可回退抢占的设备驱动综合调度算法

谢志强 [1, 2] 辛 宇 [1] 杨 静 [2]

摘 要 针对基于拟关键路径法的综合调度算法按路径长度确定工序的调度次序, 形成工序组间的并行处理, 使设备产生较多空闲时间的问题, 提出可回退抢占的设备驱动综合调度算法. 该算法以每次工序加工结束作为一次可调度工序的寻找事件, 若此时新出现的可调度工序具备抢占能力, 则产生回退事件进行重调度; 若不产生回退事件, 如果可调度工序唯一, 则调度此工序; 如果可调度工序不唯一, 选择父结点路径长的工序; 如果父结点最长路径相同, 选择用时长的工序. 由于该算法在调度工序时形成工序间的并行处理, 缩小基于拟关键路径的综合调度算法形成的并行处理单位, 进而减少加工过程中产生较多的设备空闲时间, 提高设备利用率; 同时, 由于采用抢占式的回退调度策略, 优先调度对调度结果有重要影响的长路径工序, 达到对拟关键路径法的扬长避短, 进一步提高设备利用率.

关键词 设备空闲事件, 事件驱动, 工序并行, 综合调度, 回退抢占, 工序组

DOI 10.3724/SP.J.1004.2011.01332

Machine-driven Integrated Scheduling Algorithm with Rollback-preemptive

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

Abstract Determining the scheduling sequence according to the length of path by the integrated scheduling algorithm based on allied critical path method will cause parallel processing among the sequence segment of procedures and machines with plenty of idle time. This paper presents a machine-driven integrated scheduling algorithm with rollback-preemptive. Each ending process of procedures is regarded as a searchable event of schedulable procedures in this algorithm. At this time, if a schedulable procedure which appears new has preemptive ability, it generates rollback event to reschedule. In case rollback event is not generated, we schedule this procedure if the schedulable procedure is sole; otherwise, we select the procedure which has the longest parents node path. If the longest parents node path is not sole, the procedure with the longest process time is select. As it will cause parallel processing among procedures when procedures are scheduled in this algorithm, it shortens the parallel processing unit generated by integrated scheduling algorithm based on allied critical path method and reduces excessive machine idle time caused in processing. So the machine utilization efficiency is heightened. Simultaneity, because it adopts rollback scheduling strategy with preemptive style and schedules the longest path procedure firstly which can influence scheduling result significantly, it makes best use of the advantage and bypasses the disadvantage of allied critical path method and advances the utilization efficiency further.

Key words Machines′ idle event, event-driven, procedures parallel processing, integrated scheduling, rollback-preemptive, sequence segment of procedures

对于现代工业而言, 生产线最能体现企业的制造能力. 为了提高生产线的生产效率, 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[1] 的一种求解车间调度的混合算法、张宏远等[2] 的基于约束理论的 Flow-shop 分解协调算法和徐震浩等[3] 的不确定条件下具有零等待的流水车间免疫调度算法等. 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[4] 的一种新的混合优化算法及其在车间调度中的应用、熊禾根等[5] 的考虑工序相关性的动态 Job-shop 调度问题启发式算法和魏英姿等[6] 的一种基于强化学习的作业车间动态调度方法等. 这些算法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. 随着社会对产品个性化需求的不断增加, 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分解, 采取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[7], 因此, 企业越来越需要好的算法实现加工和装配一同处理的综合调度计划. 由于拟关键路径法 (Allied c-

收稿日期 2010-12-15 录用日期 2011-05-28
Manuscript received December 15, 2010; accepted May 28, 2011
国家自然科学基金 (60873019, 61073043), 中国博士后科学基金 (20090460880), 黑龙江省自然科学基金 (F200901), 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, 2011RFXXG015) 资助
Supported by National Natural Science Foundation of China (60873019, 61073043), China Postdoctoral Science Foundation (20090460880), Natural Science Foundation of Heilongjiang Province (F200901), Heilongjiang Province Postdoctoral Science Foundation (LBH-Z09214), and Harbin Outstanding Academic Leader Foundation of Heilongjiang Province of China (2010RFXXG054, 2011RFXXG015)
1. 哈尔滨理工大学计算机科学与技术学院 哈尔滨 150080 2. 哈尔滨工程大学计算机科学与技术学院 哈尔滨 150001
1. College of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150080 2. College of Computer Science and Technology, Harbin Engineering University, Harbin 150001

ritical path method, ACPM) 较好地解决了一般综合调度问题[8], 因此, 近几年出现一些应用 ACPM 解决特殊综合调度问题的方法, 如相同设备情况[9]、非紧密衔接情况[10−11]、紧密衔接情况[12−13] 和柔性调度情况[14] 等. 这些算法的主要优点是利用包含加工和装配的产品工艺树结构具有末端分支繁衍的特点, 建立循环迭代, 优先调度对产品完成时间有重要影响的长路径上的工序且复杂度只有二次多项式. 缺点是: 1) 按路径长度确定工序的调度次序, 形成以工艺结构中唯一紧前紧后工序组为单位的并行调度. 即当某工序组调度完毕, 后续工序组才可进行调度, 工序组中的工序必须在已调度工序序列形成的空隙中, 选择合适的大小进行加工, 使得设备上产生较多无法使用的空闲时间段, 即影响设备的利用率; 2) 没有考虑同层工序间的并行处理, 影响并行效果.

为了提高综合调度的并行效率, 文献 [15] 提出考虑优先工序集等算法调度和文献 [16] 提出考虑动态关键路径, 但是由于这 2 种算法都是先确定工序的调度次序, 再按确定次序将工序调度到加工设备上, 虽然调度单位是工序, 但由于后续工序只能在先序工序的空隙中进行调度, 当空闲时间段较小时, 不仅工序延迟加工, 而且影响同时加工设备之间的并行处理, 此时, 以工序为调度单位的这两种算法也不能实现工序间的充分并行处理.

为了进一步减少综合调度的设备空闲时间段, 提高设备利用率, 形成工序间的真正并行处理, 本文利用构造事件驱动的思想[17−18], 拟采用基于设备驱动的综合调度, 为了保留 ACPM 优先调度对产品完成时间有重要影响的长路径上工序的优点, 设备驱动法优先选择可调度工序路径长的工序; 由于已调度工序的父结点路径长度可能小于新调度工序的父结点路径长度, 为了动态保留 ACPM 优先调度对产品完成时间有重要影响的长路径上工序的优点, 拟采用回退策略优先调度父结点路径长的工序, 优化综合调度. 为此, 本文提出可回退抢占的设备驱动综合调度算法. 该算法以每次工序加工结束作为一次可调度工序的寻找事件, 若此时新出现的可调度工序具有较长的父结点路径, 具备抢占能力, 则采用回退策略进行重调度; 若无被抢占工序, 则采取最大并行性选择策略选择可调度工序. 实现综合调度设备的充分利用, 最后通过实例对比分析说明所提出算法的优越性.

1 问题模型描述

研究加工和装配综合调度算法, 就是设计产品加工和装配综合调度的方法, 使产品的完工时间尽可能的短, 即产品尽早加工结束. 为了简化加工和装配综合调度的描述, 一般将加工和装配工序统一为加工, 加工和装配设备统一为设备. 综合调度的一般要求是:

1) 每道工序的加工设备唯一;

2) 同一加工设备在同一时刻只能加工一道工序;

3) 每个道序必须在其所有紧前工序均已加工完毕, 或没有紧前工序时, 才可以在其加工设备上开始加工;

4) 最后一道工序加工完毕时间为产品的总加工时间.

