Hindawi
Complexity
Volume 2021, Article ID 5570575, 10 pages
https://doi.org/10.1155/2021/5570575

Research Article

An Improved Integrated Scheduling Algorithm with Process Sequence Time-Selective Strategy

Zhen Wang, Xiaohuan Zhang, and Gang Peng

School of Computer Science and Engineering, Huizhou University, Huizhou 516007, China

Correspondence should be addressed to Xiaohuan Zhang; zhangxiaohuan@hzu.edu.cn

Received 16 February 2021; Revised 25 February 2021; Accepted 27 February 2021; Published 9 March 2021

Academic Editor: Abd E.I.-Baset Hassanien

The integrated scheduling algorithm of process sequence time-selective strategy (ISAOPSTSS) is an advanced algorithm in the field of integrated scheduling. The proposed algorithm points out the shortcomings of the process sequence time-selective strategy. Generally, there are too many "trial scheduling" times. The authors propose that there is no need to make "trial scheduling" at every "quasi-scheduling time point." In fact, the process scheduling scheme can be obtained by trial scheduling on some "quasi-scheduling time points." The scheduling result is the same as that of the sequence timing strategy. The proposed algorithm reduces the runtime of scheduling algorithm and improves the performance of the algorithm without reducing the optimization effect.

1. Introduction

Scheduling is a key factor affecting the production efficiency of manufacturing industry. Effective scheduling optimization algorithm can maximize the production efficiency on the premise of satisfying the constraints of enterprise orders, equipment, and other hardware and software resources. At present, scholars have done a lot of research on the workshop scheduling problem, and most scheduling problems are mainly divided into job shop [1–10] and flow shop [11–19]. These algorithms are mainly for scheduling optimization when the workpiece is first processed and then assembled into the product. At present, consumers have more and more demand for individual products, and manufacturing factories will face more and more orders for multivariety and small batch products. This kind of situation if the production is still in processing after assembling according to the traditional mode of production would split the product internal parallel processing and assembly relations and reduce the production efficiency. In order to seek solutions for this new research field, Zhi-Qiang et al. [20] proposed an integrated scheduling algorithm that simultaneously promotes product processing and assembly, developed a series of scheduling optimization algorithms, and expanded many new research fields.

2. Review of Related Studies

At present, in the field of general integrated scheduling research, the following research studies have been mainly carried out.

Zhi-Qiang et al. [20] firstly pointed out the important position of critical path in the process tree and emphasized that the scheduling of processes with vertical relationship in the process tree is closely related to the final scheduling result. Yang et al. [21] put forward the strategy of layer priority, short time, long path, and dynamic adjustment and pointed out that adding the parallel relationship between processes with horizontal relationship in the process tree can make the scheduling result better. Yang et al. [22] pointed out that reference [20] paid attention to the vertical structure of product tree structure and ignored the horizontal parallel processing of the same equipment process; at the same time, it is pointed out that reference [21] focuses on the horizontal structure of the product tree structure and considers the vertical path on the basis of the horizontal layer. However, the strategy of emphasizing horizontal and neglecting vertical does not conform to the mechanism of vertical-oriented product scheduling. A scheduling scheme with vertical as the main factor and horizontal as the main factor is proposed. The advantages of the algorithm are as follows: on the basis

of both vertical and horizontal, the vertical scheduling is further optimized, which is in line with the idea that the integrated scheduling is mainly vertical. The disadvantages are as follows: although the dynamic critical path idea is used to solve the problem that the serial process and parallel process are pushed forward at the same time, the idea of the algorithm is too macro, and due to the restriction of some factors, it is impossible to consider the tightness between serial processes.

Xin et al. [23] determined the scheduling sequence of process according to the length of path according to the scheduling algorithm in document [20], and forms parallel 4 processing among groups, resulting in more idle time. An integrated scheduling algorithm based on device idle event driven is proposed. On the basis of reference [23], Xin et al. [24] proposed to further optimize the scheduling results by using rollback strategy to schedule the processes with long path of parent node first; the advantages are as follows: it increases the utilization rate of equipment, maximizes the "equipment busy" principle, reduces the idle time of equipment, and makes the process more compact. The disadvantage is that "device-driven events" always look for processes in the current schedulable parallel processes. From the perspective of processes, the algorithm can be regarded as an improved "layer first" scheduling under the "leaf alignment" mode. This algorithm increases the parallelism and the processing waiting time between the serial processes and ignores the impact of vertical scheduling optimization on the scheduling results.

To sum up, the current research can optimize the parallel scheduling of processes in products, but the scheduling optimization of serial processes needs to be improved while considering the parallelism. On the basis of the above research, Xie et al. [25] proposed a time-selective integrated scheduling algorithm considering the compactness of serial operations (ISAOPSTSS). The algorithm not only inherits the advantages of the current algorithm to ensure the parallelism between processes but also optimizes the compactness between serial processes on the basis of it and further emphasizes the scheduling idea of vertical optimization. It avoids the disadvantages of previous algorithms and optimizes the scheduling results. However, ISAOPSTSS in determining the scheduling scheme of the process is more complex, resulting in operation redundancy. This paper proposes an improved algorithm, which can reduce execution time of algorithm and improve the performance of the algorithm without reducing the optimization effect of the algorithm.

