An Integrated Scheduling Algorithm for Multi-Device-Processes with the Strategy of Exchanging Adjacent Parallel Processes of The Same Device

zhen Wang
Huizhou University
xiaohuan zhang (zhangxiaohuan@hzu.edu.cn)
Huizhou University
yuxiao Cao
Nanjing Institute of Technology

Research

Keywords: Multi-Device-Process, Integrated scheduling algorithm, Adjacent parallel processes, Same processing device, Interchange strategy

Posted Date: February 22nd, 2021

DOI: https://doi.org/10.21203/rs.3.rs-212997/v1

Version of Record: A version of this preprint was published at EURASIP Journal on Wireless Communications and Networking on April 23rd, 2021. See the published version at https://doi.org/10.1186/s13638-021-01989-1.

Abstract

The current integrated scheduling algorithm ignores the influence of the pre-process on the post-process when solving the multi-device-process integrated scheduling problem, which leads to the problem of poor tightness between serial processes and low parallelism between parallel processes. This paper points out that there is no restriction of scheduling sequence between adjacent parallel processes on the same processing device, and the scheduling sequence between parallel processes on the same device can be flexibly processed to optimize the scheduling results, on the basis of the current algorithm scheduling scheme, this paper proposes the application of multi-device adjacent parallel process interchange strategy and multi-device adjacent parallel process interchange adjustment strategy, which avoid the influence of the pre-process on the post-process, improves the compactness of the serial process and the parallelism of the parallel process, and optimizes the scheduling results.

1 Introduction

The traditional process of making a product is to break it down into parts, The manufacturing-before-assembly method was adopted,For large quantities of the same products, flow-shop is adopted. Job-shop is adopted for multi-variety and small-batch products. Therefore, the corresponding pure machining scheduling and pure assembly scheduling have been deeply studied. Du et al. [1] and B S et al. [2] solve scheduling problem with genetic algorithm,Huang et al. [3] and Shen et al. [4] apply neural network to solve scheduling problem, Ying et al. [5] and C S W L A B et al. [6] solve scheduling problem with simulated annealing method, Mathlouthi et al. [7] and He et al. [8] apply abu search method to solve scheduling problem, Wu et al. [9] and Anghinolfi et al. [10] solve scheduling problem with heuristic algorithm.

With the increase of the society's demand for personalized products, the production mode of multiple varieties and small batches has attracted more and more attention. For multi-variety and small batch products, especially complex single products with tree structure, if the mode of manufacturing-before-assembly is adopted, the parallel processing relationship between product processing and assembly will be separated inevitably, which will affect the manufacturing efficiency of the product.Therefore, Xie et al. [11–12] propose a complex single-product comprehensive scheduling algorithm for processing and assembly at the same time.

The situation that a single process needs multiple devices to process together exists in the actual production of assembly manufacturing enterprises. For example, when welding two large parts, you need two devices to hold them together, and you need a welding device to weld them. At this time, the welding process needs three device to complete together.Therefore, it is necessary to solve the problem of integrated scheduling of single complex product where multiple devices work together to complete a single special process.

At present, the research on this problem is in its initial stage. Zhang et al. [13] proposes a multiple-devices-process integrated scheduling algorithm with time-selective strategy for process sequence. Based on [13], this algorithm proposes a Multi-device adjacent parallel process interchange strategy and a multi-device adjacent parallel process interchange adjustment strategy. These two strategies provide solutions to the problem in [13] of ignoring the influence of the pre-processing process on the post-processing process, which leads to poor tightness between serial processes and poor parallelism between parallel processes, and further optimize the scheduling results.

Aiming at the above problems, this paper points out that there is no restriction relation of scheduling sequence between parallel operations on the same processing device. The scheduling results of the algorithm can be optimized by flexibly handling the scheduling sequence between parallel processes of the same device. Based on [13], it is proposed to apply the multi-device adjacent parallel process interchange strategy and the multi-device adjacent parallel process interchange adjustment strategy.They avoid the influence of the pre-processing process on the post-processing process, improve the tightness of the serial processing procedure and the parallelism of the parallel processing procedure, so as to shorten the processing time of products.

The rest of this paper is organized as follows. Section 2 shows the problem description and analysis. Section 3 discusses the strategy design. The optimization algorithm is proposed in Sect. 4.designs the proposed algorithm. Section 5 shows the example analysis. Section 6 shows experimental comparison and analysis. Section 7 concludes the paper with summary.

2 Problem Description And Analysis

The process tree of complex products is a tree-like structure established according to the restriction relationship between processes of the product, the nodes in the process tree represent the process of the product, directed edges in the process tree represent the partial order relation of the process order, the root node in the process tree represents the final step of the product, when the root process is finished, it indicates that the product is finished.When multi-device processes exist in the process set, the complex single-product scheduling problem must satisfy the following six constraints simultaneously and minimize the total processing time of the product.

1. Each process must strictly comply with the partial order relationship agreed in the process tree
2. Each device can only process one procedure at any time, and the processing process can not be interrupted
3. Devices with the same functionality do not exist on the device set
4. In the process set, there are multi-device-processes which are jointly processed by multiple related devices
5. Each process must wait for the completion of all the process tree pre-process and device pre-process before processing
6. The difference between the finishing time of the latest processing procedure and the beginning time of the earliest processing procedure is the total processing time of the product.

3 Strategy Design

The strategy design in this paper is based on the scheduling results of multiple-devices-process integrated scheduling algorithm with time-selective strategy for process sequence. on this basis, several new product scheduling schemes are generated by using the multi-device adjacent parallel process interchange strategy and the multi-device adjacent parallel process interchange adjustment strategy, and select the scheduling scheme with the best scheduling result.

The multi-device adjacent parallel process interchange strategy and the multi-device adjacent parallel process interchange adjustment strategy are respectively introduced below.

3.1 Multi-device adjacent parallel process interchange strategy

This strategy will look for adjacent parallel processes on each processing device, since the problem studied in this paper is the integrated scheduling problem involving multi-device-process, the parallel process will be divided into two kinds, namely: common process and multi-device- process.

Therefore, the following four situations need to be discussed in the adjacent parallel processes:

1. Both the pre-tightening and post-tightening processes are common processes

Assume that in the current parallel adjacent processes in the same device, the pre-tightening process is process *A*, and the post-tightening process is process *B*. They are all common processes.

As shown in Fig.1, first cancel the scheduling of process *A*; And then reschedule *B*, compare the maximum finishing time of all processes of process *B* in the pre-tightening process set in the process tree with the finishing time of the pre-tightening process on its device, the maximum value is set as the processing start time of process *B*; Then process *A* is rescheduled, and the finishing time of process *B* is taken as the starting time of process *A*.

2. Pre-tightening processes are common processes, post-tightening processes are multi-deivce- processes

Assume that in the current parallel adjacent processes in the same device, the pre-tightening process is process *A*, it is common process, and the post-tightening process is process *B*, it is multi-device-process and contains several parallel sub-processes.

As shown in Fig.2, first cancel the scheduling of process *A*; And then reschedule *B*, compare the maximum finishing time of all processes in the pre-tightening process set of each parallel sub-process included in process *B* in its process tree and the finishing time of the pre-tightening process on its device respectively. The maximum of all the above times is set as the processing start time of process *B*; Then process A is rescheduled, and the finishing time of process *B* is taken as the starting time of process *A*.

3. The pre-tightening process is multi-device process, and the post-tightening process is common processes

Assume that in the current parallel adjacent processes in the same device, the pre-tightening process is process *B*, it is multi-device- process and contains several parallel sub-processes, the post-tightening process is process *A*，it is common process.

As shown in Fig.3, First of all, cancel the scheduling of all parallel sub-processes contained in process *B*; Then process *A* is rescheduled, compare the maximum finishing time of all processes in the pre-tightening process set of process *A* in the processing process tree with the finishing time of the pre-tightening process set of process *A* on its processing device, The maximum of all the above times is set as the processing start time of process *A*，Then process *B* is rescheduled, the finishing time of process *A* is respectively taken as the starting time of each parallel sub-process in process *B*.

4. Both the pre-tightening and post-tightening processes are multi-device-processes

Assume that in the current parallel adjacent processes in the same device, the pre-tightening process is process *B*, and the post-tightening process is process *A*. They are all multi-device-processes and contain several parallel sub-processes.

As shown in Fig.4, cancel the scheduling of all parallel sub-processes contained in process *B*; Then process *A* is rescheduled, compare the maximum finishing time of all processes in the pre-tightening process set of each parallel sub-process included in process *A* in its process tree and the finishing time of the pre-tightening process on its device respectively. The maximum of all the above times is set as the processing start time of process *A*; Then process *B* is rescheduled, the finishing time of process *A* is respectively taken as the starting time of each parallel sub-process in process *B*.

The algorithm steps are as follows:

Step 1. Assume that in the current parallel adjacent processes in the same device, the pre-tightening process is process *A*, and the post-tightening process is process *B*;

Step 2. Determine whether the pre-tightening process *A* is a common process, if true, go to step 3, otherwise go to step 4;

Step 3. Cancel the scheduling of process *A*, then go to Step 5;

Step 4. Cancels the scheduling of all parallel sub-processes contained in process *A*；

Step 5. Determine whether the post-tightening process *B* is a common process, go to step 6, otherwise go to step 7；

Step 6. Rescheduling Process *B*, compare the maximum finishing time of all processes in the pre-tightening process set of process *B* in the process tree with the finishing time of the pre-tightening process set of process *B* on its device, The maximum of all the above times is set as the processing start time of process *B*.