设产品由 N 道工序组成, 在 M 台设备上加工, $n\ (1 \le n \le N)$ 为工序编号, $m\ (1 \le m \le M)$ 为设备编号. 设 $\mathrm{end}(n)$ 为工序 n 的加工完毕时间, 则产品加工总时间 $T = \max(\mathrm{end}(n))$, 于是一般综合调度问题目标函数为

$$T = \min\{\max[\mathrm{end}(n)]\}$$

由于设备驱动问题只在设备空闲时刻进行调度, 设备抢占依赖于加工工序对根结点工序影响的回退判断, 并改变工序的调度次序, 为了说明此时工序的约束关系, 设

1) t_k 为设备产生空闲的时刻, 其中 $k\ (k \ge 1, t_0$ 为开始时刻);

2) X_k 为时刻 t_k 已加工完毕工序构成的集合;

3) O_k 为时刻 t_k 无紧前约束的工序集合;

4) c_{mk} 为时刻 t_k 空闲设备 m 选择加工的某一工序;

5) C_k 为该时刻开始加工的工序集合, $C_k = \{c_{mk}\}$;

6) $B(n)$ 为工序 n 的开始加工时间;

7) W_k 为时刻 t_k 空闲设备选择的工序和处于加工状态的工序集合;

8) $E_k = \{\mathrm{end}(i)\}$, 其中, $1 \le i \le n$ 为当前所有加工完毕工序的编号, E_k 记录了所有空闲时刻;

9) R_{mn} 为设备为 m 上被抢占的工序 n;

10) t_r 为回退后的时刻.

根据基于设备驱动和回退策略的综合问题的要求, 提出并建立所研究问题的目标函数及约束条件 (即数学模型) 可表示为:

目标函数为 $T = \min(\max\ (t_k))$

s. t.

$$X_k = X_{k-1} \cup \{j\}, \quad 1 \le j \le N, \quad \mathrm{end}(j) = t_k \tag{1}$$

$$\begin{aligned} &B(C_k) \ge \max[\mathrm{end}(j)], \quad j \in X_k \\ &C_k = \{c_{mk}\}, \quad C_k \subseteq O_k \end{aligned} \tag{2}$$

$$W_k = (W_{k-1} \cap \overline{(W_{k-1} \cap X_k)}) \cup C_k \tag{3}$$

$$E_k = E_{k-1} \cup \{t_k\} \tag{4}$$

$$t_{k+1} = t_k + \min\{\text{end}(i)\}, \quad i \in W_k \tag{5}$$

$$(\forall R_{mn})[R_{mn} \in (O_r \cap W_k)] \rightarrow (\exists t_r)\{[t_r = B(R_{mn})] \wedge (t_r \in E_k)\} \Rightarrow (t_r < t_k) \tag{6}$$

$$O_r = O_r - \{R_{mn}\} \tag{7}$$

其中, 式 (1) ∼ (5) 为设备驱动的约束条件, 式 (6) 和式 (7) 为回退的约束条件. 下面将对本文所建数学模型的约束条件进行详细分析.

2 模型分析

根据加工工序是否存在紧前工序, 将工序划分为不可调度工序和可调度工序.

定义 1 (不可调度工序). 紧前工序未加工完毕的工序.

定义 2 (可调度工序). 无紧前工序或紧前工序均已加工完毕的工序, O_k 即为可调度工序集.

定义 3 (设备集). 调度系统中所有加工设备所组成的集合, 其设备数量即为 M.

定义 4 (工序集). 调度系统中所有加工工序所组成的集合, 其工序数量即为 N.

当 t_k 时刻设备出现空闲, 若某一不可调度工序的紧前工序均已加工完毕, 即该工序的紧前工序在集合 X_k 中, 则该工序转化为新的可调度工序. 由分析可知, 时刻 t_k 到来时: 1) 设备出现空闲; 2) 可能产生新的可调度工序. 因此, 在时刻 t_k 空闲的设备可以驱动产生一次新的调度.

下面利用事件驱动工序调度的思想对设备驱动的约束条件进行分析.

2.1 设备驱动事件分析

定义 5 (加工完毕事件). 某工序刚结束加工时工序集产生的事件. 该事件发生时, 其加工设备产生空闲, 且该加工完毕工序的紧后工序可能成为新的可调度工序.

定义 6 (设备驱动时刻 t_k). 有工序加工完毕的时刻, 该时刻产生加工完毕事件开始一次调度. 系统设备集中的设备可以分为两种状态: 设备空闲状态、设备忙碌状态.

定义 7 (设备空闲状态). 加工设备上无加工工序时的状态.

定义 8 (设备忙碌状态). 加工设备上存在正在加工工序时的状态. 系统工序集中的工序可以分为 4 种状态: 不可调度状态、可调度状态、加工状态、加工完毕状态.

定义 9 (可调度状态). 时刻 t_k 可调度工序集合 O_k 中各工序所处的状态.

定义 10 (不可调度状态). 时刻 t_k 紧前工序未加工完毕的工序所处的状态.

定义 11 (加工状态). 时刻 t_k, W_k 中各工序所处的状态, 即工序正在加工时的状态.

定义 12 (加工完毕状态). t_k 时刻, X_k 中各工序所处的状态. 调度过程中, 当有工序加工完毕时, 即为一次设备驱动时刻 t_k, 此时工序集触发加工完毕事件, 开始一次调度;

1) 对于当前刚加工完毕的工序集合 j 即 $\text{end}(j) = t_k$, 时刻 t_k 加工完毕的工序集 $X_k = X_{k-1} \cup \{j\}, 1 \le j \le N$, $\text{end}(j) = t_k$, 即满足约束条件 1);

2) 时刻 t_k 空闲设备 m 在可调度工序集 O_k 中选择适合的工序 c_{mk} 作为该设备的加工工序. 由于空闲设备可能不唯一, 因此, 时刻 t_k 开始的加工工序集 C_k 为当前可调度工序集 O_k 的子集, 且 C_k 在其紧前工序结束后才可以开始加工, 于是 $B(C_k) \ge \max[\text{end}(j)], j \in X_k, C_k = \{c_{mk}\}, C_k \subseteq O_k$, 即满足约束条件 2);

3) 时刻 t_k 空闲设备选择工序后, 加工状态工序集 W_{k-1} 发生改变: a) 时刻 t_k 所有刚刚加工完毕的工序, 即为时刻 t_{k-1} 处于加工状态, 时刻 t_k 处于完毕状态的工序 (利用变量表示为 $\overline{(W_{k-1} \cap X_k)}$) 移出加工状态集合 $W_{k-1}, W_{k-1} = (W_{k-1} \cap \overline{(W_{k-1} \cap X_k)})$; b) 时刻 t_k 空闲设备选择的工序 C_k 加入加工状态集合 W_{k-1}, $W_k = W_{k-1} \cup C_k$. 两次更新后的 W_{k-1} 变为时刻 t_k 调度结束后加工状态工集 $W_k = (W_{k-1} \cap \overline{(W_{k-1} \cap X_k)}) \cup C_k$, 即满足约束条件 3);

4) 将该时刻 t_k 加入 E_{k-1}, 产生新的工序完成时间集, 即 $E_k = E_{k-1} \cup \{t_k\}$, 满足约束条件 4);

5) 由于下一次设备驱动时刻 t_{k+1} 为当前加工状态工序集 W_k 中最早完工工序的结束时刻, 所以 $t_{k+1} = t_k + \min\{\text{end}(i)\}$, $i \in W_k$, 即满足约束条件 5), 并在时刻 t_{k+1} 产生新的加工完毕事件.