3. Problem Description and Analysis

The integrated scheduling problem is to study how to schedule the processes to minimize the product completion time when the product is in the production mode of assembly while processing. Among them, the researchers regard the processing and assembly of each process as a whole, collectively referred to as processing. The processing time, processing equipment, and partial order of each process in the product are clearly indicated by the product processing tree. The integrated dispatching shall meet the following requirements:

(1) Each process can only be processed on one machine

(2) Each time a machine can only process one process

(3) If and only if all the preprocesses of a process are in the state of finished processing (or no preprocesses), the process can be processed

(4) The processing of a certain process cannot be interrupted

(5) The difference between the processing end time of the latest finishing process and the processing start time of the earliest starting process is the total processing time of the product

4. Analysis and Design of Scheduling Strategy

4.1. Analysis of Improved Process Sequence Time-Selective Strategy. As shown in Figure 1, the reverse order process tree of product A proposed in ISAOPSTSS is analyzed as follows:

Step 1: apply the sequencing strategy of process sequence to sort the processes in product A. According to the product process tree as shown in Figure 2, first calculate the path length of all the leaf node processes, and the results were as follows: A10: 10, A9: 16, A5: 21, A8: 20, and A11: 9. Therefore, all nodes on the path where A5 is located are selected as the first process sequence, and these processes are added to the process queue Q_u. At this time, the sequence of processes in queue Q_u is A1, A2, A3, A4, and A5; at the same time, delete these processes in the process tree of product A. At this time, the processing tree of product A becomes a forest composed of multiple subtrees. Next, the path length of leaf nodes in these subtrees is calculated in turn, and the results were as follows: A10: 9, A9: 1, A8: 20, and A11: 8. Select all nodes on the path where A8 is located as the first process sequence, and add these processes to the process queue Q_u, and the sequence of processes in queue Q_u is A1, A2, A3, A4, A5, A6, A7, and A8, and these processes are deleted in the process tree of product A. By analogy, the sequence of processes in the process queue Q_u corresponding to the process tree of final product A is A1, A2, A3, A4, A5, A6, A7, A8, A10, A11, and A9, and this sequence will be used as the scheduling sequence of processes.

Step 2: schedule the longest operation sequence in the queue Q_u to the previous operation to form the initial scheduling scheme, as shown in Figure 3.

Step 3: the improved sequencing strategy of process sequence is used to schedule the remaining processes in the process queue Q_u in turn. First, the process A6 is scheduled. The earliest start processing time of A6 is 1, and the processing device is $M3$. Because A2 and A5 have been scheduled on $M3$ device. The "quasi-scheduling time points" of A6 are the earliest start processing time 1, the end processing time 4 of process A2, and the end