Step 7. Rescheduling Process *B*, compare the maximum finishing time of all processes in the pre-process set of process *B* in the process tree with the finishing time of the pre-tightening process set of each parallel sub-process included in the process *B* on their device, the maximum of all the above times is set as the processing start time of process *B*.

Step 8. Determine whether the pre-tightening process *A* is a common process, if true, go to step 9, otherwise go to step 10;

Step 9. Take the finishing time of process *B* as the starting time of process *A*, go to step 11.

Step 10. The finishing time of process *B* is respectively taken as the starting time of each parallel sub-process in process *A*.

Step 11. End.

The algorithm flow chart is shown in Figure 5.

Fig. 5 Algorithm flow chart of multi-device adjacent parallel process interchange strategy

3.2 Multi-device adjacent parallel process interchange adjustment strategy

According to the analysis in Section 2.1, the interchange process is divided into two kinds: common process and multi-device process.

In the interchange process, the post-tightint process *P* will be changed to the front, the processing start time is earlier,

Therefore, the scheduling of post-tightening process *P* may advance the processing time of post-tightening process in its process tree and post-tightening process of device.

At the same time, the tight preprocess F will shift to the back, The processing start time will be delayed, which may affect other processes. The influence mentioned here means that the rescheduling of the previous process will result in overlapping time with the scheduling of the subsequent processes, making the processing scheme unfeasible, so it is necessary to adjust the affected subsequent processes.

Rescheduling the post-tightint process *P* can be divided into two situations:

1. Process *P* is common process

If the adjustment process is a common process, the processes affected are divided into two categories: one is the post-tightening process in the process tree; The other is the post-tightening process in the processing device. It is necessary to check whether the processing time of these two types of processes needs to be advanced.

2. Process *P* is multi-deivce-process

If the process is adjusted to a multi-device process, the number of parallel sub-processes contained in the process is first determined, then check the post-tightening process in the process tree and the post-tightening process of processing device of each parallel sub-process, determine whether the processing time of these two kinds of processes needs to be advanced.

The methods to advance the processing time of the above processes are as follows:

1. For the common process

Starting from the rescheduling end time of process *P*, the first available time point was found to be the starting time of the post-tightening of process *P* in the process tree, taking the end time of process *P* rescheduling as the starting time of the post-tightening of process *P* in its device.

2. For the multi-device-process

Starting from the rescheduling end time of process *P*, the first available time point on each equipment is found as the starting time of the post-tightening process in process tree of process P. The end time of process *P* rescheduling is taken as the start time of post-tightening process on the device.

Rescheduling the pre-tightint process *F* can be divided into two situations:

1. Process *F* is common process

If the adjustment process is a common process, there are two kinds of processes affected. One is the post-tightening process in the process tree; The other is the post-tightening process in the processing device. The two kinds of processes need to be inspected respectively to determine whether they are affected.

2. Process *F* is multi-deivce-process

If the adjustment process is a multi-device-process, first of all, it is determined that it contains parallel sub-processes, and then separately check the post-tightening process in its process tree and the post-tightening process on the devices of each parallel sub-process to determine whether it is affected. If affected, rescheduling is required.

The above mentioned processes involved in the adjustment are divided into two situations:

1. For the common process

For the post-processing process in the process tree of procedure *F*, set its processing start time as the maximum of the processing end time of the pre-tightening process on the device and the processing end time of process *F* in the current scheme, For the post-tightening process on the device, its processing start time is set as the processing end time of the process *F*.

2. For the multi-device-process

Due to the multi-device-process includes several parallel sub-processes, Adjust the post-tightening process of process *F* in the process tree to ensure that the starting time of each sub-process is consistent, calculate the finishing time of process *F* and the finishing time of the pre-processing process of each sub-process on its processing device in the current scheme, respectively, then, the maximum value is set as the processing start time of the multi-device process. For the post-tightening process on the device, the finishing time of process F is set as the starting time of each parallel sub-process contained in it.

In addition, the adjustment of the above processes may affect other processes, and then the adjustment will affect their subsequent processes, so it is necessary to check all the processes that may be affected in turn until all the affected processes are adjusted, and finally generate the scheduling scheme.

The implementation of this policy relies on the queue data structure, first set up the adjustment process queue. Initially, the pre-tightening and post-tightening processes that need to be interchanged are separately queued, and then determine whether the process is a common process or a multi-device-process. If it is a common process, then judge whether the post-tightening process in its process tree and the post-tightening process of the same device are affected, as shown in Figure 6 (a). If so, adjust them separately and put them into the queue so as to check and adjust the subsequent sequence processes affected by them;

If it is a multi-device-process, then judge whether the post-tightening process in its process tree and the post-tightening process of each parallel sub-process on their device are affected, as shown in Figure 6 (b).If so, adjust them separately and put them into the queue so as to check and adjust the subsequent sequence processes affected by them.

Fig. 6 Schematic diagram of multi-device adjacent parallel process interchange adjustment strategy

The algorithm steps are as follows:

Step 1. Adjacent interchange processes on the same device are respectively added to the adjustment queue;

Step 2. The adjustment queue is queued and the result is stored in *P*[];

Step 3. Judge whether *P* is empty, if not, go to step 4, otherwise go to step 32;

Step 4. Determine whether *P* is a common process, If it is, go to step 5; otherwise, go to step 18;

Step 5. Judge whether *P* is processed in advance, If it goes to step 6, otherwise, go to step 12;

Step 6. Look for the post-tightening process *FT* of *P* in the process tree, If any, go to step 7, otherwise go to step 9;

Step 7. Taking the end time of process *P* rescheduling as the starting point, the first available time point was found as the starting time of process *FT*;

Step 8. Add process *FT* into the adjustment queue, If it is a multi-device-process, then add each parallel sub-procedure into the adjustment queue in turn;

Step 9. Look for the post-tightening process *TD* of *P* on the device, If any, go to step 10, otherwise go to step 12;

Step 10. Taking the end time of process *P* rescheduling as the starting point, the first available time point was found as the starting time of process *TD*;

Step 11. Add process *TD* into the adjustment queue, If it is a multi-device-process, then add each parallel sub-process into the adjustment queue in turn, go to step 2;

Step 12. Look for the post-tightening process *AT* of *P* in the process tree, If any, go to step 13, otherwise go to step 15;

Step 13. The maximum value of the finishing time of pre-tightening process of *AT* and the finishing time of process *P* is set as the starting time of process *AT*;

Step 14. Add process *AT* into the adjustment queue, If it is a multi-device-process, then add each parallel sub-process into the adjustment queue in turn;

Step 15. Look for the post-tightening process *AD* of *P* on the device, If any, go to step 16, otherwise go to step 18;

Step 16. The finishing time of the current process is set as the processing start time of *AD* process;

Step 17. Add process *AD* into the adjustment queue, If it is a multi-device-process, then add each parallel sub-process into the adjustment queue in turn, go to step 2;

Step 18. Judge whether *P* is processed in advance, If it is, go to step 19; otherwise, go to step 25;

Step 19. Look for the post-tightening process *MFT* of *P* in the process tree, If any, go to step 20, otherwise go to step 22;

Step 20. Taking the end time of process *P* rescheduling as the starting point, the first available time point was found as the starting time of process *MFT*;

Step 21. Add process *MFT* into the adjustment queue, If it is a multi-device-process, then add each parallel sub-process into the adjustment queue in turn;

Step 22. Look for the post-tightening processes on the device of each parallel sub-process included in the process *P*, and store it in *MFD*[], If any, go to step 23, otherwise go to step 25;

Step 23. The finishing time of each parallel sub-process in procedure P is taken as the starting time of each process in its corresponding process *MFD*[];

Step 24. Add each process in *MFD*[] to the adjustment queue, If one of the process is a multi-device procedure, it is necessary to add each parallel sub-process into the adjustment queue successively, go to step 2;

Step 25. Look for the post-tightening process *MAT* of *P* in the process tree, If any, go to step 26, otherwise go to step 28;

Step 26. Calculate the finishing time of the pre-processing process on the device and the finishing time of the pre-processing process in the process tree of each parallel sub-process, respectively, Then, the maximum value is set as the start time of *MAT*;

Step 27. Add process *MAT* into the adjustment queue, If it is a multi-device-process, then add each parallel sub-process into the adjustment queue in turn;

Step 28. Look for the post-tightening processes on the device of each parallel sub-process included in the process *P*, and store it in *MAD*[], If any, go to step 29, otherwise go to step 31;

Step 29. The finishing time of each parallel sub-process in procedure *P* is taken as the starting time of each process in its corresponding process *MAD*[];

Step 30. Add each process in *MAD*[] to the adjustment queue, If one of the process is a multi-device procedure, it is necessary to add each parallel sub-process into the adjustment queue successively;

Step 31. Go to step 2。

Step 32. End the adjustment, calculate the total processing time of the current scheme, exit.

The algorithm flow chart of multi-device adjacent parallel process interchange adjustment strategy is shown in Figure 7.

4 Algorithm Design

Step 1. The basic scheduling scheme is developed by using multiple-devices-process integrated scheduling algorithm with time-selective strategy for process sequence;

Step 2. Assume that the number of processing device is *M*;

Step 3. *k*=1。

Step 4. Determine *k* < = *M*, If it is established, go to step 5; if it is not established, go to step 14;

Step 5. Assume that the number of device *k* is *Nk*;

Step 6. *i*=1。

Step 7. Determine *i*<*Nk*。If it is established, go to step 8; if it is not established, go to step 13;

Step 8. Determine whether the *i* process and *i* +1 process are parallel processes, If so, go to 9, not go to step 12;

Step 9. Start the multi-device adjacent parallel process interchange strategy, Interchange the *i* process with the *i*+1 process;

Step 10. Start themulti-device adjacent parallel process interchange adjustment strategy, Adjust the process affected by the step 9;

Step 11. Generate a new product scheduling scheme and add it to the product scheduling set;

Step 12. *i*++, go to step 7。

Step 13. *k*++,go to step 4。

Step 14. The total processing time of each product scheduling scheme in the product scheduling scheme set is calculated respectively;

Step 15. The scheme with minimum processing time was selected as the final scheduling scheme of the product;

Step 16. Outputs the Gantt chart of scheduling results;

Step 17. End.