以上设备驱动过程形成如图 1 所示的由上至下的加工完毕事件顺序图, 即通过对当前设备驱动时刻的处理, 算出下一次设备驱动时刻, 并在下一次设备驱动时刻触发新的加工完毕事件. 当后续设备驱动时刻 t_p 到来时, 所有工序均已加工完毕, 即 $X_p = 1, 2, \cdots, N$, 则时刻 t_p 即为产品加工完成时刻 T. 为了更详细描述加工完毕事件顺序图中设备驱动时刻系统状态, 通过给出与图 1 相应的时刻 t_p 系统状态转化图 (见图 2), 图 2 中阴影部分为设备集状态, 进一步说明所设计的设备驱动方案约束条件的合理性.

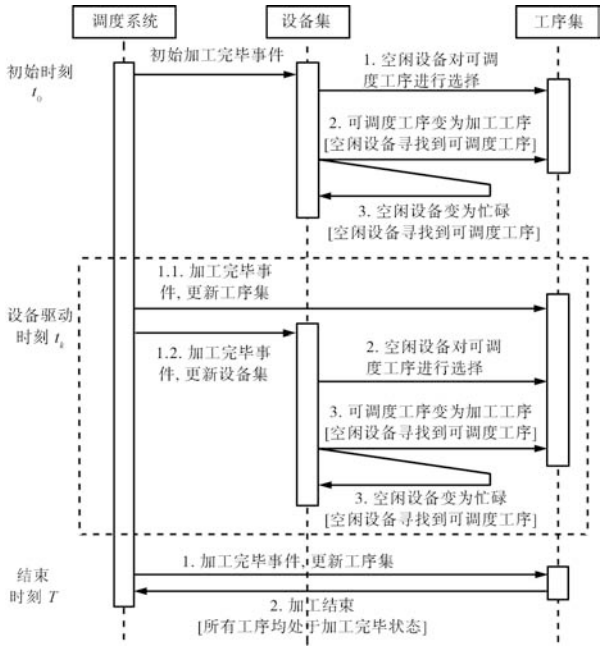


图 1　加工完毕事件顺序图

Fig. 1　The events sequence diagram of scheduling system

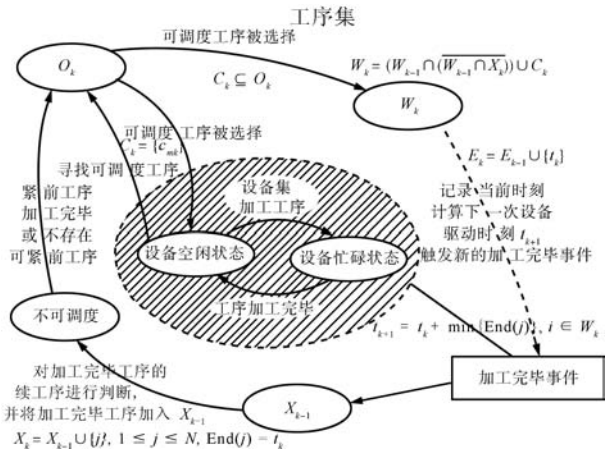


图 2　调度系统中设备集和工序集状态转化图

Fig. 2　The states and events conversion graph of scheduling system

2.2　工序抢占分析

当工序 F 加工完毕时, 产生设备驱动时刻 t_k, 若此时设备 m 有正加工工序 A 和新的可调度工序 D, 由于工序 A 没加工完, 设 A 已加工部分为 A[1], A 未加工部分为 A[2], 如图 3 所示. 由于 A 和 D 具有设备相关性, 此时有两种调度方案: 1) 正常加工, 即 D 作为 A 的后续; 2) 抢占, 即推迟加工工序 A, 使 A 作为 D 的后续. 两种方案加工模型如图 4 所示, 调度方案如图 5 所示, 为了说明优先调度父结点路径长的工序可以提高调度效率, 下面分析两种不同的调度方案的调度结果.

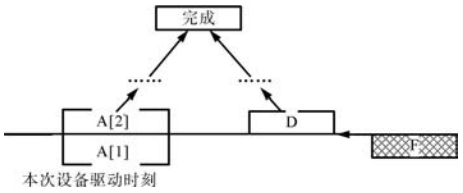


图 3　工序 F 加工完毕时产品加工模型

Fig. 3　The processing model at process F finished

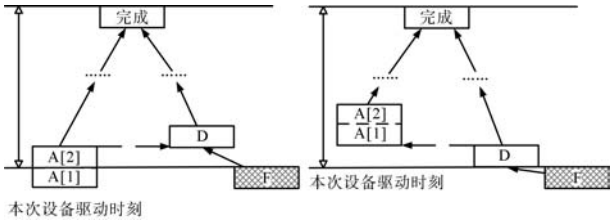


图 4　两种工艺模型

Fig. 4　Two kinds of process models

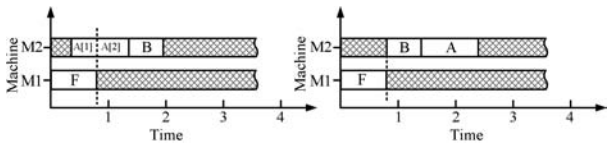


图 5　两种加工结果甘特图

Fig. 5　Two kinds of processing result Gantt chart

设工序 A 的加工时间为 G_A, 于是 $G_A = G_{A[1]} + G_{A[2]}$; 工序 A 和工序 D 到根结点的路径长度分别为 G_A 和 G_D; 方案 1): D 在 A 加工完毕后开始加工, 由于 D 可与工序 A 的后续工序并行, 产品完成时间为 $T_1 = \max[(t_k + G_{A[2]} + Q_A), (t_k + G_{A[2]} + G_D + Q_D)]$; 方案 2): 若 D 抢占 A, 则 A 在 D 加工完毕后开始加工, 且可与 D 的后续工序并行, 此时产品完成时间为 $T_2 = \max[(t_k + G_D + Q_D), (t_k + G_D + G_A + Q_A)]$. 于是两种方案中最优结果为 $\min\{\max[(t_k + G_{A[2]} + Q_A), (t_k + G_{A[2]} + G_D + Q_D)], \max[(t_k + G_D + Q_D), (t_k + G_D + G_A + Q_A)]\}$, 即时刻 t_k 产品完成预测时间 $T_k = \min(T_1, T_2)$. 为了比较 T_1 和 T_2, 需判断 $T_1 - T_2$.

$$T_1 - T_2 = (G_{A[2]} - G_D) + \{\max[Q_A, (G_D + Q_D)] - \max[Q_D, (G_A + Q_A)]\}$$

根据上式中的各种取大可能, 共有以下 4 种情况;

$$\begin{cases} Q_A > (G_D + Q_D), Q_D > (G_A + Q_A) & 1) \\ Q_A > (G_D + Q_D), Q_D < (G_A + Q_A) & 2) \\ Q_A < (G_D + Q_D), Q_D > (G_A + Q_A) & 3) \\ Q_A < (G_D + Q_D), Q_D < (G_A + Q_A) & 4) \end{cases}$$

根据情况 1), 有 $Q_A > Q_D > Q_A$, 即情况 1) 矛盾,

T_1 和 T_2 在另外 3 种情况的结果如下:

$$\begin{cases} 2) \Rightarrow \begin{cases} T_1 = t_x + G_{A[2]} + Q_A \\ T_2 = t_x + G_D + (G_A + Q_A) \end{cases} \\ 3) \Rightarrow \begin{cases} T_1 = t_x + G_{A[2]} + (G_D + Q_D) \\ T_2 = t_x + G_D + Q_D \end{cases} \\ 4) \Rightarrow \begin{cases} T_1 = t_x + G_{A[2]} + (G_D + Q_D) \\ T_2 = t_x + G_D + (G_A + Q_A) \end{cases} \end{cases}$$

计算 $T_1 - T_2$, 结果如下:

$$\Rightarrow \begin{cases} -(G_{A[2]} + G_D) & 2) \\ G_{A[2]} & 3) \\ Q_D - (Q_A + G_{A[1]}) & 4) \end{cases}$$

因为情况 2) 时, $T_1 < T_2$, 此时选择方案 1); 情况 3) 时, $T_1 > T_2$, 此时选择方案 2) 进行抢占; 情况 4) 时, 若 $(Q_A + G_{A[1]}) < Q_D$, $T_1 > T_2$, 此时选择方案 2) 进行抢占, 若 $(Q_A + G_{A[1]}) > Q_D$, $T_1 < T_2$ 此时选择方案 1).