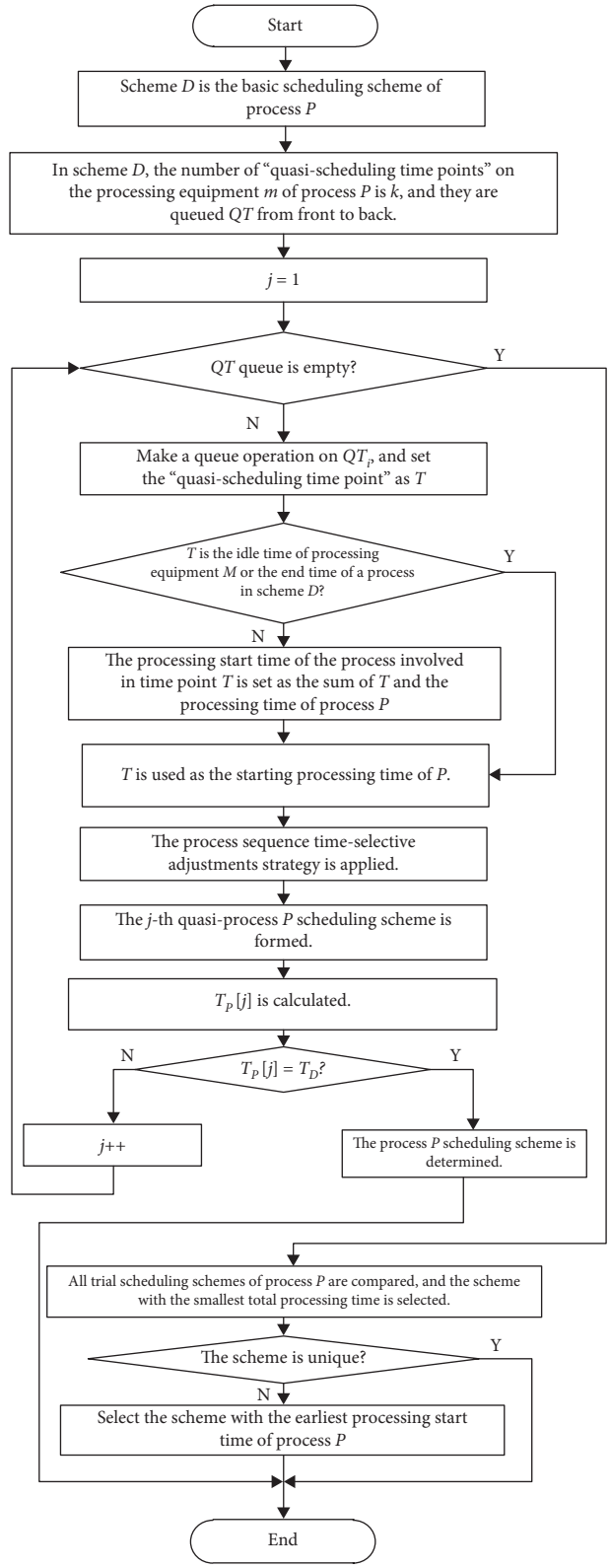

Figure 1: The flowchart of the process sequence time-selective scheduling strategy.

processing time 21 of process A5. Try to schedule A6 at these three time points, and get three trial scheduling schemes as shown in Figure 4.

In the A6 "quasi-scheduling scheme" shown in Figure 4, ISAOPSTSS calculated the total processing time of each "quasi-process scheduling scheme" formed at each "quasi-scheduling time point" and the scheme shown in Figure 4(b)

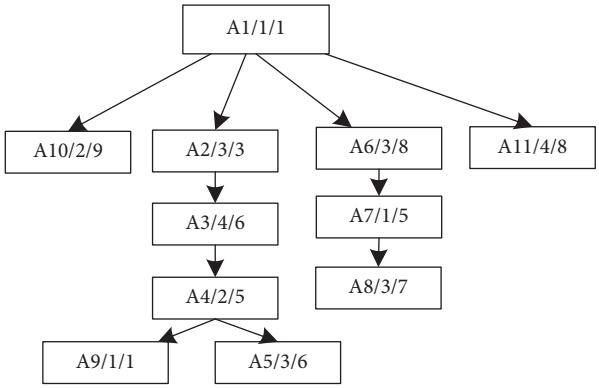

Figure 2: Process tree of product A.

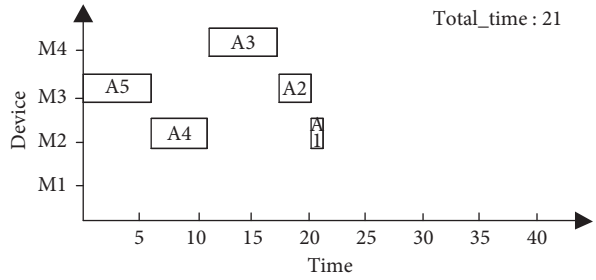

Figure 3: Gantt chart of initial scheduling scheme for product A.

is selected as the scheduling scheme of A6. scheme with the least total processing time as the A6 scheduling scheme.

In fact, according to the characteristics of the process sequence time-selective algorithm, it is not necessary to calculate the total processing time of each "quasi-process scheduling scheme" formed on each "quasi-scheduling time point."

First, the total processing time of the current basic scheduling scheme is T_D, and the processing time of the current scheduling process is t. The total processing time of the current process P scheduling is T_p.

$$T_D \leq T_p \leq T_D + t. \tag{1}$$

According to formula (1), the total processing time of the current process P quasi-scheduling scheme is T_p, the minimum processing time is T_D, and the maximum processing time is the sum of the total processing time t of the current basic scheduling scheme and the processing time t of the current scheduling process. It can be seen that if the "quasi-scheduling time point" is used to schedule the processes in the order from front to back and if the total processing time of a quasi-process scheduling scheme is T_D, it can be determined that the quasi-process scheduling scheme on the quasi-scheduling time point is the current process scheduling scheme, and there is no need to calculate the total processing time of the quasi-process scheduling scheme on the quasi-scheduling time point after the time point; of course, the total processing time T_D of the "quasi-operation scheduling scheme" generated from the "quasi-scheduling time point" is equal to the total processing time T_D of the basic scheduling scheme; however, due to the fact that the "quasi-scheduling time point" is scheduled in the order from the front to the back in the trial scheduling process, the former "quasi-scheduling time point" is better than the latter in the case of the same total processing time from the perspective of interprocess compactness.

Therefore, the current process scheduling scheme will be discussed in the following two situations:

(1) If the total processing time of the "quasi-process scheduling scheme" generated at the current "quasi-scheduling time point" is greater than that of the current basic scheduling scheme, the total processing time of the "quasi-process scheduling scheme" generated at the next "quasi-scheduling time point" will continue to be determined.