The algorithm flow chart of this paper is shown in Figure 8.

5 The Example Analysis

In order to facilitate readers to understand the algorithm, the following through the example analysis. Product *P* is shown in Fig. 9, where each processing node represents one processing procedure, which can be divided into common procedure and multi-equipment procedure. For example, *P*1/*M*1/10 means that the procedure is named *P*1 and processed on equipment *M*1, which is a common procedure with a processing duration of 10; *P*2 / *M*1*M*2 /20, which means that the process name is *P*2, and it is a multi-equipment process, which needs to be processed on equipment *M*1 and *M*2 at the same time, and the processing time is 20. According to the algorithm proposed in literature [13], the total processing time of product *P* is 265, and its scheduling Gantt chart is shown in Figure 10. Meanwhile, according to the algorithm proposed in this paper, the scheduling process of product *P* is shown in Table 1, and the total processing time of product *P* is 260, and the final product *P* scheduling Gantt chart is shown in Figure 11.

Table 1 The scheduling process of product *P* scheduled by the proposed algorithm

Device id	Exchange process	Each process scheduling time point in the current scheme
*M*1	*P*23、*P*25	*P28:0,P26:20,P23:60,P18:90,P13:115,P8:145,P6:175,P2:235,P27:20,P25:40,P20:115,P14:175,P9:200,P21:55,P16:200,P10:245,P7:200,*
*M*1	*P*19、*P*20	*P28:0,P26:20,P23:40,P18:70,P13:110,P8:140,P6:170,P2:230,P27:20,P25:70,P20:95,P14:170,P9:195,P21:55,P16:195,P10:240,P7:195,P*
*M*1	*P*20、*P*8	*P28:0,P26:20,P23:40,P18:70,P13:110,P8:140,P6:170,P2:255,P27:20,P25:70,P20:170,P14:195,P9:220,P21:55,P16:220,P10:265,P7:220,*
*M*1	*P*14、*P*8	*P28:0,P26:20,P23:40,P18:70,P13:110,P8:160,P6:190,P2:230,P27:20,P25:70,P20:110,P14:135,P9:160,P21:55,P16:190,P10:230,P7:190,*
*M*1	*P*14、*P*7	*P28:0,P26:20,P23:40,P18:70,P13:110,P8:140,P6:170,P2:230,P27:20,P25:70,P20:110,P14:185,P9:210,P21:55,P16:210,P10:255,P7:170,*
*M*1	*P*7、*P*22	*P28:0,P26:20,P23:40,P18:70,P13:110,P8:140,P6:170,P2:235,P27:20,P25:70,P20:110,P14:170,P9:195,P21:55,P16:195,P10:245,P7:210,*
*M*1	*P*22、*P*1	*P28:0,P26:20,P23:40,P18:70,P13:110,P8:140,P6:170,P2:230,P27:20,P25:70,P20:110,P14:170,P9:195,P21:55,P16:195,P10:240,P7:195,*
*M*2	*P*23、*P*17	*P28:0,P26:20,P23:90,P18:120,P13:145,P8:185,P6:215,P2:275,P27:20,P25:120,P20:160,P14:215,P9:240,P21:55,P16:240,P10:285,P7:2*
*M*2	*P*17、*P*13	*P28:0,P26:20,P23:40,P18:70,P13:95,P8:145,P6:175,P2:235,P27:20,P25:70,P20:110,P14:175,P9:200,P21:55,P16:200,P10:245,P7:200,P*
*M*2	*P*13、*P*15	*P28:0,P26:20,P23:40,P18:70,P13:155,P8:185,P6:215,P2:230,P27:20,P25:70,P20:110,P14:135,P9:185,P21:55,P16:160,P10:255,P7:215,*
*M*2	*P*15、*P*6	*P28:0,P26:20,P23:40,P18:70,P13:110,P8:140,P6:170,P2:230,P27:20,P25:70,P20:110,P14:170,P9:210,P21:55,P16:195,P10:255,P7:195,*
*M*2	*P*9、*P*6	*P28:0,P26:20,P23:40,P18:70,P13:110,P8:140,P6:220,P2:240,P27:20,P25:70,P20:110,P14:170,P9:195,P21:55,P16:195,P10:260,P7:195,*
*M*2	*P*9、*P*3	*P28:0,P26:20,P23:40,P18:70,P13:110,P8:140,P6:170,P2:230,P27:20,P25:70,P20:110,P14:170,P9:235,P21:55,P16:195,P10:260,P7:195,*
*M*2	*P*3、*P*10	*P28:0,P26:20,P23:40,P18:70,P13:110,P8:140,P6:170,P2:230,P27:20,P25:70,P20:110,P14:170,P9:195,P21:55,P16:195,P10:220,P7:195,*
*M*3	*P*24、*P*21	*P28:0,P26:20,P23:40,P18:70,P13:110,P8:140,P6:170,P2:230,P27:20,P25:70,P20:110,P14:170,P9:195,P21:35,P16:195,P10:240,P7:195,*
*M*3	*P*21、*P*18	*P28:0,P26:20,P23:40,P18:70,P13:130,P8:160,P6:190,P2:250,P27:20,P25:70,P20:110,P14:190,P9:195,P21:95,P16:135,P10:260,P7:215,*
*M*3	*P*18、*P*20	*P28:0,P26:20,P23:40,P18:115,P13:140,P8:170,P6:200,P2:255,P27:20,P25:70,P20:90,P14:140,P9:220,P21:55,P16:70,P10:110,P7:200,P*
*M*3	*P*20、*P*8	same as *M*1
*M*3	*P*8、*P*14	same as *M*1
*M*3	*P*14、*P*16	*P28:0,P26:20,P23:40,P18:70,P13:110,P8:140,P6:170,P2:250,P27:20,P25:70,P20:110,P14:190,P9:215,P21:55,P16:170,P10:260,P7:215,*
*M*3	*P*16、*P*3	*P28:0,P26:20,P23:40,P18:70,P13:110,P8:140,P6:170,P2:230,P27:20,P25:70,P20:110,P14:170,P9:230,P21:55,P16:230,P10:255,P7:195,*
*M*3	*P*3、*P*5	*P28:0,P26:20,P23:40,P18:70,P13:110,P8:140,P6:170,P2:230,P27:20,P25:70,P20:110,P14:170,P9:195,P21:55,P16:195,P10:255,P7:195,*
*M*3	*P*23、*P*25	same as *M*1
*M*4	*P*25、*P*17	*P28:0,P26:20,P23:40,P18:70,P13:110,P8:155,P6:185,P2:245,P27:20,P25:90,P20:130,P14:185,P9:210,P21:55,P16:210,P10:255,P7:210,*
*M*4	*P*17、*P*13	same as *M*2
*M*4	*P*8、*P*12	*P28:0,P26:20,P23:40,P18:70,P13:110,P8:155,P6:185,P2:245,P27:20,P25:70,P20:110,P14:185,P9:210,P21:55,P16:210,P10:255,P7:210,*
*M*4	*P*12、*P*11	*P28:0,P26:20,P23:40,P18:70,P13:110,P8:140,P6:170,P2:230,P27:20,P25:70,P20:110,P14:170,P9:195,P21:55,P16:195,P10:240,P7:195,*
*M*4	*P*11、*P*4	*P28:0,P26:20,P23:40,P18:70,P13:110,P8:140,P6:170,P2:250,P27:20,P25:70,P20:110,P14:170,P9:195,P21:55,P16:195,P10:240,P7:195,*
*M*4	*P*4、*P*2	*P28:0,P26:20,P23:40,P18:70,P13:110,P8:140,P6:170,P2:205,P27:20,P25:70,P20:110,P14:170,P9:195,P21:55,P16:195,P10:240,P7:195,*

6 Experimental Comparison And Analysis

To verify the effectiveness of the proposed algorithm,four sets of data were randomly generated, with 50 products in each set. The parameters of products were randomly generated.The software used on the experimental platform is Windows 10, 64 bit, GCC5.5, and the hardware of the experimental platform is an Intel Core I7-860 processor with 32 GB of memory. Five groups of experiments were designed as follows: Four groups of experiments were designed as follows: The randomly generated data in Experiment 1 were that the number of processes was less than 20，the total processing time of the workpiece was less than 50 and the number of devices is 2, 3 and 5 in Multiple-Devices-Process respectively；The randomly generated data in Experiment 2 were that the number of processes was between 20-30，the total processing time of the workpiece was between 50-100 and the number of devices is 2, 3 and 5 in Multiple-Devices-Process respectively；The randomly generated data in Experiment 3 were that the number of processes was between 30-50，the total processing time of the workpiece was between 100-200 and and the number of devices is 2, 3 and 5 in Multiple-Devices-Process respectively；The randomly generated data in Experiment 4 were that the number of processes was beteen 50-60，the total processing time of the workpiece was betweetn 200-300 and and the number of devices is 2, 3 and 5 in Multiple-Devices-Process respectively.