通过以上分析, 情况 3) 选择方案 2) 进行抢占, 情况 4) 部分条件选择方案 2) 进行抢占. 为了将进行抢占的条件统一起来, 便于调度时进行抢占判断, 分析情况 3) 和 4) 能进行抢占的条件如下:

$$\begin{cases} 3) \Rightarrow \begin{cases} G_{A[2]} > 0 \\ Q_A < (G_D + Q_D) \\ Q_D > (G_A + Q_A) \end{cases} \\ 4) \Rightarrow \begin{cases} [Q_D - (Q_A + G_{A[1]})] > 0 \\ G_A < (G_D + Q_D) \\ Q_D > (G_A + Q_A) \end{cases} \end{cases}$$

$$\Rightarrow (G_{A[1]} + Q_A) < Q_D$$

通过分析, 进行抢占的条件是: 正在加工工序 (被抢占工序) 的已加工时间与其父结点路径长度之和小于新可调度工序 (抢占工序) 的父结点路径长度. 例如, 对图 6 所示的产品 F 进行加工, 如果采取设备驱动方法进行调度, 当可调度工序不唯一时, 采取最大并行性选择策略调度, 即优先调度父结点路径长的工序, 如果父结点最长路径相同, 选择用时长的工序. 于是产品 F 形成的结果甘特图如图 7 所示, 加工总时间为 22 个时间单位.

对于以上方法, 当 $t_1 = 4$ 时, 工序 F_6 成为可调度工序, 由于 F_5 已加工时间为 4, 父结点路径长度为 7, 工序 F_6 父结点路径长度为 13, $(Q_{F5} + G_{F5[1]}) = 11$, $Q_{F6} = 13$ 满足抢占条件, 于是当 F_6 抢占 F_5 时, 形成新的产品加工甘特图如图 8 所示, 加工总时间为 20. 说明抢占回退可以优化产品调度结果.

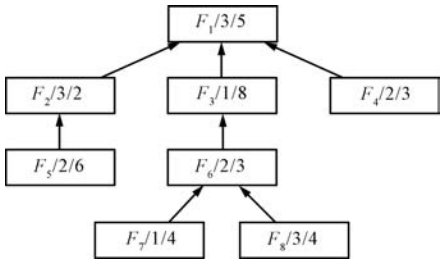


图 6 产品 F 加工工艺图

Fig. 6 Processing flow chart of product F

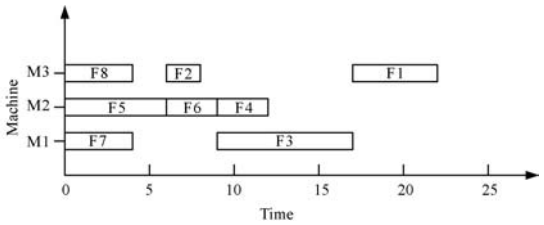


图 7 产品 F 加工甘特图

Fig. 7 Processing Gantt chart of product F

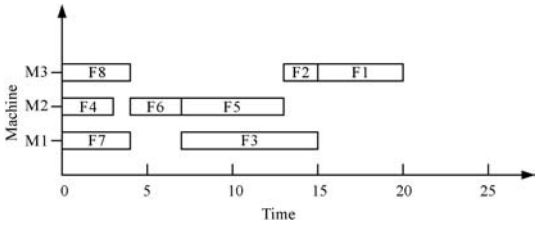


图 8 推迟调度工序 F_5 产品 F 加工甘特图

Fig. 8 Gantt chart for postponing procedure F_5 of product F

2.3 回退分析

定义 13 (被抢占工序). 设时刻 t_k 设备 m 上加工的工序 F_{mn} 处于加工状态, 若此时用新的可调度工序 F_{mj} 替换工序 F_{mn} 可以得到更优的调度结果, 则工序 F_{mn} 称为被抢占工序 (即上文描述中 R_{mn}).

定义 14 (回退事件). 在某一设备驱动时刻 t_k, 由于替换被抢占工序, 整个调度过程分析前移的事件.

定义 15 (回退时刻). 一次回退事件前移到的时刻, 即被抢占工序 R_{mn} 的开始加工时刻 $B(R_{mn})$.

定义 16 (状态栈). 该栈记录各设备驱动时刻设备集、工序集中元素所处的状态. 当加工完毕事件发生, 设备集和工序集状态更新完毕后, 系统将设备集和工序集中各元素的状态压入状态栈中. 当回退事件发生时, 状态栈根据记录的设备驱动时刻进行退栈, 还原系统设备集和工序集中各元素状态.

由回退问题分析可知, 当有回退事件发生时, 系统重新对已调度工序进行重新选择, 其过程为:

1) 设备驱动时刻 t_k, 若存在被抢占工序 R_{mn}, 必有 $t_r \in E_k$ 且 $t_r = B(R_{mn})$. 满足约束条件 6). 当系统回退到时刻 t_r, 利用状态栈恢复时刻 t_r 各元素的状态;

2) 在系统回退时刻 t_r, 为了避免被抢占工序 R_{mn} 再次调度, 无法达到回退的目的, 采取对工序 R_{mn} 睡眠的策略, 即排除对被抢占工序 R_{mn} 的一次选择 $O_r = O_r - \{R_{mn}\}$, 即满足约束条件 7). 随后按加工完毕事件的处理过程完成时刻 t_r 工序调度, 通过更新后的 W_r 计算下一设备驱动时刻 t_{r+1}, 并在设备 m 下一次空闲时, 将工序 R_{mn} 的状态转化为可调度状态.

通过第 2.1 节和第 2.3 节的分析, 所建问题数学模型中所有约束条件均符合设备驱动和回退策略的实际约束.

3 问题求解

由模型分析可知, 时刻 t_k 回退事件的发生依赖两个条件: 1) 产生新的可调度工序; 2) 新的可调度工序的加工设备忙碌. 因此, 在时刻 t_k 应该先对忙碌设备进行回退判断, 若产生回退事件, 则进行回退处理, 否则驱动空闲设备进行可调度工序的寻找.

因此, 可回退抢占的设备驱动综合调度需要解决: 1) 空闲设备可调度工序的寻找; 2) 忙碌设备回退判断. 为了提高调度效率, 分别对两个需要解决的问题提出了最大并行性选择策略和退回策略.

3.1 最大并行性选择策略与实现

当设备驱动时, 若空闲设备仅有一个可调度工序, 则选择该可调度工序; 若空闲设备无可调度工序, 该空闲设备仍处于空闲状态; 若空闲设备有多个可调度工序, 则选择调度父结点路径最长的工序, 原因如下.

若设备驱动时刻 t_k, 空闲设备 i 的可调度工序不唯一, 有 $F_{i,1}, F_{i,2}, \cdots, F_{i,j}$, 设对应各可调度工序的父结点路径长度分别为 $Q_{i,1}, Q_{i,2}, \cdots, Q_{i,j}$, 各可调度工序的加工时间分别为 $I_{i,1}, I_{i,2}, \cdots, I_{i,j}$. 由于在同一设备上可同时加工的工序必须串行加工, 因此, 需要考虑串行加工对调度结果的影响. 由于选择工序的后续工序可与其他串行工序并行处理, 所以优先调度父结点路径长的工序可增加设备的并行处理, 提高调度效率. 比较分析如下.

