots$, n/m 次, 即全部标准工序需要比较的总次数为 $(n+k+1)\times(n-k)/(2\times m)=(n^2-k^2+n-k)/(2\times m)$, 由于 $1<m\ll n$, 所以确定所有标准工序开始时间的复杂度为 $\mathrm{O}(n^2-k^2)$.

对于工序组中的一个工序, 使用首次适应调度算法进行加工时, 最坏情况为全部标准工序均已加工完毕, 已经加工的工序组中工序个数为 p, 当前加工工序组中工序已经调度的个数为 q, 由于有 m 台加工设备, 所以需要比较的次数为 $(n-k+p+q)/m$; 该工序确定在设备上的调度加工后, 需要与其前序工序组中的工序的结束时间进行 1 次比较, 当不满足紧密衔接的约束条件时, 即 $[N.P].T_e<N.Tb$, 该工序组已经加工的全部工序需要依次向后移动, 此时需要向后移动的工序数为 q, 如果某一次移动无法实现紧密衔接的要求, 需要当前工序向后移动 1 次, 相应的所有该工序组的前序工序向后移动的总数为 q, 满足当前工序为工序组中工序的要求需要移动的次数 $(n-k+p+q)\ (q+1)/m$, 确定工序组中一个工序需要比较和移动的总次数为 $(n-k+p+q)/m+1+(n-k+p+q)\,(q+1)/m$. 因为 $k\leq p+q$, 所以确定工序组中一个工序最多需要处理次数为 $n/m+1+nk/m$; 由于有 k 个工序组中的工序, 所以确定全部工序组中工序的开始时间需要比较和移动的总次数为 $k\ (n/m+1+nk/m)=kn/m+k+k^2n/m$, 因为 $1<m\ll n$, 所以确定全部工序组中工序的开始时间需要处理的次数的时间复杂度为 $\mathrm{O}(k^2n)$.

所以确定所有工序开始加工时间的时间复杂度为 $\max\{\mathrm{O}(n^2-k^2),\ \mathrm{O}(k^2n)\}$.

综合以上四点, 本文提出算法总的处理次数为以上四点的和, 该算法的时间复杂度为 $\max\{\mathrm{O}(n^2-k^2),\ \mathrm{O}(k^2n)\}$, 因为 $0\leq k\leq n$, 所以本文提出算法的最大时间复杂度为 $\mathrm{O}(n^3)$, 当 k 值较小时, 该算法的时间复杂度一般为 $\mathrm{O}(n^2)$.

由于本文所采用的算法优先调度对设备要求较高的紧密衔接工序组中工序, 使工序组和标准工序在确定加工时间后不再调整, 避免了文献 [19] 方法中由于全部工序加工结束后, 紧密衔接工序组中工序移动调整引起的连锁反应. 由于工序组中工序移动调整的复杂度是 2 次的, 而每次移动调整引起的连锁反应的复杂度也是 2 次的, 所以本文提出算法的时间复杂度比文献 [19] 中方法的时间复杂度 $\mathrm{O}(n^4)$ 至少低一个数量级.

4 实例

产品 A 是由 16 个存在约束关系的工件组成的复杂产品, 共 27 个工序, 可在 4 台设备上加工, 产品 A 的扩展加工工艺树模型如图 1 所示.

根据路径上属于工序组的工序的多少确定工序组的调度顺序, 有 3 个路径含有工序组中的工序, 包含工序组中工序的情况为: $\{A11,\ A13,\ A14,\ A15\}$, $\{A6,\ A8,\ A15\}$, $\{A20,\ A21\}$, 所以优先调度工序

组 {$A11$, $A13$, $A14$, $A15$}, 最后调度工序组 {$A20$, $A21$}. 因为工序组 {$A11$, $A13$, $A14$, $A15$} 的前序工序为 $A10$, $A12$, 和树 {$A1 \sim A9$}, 工序 $A10$ 为工序 $A11$ 的紧前工序, $A12$ 为 $A13$ 的紧前工序, 工序树 {$A1 \sim A9$} 为工序 $A15$ 的紧前工序树, 工序组内的调度顺序为: $A11$, $A13$, $A14$, $A15$, 所以工序组前序工序调度顺序为: $A10$, $A12$, 工序树 {$A1 \sim A9$}, 工序组 {$A11$, $A13$, $A14$, $A15$}.