(2) If the total processing time of the "quasi-process scheduling scheme" generated from the current "quasi-scheduling time point" is equal to the total processing time of the current basic scheduling scheme, the calculation of the "quasi-scheduling time point" behind will be stopped, and the "quasi-process scheduling scheme" generated from the current "quasi-scheduling time point" will be taken as the current process scheduling scheme.

4.2. Algorithm Design of Improved Process Sequence Time-Selective Scheduling Strategy. The specific steps of the improved algorithm are as follows:

Step 1: set the basic scheduling scheme of process P as D.

Step 2: in scheme D, k "quasi-scheduling time points" on the processing equipment m of process P are found, and they are queued QT from front to back, $j = 1$.

Step 3: judge whether the QT queue is empty. If it is not empty, make a queue operation on the QTi queue. Take out the "quasi-scheduling time point" T and go to step 4. If it is empty, go to step 12.

Step 4: judge whether T is the idle time of processing equipment M or the end time of a process in scheme D. If not, go to step 5 and if so, go to step 6.

Step 5: the processing start time of the process involved in time point T (the process being processed at time point T) is set as the sum of T and the processing time of process P.

Step 6: time point T is used as the starting processing time of process P to schedule process P.

Step 7: the process sequence time-selective adjustment strategy [25] is used to adjust the process affected by scheduling process P, forming the j-th quasi-process P scheduling scheme.

Step 8: the total processing time $T_P[j]$ of the j-th quasi-process P scheduling scheme is calculated.

Step 9: judge whether $T_P[j] = T_D$ is true. If yes, go to step 10; otherwise, go to step 11.

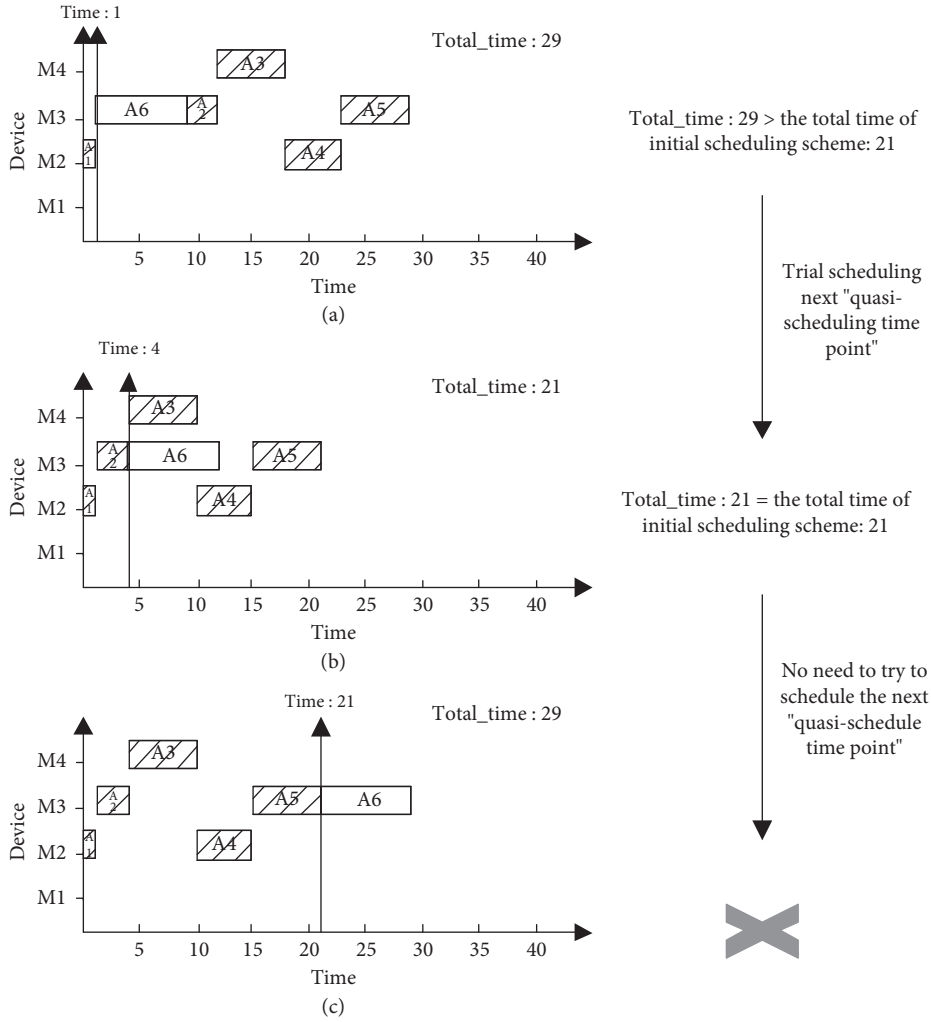

Figure 4: Gantt chart of each trial scheduling scheme of process A6. (a) The scheduling scheme is obtained by trial scheduling at quasi-scheduling time point "1." (b) The scheduling scheme is obtained by trial scheduling at quasi-scheduling time point "4." (c) The scheduling scheme is obtained by trial scheduling at quasi-scheduling time point "21."