先考虑对只有两个可调度工序 $F_{i,1}$, $F_{i,2}$ 的情况, 此时有 2 种方案, $F_{i,1}$ 或 $F_{i,2}$ 先加工.

方案 1. 选择 $F_{i,1}$, 则两个可调度工序及其后续工序的最早完成时间 $I_{F1} = I_{i,1} + \max[Q_{i,1}, (I_{i,2} + Q_{i,2})]$;

方案 2. 选择 $F_{i,2}$, 则两个可调度工序及其后续工序的最早完成时间 $I_{F2} = I_{i,2} + \max[Q_{i,2}, (I_{i,1} + Q_{i,1})]$.

式中, $\max(a_1, a_2)$ 表示 a_1 和 a_2 并行处理的最晚结束时间. 此时, 为了产品加工尽早结束, 设计同一设备选择调度工序的目标函数 $S = \min(I_{F1}, I_{F2})$.

为了比较 I_{F1} 和 I_{F2}, 判断:

$$I_{F1} - I_{F2} = (I_{i,1} - I_{i,2}) + \{\max[Q_{i,1}, (I_{i,2} + Q_{i,2})] - \max[Q_{i,2}, (I_{i,1} + Q_{i,1})]\}$$

根据上式中的各种取大可能进行以下 4 种情况分析:

$$\begin{cases} Q_{i,1} > (I_{i,2}), Q_{i,2} > (I_{i,1} + Q_{i,1}) & 1) \\ Q_{i,1} > (I_{i,2}), Q_{i,2} < (I_{i,1} + Q_{i,1}) & 2) \\ Q_{i,1} < (I_{i,2}), Q_{i,2} > (I_{i,1} + Q_{i,1}) & 3) \\ Q_{i,1} < (I_{i,2}), Q_{i,2} < (I_{i,1} + Q_{i,1}) & 4) \end{cases}$$

根据情况 1), 有 $Q_{i,1} \geq Q_{i,2} \geq Q_{i,1}$, 即情况 1) 矛盾, I_{F1} 和 I_{F2} 在另外 3 种情况的结果如下:

$$\begin{cases} 2) \Rightarrow \begin{cases} I_{F1} = I_{i,1} + Q_{i,1} \\ T_{F2} = I_{i,2} + I_{i,1} + Q_{i,1} \end{cases} \\ 3) \Rightarrow \begin{cases} I_{F1} = I_{i,1} + I_{i,2} + Q_{i,2} \\ I_{F2} = I_{i,2} + Q_{i,2} \end{cases} \\ 4) \Rightarrow \begin{cases} I_{F1} = I_{i,1} + I_{i,2} + Q_{i,2} \\ T_{F2} = I_{i,2} + I_{i,1} + Q_{i,1} \end{cases} \end{cases}$$

计算 $I_{F1} - I_{F2}$, 结果如下:

$$\Rightarrow \begin{cases} -I_{i,2} & 2) \\ I_{i,1} & 3) \\ Q_{i,2} - Q_{i,1} & 4) \end{cases}$$

情况 2) 时, $I_{F1} < I_{F2}$, 选择方案 1; 情况 3) 时, $I_{F1} > I_{F2}$, 选择方案 2;

情况 4) 时, 若 $Q_{i,2} < Q_{i,1}$, 选择方案 1, 若 $Q_{i,2} > Q_{i,1}$, 选择方案 2. 通过以上分析, 情况 3) 选择方案 2, 情况 4) 部分条件选择方案 2. 将条件统一起来得出选择方案 2 的条件如下:

$$\begin{cases} 3) \Rightarrow \begin{cases} I_{i,1} > 0 \\ Q_{i,1} < (I_{i,2} + Q_{i,2}) \\ Q_{i,2} > (I_{i,1} + Q_{i,1}) \end{cases} \\ 4) \Rightarrow \begin{cases} (Q_{i,2} - Q_{i,1}) > 0 \\ Q_{i,1} < (I_{i,2} + Q_{i,2}) \\ Q_{i,2} > (I_{i,1} + Q_{i,1}) \end{cases} \end{cases}$$

$\Rightarrow (Q_{i,1} < Q_{i,2})$

即优先选择调度父结点路径最长的工序.

当 $Q_{i,1} = Q_{i,2}$, 根据动态关键路径法[16], 优先调度加工时间长的工序.

所以从两个工序中选择的最大并行性选择策略是: 优先调度父结点路径最长的工序, 当父结点路径长度相同时, 优先调度加工时间长的工序.

对于有多个可调度工序的情况, 可推广两个工序的最大并行性选择策略, 优先调度父结点路径长的工序, 当父结点路径长度相等时, 优先调度加工时间长的工序. 为了利用最大并行性选择策略确定一个空闲设备上的可调度工序, 首先, 需要确定设备的空闲时刻, 为此, 设计算法 1 确定设备空闲时刻, 并在设备空闲事件发生时, 寻找可调度工序和算法 2 确定一个空闲设备上的可调度工序.

算法 1. $select\_operations()$

以正在加工工序最早结束时间为下一设备空闲时刻, 并在设备空闲事件发生时寻找可调度工序.

$\#define MAXTIME 32678$

Begin

01 $NT = MAXTIME$

NT 为下一次空闲事件发生的时刻, D 为加工设备 S_O 为设备上所选择的工序.

02 For $(d; 0 < d \leq; d++)\{$

03 If $(D[d].idle ==$ false)

$Device[d]$ 为处于忙碌状态的设备

04 If $(D[d].S\_O.endtime \leq NT)$

05 $NT = D[d].S\_O.endtime$

$selected\_operation.endtime$ 为正在加工工序的结束时间.

06 For $(d; 0 < d \leq D; d++)$

07 If $(D[d].idle \leq$ true)

$Device[d]$ 为空闲设备

08 $get\_schedulable\_operations(d)$

为空闲设备上的可调度工序.

End

算法 2. $get\_schedulable\_operations(d)$

输入: 对空闲设备 d 进行一次可调度工序的寻找;

输出: 为空闲设备 d 分配一个可调度工序, 计算该工序的加工结束时刻, 并将该空闲设备标记为非空闲.

H 表示设备最大编号, 设备集中各设备编号为 $1, 2, \cdots, H$;

T 表示当前设备驱动时刻 (初始时刻 $T = 0$ 时, 所有设备均为空闲设备); Device[·] 表示设备集数组; S_O 表示可在设备 i 上加工的可调度工序集合.

Begin

01 $SO = 0$

02 If (size of$(SO[i])$)

存在可在设备 i 上加工的可调工序

03 For $(p = 0; p <$ size of$(SO[i]); p++)$

04 If $(D[d].idle ==$ true)

$Device[d]$ 为空闲设备

05 $get\_schedulable\_operationS(d)$

利用最大并行性选择策略确定可调度工序

06 $D[d].S_O = selected\_operation$

为空闲设备 d 分配一个可调度工序

07 $D[d].S_O.endtime = T + S_O.worktime$

计算该工序的加工结束时刻

08 $D[d].idle =$ false

将该空闲设备标记为非空闲

End

3.2 回退策略和可靠性分析

由于回退的条件是抢占条件成立, 所以回退策略是: 如果有抢占工序, 调度系统状态回退到被抢占工序开始加工时刻, 并将被抢占工序睡眠至下一设备驱动时刻, 其他可调度工序正常调度.

