DOI：10.3901/JME.2018.06.191

考虑串行工序紧密度的择时综合调度算法*

谢志强 张晓欢 高一龙 辛 宇
(哈尔滨理工大学计算机科学与技术学院 哈尔滨 150080)

摘要：针对目前综合调度算法在处理一般综合调度问题时为了兼顾加工工艺树中工序的并行处理，忽略串行工序之间紧密度，影响调度结果的问题，提出考虑串行工序紧密度的择时综合调度算法。该算法提出工序序列排序策略，从工艺树的整体结构出发,将其划分成内部工序只具有串行关系的工序序列，根据工序序列的长短确定其调度顺序；提出择时调度策略，结合工艺树自身特点，为调度工序选择若干合法加工时间点，分别在每个时间点调度工序，得到该工序的试调度方案集合，从其中选择加工总用时最小的方案作为工序调度方案，若不唯一，则选择工序加工时间最早的方案。实例表明，该算法既保证并行工序的并行处理，又有效提高串行工序的紧密度，优化了综合调度的结果。

关键词：工序序列排序；择时；串行工序；紧密度；综合调度算法
中图分类号：TP278

Time-selective Integrated Scheduling Algorithm Considering The Compactness of Serial Processes

XIE Zhiqiang ZHANG Xiaohuan GAO Yilong XIN Yu
(College of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150080)

Abstract: Aiming at the problem of ignoring the compactness of serial processes when being compatible with parallel processing of processes that leads to the imprecision result,an time-selective integrated scheduling algorithm considering the compactness of serial processes is presented.The strategy of process sequence sorting is proposed, and it divide the processes into several process sequences based on the integral structure of the processing tree, and the processes in these sequences only has serial relation.Determine the scheduling order according to the length of the process sequence; the time-selective strategy is proposed, and some legal processing time points are selected for the scheduling process according to the characteristics of the process tree. trial scheduling process at each time point respectively, a set of trial scheduling scheme for the process is obtained, The scheme which's total scheduling time is least in the set is chosen as the process scheduling scheme. If not unique, the scheme in which the start time of the process is earliest should be selected. Example shows that the proposed algorithm can raise the compactness of serial processes when being compatible with parallel processing of processes, and make the scheduling result more accurate.

Key words: process sequence sorting; time-selective; serial process; compactness; integrated scheduling algorithm

0 前言

调度是影响制造业生产效率的关键因素，有效的调度优化算法能在满足企业订单、设备及其它软硬件资源限制的前提下，使生产效率最大化。目前，学者们针对车间调度问题做了大量的研究[1-6]，这些算法主要针对先加工工件，后将各工件组装成产品的情况进行调度优化。随着社会对产品的需求不断向个性化、多样化转变，生产制造行业将面临越来越多的多品种、小批量产品定单，这种局势下如果生产仍然按照传统的先加工后装配的生产方式进行生产，势必割裂产品内在的加工与装配的并行关系，降低生产效率。为了寻求这一新研究领域的解决方案，谢志强等[7]提出了将产品的加工和装配同时推进的综合调度，研究产生一系列调度优化算法，并拓展出很多新的研究领域。目前，在一般综合调度研究领域中，主要从以下两个方面进行了研究。

(1) 基于工序的研究。文献[7]首先指出关键路径在加工工艺树中的重要地位，强调加工工艺树中

* 国家自然科学基金(61370086, 61602133，61772160)、黑龙江省教育厅科技(12531105)、黑龙江省博士后科研启动(LBH-Q13092)、中国博士后(2016M591541)、黑龙江省博士后(LBH-Z15096)和全国高等学校计算机教育研究会(研究生创新)(ER2014018)资助项目。20160903 收到初稿，20170506 收到修改稿

具有纵向关系工序的调度与最终调度结果有着十分密切的联系；文献[8]提出层优先、短用时、长路径和动态调整策略，指出增加工艺树中具有横向关系工序之间的并行性能使调度结果更优；文献[9]指出文献[7]注意产品树状结构的纵向，忽略横向其他路径上不同设备工序可并行处理，同时指出文献[8]重点考虑产品树状结构的横向，在横向层的基础上考虑纵向路径，但其重横轻纵的策略不符合树状结构产品调度以纵向为主的机理，提出以纵为主兼顾横向的调度方案。算法的优点是：在纵横兼顾的基础上，进一步优化了纵向调度，符合综合调度以纵向为主的思想；缺点是：其采用动态关键路径思想虽然较好解决了串行工序与并行工序同时推进的问题，但该算法思想过于宏观，没有考虑调度过程中由于工序加工设备等因素的制约，造成串行工序之间紧密度差的细节问题。

(2) 基于设备的研究。文献[10]针对文献[7]中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，提出基于设备空闲事件驱动的综合调度算法；文献[11]在文献[10]的基础上提出采用回退策略优先调度父节点路径长的工序来进一步优化调度结果，优点是：增加设备利用率，最大化了“设备忙”原则，减少设备空闲时间，使工序加工更加紧凑；缺点是：“设备驱动事件”寻找工序时，总是在当前可调度的并行工序当中寻找，从工序的角度分析该算法可将其视为“叶对齐”方式下的改进“层优先”调度，增加并行性的同时，势必拉大串行工序之间的加工等待时间，忽略纵向调度优化对调度结果的影响。

综上所述，目前的研究能够对产品中工序的并行调度做出较好的优化，但在兼顾并行性的同时，对串行工序的调度优化还有待提高。提出考虑串行工序紧密度的择时综合调度算法。调度工序时，运用择时策略在若干符合工艺约束关系的时间点试调度工序，得到若干个准工序调度方案，在其中选择该工序的“最佳调度方案”(即当前所有已调度工序加工总用时最小的方案)，当“最佳调度方案”不唯一时，则选择工序加工开始时间最早的方案。

该算法既继承了现在算法保证工序间并行性的优点，又在其基础上优化了串行工序间的紧密度，进一步强调了纵向优化为主的调度思想，避免了之前算法中顾此失彼的弊端，优化了调度结果。

1 问题描述分析

综合调度问题就是研究当产品处于边加工边装配的生产模式下如何调度其中工序能够使得产品完成时间最小。其中，我们把每道工序的加工和装配视为一个整体，统称为加工。采用产品加工工艺树来明确指出产品中各个工序的加工时间，加工设备以及其加工的偏序关系。综合调度时须满足以下要求。

(1) 每个工序只能在一台机器上进行加工。

(2) 每一时刻某一台机器只能加工一个工序。

(3) 当且仅当一个工序的所有前序工序都处于加工完毕状态(或无前序工序)时，该工序才能够被加工。

(4) 产品某一工序的加工不能被间断。

(5) 产品的最晚结束加工工序的加工结束时间与最早开始加工工序的加工开始时间之差为产品的加工总用时。

定义 1：准调度时间点。某一加工工序在其加工设备上的其紧前工序加工结束时间点以及其后所有已调度工序的加工结束时间点都是该工序的准调度时间点。

定义 2：路径长度。在当前的加工工艺树中，叶子节点及其当前所有前序工序的加工时间之和。

定义 3：工序序列。彼此之间具有串行关系的工序集合，且每道工序在该集合中最多具有一个紧前工序和一个紧后工序。

定义 4：工序队列。其中存放了按工序序列排序策略排序后的工序。

定义 5：调度方案。其中记录各个设备上目前已调度的工序信息。

定义 6：初始调度方案。只调度工艺树中路径最长的工序序列上工序所形成的调度方案。

定义 7：工序调度方案。某一工序调度完成时刻所产生的调度方案。

定义 8：产品调度方案。产品中的所有工序调度完成所产生的调度方案。

定义 9：基础调度方案。调度工序 i 时，其工序队列中的紧前工序 i–1 的调度方案 P_{i-1} 就是工序 i 的基础调度方案。

设产品由 Num 道工序组成且公式 $Num=n+N$ 成立，其中 n 为最长工序序列上的工序数，N 为非最长工序序列上的工序总数且在工序队列 Q_u 中的序号为 $i(1\leqslant i\leqslant N)$，在 M 台设备上加工，以初始调度方案为起点，之后每调度一个工序经过择时策略形成一个工序调度方案，调度过程中共形成 N+1 个调度方案，第 N+1 个方案的加工总用时即为产品的加工总用时。设初始调度方案为 P_0，工序队列中工序 i 的工序调度方案为 P_i，工序 N 的工序调度方案为 P_N，P_N 即为产品调度方案，则目标函数为

$$T=\min(P_N.total\_time)$$

s.t.