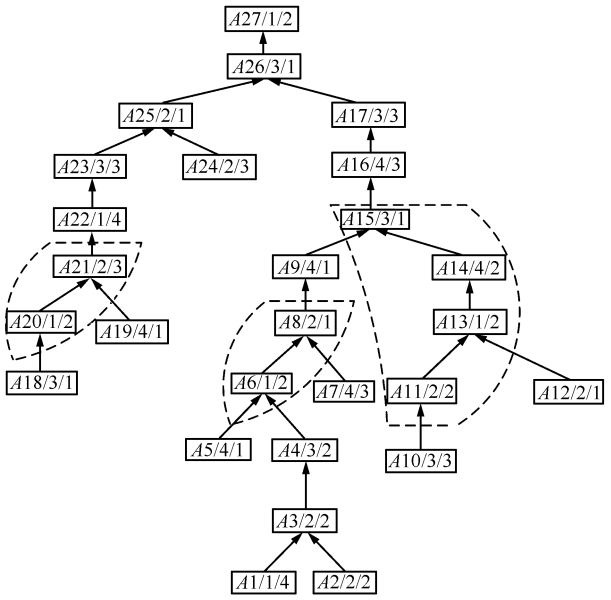


图 1　产品 A 的扩展加工工艺树模型图

Fig. 1　Expansion processing tree of product A

因为工序树 {$A1 \sim A9$} 中存在工序组, 所以优先确定工序组的调度顺序, 因为工序 $A7$ 为 $A8$ 的紧前工序, 工序 $A5$ 和工序树 {$A1 \sim A4$} 为工序 $A6$ 的紧前工序, 工序组内工序 $A6$ 先于工序 $A8$ 调度, 所以优先调度工序 $A5$ 和工序树 {$A1 \sim A4$}, 按拟关键路径法优先调度工序树 {$A1 \sim A4$} 和工序 $A5$, 所以调度顺序为: {$A1$, $A2$, $A3$, $A4$, $A5$, $A7$, $A6$, $A8$, $A9$}.

工序组 {$A20$, $A21$} 中工序 $A18$ 为工序 $A20$ 的紧前工序, 工序 $A19$ 为工序 $A21$ 的紧前工序, 工序组中调度顺序为: $A20$, $A21$, 所以优先调度工序 $A19$, 所以调度顺序为: {$A18$, $A19$, $A20$, $A21$}.

所以全部工序组的调度顺序为: {$A10$, $A12$, $A1$, $A2$, $A3$, $A4$, $A5$, $A7$, $A6$, $A8$, $A9$, $A11$, $A13$, $A14$, $A15$, $A18$, $A19$, $A20$, $A21$}.

工序组调度结束后, 剩余工序按拟关键路径法确定工序的调度顺序. 剩余工序的关键路径为: {$A22$, $A23$, $A25$, $A26$, $A27$}. 调度顺序为: {$A22$, $A23$, $A24$, $A25$, $A16$, $A17$, $A26$, $A27$}.

所以产品 A 的调度顺序为: {$A10$, $A12$, $A1$, $A2$, $A3$, $A4$, $A5$, $A7$, $A6$, $A8$, $A9$, $A11$, $A13$, $A14$, $A15$, $A18$, $A19$, $A20$, $A21$, $A22$, $A23$, $A24$, $A25$, $A16$, $A17$, $A26$, $A27$}.

因为工序 $A10$, $A12$, $A1$, $A3$, $A4$, $A5$, $A9$ 不需要等待, 直接调度该工序, 工序 $A2$, $A7$ 在最早可以开始加工时间设备占用, 向后寻找可以加工的空闲时间段至末尾, 在末尾进行调度加工, 即分别在 $1 \sim 3$, $1 \sim 4$ 时间段进行调度加工.

工序 $A6$, $A8$ 在最早可以开始加工时间可以调度加工, 且满足紧密衔接约束条件, 所以直接调度加工.

工序 $A11$ 在最早可以调度时间段设备占用, 所以向后寻找可以加工的空闲时间段, 在时间段 $6 \sim 8$ 进行调度加工. 工序 $A13$ 在最早可以调度时间段设备占用, 向后寻找可以加工的空闲时间段, 在时间段 $10 \sim 12$ 进行调度. 因为工序 $A11$ 的结束时间为 8, 不满足 $A11$, $A13$ 的紧密衔接约束条件, 所以工序 $A11$ 的开始加工时间后移, 工序 $A11$ 的加工时间为 $8 \sim 10$. 工序 $A14$, $A15$ 可以在最早可以开始加工时间进行调度加工, 且满足紧密衔接约束条件, 所以直接调度加工.