由于本算法提出的时间状态栈和抢占条件可以避免回退事件, 可能形成 3 种情况: 1) 回退的死循环; 2) 调度结果更差; 3) 连续回退, 说明所提出的回退策略可靠性好, 下面予以说明.

回退事件可能会使后续调度过程多次回退到同一时刻, 形成回退的死循环. 例如, 图 9 所示的 P1 和 P2 均为时刻 t_1 可在设备 M1 上加工的可调度工序, 当工序 F 在时刻 t_2 成为新可调度工序时, 可抢占工序 P1 或 P2. 使调度从时刻 t_2 回退到时刻 t_1, 设备 M1 在 t_1 重新调度工序 P1 或 P2, 在时刻 t_2 再被 F 抢占, 回退到 t_1, 形成 t_2 回退到 t_1 的死循环.

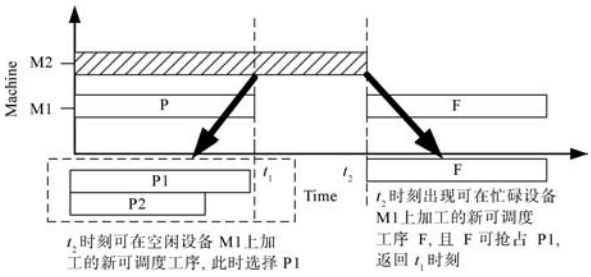


图 9 t_2 到来时的调度结果

Fig. 9 Schedule result when t_2 arrives

1) 由于本文采用了基于时间的状态栈回退策略, 每次回退事件发生时, 被抢占工序在其原有开始时间为睡眠状态, 使被抢占的工序无法参与其原有开始时间的调度. 若后续回退事件 K 次回退到同一时刻, 则该时刻的睡眠工序数量为 K, 即每次回退排除一个可调度工序, 由于可调度工序有限, 经过若干次回退后可筛选出最优的调度方案. 图 10 表示经过两次回退后, 时刻 t_1, P1 和 P2 均被标记为睡眠, 不参与时刻 t_1 调度, 即不会出现回退调度的死循环.

2) 对于调度结果更差的问题, 例如经过两次回退后, 时刻 t_2, P1 和 P2 均被唤醒成为可调度工序, 空闲设备 M1 上的可调度工序为 P1、P2、F. 此时, 若 M1 选择 P1 或 P2, 会导致调度结果更

差. 由于工序 F 可抢占工序 P1 和 P2, 根据回退策略的判断条件可知满足 $(Q_{P1} + G_{P1[1]}) < Q_F$ 和 $(Q_{P2} + G_{P2[1]}) < Q_F$, 于是有 $Q_{P1} < Q_F$ 和 $Q_{P2}\ Q_F$, 即 F 的父结点路径长度最大, 根据最大并行性选择策略, 时刻 t_2 M1 优先选择 F, 所以本算法不会出现调度结果更差的情况.

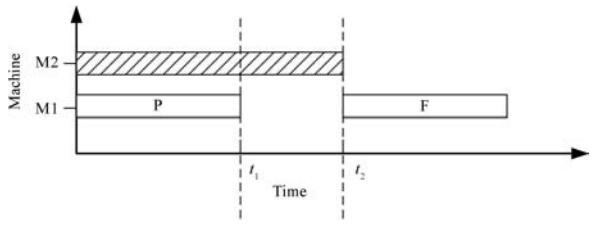


图 10　经过多次回退 t_1 后的调度结果

Fig. 10　Schedule result after many rollbacks of t_1

3) 由上文中目标函数的约束条件 5)～7) 可知, 若 t_r 为由 t_k 回退的时刻, 则在时刻 t_r 不再发生连续回退事件. 事实上, 由时刻 t_r 到时刻 t_k, 时刻 t_r 已完成回退判断, 即不再回退, 又由约束条件 7) 可知, 在回退时刻 t_r 将被抢占工序标记为睡眠状态, 缩小了时刻 t_r 空闲设备可调度的工序集, 因此, 不会改变时刻 t_r 回退判断结果. 即由于不再发生连续回退事件, 不会出现无限回退问题.

4　算法设计及复杂度分析

4.1　算法设计

由于所研究的调度系统由工序集、设备集和状态栈构成, 在设备驱动时刻, 设备集和工序集分别通过触发回退事件和加工完毕事件反馈调度系统, 所以根据不同事件影响调度系统设计调度算法.

算法设计思想如下:

1) 初始时刻 t_0 所有设备均空闲. 系统驱动空闲设备利用最大并行性选择策略对可调度工序进行选择, 并计算下一设备驱动时刻 t_1, 并在 t_1 时刻由工序集触发加工完毕事件.

2) 若所有工序处于加工完毕状态, 转 6).

3) 根据目标函数的约束条件 1)～5) 更新设备集和工序集中各元素状态, 并将设备集和工序集的状态压入状态栈中, 对新可调度工序通过抢占分析进行回退判断: a) 若满足回退条件, 利用回退策略计算下一设备驱动时刻, 转 4); b) 若不满足回退条件, 转 5).

4) 状态栈根据目标函数的约束条件 6) 和 7), 恢复设备集和工序集中各元素为回退时刻状态.

5) 利用最大并行性选择策略选择调度工序并计算下一设备驱动时刻, 并在下一设备驱动时刻由工序集触发加工完毕事件, 转 2).

6) 产品加工结束, 将系统当前状态压入栈中, 并记录当前的时间为产品加工总时间 T.

算法流程图如图 11 所示.

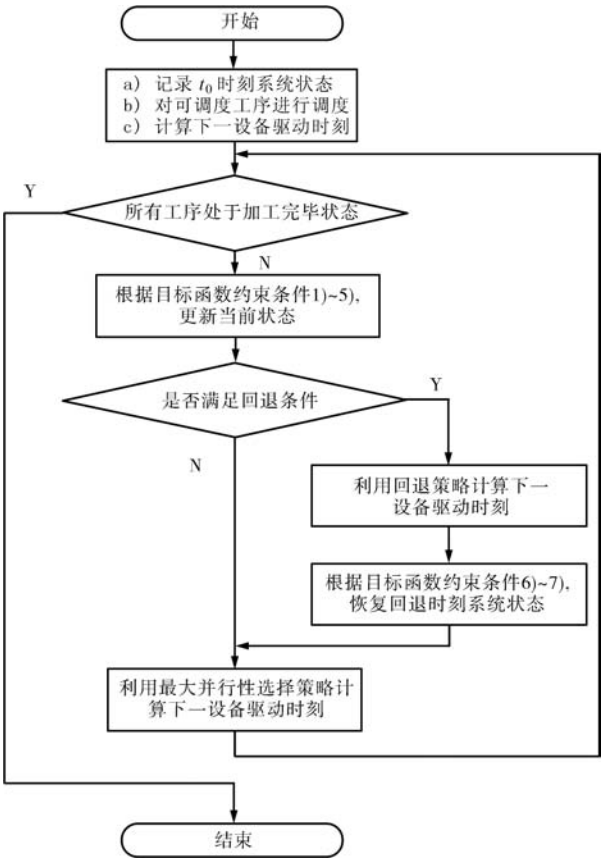


图 11　算法流程图

Fig. 11　The flow diagram of the algorithm

4.2　复杂度分析

本算法复杂之处在于, 设备驱动时刻空闲设备利用最大可并行策略对可调度工序的选择和回退策略对被抢占工序的判定. 具体分析如下:

设调度系统中工序集中工序数为 n, 设备集中设备总数为 M.

1) 调度系统初始时, 工序排序:

将所有工序父结点路径长度按 Quick-sort 快速排序法排序, 工序排序复杂度为 $\mathrm{O}(n\lg n)$.