$$w.work\_starttime \geqslant w.prio.work\_finishtime \quad (1)$$

$$P_i.total\_time=\max(machine[j].maxfinishtime)$$

$$1 \leqslant j \leqslant M \quad 0 \leqslant i \leqslant N \quad (2)$$

$$P_i.total\_time \leqslant P_{i+1}.total\_time \quad 0 \leqslant i \leqslant N-1 \quad (3)$$

其中，式(1)表示某一工序的加工开始时间一定不小于其工艺树中紧前工序的加工结束时间。式(2)表示某一工序调度方案的加工总用时为此刻所有设备加工完成时间的最大值；式(3)表示某一工序调度方案的加工总用时一定不大于其在工序队列中的紧后工序调度方案的加工总用时。

2　算法设计思想

目前存在一般综合调度算法往往根据预先定义的规则对工序进行排序，默认工序的加工开始时间为当前状态下可行的最早加工开始时间，算法的执行过程就是工序的排序过程，工序的排序结果即为产品调度方案。这种从工艺树的一端进行调度的算法往往只考虑路径长短、加工用时长短和是否处于可调度状态等因素，没有考虑工序在调度过程中相互制约，先调度工序对后序工序造成的影响。与之不同，所提调度算法则是先确定工序的加工顺序，然后再依次为其选择加工开始时间，目的是为工序寻找最契合工艺树特点的加工时间点，使调度工序能与已调度工序相互配合，兼顾并行性的同时，保证串行工序紧密度，实现产品加工总用时最小的目标。首先，从工艺树整体结构出发，将其划分成若干工序序列，按工序序列由长到短的顺序确定各工序的调度次序；调度工序时，将工序在符合工艺约束关系的若干“准调度时间点”上进行试调度，形成若干“准工序调度方案”，并在其中找到当前加工总用时最小的“准工序调度方案”确定为该工序调度方案。依此类推，直到所有工序调度完毕，得到产品调度方案。

具体实现方法和过程描述如下：创建工序队列和工序调度方案链表。应用工序序列排序策略将工艺树中的所有工序依次入工序队列；将最长工序序列上工序出队，并调度其形成初始调度方案；将初始调度方案作为方案链表中的第一个节点；调度工序时，出队一个工序，以当前链表中最后一个调度方案作为基础调度方案，应用择时策略生成该工序的调度方案；将这个调度方案做为新的链表节点追加到调度方案链表中。依次循环操作直至工序队列为空。图 1 为产品调度过程示意图。

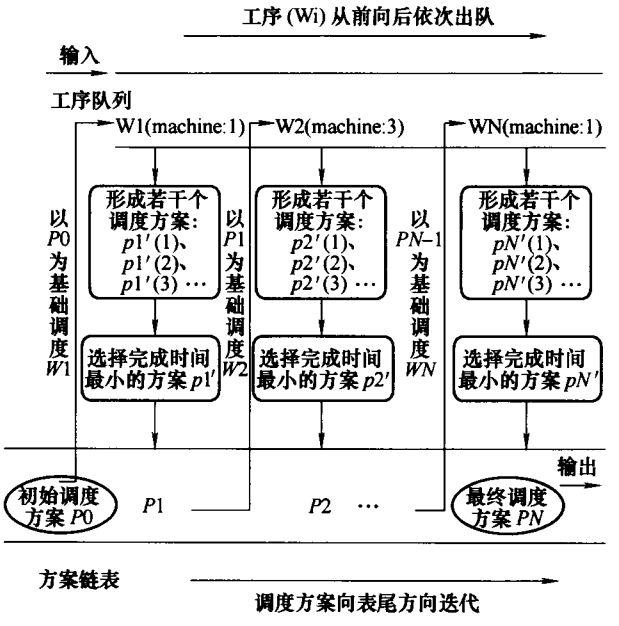


图 1　产品调度过程示意图

3　调度策略分析与设计

下面给出算法中所用到的类和数据结构的描述及说明。

(1) 算法中所涉及到的类描述如下。

工序类 W 的描述：

```
class W{
private:
int work_id;//工序号
int work_time;//该工序加工所需时间
int work_starttime;//该工序的加工开始时间
int work_finishtime;//该工序的加工结束时间
int machine_ id;//工序的加工设备
bool is_workinlongpath;//是否是最长工序序列上工序
bool is_scheduled;//是否已经被调度
W *tree_next;  //逆序加工工艺树中紧后工序链表
W   tree_prio;//逆序加工工艺树中紧前工序
W *next;//工序 w 在当前队列中紧后工序的工序号
public:
... ...
};
```

设备类 Machine 的描述：

```
class Machine{
private:
int id; //加工设备号
W *work;//设备上已经调度的加工工序的链表，以工序加工开始时间排序
int maxfinishtime;//该设备当前最大加工完成时间
```

```
public:
... ...
};
```

方案类 Scheme 的描述：

```
class Scheme{
private:
int id;//调度轮次
int total_time;//方案中工序加工总用时
Machine   *mach;//调度设备
public:
... ...
};
```

其中，工序类、设备类和方案类彼此之间存在着聚合关系，如图 2 所示。

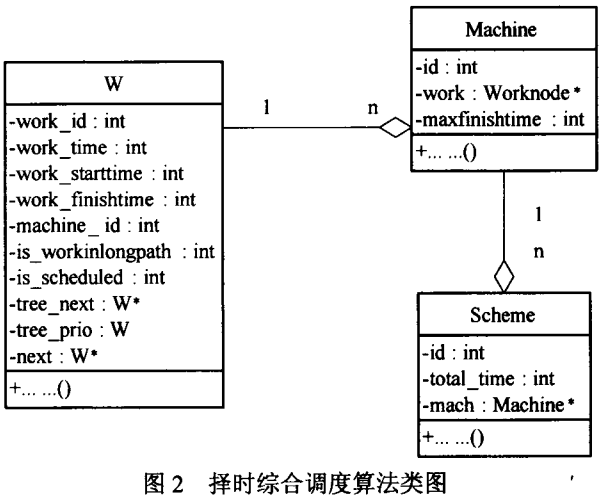


图 2 择时综合调度算法类图

(2) 算法中所涉及的数据结构描述如下。

1) S 栈逆置工序序列。

2) Q_u 队列存放排序后工序。

3) QT_i 队列存放第 i 个工序的所有“准调度时间点”。

4) adjust_queue 队列存放因某工序的调度引起冲突而被调整的工序。

5) P 链表存放每个工序的调度方案。

下面对工序序列排序策略和择时调度策略分别作出介绍。

3.1 工序序列排序策略

3.1.1 工序序列排序策略分析

文献[7]中提出将加工工艺树及其中各个子树中的工序分成具有唯一紧前、紧后相关工序的工序和独立工序两类，目的是一方面使得具有唯一紧前、紧后相关工序之间紧密度提高；另一方面灵活调度对产品调度结果影响较小的独立工序，从而提高工序间的并行性，优化产品调度结果。但由于拟关键路径算法按子树对工艺树中工序进行调度，这一思想应用有一定的局限性，使得最终产品调度结果并不十分理想。工序序列排序策略是对文献[7]中工序分类思想的继承和改进，优点如下。