工序 $A18$ 在最早可以开始加工时间设备占用, 向后寻找最早可以加工的空闲时间段, 在 $3 \sim 4$ 时间段进行调度加工. 工序 $A19$ 在最早可以开始加工时间设备占用, 向后寻找最早可以加工的空闲时间段, 在 $4 \sim 5$ 时间段进行调度加工.

工序 $A20$ 在最早可以开始时间可以进行调度加工, 即时间段 $4 \sim 6$, 工序 $A21$ 在最早可以开始加工时间设备占用, 向后寻找最早可以加工的空闲时间段, 只能在 $11 \sim 14$ 时间段进行调度加工, 但由于工序 $A20$ 与 $A21$ 为紧密衔接工序, 而工序 $A20$ 向后移动不能满足条件, 所以, 将工序 $A20$ 与 $A21$ 一同放到末尾进行调度加工, 在设备末尾, 设备 1 的结束时间为 12, 设备 2 的结束时间为 11, 早于设备 1, 所以直接将工序 $A20$, $A21$ 放到设备末尾进行调度加工. 紧密衔接工序组及相关工序的全部调度时间为 17.

对于剩余工序: 工序 $A22$, $A23$, $A25$, $A16$, $A17$, $A26$, $A27$ 不需要等待, 直接调度该工序; 工序 $A24$ 在最早可以调度时间段设备占用, 所以向后寻找可以加工的空闲时间段, 在时间段 $11 \sim 14$ 进行调度加工, 总加工时间为 28. 调度的甘特图如图 2 所示.

如果使用文献 [19] 的算法, 在无紧密衔接条件下调度的产品总加工时间为 28, 工序 $A6$, $A8$ 紧密衔接, 满足紧密衔接条件, 不需要移动, 工序组 $A11$, $A13$, $A14$, $A15$ 中 $A11$ 与 $A13$ 不能满足紧密衔接约束条件, 所以向后移动工序 $A11$, 工序组 $A20$, $A21$ 不能满足紧密衔接约束条件, 将工序 $A20$ 向后移动,

并将其全部后续工序向后移动, 总加工时间为 31.

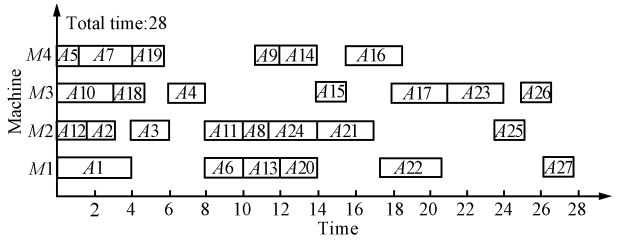


图 2 包含紧密衔接工序调度所得甘特图

Fig. 2 Gantt chart of FFSM scheduling algorithm by no-wait constraint operations

使用文献 [19] 的算法调度的产品甘特图如图 3, 无紧密衔接条件下的调度产品甘特图如图 4.

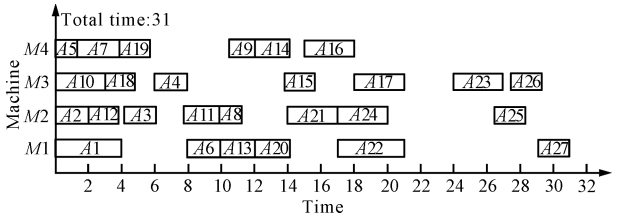


图 3 使用文献 [19] 调度所得甘特图

Fig. 3 Gantt chart of scheduling algorithm according to [19]