2) 设备驱动时刻, 若不产生回退事件系统需要按如下顺序进行操作:

a) 加工完毕工序转入加工完毕工序集 X_k;

b) 判断是否唤醒睡眠工序;

c) 新的可调度工序转入可调度工序集 O_k;

d) 对忙碌设备进行回退判断;

e) 空闲设备选择可调度工序;

f) 将该设备驱动时刻记录到空闲时刻集 E_k 中;

g) 计算下一设备驱动时刻 t_{k+1}.

由于空闲设备最多为 M 台, 所以操作 a) ∼ d), 每步最多进行 M 次; 由于已对所有工序的父结点路径长度排序, 单一空闲设备选择可调度工序只需 1 次操作, 因此操作 e) 最多进行 M 次; 操作 f) 只进行 1 次; 由于处于加工状态的工序最多为 M, 操作 g) 确定 M 个工序的最早结束时间, 最多进行 $M-1$ 次比较. 因此, 每一次设备驱动时刻若不产生回退事件最多进行 $6M$ 次操作, 所有工序按设备驱动正向调度的次数为 $6Mn$.

3) 产生回退事件系统需要按顺序进行如下操作:

a) 计算回退时刻 t_r;

b) 根据时刻 t_r 状态栈中的记录, 将从时刻 t_r 到 t_k 时刻状态发生改变的工序, 移出时刻 t_k 的状态集并加入到时刻 t_r 的状态集;

c) 将回退工序标记为睡眠.

其中, 操作 a) 和 c) 只需进行 1 次, 从时刻 t_r 到时刻 t_k 状态发生改变的工序数 $\leq n$, 因此, 操作 b) 最多进行 n 次状态调整. 所以每一次设备驱动时刻若产生回退事件最多进行 $2+n$ 次操作, 如果该回退重新调度所有工序, 那么该回退产生的全部操作最多为 $6Mn+2+n$ 次. 设 n 个工序都调度完后发生全回退, 调度全部操作最多为 $f(n)$;

设 n 个工序都调度完后发生全回退, 调度全部操作最多为 $f(n)$:

$$f(n) = n(6Mn+2+n)+6Mn$$

对于前续设备驱动时刻, 最多调度完 $n-1$ 个工序, 则该时刻调度全部操作最多为 $f(n-1)$, 因此, 可抢占回退的设备驱动综合调度总操作次数为

$$\begin{aligned} &f(n)+f(n-1)+\cdots+f(1)= \\ &\quad (M+1)[n^2+(n-1)^2+\cdots+1^2]+ \\ &\quad (6M+2)[n+(n-1)+\cdots+1]= \\ &\quad \left(2M+\frac{1}{6}\right)^3+\left(3M+\frac{3}{2}\right)n^2+(1-3M)n \end{aligned}$$

所以, 可回退抢占的设备驱动综合调度算法的复杂度为 $\max[\mathrm{O}(n^3), \mathrm{O}(n\lg n)] = \mathrm{O}(n^3)$.

5 实例分析

对于文献 [8] 中如图 12 所示的产品 A, 分别采用文献 [8] 考虑关键设备紧凑, 文献 [15] 考虑优先工序集, 文献 [16] 考虑动态关键路径等算法调度和一般设备驱动算法调度, 得到如图 13 所示的调度结果甘特图.

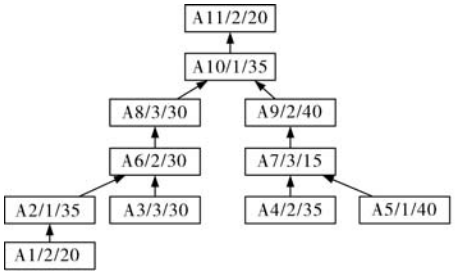
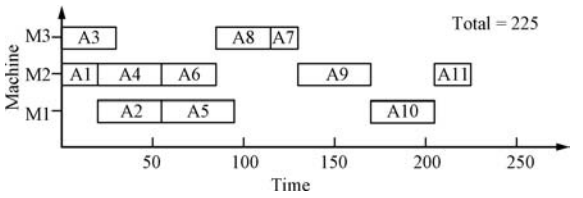


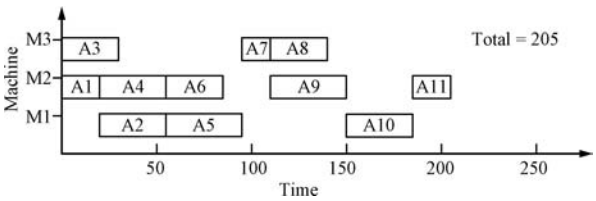
图 12 产品 A 加工工艺图

Fig. 12 Process flow chart of product A



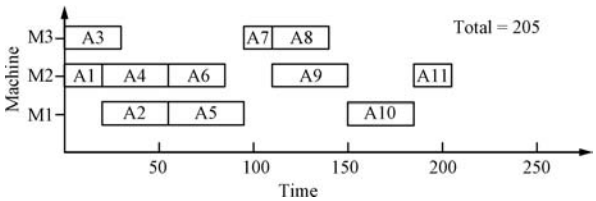
(a) 文献 [8] 算法结果

(a) The result of [8]



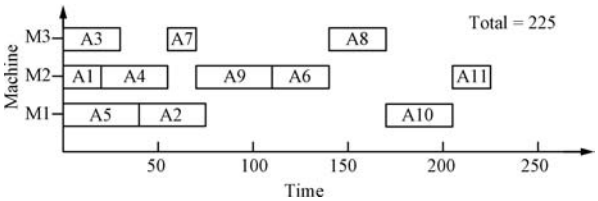
(b) 文献 [15] 算法结果

(b) The result of [15]



(c) 文献 [16] 算法结果

(c) The result of [16]



(d) 无回退设备驱动算法

(d) The result of no-rollback scheduling algorithm

图 13 不同算法对产品 A 调度结果的甘特图

Fig. 13 Gantt charts obtained by using different algorithms to schedule product A

采用本文算法的调度步骤说明如表 1 所示, 调度结果甘特图如图 14 所示, 对比调度结果得到如表 2 所示的对比表.

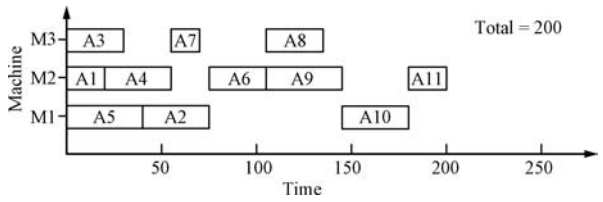


图 14 利用本文算法实现的对产品 A 进行加工的调度甘特图

Fig. 14 Gantt chart obtained by using the algorithm proposed by this paper to schedule product A

由表 2 可知, 本文算法的产品完成时间最少, 之所以本文算法调度结果更优, 是因为:

文献 [8] 采用 ACPM 法调度, 调度单位是工序组, 形成工序组间的并行处理, 由于并行处理单位较大, 使设备产生较多空闲时间段, 影响设备利用率. 例如该例工序组序列: 1) A1, A2; 2) A3, A6, A8; 3) A5; 4) A4, A7, A9, A10, A11. 文献 [8] 中算法由于先调度了工序 A2, 在图 13 (a) 的 M1 上产生了不可利用的空闲 (0) ∼ (20), 使得后续工序 A5 只能在 A2 结束后加工, 从而降低了产品的并行性.