(1) 将产品加工工艺树划分成若干工序序列，而不是若干棵子树，且调度时以工序序列为单位，有效的提高了工艺树中串行工序的紧密度。

(2) 将工序按其所在工序序列的路径长度从大到小的顺序进行排序，并按此顺序调度工序，保证优先调度对产品调度结果影响较大的工序。

3.1.2 工序序列排序策略算法描述

工序序列排序策略的算法步骤如下所示。

(1) 判断加工工艺树是否为空，是则转至步骤(7)，否则转至步骤(2)。

(2) 为所有现存叶子节点计算路径长度。

(3) 找到路径长度最长的叶子节点，如果不唯一，转至步骤(4)，否则转至步骤(5)。

(4) 选择路径上工序最多的叶子节点。

(5) 从该叶子节点开始，将其和其当前工艺树中的全部前序工序依次入栈 S 并在工艺树中删除。

(6) 将 S 栈中的节点依次弹栈，入队 Q_u，转至步骤(1)。

(7) 退出，返回队列 Q_u。

工序序列排序策略算法流程图如图 3 所示。

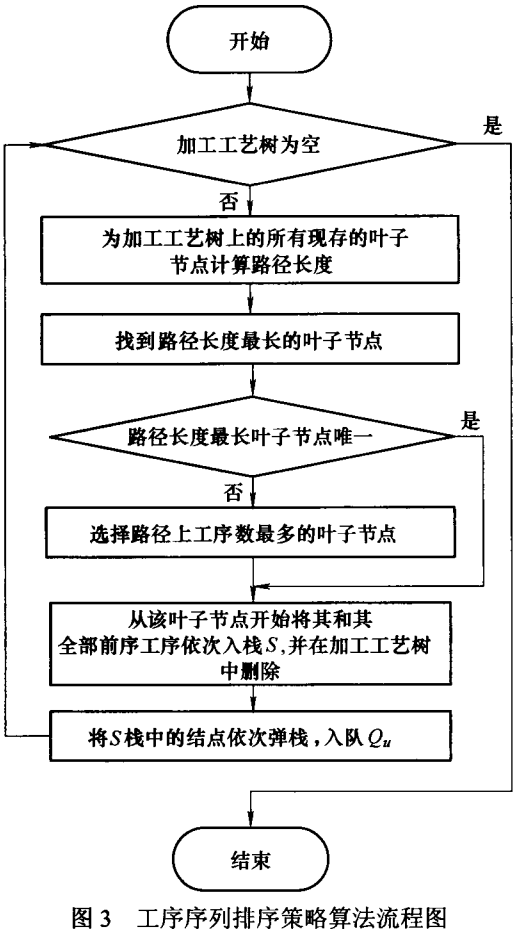


图 3 工序序列排序策略算法流程图

工序序列排序策略的序列划分示意图如图 4 所示。其操作步骤为：第 1 步，为工艺树中所有叶子节点 W10，W9，W5，W8 和 W11 计算路径长度，得到 W5 所在路径长度最长为 21，选择其所在路径上所有工序 W1、W2、W3、W4 和 W5 组成第 1 个工序序列，并将其按照从根节点到叶节点的顺序依次加入到工序队列，同时将其在工艺树中删除；第 2 步，为工艺树中其余叶子节点 W10，W9，W8 和 W11 计算当前(删除第 1 工序序列节点后)所在路径长度，得到 W8 所在路径长度最长为 20，选择其所在路径上所有工序 W6、W7 和 W8 组成第 2 个工序序列，并将其按照从根节点到叶节点的顺序依次加入到工序队列，同时将其在工艺树中删除；同理得到第 3 工序序列 W10，第 4 工序序列 W11 和第 5 工序序列 W9，且依次加入到工序队列当中，工序队列工序顺序为：W1、W2、W3、W4、W5、W6、W7、W8、W10、W11 和 W9，该顺序为工序的调度次序。接下来应用择时调度策略依次调度工序队列中工序。

3.2　择时调度策略

3.2.1　择时调度策略分析

3.2.1.1　逆序调度顺序的确定

择时策略的目的是在工序 i 的基础调度方案 P_{i-1} 上找到加工总用时 P_i。totaltime 最小的调度方案 P_i。首先需要确定工序 i 的最早加工开始时间 t，且以 t 为起点，寻找工序 i 的各个“准调度时间点”。如果采用正序调度，在调度工艺树中某工序序列中的叶节点工序时，往往较难确定其在加工设备上的第 1 个“准调度时间点”。

如图 4 所示产品，若按正序调度，调度工序

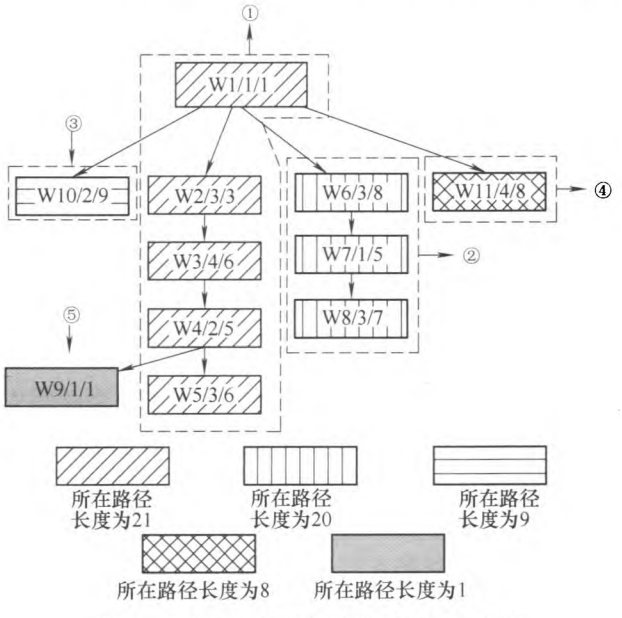


图 4　工序序列排序策略序列划分示意图

W8 时将在如图 5 所示的基础调度方案上确定其“准调度时间点”。工序 W8 与基础方案中 W5、W4、W3 和 W2 均为并行关系，与工序 W1 为串行关系。其“准调度时间点”的确定范围为(−∞,20]，其中“20”为工序 W2 的加工结束时间，在这一区间的下限为“−∞”，很难确定工序 W8 的第 1 个(即加工开始时间最早的)“准调度时间点”。

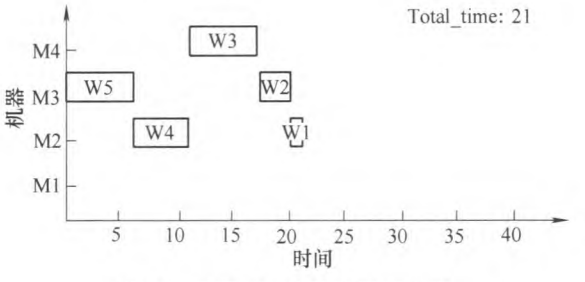


图 5　工序 W8 的基础调度方案

如果将工艺树中所有偏序关系逆置，工艺树中除根节点外，每一工序均有紧前工序(根节点由于一定属于最长工序序列，在形成初始调度方案时已经被调度，这里不需要考虑)，所以工序第 1 个“准调度时间点”(即最早加工开始时间)很容易被确定，即为其工艺树中紧前工序加工结束时间。

图 6 为将图 4 产品逆置后所得初始调度方案，按照逆序调度思想，以其为基础调度方案，下一调度工序为 W6，W6 与其中的工序 W1 为串行关系，与工序 W2，W3，W4 和 W5 均为并行关系。W6 的“准调度时间点”确定范围为[1,+∞)，其中“1”为工序 A1 的加工结束时间，这一区间的下限为工序 W6 的第 1 个“准调度时间点”，即为“1”。综上所述，将加工工艺树逆置更有利于算法的实现。