7 Conclusion

The current advanced algorithm--multiple-devices-process integrated scheduling algorithm with time-selective strategy for process sequence is used as the basic scheduling scheme; Several new scheduling schemes are generated on the basis of the basic scheduling scheme by using multi-device adjacent parallel process interchange strategy and a multi-device adjacent parallel process interchange adjustment strategy, then the scheme with the shortest processing time is selected to further optimize the multi-device process scheduling algorithm.

Declarations

Acknowledgements

The authors acknowledged the anonymous reviewers and editors for their efforts in valuable comments and suggestions.

Funding

This work was supported by the Project of Educational Commission of Guangdong (Grant No. 2019KTSCX177), Science and Technology Planning Project of Guangdong (Grant No. 2020A1414010235), Science and Technology Planning Project of Huizhou(Grant No. 2020SC0306023) and Professorial and Doctoral Scientific Research Foundation of Huizhou University (Grant No. 2019JB014, 2018JB007).

Availability of data and materials

Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study.

Authors' contributions

Z. Wang proposes the innovation ideas and theoretical analysis, and X. Zhang carries out experiments and data analysis. Y. Cao conceived of the study, and participated in its design and coordination and helped to draft the manuscript. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Author details

[1]School of Computer Science and Engineering, Huizhou University, Huizhou, 516007, P. R. China

[2]School of Computer Engineering, Nanjing Institute of Technology, Nanjing, 211167, P. R. China

References

1. Du , P. Dong, V. Sugumaran, et al, Dynamic decision support framework for production scheduling using a combined genetic algorithm and multiagent model. Expert Systems 2020(5).
2. B S Y A , B T L A , B B W A, A co-evolutionary genetic algorithm for the two-machine flow shop group scheduling problem with job-related blocking and transportation times. Expert Systems with Applications 152.
3. Huang, L. Gao, A Time Wave Neural Network Framework for Solving Time-Dependent Project Scheduling Problems. IEEE Transactions on Neural Networks and Learning Systems 2019, 1-10.

4. Shen, S. Huo, H. Yan, et al. Distributed Dissipative State Estimation for Markov Jump Genetic Regulatory Networks Subject to Round-Robin Scheduling. IEEE Transactions on Neural Networks and Learning Systems 2019(99), 1-10.
5. Ying, S. Lin, Solving no-wait job-shop scheduling problems using a multi-start simulated annealing with bi-directional shift timetabling algorithm. Computers & Industrial Engineering 2020, 146:106615.
6. C S W L A B , D C Y C , D P P , et al, Multi-temperature simulated annealing for optimizing mixed-blocking permutation flowshop scheduling problems - ScienceDirect. Expert Systems with Applications, 165.
7. Mathlouthi, M. Gendreau, J. Potvin, A Metaheuristic based on Tabu Search for Solving a Technician Routing and Scheduling Problem. Computers & Operations Research 2020:105079.
8. He, M. Weerdt, N. Yorke-Smith, Time/sequence-dependent scheduling: the design and evaluation of a general purpose tabu-based adaptive large neighbourhood search algorithm. Journal of Intelligent Manufacturing 2020(10).
9. Wu, J. Cheng, F. Chu, Large-scale energy-conscious bi-objective single-machine batch scheduling under time-of-use electricity tariffs via effective iterative heuristics. Annals of Operations Research 2019(3).
10. Anghinolfi, M. Paolucc, R. Ronco, A bi-objective heuristic approach for green identical parallel machine scheduling. European Journal of Operational Research 2020.
11. Xie, S. Liu, P Qiao, Dynamic job-shop scheduling algorithm based on ACPM and BFSM. Journal of Computer Research and Development, 2003(40)977–983.
12. Xie, J. Yang, Y. Zhou, et al. Dynamic critical paths multi-product manufacturing scheduling algorithm based on operation set. Chinese Journal of Computers. 2011(34) 406–412.
13. X. Zhang, D. Zhang, Z Wang, Y Xin, Multiple-Devices-Operation Integrated Scheduling Algorithm with Time-selective for Process Sequence. Complexity 2020(10).

Figures

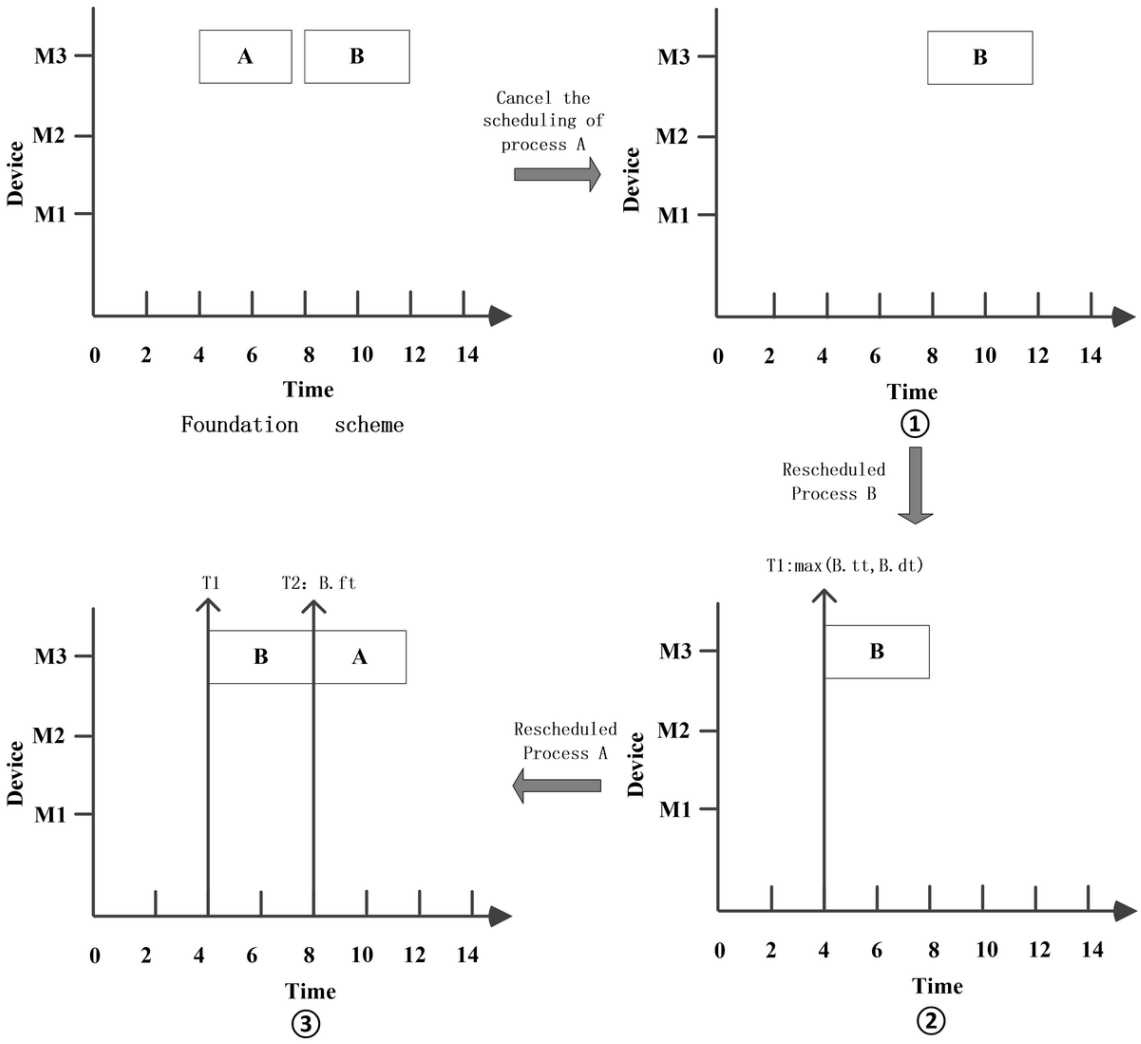


Figure 1

Schematic diagram of the interchange when the pre - and post-tightening processes are common processes