Step 10: the scheduling scheme obtained at time point T is taken as the scheduling scheme of process P, and go to step 14.

Step 11: j++, go to Step 3.

Step 12: the total processing time of j quasi-process P scheduling schemes is compared, and the scheme with the smallest total processing time is selected.

Step 13: judge whether the scheme is unique. If it is unique, select it. If it is not unique, select the scheme with the earliest processing start time of process P.

Step 14: exit.

The flowchart of the process sequence time-selective scheduling strategy is shown in Figure 1.

5. Algorithm Design

The implementation steps of the improved algorithm are as follows:

Step 1: the reverse order processing tree is obtained by reversing the partial order relationship of processing processes in the processing tree.

Step 2: the operation queue Q_u is obtained by using the process sequence sorting strategy.

Step 3: all processes in the longest process sequence on Q_u are queued and scheduled to form the initial scheduling scheme P_0.

Step 4: $i = 1$.

Step 5: judge whether Q_u is empty. If it is empty, go to step 8; otherwise, go to step 6.

Step 6: the queue Q_u is queued to obtain the current scheduling process P; the processing time is t, and the processing equipment is M.

Step 7: the improved process sequence time-selective strategy is applied to schedule P, and the process P scheduling scheme is obtained; i++, go to step 5.

Step 8: form product scheduling Gantt chart and output.

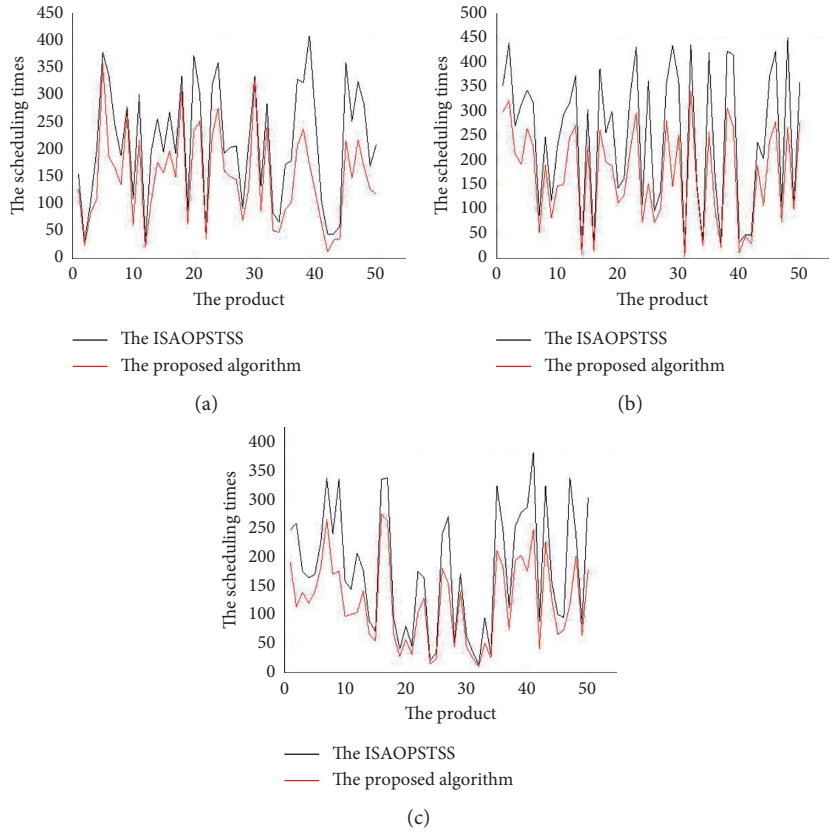

Figure 5: Comparison of scheduling times when the total number of processes is 30. (a) The scheduling times of 30 processes in 3 devices. (b) The scheduling times of 30 processes in 6 devices. (c) The scheduling times of 30 processes in 9 devices.

6. Experimental Method

In order to verify the performance of the algorithm, a group of experiments is designed for comparison. The algorithm performance is affected by the structure of product process tree, the number of processing equipment, and the processing time. Therefore, the proposed algorithm is compared with the ISAOPSTSS algorithm from the perspective of different scale parameters. First, each experiment will randomly generate 50 products. The parameters in the process tree are randomly generated. The parameters are as follows: the structure of the process tree (including the total number of layers of the process tree, the number of processes in each layer, and the relationship between the front and back of the process), the processing time and equipment number of the process in the process tree, and the total number of processing equipment of the process. Each group of experiments will randomly generate 50 products because the product structure is random, which can effectively prove the effectiveness of the algorithm in different cases. The above two algorithms are implemented in C++ language by dev c++ 4.9.9.2. Schedule the randomly generated product process trees, set counters in the two algorithms to monitor the times of "trial scheduling" in each algorithm, and record them for comparison. Since the algorithm proposed in this paper is an improvement on the algorithm proposed in ISAOPSTSS, if the number of "trial scheduling" of the algorithm proposed in this paper is less than that in ISAOPSTSS under different parameter conditions, the effectiveness of the algorithm proposed in this paper can be proved.