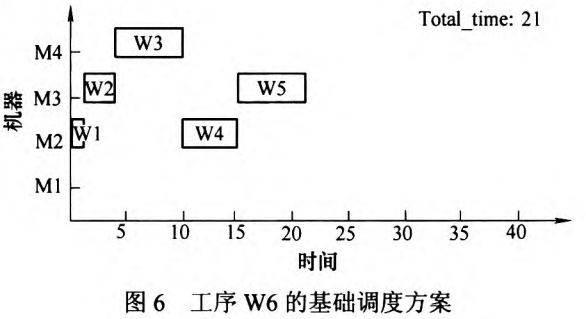


图 6　工序 W6 的基础调度方案

3.2.1.2　初始调度方案的确定

首先调度最长工序序列上的工序形成初始调度方案，并在此基础上通过择时策略依次调度工艺树上的其他工序，理由如下。

(1) 由于采用逆序调度，该初始调度方案为工序队列中其它各个工序序列中的首个调度工序提供

了最早加工开始时间(即第 1 个“准调度时间点”)。

(2) 形成初始调度方案时不需要对其中工序应用择时策略。多数情况下，最长工序序列上的工序数量也较其它工序序列多，选择调度其上工序形成初始调度方案节省了算法的时间开销。

以下是形成初始调度方案的算法步骤。

(1) 设 i=1，对工序队列 Q_u 作出队操作，取得当前调度工序 W_i，其加工时间为 t_i，加工设备为 M_i，判断工序 W_i 是否是最长工序序列上的工序，是则转至步骤(2)，否则转至步骤(3)。

(2) 该调度工序的紧前工序加工结束时间 W_{i-1}.work_finishtime 作为其加工开始时间 W_i.work_starttime，安排到设备 M_i 上，刷新工序 W_i 信息和设备 M_i 信息，i++，转至步骤(1)。

(3) 目前已形成初始调度方案 P_0，采用择时策略调度该工序。

形成初始调度方案算法的流程图如图 7 所示。

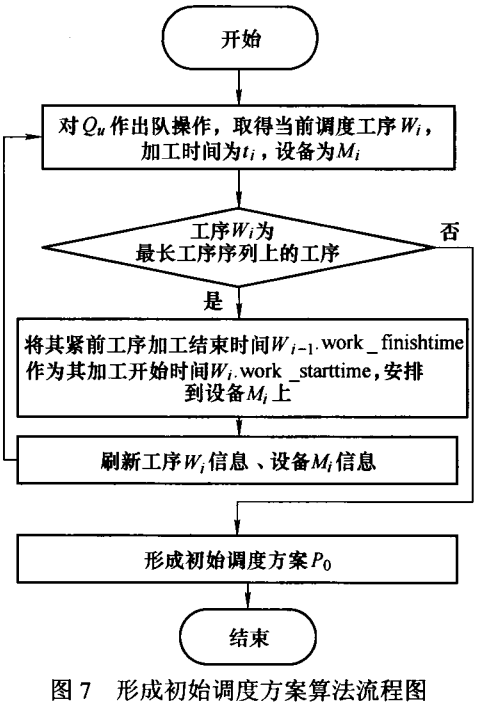


图 7 形成初始调度方案算法流程图

3.2.1.3 确定“准调度时间点”

对工序 w 进行调度时，首先要在其加工设备上确定若干个“准调度时间点”，方法是以工序 w 的紧前工序加工结束时间作为第 1 个“准调度时间点”，并以此时间点为起点在其加工设备上寻找每个工序的加工结束时间点分别作为工序 w 的“准调度时间点”，且依次放入 QT_i 队列中。

这里需要说明的是，第 1 个“准调度时间点”是由工序 w 在工艺树中紧前工序加工结束时间来决定的，在此时间点调度工序 w 往往会影响已调度工序。如图 8 和图 9 所示，设工序 A2 与 A3 在同一路径，且 A3 是 A2 的紧后工序，在调度 A3 时，以 A2 的加工结束时间 $T1$ 作为第 1 个“准调度时间点”。存在以下几个时刻/时间段。

$T0$：A1 工序的加工开始时刻。

$T1$：工序 A2 的加工结束时刻、工序 A3 的“准调度时间点”。

$T2$：工序 A1 的加工结束时刻。

$T3$：工序 A3 以 $T1$ 作为加工开始时刻时的加工结束时刻。

Δt：工序 A1 与工序 A3 发生冲突的时间段。

具体情况如图 6a 所示。分析如下。

当Δt>0 时，一定至少满足以下两种情况之一。

$$T0 \leqslant A3.work\_starttime < T2 \tag{4}$$

$$T0 < A3.work\_finishtime \leqslant T2 \tag{5}$$

式(4)如图 8a 图所示，式(5)如图 9a 图所示。确认上述情况是否成立，需要对 A3 在 $T1$ 时刻进行试调度。若以上两种情况都不满足，则不需要调整；若满足其中之一，为了避免冲突有两种解决方案：其一为将 A3 的加工开始时间延后至 $T2$，其二为将 A3 在 $T1$ 时刻加工，将 A1 的加工开始时间延后至 $T3$ 时刻。由于择时策略后续将选择该设备上 A3 开始加工时刻后每一工序的加工结束时刻作为工序 A3 的“准调度时间点”，且 $T2$>$T1$，$T2$ 点仍然会被选作“准调度时间点”。所以方案二比方案一考虑的更全面，使问题的解空间更大，采用方案二作为解决工序加工冲突的方法，具体如图 8b、9b 所示。

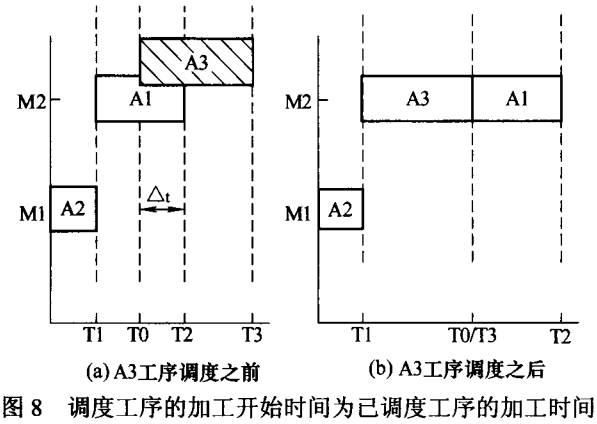


图 8 调度工序的加工开始时间为已调度工序的加工时间

具体处理方法如下：以 A2 的加工结束时间作为 A3 的加工开始时间，将 A3 插入到其设备的已调度工序链表 *work*(链表中的节点工序以其加工开始时间按升序排序)当中，位置在 A1 工序之前。此时不需对 A1 工序加工开始时间进行更改，至于由此产生的工序调度冲突问题由后面的择时调整策略一并解决。

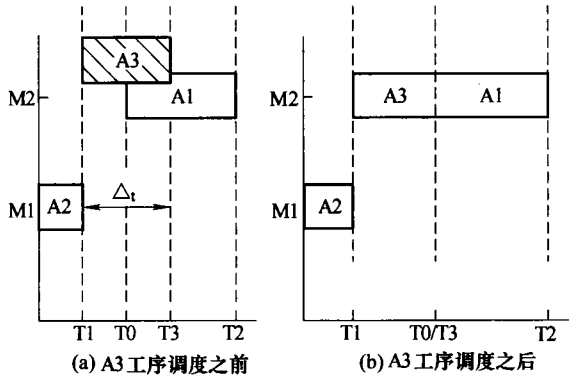


图 9　调度工序的加工结束时间为已调度工序的加工时间

3.2.2　择时调整策略

选择“准调度时间点”时只考虑工序之间的加工顺序，而没有考虑工序的调度是否会对该设备上已调度工序造成影响。所以，在调度工序后，需要对其可能影响的已调度工序进行检查，当调度工序的加工完成时间大于同设备上后面工序的加工开始时间或其工艺树中紧后工序的加工开始时间时，需启动择时调整策略。