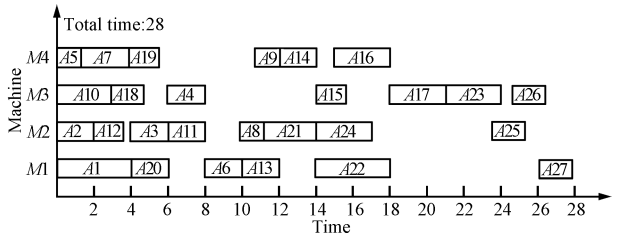


图 4 一般工序调度所得甘特图

Fig. 4 Gantt chart of scheduling algorithm by standard operations

5 结论

本文提出的具有紧密衔接约束条件的复杂产品综合调度算法, 通过分步确定工序的调度顺序和在设备上的开始加工时间, 并对具有紧密衔接约束条件的工序组中的工序采取提前联动调度, 不仅减少相关设备上的已加工工序对工序组中工序的影响, 使工序组尽早完工, 而且不会产生工序组中工序移动对标准工序的连锁反应, 所以该算法不仅时间复杂度比目前已有算法的时间复杂度低, 而且调度结果更优. 对于具有固定延迟条件的问题, 通过将延迟条件转换成包含虚拟紧密衔接的工序, 该算法也可扩展解决. 因此, 该算法为进一步优化具有紧密衔接工序的复杂产品综合调度问题及相关问题提供了参考方案, 有一定的理论和实用价值.

References

1 Zhang Chang-Sheng, Sun Ji-Gui, Yang Qing-Yun, Zheng Li-Hui. A hybrid algorithm for flowshop scheduling problem. *Acta Automatica Sinica*, 2009, **35**(3): 332−336
(张长胜, 孙吉贵, 杨轻云, 郑黎辉. 一种求解车间调度的混合算法. 自动化学报, 2009, **35**(3): 332−336)

2 Hu Rong, Qian Bin. A hybrid differential evolution algorithm for stochastic flow shop scheduling with limited buffers. *Acta Automatica Sinica*, 2009, **35**(12): 1580−1586
(胡蓉, 钱斌. 一种求解随机有限缓冲区流水线调度的混合差分进化算法. 自动化学报, 2009, **35**(12): 1580−1586)

3 Yan Li-Jun, Li Zong-Bin, Wei Jun-Hu, Du Xuan. A new hybrid optimization algorithm and its application in job shop scheduling. *Acta Automatica Sinica*, 2008, **34**(5): 604−608
(闫利军, 李宗斌, 卫军胡, 杜轩. 一种新的混合优化算法及其在车间调度中的应用. 自动化学报, 2008, **34**(5): 604−608)

4 Wei Ying-Zi, Zhao Ming-Yang. A reinforcement learning-based approach to dynamic job-shop scheduling. *Acta Automatica Sinica*, 2005, **31**(5): 765−771

5 Jansen K, Mastrolilli M, Solis-Oba R. Approximation algorithms for flexible job shop problems. *Lecture Notes in Computer Science*, 2000, **1776**: 68−77

6 Gerhard G J, Woeginger J. Inapproximability results for no-wait job shop scheduling. *Operations Research Letters*, 2004, **32**(4): 320−325

7 Thomalla C S. Job shop scheduling with alternative process plans. *International Journal of Production Economics*, 2001, **74**(1−3): 125−134

8 Framinan J M, Schuster C. An enhanced timetabling procedure for the no-wait job shop problem: a complete local search approach. *Computers and Operations Research*, 2006, **33**(5): 1200−1213

9 Qian Bin, Wang Ling, Huang De-Xian, Jiang Yong-Heng, Wang Xiong. Rolling strategy and optimization algorithm for dynamic no-wait flow shop scheduling problem. *Control and Decision*, 2009, **24**(4): 481−487
(钱斌, 王凌, 黄德先, 江永亨, 王雄. 动态零等待流水线调度问题的滚动策略及优化算法. 控制与决策, 2009, **24**(4): 481−487)

10 Xu Zhen-Hao, Gu Xing-Sheng. Immune scheduling algorithm for flow shop under uncertainty with zero wait. *Computer Integrated Manufacturing Systems*, 2004, **10**(10): 1247−1251
(徐震浩, 顾幸生. 不确定条件下具有零等待的流水车间免疫调度算法. 计算机集成制造系统, 2004, **10**(10): 1247−1251)

11 Xie Zhi-Qiang. Study on Operation Scheduling of Complex Product with Constraint among Jobs [Ph. D. dissertation], Harbin University of Science and Technology, China, 2009
(谢志强. 工件间有约束的复杂产品工序调度研究 [博士学位论文], 哈尔滨理工大学, 中国, 2009)