7. Results and Discussion

Six groups of experiments were designed as follows. In Experiment 1, as shown in Figure 5, the proposed algorithm is compared with the algorithm in ISAOPSTSS. In order to compare, 50 product process trees were randomly generated, the total number of processes is 30, and the total number of processing equipment is 3, 6, and 9, respectively. In Experiment 2, as shown in Figure 6, the proposed algorithm is compared with the algorithm in ISAOPSTSS. The comparison data are used to randomly generate 50 groups, the total number of processes is 50, and the total number of processing equipment is 3, 6, and 9, respectively. In Experiment 3, as shown in Figure 7, the proposed algorithm is compared with the algorithm in ISAOPSTSS. The comparison data are used to randomly generate 50 groups, the total number of processes is 80, and the total number of processing equipment is 3, 6, and 9, respectively. In Experiment 4, as shown in Figure 8, the proposed algorithm is compared with the algorithm in ISAOPSTSS. The comparison data are used to randomly generate 50 groups, the total number of processes is 100, and the total number of processing equipment is 3, 6, and 9, respectively. As shown in Figure 9, Experiment 5 shows the average scheduling

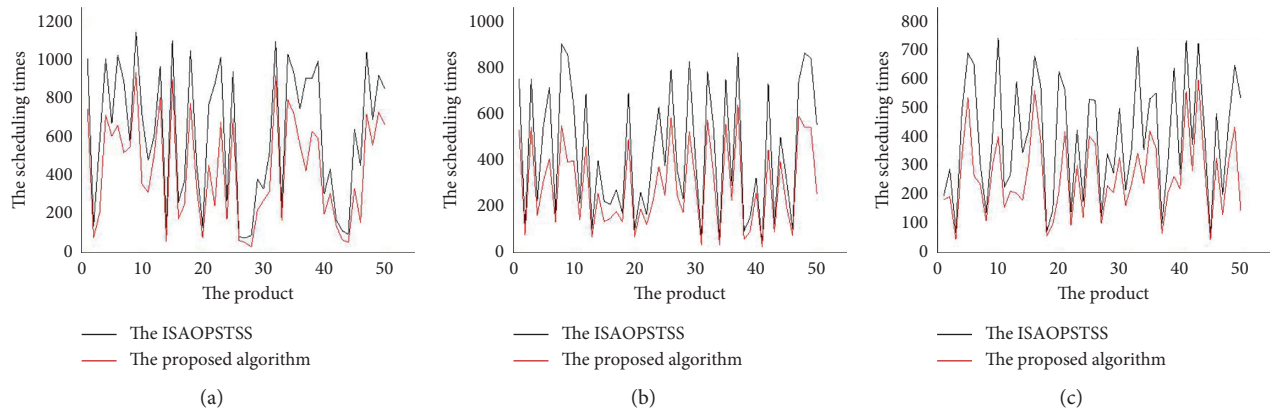

Figure 6: Comparison of scheduling times when the total number of processes is 50. (a) The scheduling times of 50 processes in 3 devices. (b) The scheduling times of 50 processes in 6 devices. (c) The scheduling times of 50 processes in 9 devices.

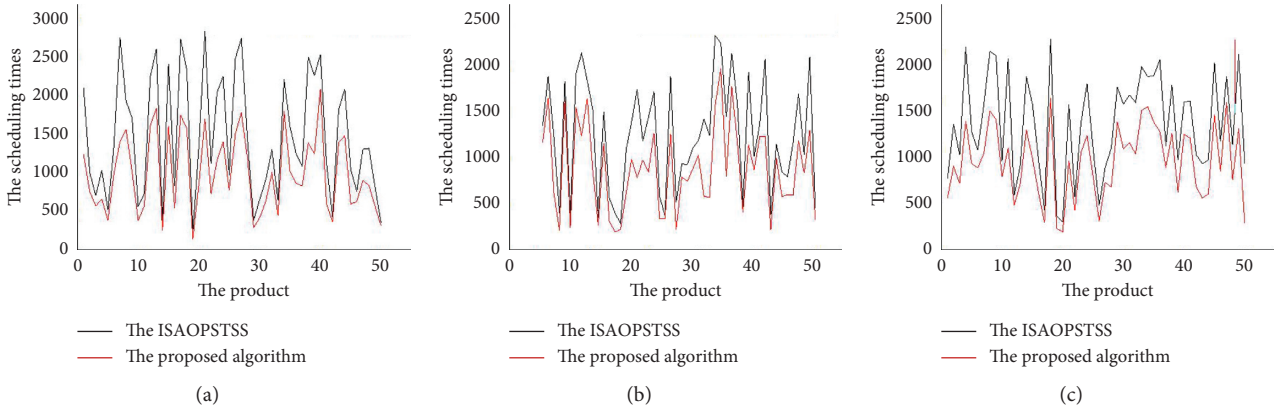

Figure 7: Comparison of scheduling times when the total number of processes is 80. (a) The scheduling times of 80 processes in 3 devices. (b) The scheduling times of 80 processes in 6 devices. (c) The scheduling times of 80 processes in 9 devices.