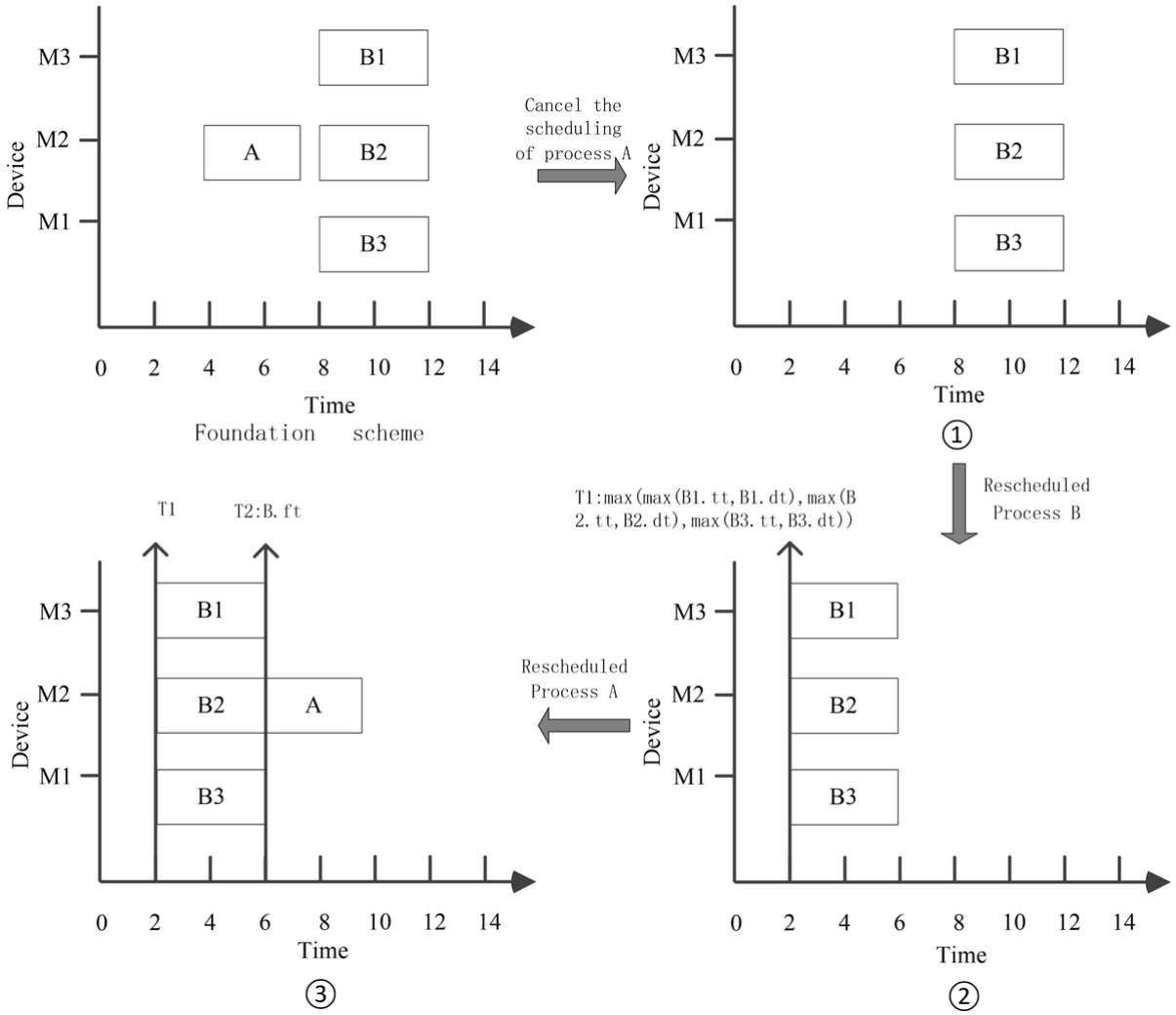


Figure 2

Schematic diagram of the interchange when the pre-tightening process is a common process and the post-tightening process is a multi-device process

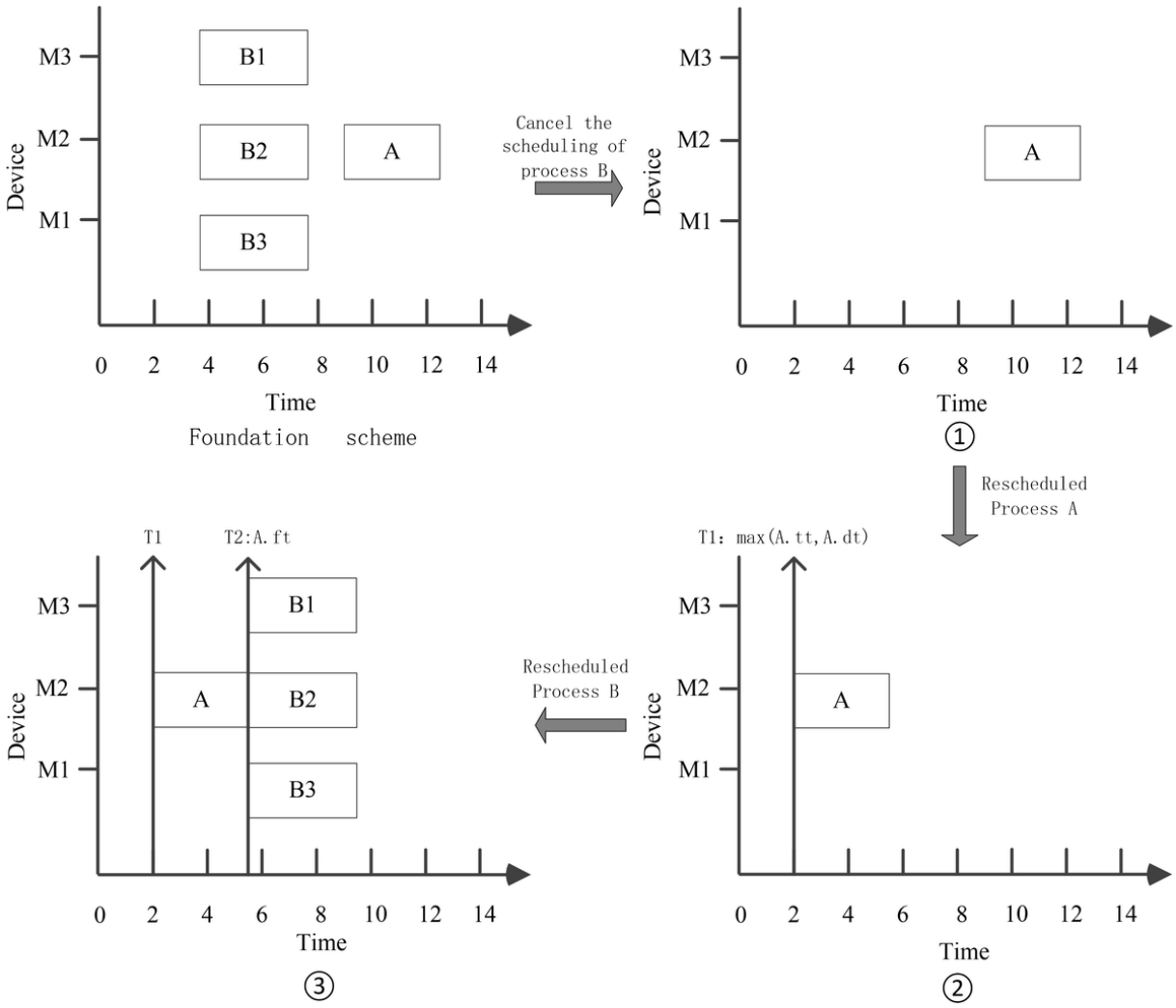


Figure 3

Schematic diagram of the interchange when the pre-tightening process is a multi-device process and the post-tightening process is a common process

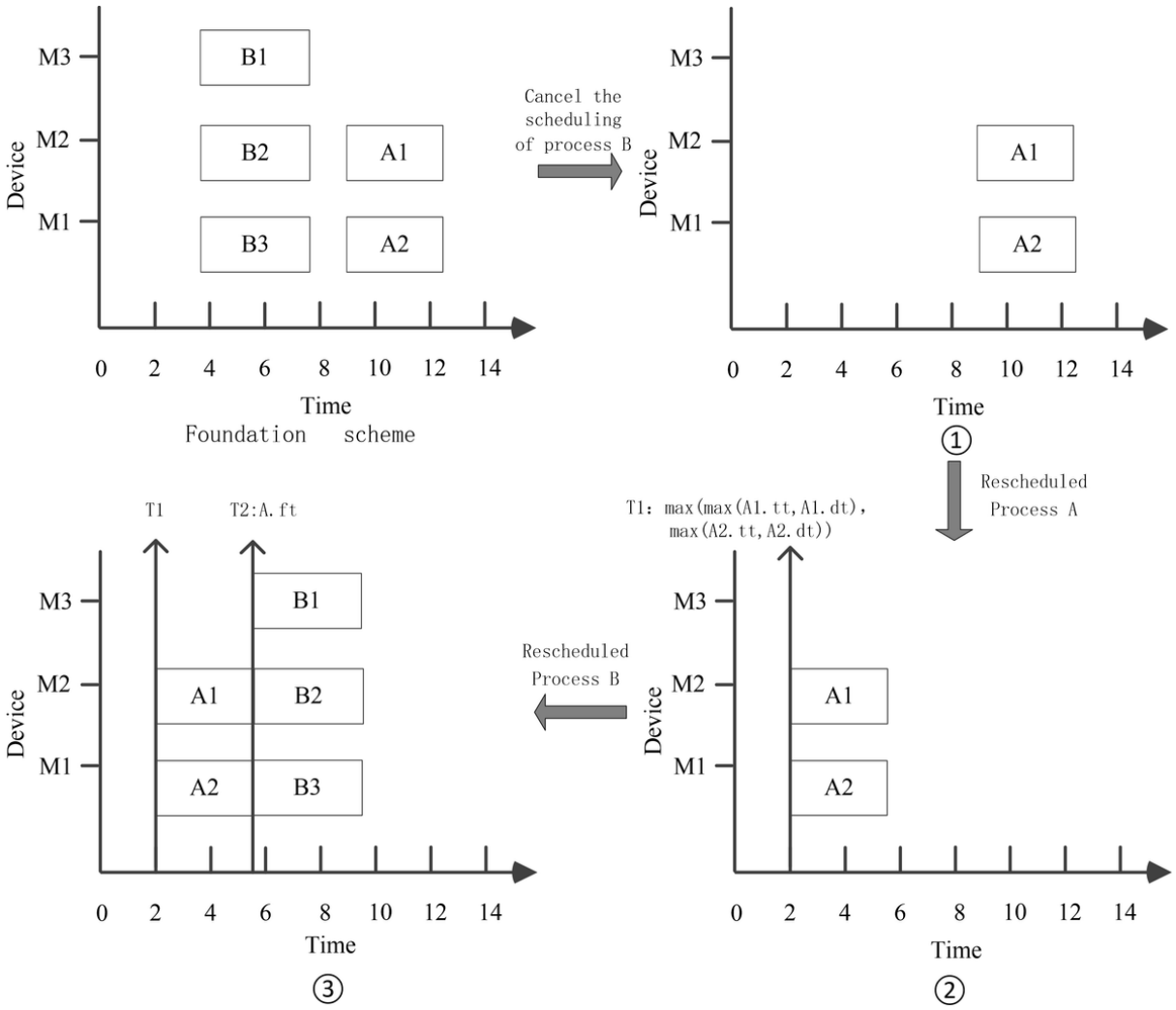


Figure 4

Schematic diagram of the interchange when the pre - and post-tightening processes are multi-device processes