12 Garey M R, Johnson D S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: W.H. Freeman and Company, 1979

13 Huang Qi-Chun, Chen Qi, Yu Rui-Zhao. A fast job oriented scheduling algorithm. *Journal of Software*, 1999, **10**(10): 1073−1077
(黄启春, 陈奇, 俞瑞钊. 一种面向作业的快速调度算法. 软件学报, 1999, **10**(10): 1073−1077)

14 Xie Zhi-Qiang, Mo Tao, Tan Guang-Yu. Dynamic job-shop scheduling algorithm of the non-close-joining operations. *Chinese Journal of Mechanical Engineering*, 2008, **44**(1): 155−160
(谢志强, 莫涛, 谭光宇. 非紧密衔接工序动态车间调度算法. 机械工程学报, 2008, **44**(1): 155−160)

15 Xie Z Q, Hao S Z, Ye G J, Tan G Y. A new algorithm for complex product flexible scheduling with constraints between jobs. *Computers and Industrial Engineering*, 2009, **57**(3): 766−772

16 Xie Zhi-Qiang, Yang Jing, Yang Guang, Tan Guang-Yu. Dynamic job-shop scheduling algorithm with dynamic set of operation having priority. *Chinese Journal of Computers*, 2008, **31**(3): 502−508
(谢志强, 杨静, 杨光, 谭光宇. 可动态生成具有优先级工序集的动态 Job-Shop 调度算法. 计算机学报, 2008, **31**(3): 502−508)

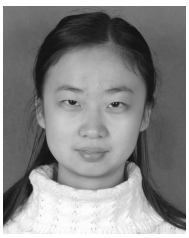
17 Xiong He-Gen, Li Jian-Jun, Kong Jian-Yi, Yang Jin-Tang, Jiang Guo-Zhang. Heuristic method for dynamic job shop scheduling problem with operation relativity. *Chinese Journal of Mechanical Engineering*, 2006, **42**(8): 50−55
(熊禾根, 李建军, 孔建益, 杨金堂, 蒋国璋. 考虑工序相关性的动态 Job Shop 调度问题启发式算法. 机械工程学报, 2006, **42**(8): 50−55)

18 Schuster C J. No-wait job shop scheduling: tabu search and complexity of subproblems. *Mathematical Methods of Operations Research*, 2006, **63**(3): 473−491

19 Xie Zhi-Qiang, Li Zhi-Min, Hao Shu-Zhen, Tan Guang-Yu. Study on complex product scheduling problem with no-wait constraint between operations. *Acta Automatica Sinica*, 2009, **35**(7): 983−989
(谢志强, 李志敏, 郝淑珍, 谭光宇. 工序间存在零等待约束的复杂产品调度研究. 自动化学报, 2009, **35**(7): 983−989)

20 Xie Zhi-Qiang, Liu Sheng-Hui, Qiao Pei-Li. Dynamic job-shop scheduling algorithm based on ACPM and BFSM. *Journal of Computer Research and Development*, 2003, **40**(7): 977−983
(谢志强, 刘胜辉, 乔佩利. 基于 ACPM 和 BFSM 的动态 Job Shop 调度算法. 计算机研究与发展, 2003, **40**(7): 977−983)

谢志强 哈尔滨工程大学博士后, 哈尔滨理工大学教授. 主要研究方向为企业智能计算, 数据库与知识工程. 本文通信作者. E-mail: xzq011@tom.com
(**XIE Zhi-Qiang** Postdoctor at Harbin Engineering University and professor at Harbin University of Science and Technology. His research interest covers enterprise intelligence computing, database, and knowledge engineering. Corresponding author of this paper.)



滕宇峥 哈尔滨理工大学计算机科学与技术学院硕士研究生. 2007 年获山西财经大学信息管理学院学士学位. 主要研究方向为企业智能计算.
E-mail: tengyuzheng@163.com
(**TENG Yu-Zheng** Master student at the School of Computer Science and Technology, Harbin University of Science and Technology. She received her bachelor degree from Shanxi University of Finance and Economics in 2007. Her main research interest is enterprise intelligence computing.)

杨 静 哈尔滨工程大学教授. 主要研究方向为企业智能计算, 数据库与知识工程. E-mail: yangjing@hrbeu.edu.cn
(**YANG Jing** Professor at Harbin Engineering University. Her research interest covers enterprise intelligence computing, database, and knowledge engineering.)