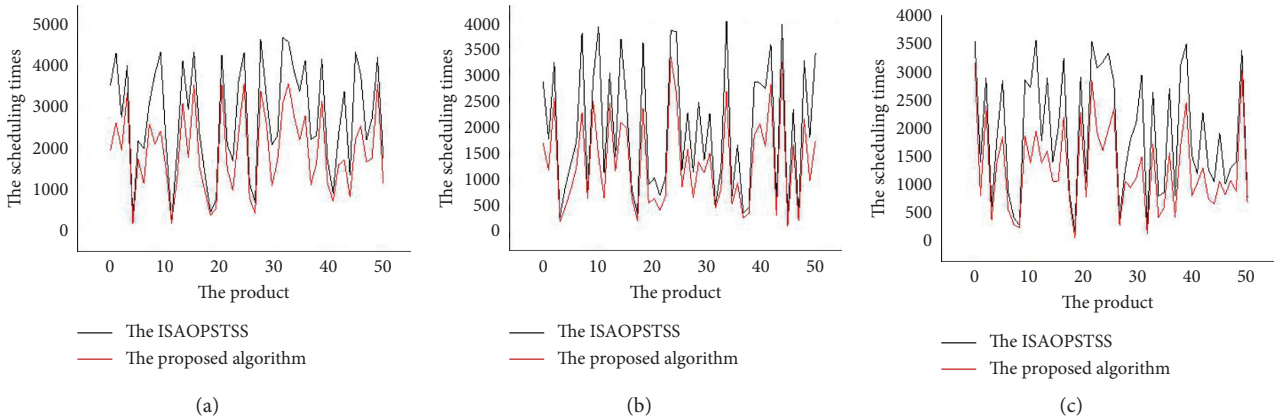

Figure 8: Comparison of scheduling times when the total number of processes is 100. (a) The scheduling times of 100 processes in 3 devices. (b) The scheduling times of 100 processes in 6 devices. (c) The scheduling times of 100 processes in 9 devices.

times comparison between the proposed algorithm and ISAOPSTSS algorithm when the total number of processes is 30, 50, 80, and 100, respectively. As shown in Figure 10, Experiment 6 shows that in Experiment 1, when the total number of devices is 3, 6, and 9, respectively, the average scheduling times of the proposed algorithm are compared with those of ISAOPSTSS.

Analysis of the above experimental data shows that the number of trial scheduling times of the algorithm in this paper is significantly reduced compared with the number of