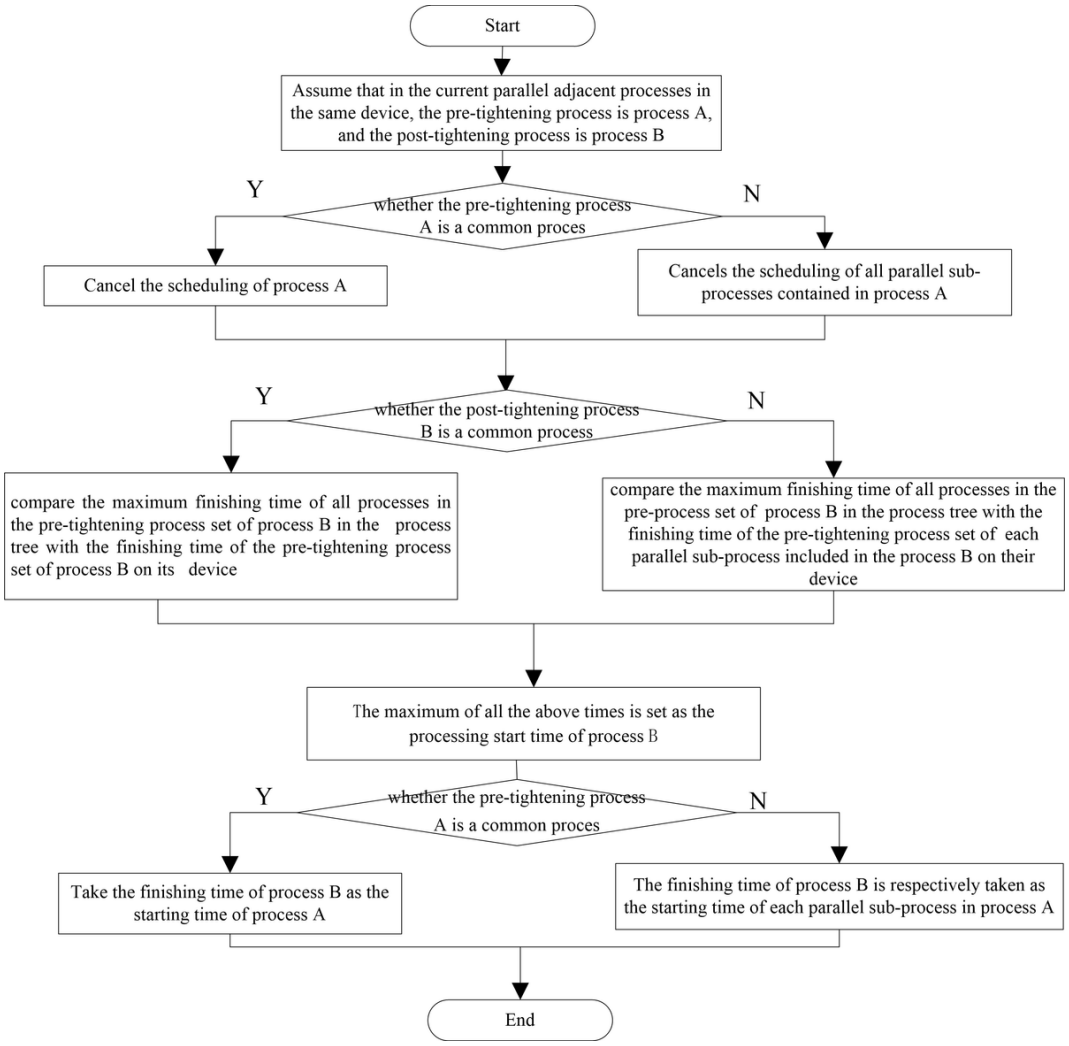
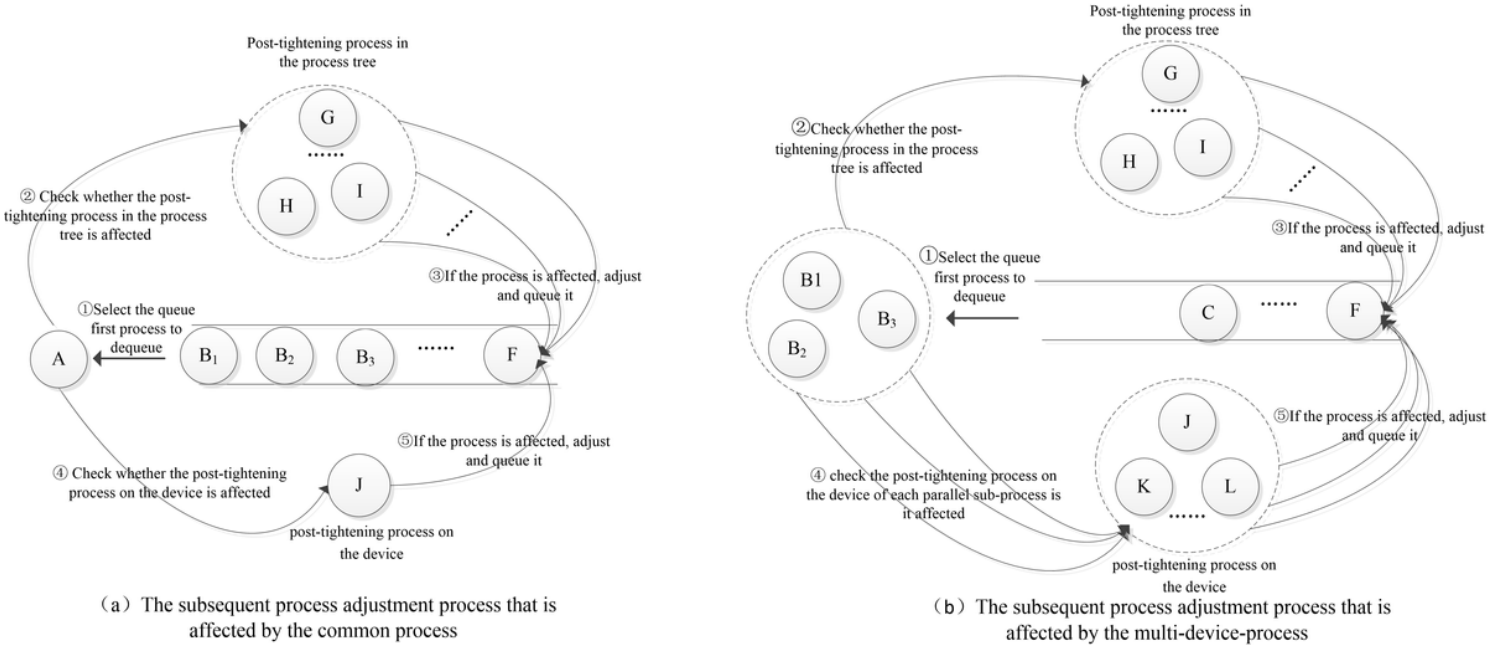


Figure 5

Algorithm flow chart of multi-device adjacent parallel process interchange strategy



（a）The subsequent process adjustment process that is affected by the common process

（b）The subsequent process adjustment process that is affected by the multi-device-process

Figure 6

Schematic diagram of multi-device adjacent parallel process interchange adjustment strategy

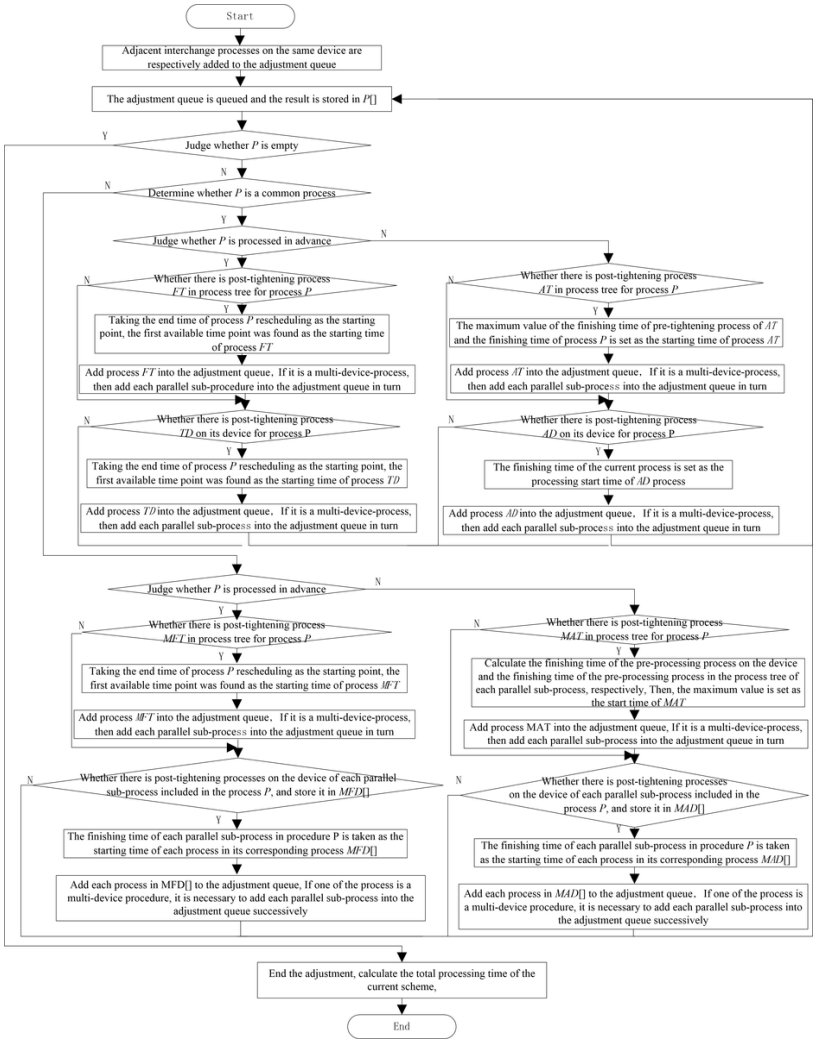


Figure 7

Flowchart of multi-device adjacent parallel process interchange adjustment strategy