择时调整策略示意图如图 10 所示。

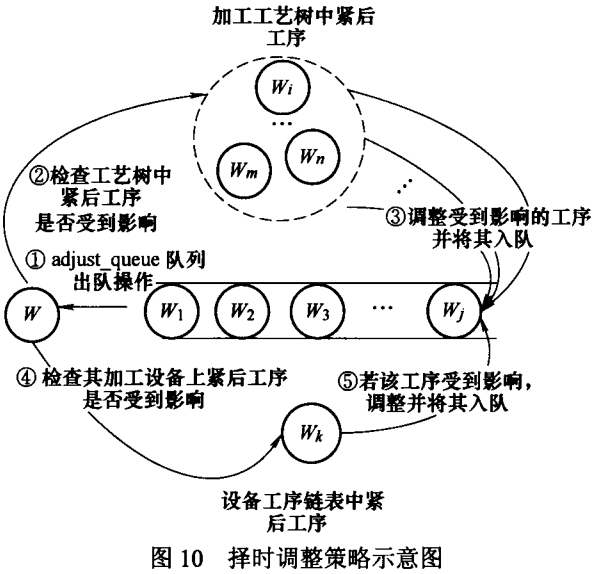


图 10　择时调整策略示意图

下面给出择时调整策略算法的步骤。

(1) 将当前调度工序入队 adjust_queue。

(2) 对 adjust_queue 做出队操作，并将结果存入 W 中。

(3) 判断 W 是否为空，不为空转至步骤(4)，否则转至步骤(10)。

(4) 设 W 在其工艺树中存在 k 个紧后工序，n=1。

(5) 判断 $n>k$ 是否成立，如果成立转至步骤(8)，不成立转至步骤(6)。

(6) 判断 W 的加工完成时间是否大于其工艺树中紧后工序 W_n 的加工开始时间(未被调度的工序其开始时间默认为+∞)，如果是转至步骤(7)，否则转至步骤(8)。

(7) 将 w.work_finishtime 作为 wn.work_ starttime，将 w_n 入队 adjust_queue，n++。

(8) 判断 w 在其加工设备的已调度工序链表上是否存在紧后工序 w_m 并且 w 的加工完成时间大于 w_m 的加工开始时间，如果是转至步骤(9)，否则转至步骤(2)。

(9) 将 w.work_finishtime 作为 wm.work_ starttime，将 wm 入队 adjust_queue，转至步骤(2)。

(10) 调整完毕，计算当前方案的加工总用时，退出。

择时调整策略算法流程图如图 11 所示。

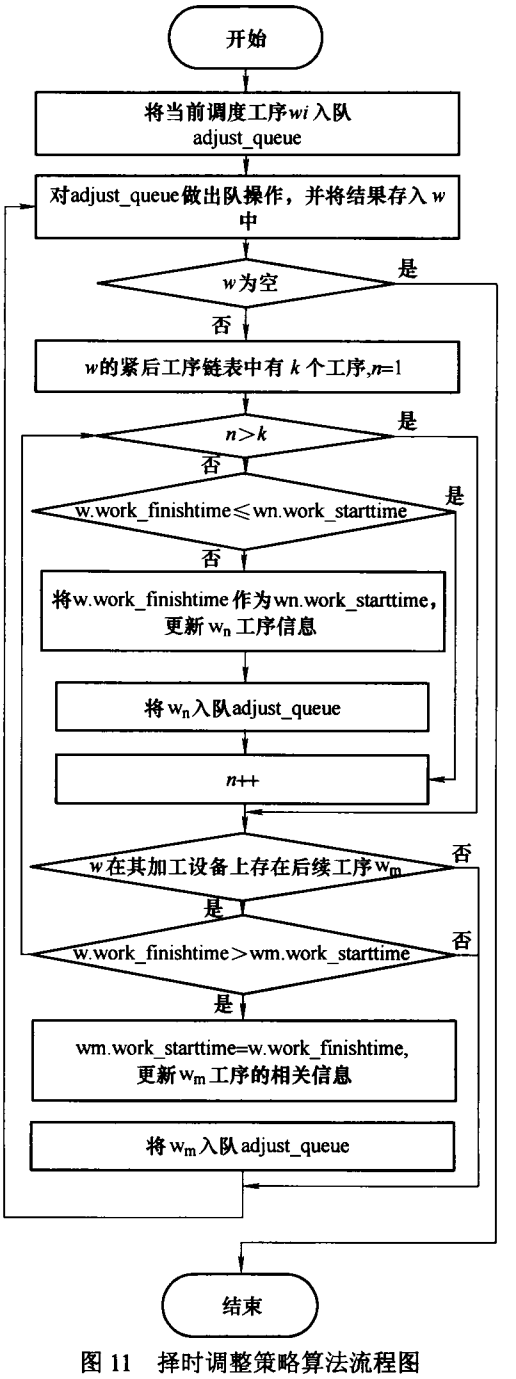


图 11　择时调整策略算法流程图

3.2.3 择时调度策略的算法描述

以下为择时调度策略算法的具体步骤。

(1) 找到工序 W_i 的基础调度方案 P_{i-1}。

(2) 在方案 P_{i-1} 上寻找工序 W_i 的加工设备 M_i 上的 k 个“准调度时间点”，按从前向后的顺序依次入队 QT_i，j=1。

(3) 判断 QT_i 队列是否为空，不为空则对 QT_i 队列做出队操作取出“准调度时间点”T，转至步骤(4)，为空则转至步骤(8)。

(4) 判断 T 是否是方案 P_{i-1} 中加工设备 M_i 上的空闲时间或某个工序的加工结束时间，如果否转至步骤(5)，如果是转至步骤(6)。

(5) 将时间点 T 所涉及的工序(“T”时间点正在加工的工序)向后移动到时间点 $T+t_i$ 开始加工。

(6) 以时间点 T 作为工序 W_i 的起始加工时间对工序 W_i 进行调度。

(7) 启用择时调整策略对由于调度工序 W_i 而影响到的工序进行调整(即对调度方案进行更新)，形成准工序 W_i 调度方案 P_{ij}，k=j++，转至步骤(3)。