由于文献 [15] 中的优先级算法和文献 [16] 中的动态关键路径算法都是先确定工序的调度次序, 再按确定次序将工序调度到加工设备上, 虽然这两种调度算法的调度单位是工序, 但由于后续工序只能在先序工序的空隙中进行调度, 当空闲时间段较小时, 不仅工序延迟加工, 而且影响同时加工设备之间的并行处理, 此时, 以工序为调度单位的这两种算法也不能实现工序间的充分并行处理. 如文献 [15] 中算法的加工次序为

A1, A3, A2, A4, A5, A7, A6, A9, A8, A10, A11

文献 [16] 中算法的加工次序为

A1, A2, A5, A3, A4, A7, A6, A9, A8, A10, A11

由于文献 [15] 和文献 [16] 中算法都先调度工序 A2, 在图 13 (b) ∼ (c) 的 M1 上产生了不可利用的空闲时间段 (0) ∼ (20), 降低了设备的利用率.

表 1 本文算法调度步骤说明

Table 1 Scheduling process statistics for the algorithm proposed by this paper

设备驱动时刻	空闲设备	新可调度工序	回退时刻	M1 加工	M2 加工	M3 加工
t_0 0	M1, M2, M3	A1, A3, A4, A5		A5	A1	A3
t_1 20	M3	A2		A5	A4	A3
t_2 30	M2, M3			A5	A4	
t_3 40	M1, M3			A2	A4	
t_4 55	M2, M3	A7		A2		A7
t_5 70	M2, M3	A9		A2	A9	
t_6 75	M1, M3	A6	t_5			
t_5 70	M2, M3			A2		
t_6 75		A6			A6	
t_7 105	M1, M2, M3	A8, A9			A9	A8
t_8 135	M1, M3				A9	
t_9 145	M1, M3	A10		A10		
t_{10} 180	M1, M2, M3	A11			A11	
t_{11} 200 结束	M1, M2, M3					

表 2 调度结果对比表

Table 2 Contrast table of scheduling results

各类算法	设备空闲时间和	最晚结束设备 M1 上的空闲时间	完工时间
本文算法	185	55	200
关键设备紧凑	200	80	225
动态关键路径	200	60	205
优先工序集	200	60	205
一般设备驱动算法	240	80	225

由于本文算法在加工设备空闲时, 无需按预先确定的次序调度工序, 只需选择合适的可调度工序, 所以可以形成真正工序间的并行操作且不存在工序比较插入空闲时间段的操作, 避免产生文献[8, 15−16] 算法中出现的设备不可利用空闲时间段. 又由于在后续调度过程中利用回退策略优先调度对调度结果有重要影响的动态长路径上的工序, 所以利用回退策略可使调度结果进一步优化, 例如图 14 中, 由于工序 A6 采用回退策略对工序 A9 抢占, 使本文算法的调度结果图 14 比一般设备驱动算法的调度结果图 13 (d) 减少了 25 个工时.

通过以上分析可知, 以往算法先确定了工序的调度次序, 或者以工序组作为调度单位, 或者需要工序比较插入空闲时间段, 不仅影响设备的充分利用, 而且增加确定工序开始时间的比较次数. 而本文算法是在设备空闲的时刻对可调度工序进行选择, 既不存在工序组间的并行调度, 也不存在工序比较插入空闲时间段的操作, 通过工序间的真正并行处理, 实现大幅度减少设备空闲时间. 又由于采用回退策略, 优先调度对调度结果有重要影响的动态长路径上的工序, 进一步缩短产品总加工时间. 所以可回退抢占的设备综合调度算法可以实现对调度结果的优化.

6 结论

本文利用事件驱动的思想, 提出了解决综合调度问题的新方法, 并对产品实例进行了软件仿真实现. 仿真结果表明该方法在解决综合调度这一 NP 问题上, 比以往的 APCM 算法调度结果更优, 结论如下:

优点: 1) 以工序为调度单位, 并行处理单位小, 并行效果更好, 设备利用率更高, 提高加工效率; 2) 抢占式的回退调度策略, 优先调度对调度结果有重要影响的长路径工序; 3) 由于被抢占工序采取睡眠策略, 避免调度循环回退.

缺点: 算法复杂度比以往的 APCM 算法 (复杂度为二次多项式) 高 1 次.

在计算机飞速发展的今天, 通过牺牲少量计算机资源获取更好的计算结果已成为算法设计者的共识, 因此优于 ACPM、调度优先级和动态关键路径等方法的本文算法和所实现的软件, 对深入研究综合调度问题的解决方案提供了新的启示, 具有一定的理论和实际意义.

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 Zhang Hong-Yuan, Xi Yu-Geng, Gu Han-Yu. A decomposition and coordination scheduling method for flow-shop problem based on TOC. *Acta Automatica Sinica*, 2005, **31**(2): 182−187
(张宏远, 席裕庚, 谷寒雨. 基于约束理论的 Flow-shop 分解协调算法. 自动化学报, 2005, **31**(2): 182−187)

3 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)

4 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)

5 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)

6 Wei Ying-Zi, Zhao Ming-Yang. A reinforcement learning-based approach to dynamic job shop scheduling. *Acta Automatica Sinica*, 2005, **31**(5): 765−771
(魏英姿, 赵明扬. 一种基于强化学习的作业车间动态调度方法. 自动化学报, 2005, **31**(5): 765−771)

7 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)

8 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. 计算研究与发展, 2003, **40**(7): 977−983)

9 Xie Z Q, Ye G J, Zhang D L, Tan G Y. New nonstandard job shop scheduling algorithm. *Chinese Journal of Mechanical Engineering*, 2008, **21**(4): 97−100

10 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)

11 Xie Zhi-Qiang, Shao Xia, Yang Jing. Algorithm for integrated flexible scheduling with device-independence deferred constraint. *Journal of Mechanical Engineering*, 2011, **47**(4): 177−185
(谢志强, 邵侠, 杨静. 存在设备无关延迟约束的综合柔性调度算法. 机械工程学报, 2011, **47**(4): 177−185)

12 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)

13 Xie Zhi-Qiang, Teng Yu-Zheng, Yang Jing. Integrated scheduling algorithm with no-wait constraint operation group. *Acta Automatica Sinica*, 2011, **37**(3): 371−379
(谢志强, 滕宇峥, 杨静. 紧密衔接工序组联动的综合调度算法. 自动化学报, 2011, **37**(3): 371−379)

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

15 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)

16 Xie Zhi-Qiang, Yang Jing, Zhou Yong, Zhang Da-Li, Tan Guang-Yu. Dynamic critical paths multi-product manufacturing scheduling algorithm based on operation set. *Chinese Journal of Computers*, 2011, **34**(2): 406−412
(谢志强, 杨静, 周勇, 张大力, 谭光宇. 基于工序集的动态关键路径多产品制造调度算法. 计算机学报, 2011, **34**(2): 406−412)

17 Yang Zhi-Yi, Yang Gang, Zhang Hai-Hui. An approach for implementing service-oriented and event-driven information integration platform. *Journal of Computer Research and Development*, 2008, **45**(10): 1799−1806
(杨志义, 杨刚, 张海辉. 一种面向服务的事件驱动架构信息集成平台构造方法. 计算机研究与发展, 2008, **45**(10): 1799−1806)

18 Liu Jia-Hong, Wu Quan-Yuan. An event-driven service-oriented computing platform. *Chinese Journal of Computers*, 2008, **31**(4): 588−598
(刘家红, 吴泉源. 一个基于事件驱动的面向服务计算平台. 计算机学报, 2008, **31**(4): 588−598)

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** Postdoctoral 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.)

辛 宇 哈尔滨理工大学计算机科学与技术学院硕士研究生. 2008 年获得哈尔滨理工大学学士学位. 主要研究方向为企业智能计算.
E-mail: xin_yu_xy@yahoo.cn
(**XIN Yu** Master student at the College of Computer Science and Technology, Harbin University of Science and Technology. He received his bachelor degree from Harbin University of Science and Technology in 2008. His 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.)