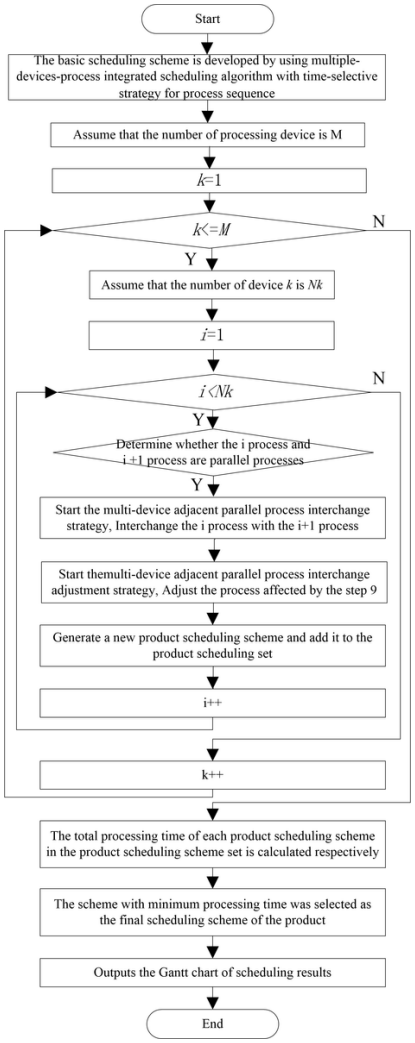


Figure 8

Algorithm flow chart of this paper

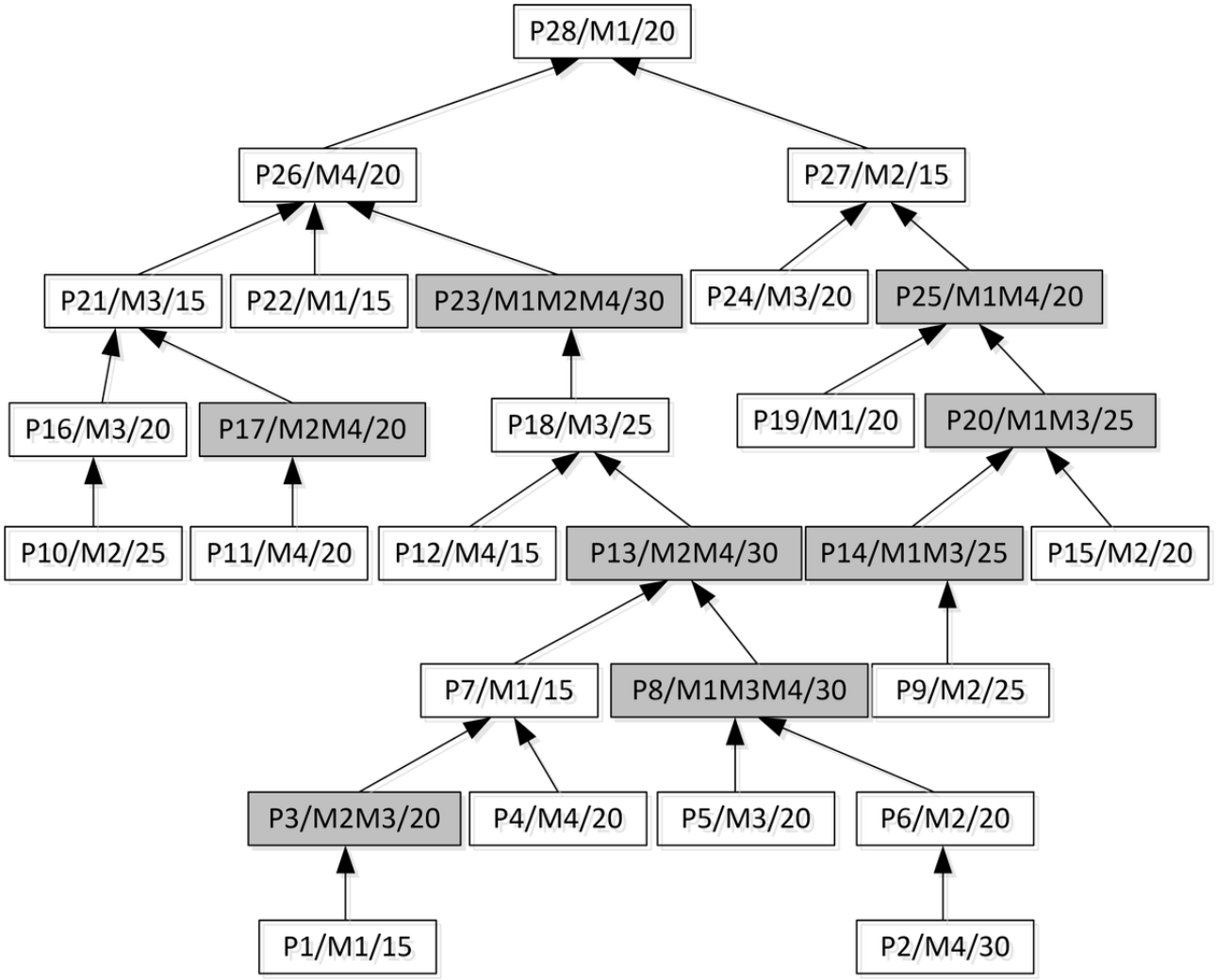


Figure 9

Process tree of product P

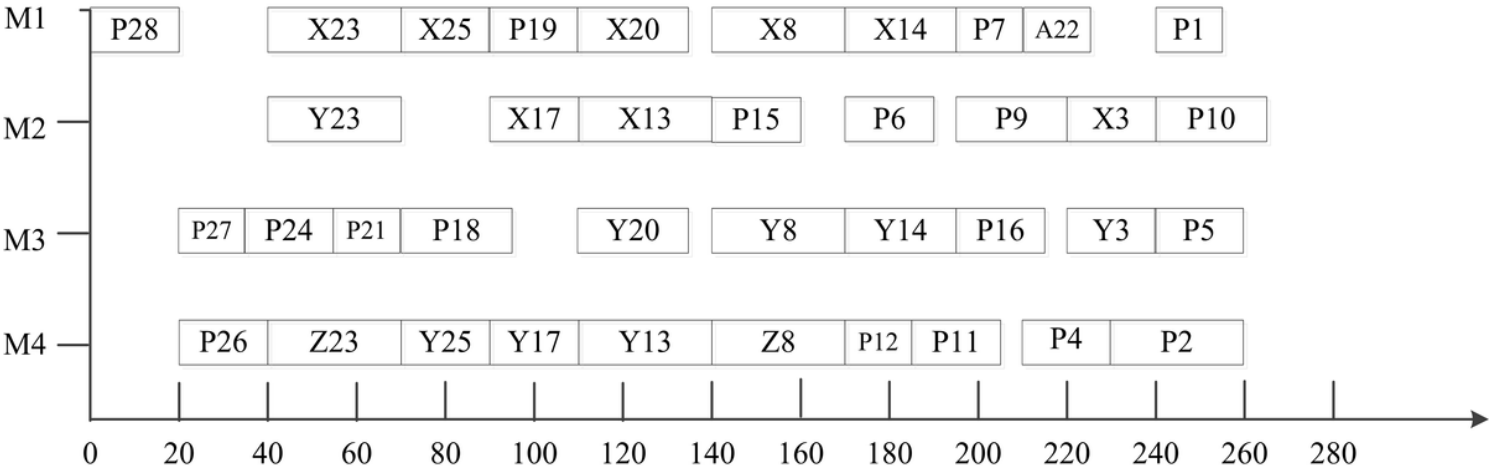


Figure 10

Gantt Chart of Algorithm Scheduling Results in Literature [13]