(8) 对 k 种方案(P_{ij})的加工总用时进行比较，选择加工总用时最小的方案。

(9) 判断方案是否唯一，唯一则选择之，不唯一则选择工序 W_i 加工开始时间最早的方案 P_{io} (o=1,2,⋯,k)。

(10) 刷新工序 W_i 信息、设备 M_i 信息和调度方案 P_i 信息，退出。

择时调度策略算法流程图如图 12 所示。

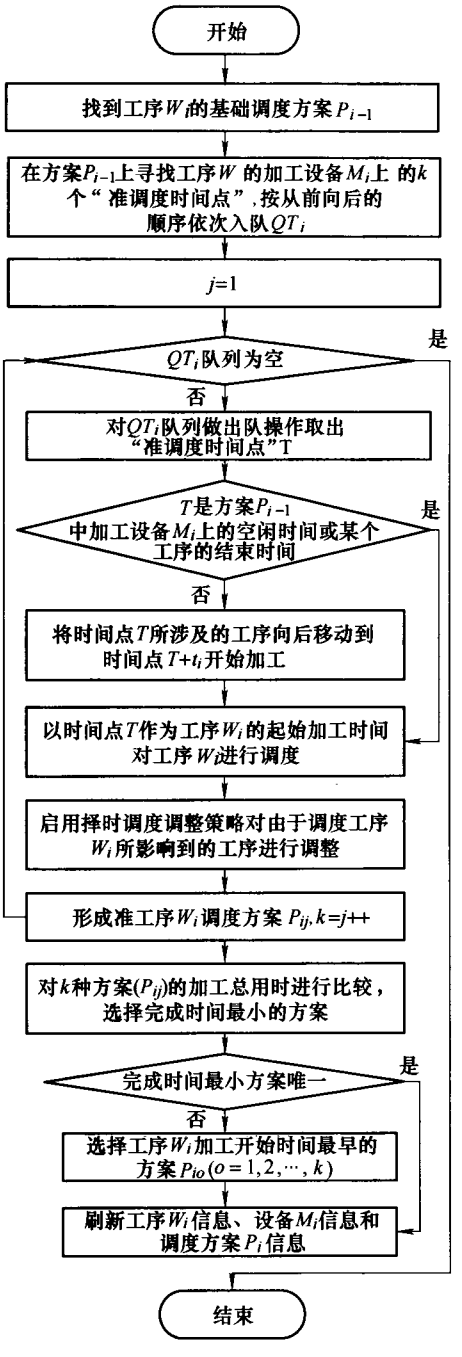


图 12 择时调度策略算法流程图

4 算法设计及复杂度分析

4.1 算法设计

考虑串行工序紧密度的择时调度算法实现步骤如下所示。

(1) 将加工工艺树中加工工序的偏序关系取反，得到逆序加工工艺树。

(2) 应用工序序列排序策略得到工序队列 Q_u。

(3) 将 Q_u 上的最长工序序列中所有工序依次出队并调度形成初始调度方案 P_0。

(4) i=1。

(5) 判断 Q_u 是否为空，为空则转至步骤(8)，否则转至步骤(6)。

(6) Q_u 出队列，取得当前调度工序 W_i，其加工时间为 t_i，加工设备为 M_i。

(7) 应用择时调度策略调度 W_i，得到方案 P_i，i++，转至步骤(5)。

(8) 形成产品调度甘特图并输出。

考虑串行工序紧密度的择时调度算法流程图如图 13 所示。

4.2 算法复杂度分析

设产品工序总数为 Num，最长工序序列上的工序数为 n，其他工序数为 N，等式 Num=n+N 成立，产品加工设备数为 M。

4.2.1 将加工工艺树中的加工偏序关系逆置

将每一道工序的紧前(紧后)工序更改为其紧后(紧前)工序，得到逆序加工工艺树。由于产品的工序总数为 Num，更改每一道工序的紧前/紧后工序的复杂度为 $O(Num)$。

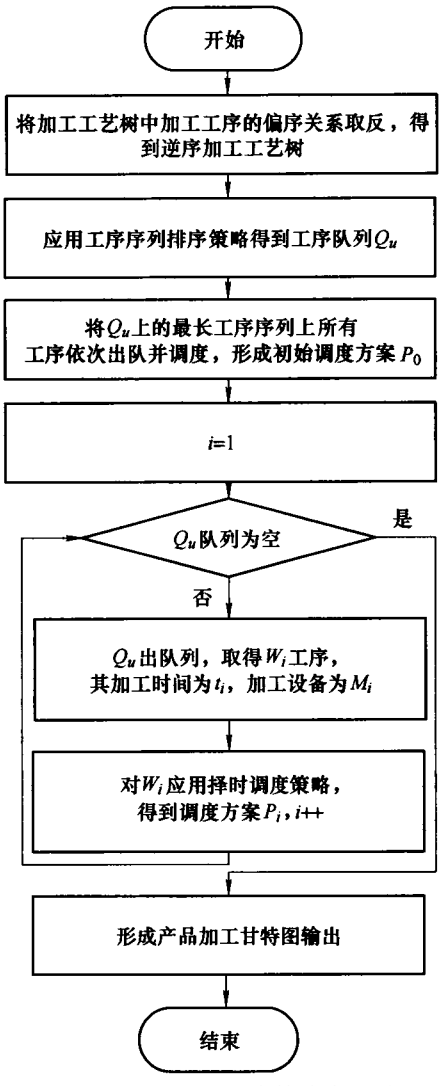


图 13　择时综合调度算法流程图

4.2.2　工序序列排序策略

工序序列排序是以工序序列为基本单位的，工序序列的个数就是产品加工工艺树中叶节点的个数，并按工序序列的路径长短对其进行排序，根据工序序列排序策略的规则，每一轮所选择出最长工序序列后，需将其中工序从工艺树中删除掉，这势必影响剩余叶节点所在工序序列的长度。所以本算法需进行若干轮筛选，每轮通过计算路径长度和比较路径长度，选择出最长工序序列。

分析如下，设叶节点的个数为 ln，那么第一轮筛选时计算路径长度的次数为 ln 次，并且每次计算路径长度时所做的加法运算次数为 $k(0\leqslant k\leqslant Num-ln)$ 次，其中 ln 最小为 1，即 k 最大可取 $Num-1$，第一轮计算的次数为 ln 次，比较的次数为 $ln-1$ 次；第二轮计算的次数为 $ln-1$ 次，比较的次数为 $ln-2$ 次，依此类推，最后一轮计算的次数为 2 次，比较的次数为 1 次。共计进行 $ln-1$ 轮计算和比较。由于每轮计算的次数要多于比较的次数，这里依据计算的次数计算复杂度：共进行 $ln-1$ 轮筛选，每轮筛选最多进行 ln 次计算，每次计算最多进行 $Num-1$ 次累加。所以所做运算次数最多为$(ln-1)\times ln\times(Num-1)$次，即工序序列排序策略的复杂度为 $O(Num\times ln^2)$，其中 $ln<<Num$。

4.2.3　形成初始调度方案

初始调度方案由最长工序序列上的 n 个工序调度形成，在调度过程中不需要使用择时策略，所以此项操作的复杂度为 $O(n)$。

4.2.4　择时策略

由于总工序数为 Num，加工设备数为 M，所以一台设备上最多可能加工 $Num-M+1$ 个工序，且择时次数最多的工序为工序队列中排在最后的工序，该工序加工之前，最坏的情况是，其加工设备上最多已有 $Num-M$ 道工序已经调度,此刻其调度的方案共有 $Num-M+1$ 种，且这些工序调度后都需要启动择时调整策略，且调整涉及工序队列中排在该工序之前的所有工序，此时需进行 $Num-1$ 次处理。所以择时策略最多进行$(Num-M+1)\times(Num-1)$次处理，其复杂度为 $O(Num^2)$。由于上述分析是针对某一个工序的，实际调度过程中需要处理的工序为 N 个，即得算法的时间复杂度为 $O(N\times Num^2)$。

上述各部分操作为串行操作，本文算法的时间复杂度取上述各项操作的最大值为 $O(N\times Num^2)$。

5　实例分析

由于以上算法的提出是理论分析的结果，并没有结合具体实例，具有普遍意义。为了方便读者了解该算法，下面借助产品调度实例进一步说明。某加工制造企业单件订单产品 A，由 11 个工序组成，在 4 台设备上加工，其产品加工工艺树如图 14 所示，下面将所提算法分别与目前该研究领域的所有同类算法做出对比。