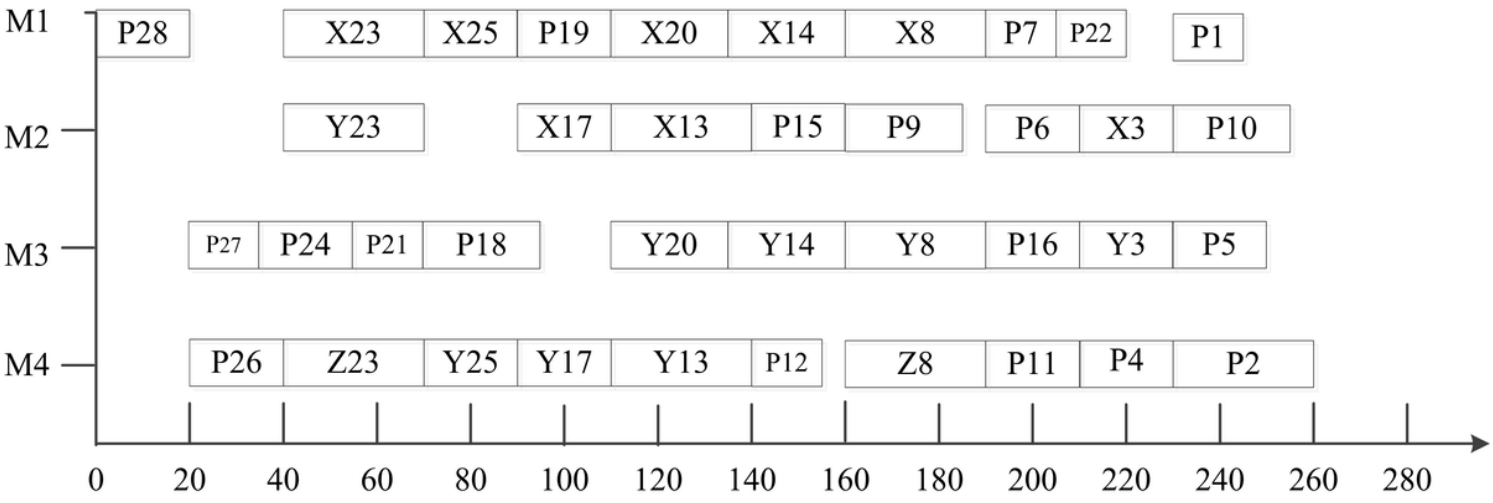


Figure 11

Gantt chart of scheduling results of the algorithm in this paper

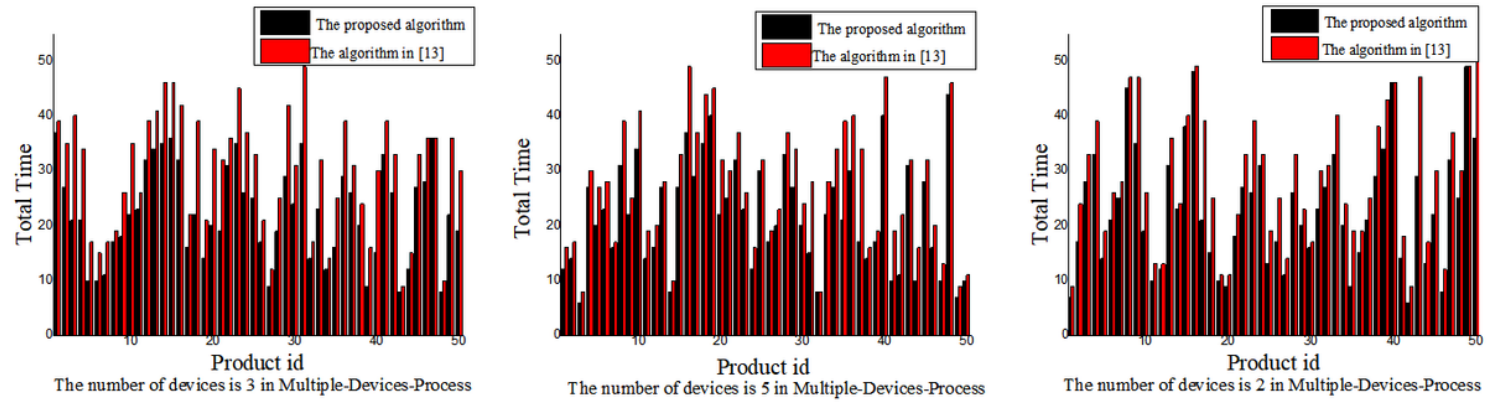


Figure 12

Experimental comparison diagram of the number of processes was less than 20 and the total processing time of the workpiece was less than 50

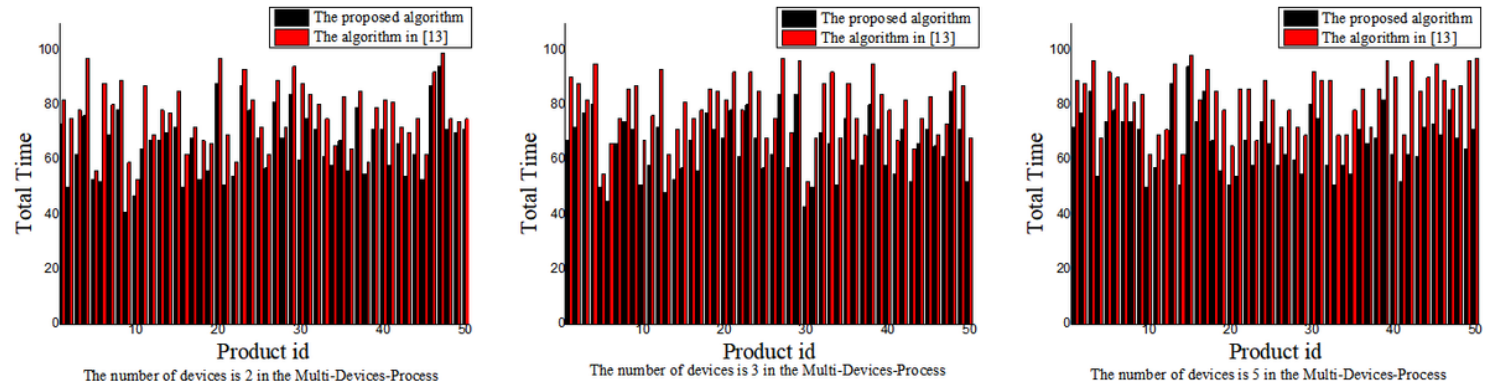


Figure 13

Experimental comparison diagram of the number of processes was between 20-30 and the total processing time of the workpiece was between 50-100

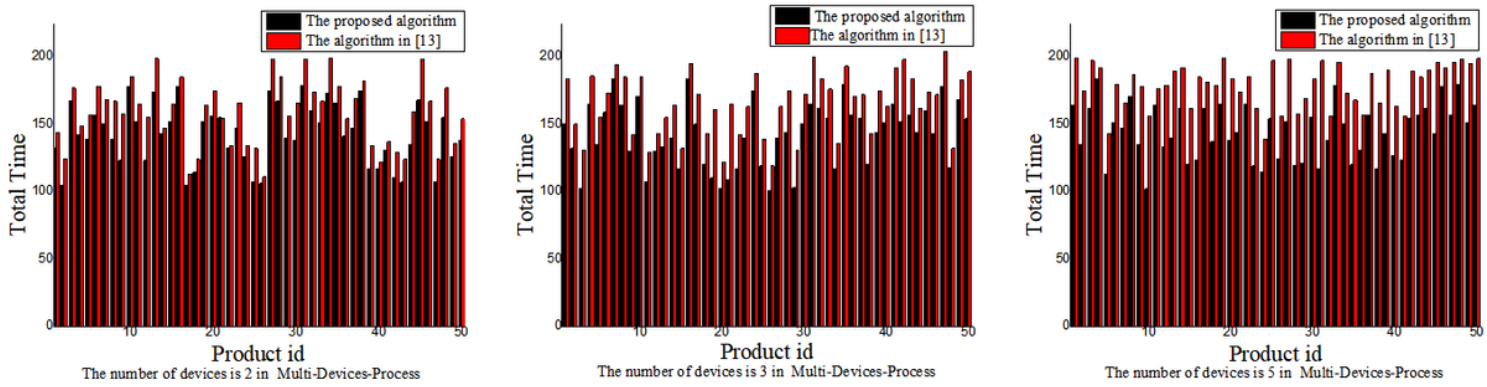


Figure 14

Experimental comparison diagram of the number of processes was between 30-50 and the total processing time of the workpiece was between 100-200

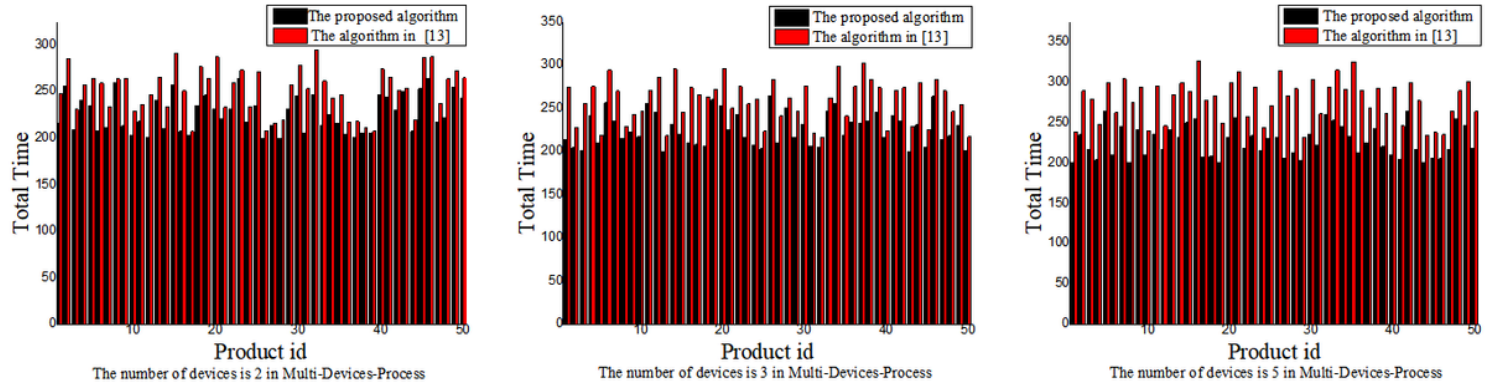


Figure 15

Experimental comparison diagram of the number of processes was between 50-60 and the total processing time of the workpiece was between 200-300