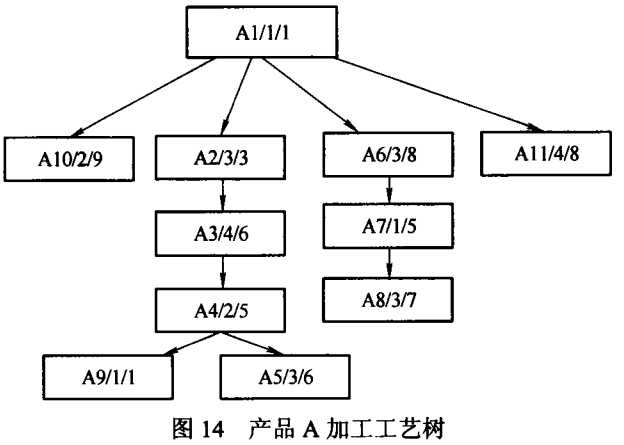


图 14　产品 A 加工工艺树

使用文献[7-11]中算法调度产品 A，得到调度结果如图 15～19 所示。上述算法中属于基于工序调度

的算法为文献[7-9]所提出的算法，其中调度结果最好的是文献[9]所提出动态关键路径算法。该算法根据每个工序在加工工艺树中的路径长度确定其调度的次序，而没有考虑到加工设备冲突导致工序间紧密度差的细节问题。例如，该算法中工序的调度顺序为：A5、A8，A11、A9、A4、A7、A3、A10，A6、A2 和 A1。其中，由调度工序 A11 占用设备 M4 导致 A3 延时加工，调度工序 A6 占用设备 M3 导致 A2 延时加工，最终导致调度结果较差；属于设备驱动调度的算法为文献[10]和文献[11]所提出的算法。其中，文献[11]中所提算法较文献[10]所提算法考虑全面，在调度产品 A 时由于调度工序 A11 占用设备 M4 导致 A3 延时加工，调度工序 A6 占用设备 M3 导致 A2 延时加工，最终导致调度结果较差，导致了最终调度结果不理想。

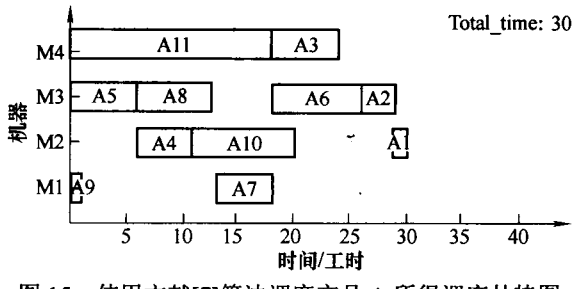


图 15 使用文献[7]算法调度产品 A 所得调度甘特图

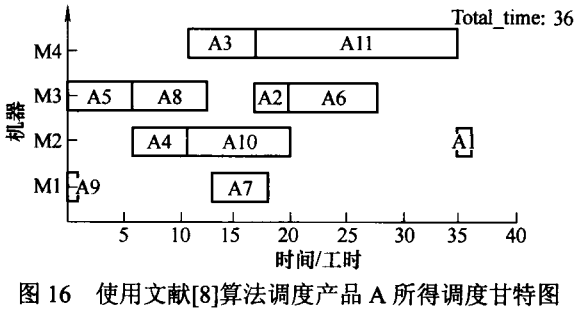


图 16 使用文献[8]算法调度产品 A 所得调度甘特图

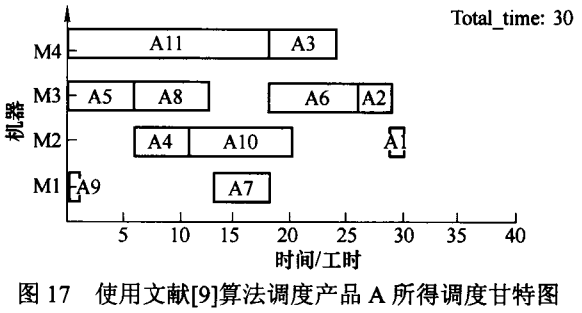


图 17 使用文献[9]算法调度产品 A 所得调度甘特图

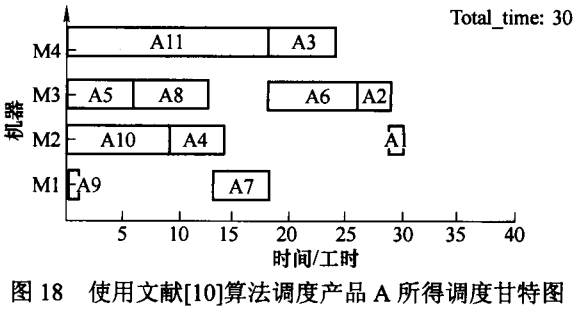


图 18 使用文献[10]算法调度产品 A 所得调度甘特图

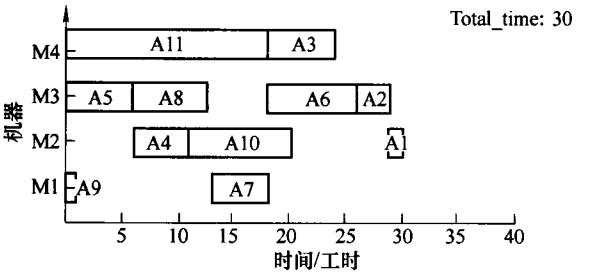


图 19 使用文献[11]算法调度产品 A 所得调度甘特图

下面使用所提算法调度产品 A，步骤如下。

(1) 应用工序序列排序策略，对产品 A 中各工序进行排序：根据如图 14 所示产品加工工艺树，首先计算其所有叶子节点工序的路径长度，结果分别为：A10：10，A9：16，A5：21，A8：20，A11：9。所以选择 A5 所在路径上的所有节点做入栈 S、出栈 S 和入队 Q_u 的操作，操作结果为：队列 Q_u 中工序顺序为 A1→A2→A3→A4→A5，并在产品 A 加工工艺树中将这些工序删除掉。这时，产品 A 加工工艺树变成了由多个子树组成的森林，接下来依次计算这些子树中叶子节点的路径长度，结果分别为：A10：9，A9：1，A8：20，A11：8。选择 A8 所在路径上的所有节点工序做入栈 S、出栈 S 和入队 Q_u 的操作，操作结果为：队列 Q_u 中工序顺序为 A1→A2→A3→A4→A5→A6→A7→A8，并在产品 A 加工工艺树中将这些工序删除掉。依次类推，产品 A 加工工艺树所对应的工序队列 Q_u 中工序顺序为 A1→A2→A3→A4→A5→A6→A7→A8→A10→A11→A9，且该顺序将作为工序的调度顺序。

(2) 调度序队列 Q_u 中最长工序序列上工序，形成初始调度方案，如图 20 所示。

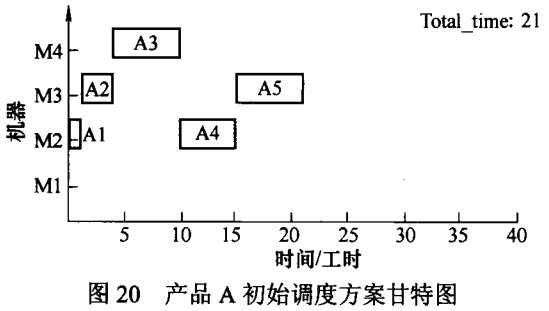
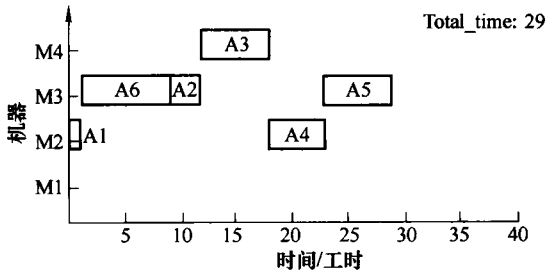
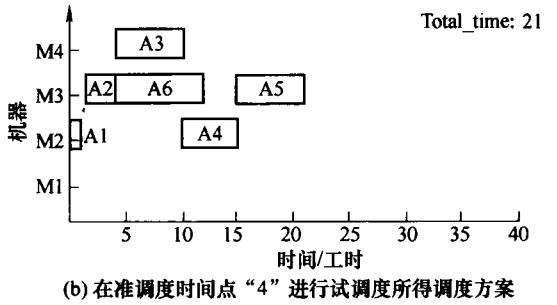


图 20 产品 A 初始调度方案甘特图

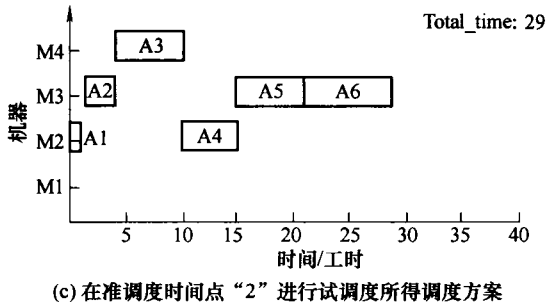
(3) 依次采用择时策略调度工序队列 Q_u 中剩余工序，首先调度工序 A6，A6 的最早开始加工时间为 1，加工设备为 M3，由于 M3 设备上已经调度了 A2 和 A5。A6 的准调度时间点分别为最早开始加工时间 1，工序 A2 的加工结束时间 4 和工序 A5 的加工结束时间 21。分别在这 3 个时间点上试调度 A6，得到如图 21 所示的 3 种试调度方案。



(a) 在准调度时间点“1”进行试调度所得调度方案



(b) 在准调度时间点“4”进行试调度所得调度方案



(c) 在准调度时间点“2”进行试调度所得调度方案

图 21　工序 A6 试调度方案甘特图

在图 21 所示的 A6“准调度方案”中，选择加工总用时最少的(b)方案作为 A6 的调度方案。并依次调度后序工序得产品调度方案，其甘特图如图 22 所示。

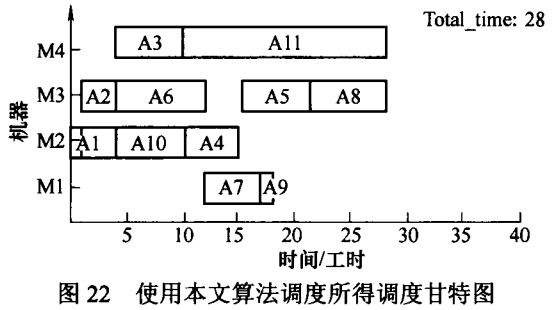


图 22　使用本文算法调度所得调度甘特图

通过以上产品调度甘特图的对比分析可知，该算法应用于一般综合调度问题的优化效果在这些算法中是最好的。

6　结论

考虑串行工序紧密度的择时综合调度算法通过提出的 2 个策略达到优化综合调度的目的，主要结论如下。

(1) 全面考虑工序的调度对工艺树整体的影响，避免了先调度工序对后调度工序的影响，提高了串行工序之间紧密度。

(2) 工序序列排序策略和择时调度策略扩大了问题的解空间，使现存各算法的解成为其的子集，摆脱了固定规则的局限，在其中灵活选择最优解。

(3) 算法复杂度不超过三次多项式。

综上所述，考虑串行工序紧密度的择时综合调度算法为解决一般综合调度问题提供了新的方法，并为进一步深入研究综合调度拓展了思路，具有一定的理论和实际意义。

参 考 文 献

[1] 王圣尧, 王凌, 许烨, 等. 求解混合流水车间调度问题的分布估计算法[J]. 自动化学报, 2012, 38(3)：437-443.
WANG Shengyao, WANG Ling, XU Ye, et al. An estimation of distribution algorithm for solving hybrid flow-shop scheduling problem[J]. Automatica Sinica, 2012, 38(3)：437-443.

[2] ZHAO Ning, YE Song, LI Kaidian, et al. Effective iterated greedy algorithm for flow-shop scheduling problems with time lags[J]. Chinese Journal of Mechanical Engineering, 2017，30(3)：652-662.

[3] 彭建刚, 刘明周, 张铭鑫, 等. 基于改进非支配排序的云模型进化多目标柔性作业车间调度[J]. 机械工程学报, 2014, 50(12)：198-205.
PENG Jiangang, LIU Mingzhou, ZHANG Mingxin, et al. Cloud model evolutionary multi-objective flexible job-shop scheduling based on improved non-dominated sorting[J]. Journal of Mechanical Engineering, 2014, 50 (12)：198-205.

[4] 张洁, 秦威, 宋代立. 考虑工时不确定的混合流水车间滚动调度方法[J]. 机械工程学报, 2015, 51(11)：99-108.
ZHANG Jie, QIN Wei, SONG Daili. Rescheduling algorithm based on rolling horizon procedure for a dynamic hybrid flow shop with uncertain processing time[J]. Journal of Mechanical Engineering, 2015, 51(11)：99-108.

[5] 贾文友, 江志斌, 李友. 面向产品族优化时间窗下可重入批处理机调度[J]. 机械工程学报, 2015, 51(12)：192-201.
JIA Wenyou, JIANG Zhibin, LI You. Family-oriented to optimize scheduling problem of re-entrant batch processing machine with due window[J]. Journal of Mechanical Engineering,2015,51(12)：192-201.

[6] TANG Dunbing, DAI Min. Energy-efficient approach to minimizing the energy consumption in an extended job-shop scheduling problem[J]. Chinese Journal of Mechanical Engineering, 2015,28 (5)：1048-1055.

[7] 谢志强, 刘胜辉, 乔佩利. 基于 ACPM 和 BFSM 的动态 Job-Shop 调度算法[J]. 计算机研究与发展, 2003, 40(7): 977-983.
XIE Zhiqiang, LIU Shenghui, QIAO Peili. Dynamic job-shop scheduling algorithm based on ACPM and BFSM[J]. Journal of Computer Research and Development, 2003, 40(7): 977-983.

[8] 谢志强, 杨静, 杨光, 等. 可动态生成具有优先级工序集的动态 Job-Shop 调度算法[J]. 计算机学报, 2008, 31(3): 502-508.
XIE Zhiqiang, YANG Jing, YANG Guang, et al, Dynamic job-shop scheduling algorithm with dynamic set of operation having priority[J]. Chinese Journal of Computers, 2008, 31(3): 502-508.

[9] 谢志强, 杨静, 周勇, 等. 基于工序集的动态关键路径多产品制造调度算法[J]. 计算机学报, 2011, 34(2): 406-412.
XIE Zhiqiang, YANG Jing, ZHOU Yong, et al. Dynamic critical paths multi-product manufacturing scheduling algorithm based on operation set[J]. Chinese Journal of Computers, 2011, 34 (2): 406-412.

[10] 谢志强,辛宇,杨静. 基于设备空闲事件驱动的综合调度算法[J]. 机械工程学报, 2011, 47(11): 139-147.
XIE Zhiqiang, XIN Yu, YANG Jing. Integrated scheduling algorithm based on event driven by machines' idle[J]. Journal of Mechanical Engineering, 2011, 47(11): 139-147.

[11] 谢志强, 辛宇, 杨静. 可回退抢占的设备驱动综合调度算法[J]. 自动化学报, 2011, 37(11): 1332-1343.
XIE Zhiqiang, XIN Yu, YANG Jing. Machine-driven integrated scheduling algorithm with rollback-preemptive[J]. Automatica Sinica, 2011, 37(11): 1332-1343.

作者简介：谢志强(通信作者)，男，1962 年出生，博士后，教授，博士研究生导师。主要研究方向为企业智能计算与调度优化。
E-mail：xiezhiqiang@hrbust.edu.cn
张晓欢(通信作者)，女，1983 年出生，博士。主要研究方向为企业智能计算。
E-mail：huanhuan291@126.com
高一龙，男，1991 年出生。主要研究方向为企业智能计算。
E-mail：1033634051@qq.com
辛宇，男，1987 年出生，博士后。主要研究方向为企业智能计算、云计算。
E-mail：xin_yu_xy@yahoo.cn