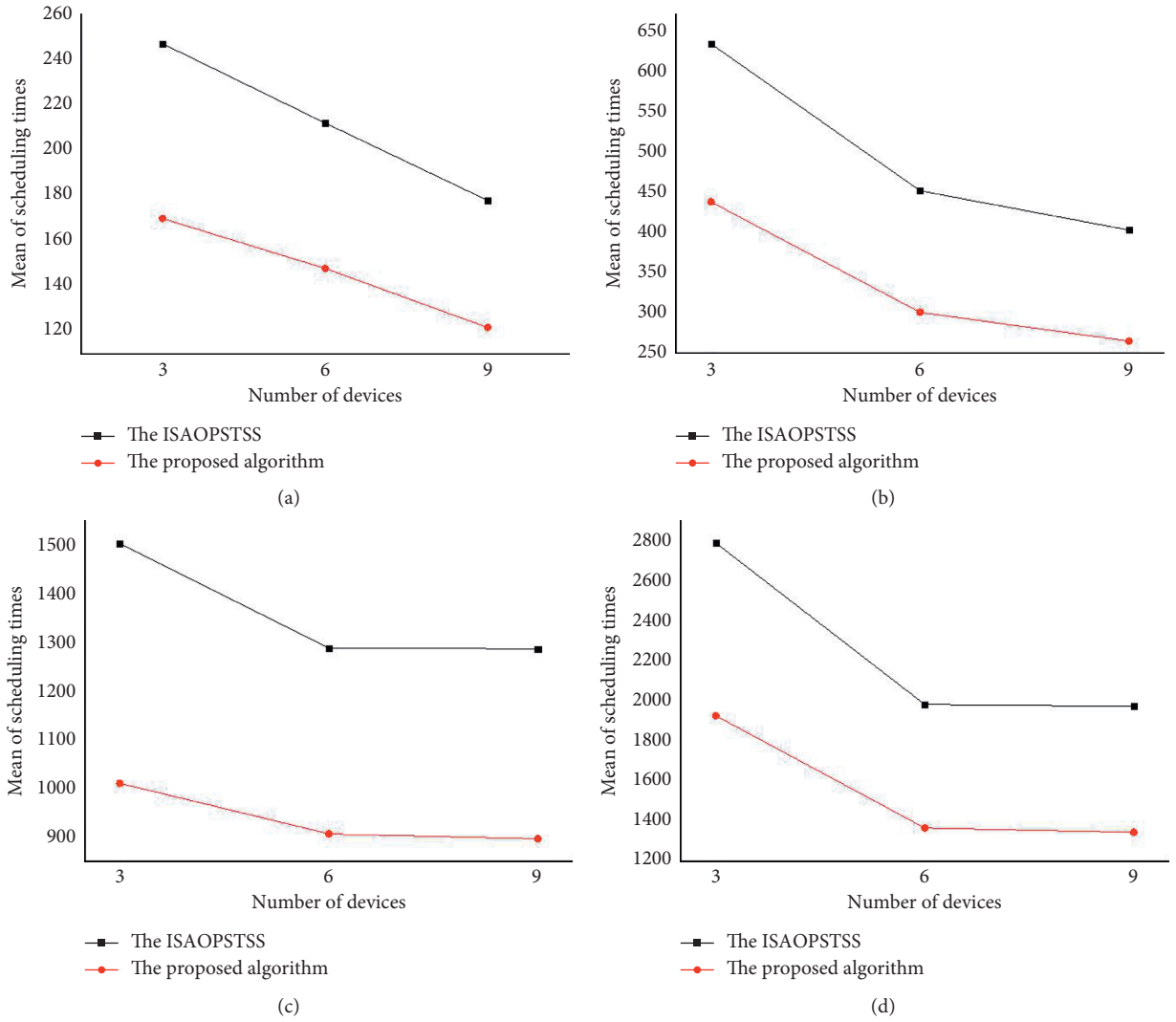

Figure 9: The comparison chart of the average scheduling times when the total number of processes is 30, 50, 80, and 100, respectively. (a) Comparison chart of mean scheduling times when the number of processes is 30. (b) Comparison chart of mean scheduling times when the number of processes is 50. (c) Comparison chart of mean scheduling times when the number of processes is 80. (d) Comparison chart of mean scheduling times when the number of processes is 100.

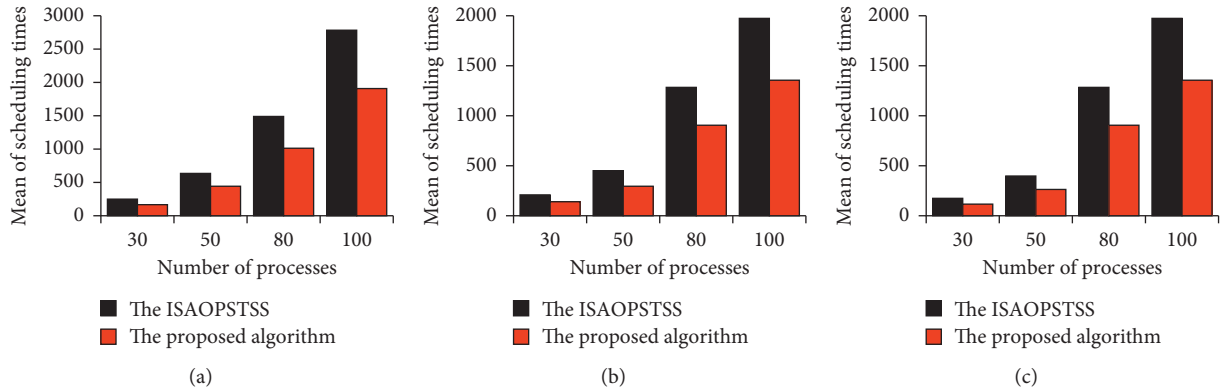

Figure 10: The comparison chart of the average scheduling times when the total number of devices is 3, 6, and 9, respectively. (a) Comparison chart of mean scheduling times when the number of devices is 3. (b) Comparison chart of mean scheduling times when the number of devices is 6. (c) Comparison chart of mean scheduling times when the number of devices is 9.

pilot scheduling in ISAOPSTSS, and the reduction ratio is about 30%.

8. Conclusions

The algorithm proposed in this paper is a further optimization of the operation sequence timing algorithm. On the premise of ensuring the optimization results, it simplifies the algorithm steps, reduces the algorithm execution time, and improves the algorithm performance. At present, the ISAOPSTSS algorithm has been applied in batch processing scheduling and two-job-shop scheduling and other fields. It may be the next step to apply the proposed algorithm in these fields to improve the performance of the algorithm.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

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

References

[1] Z. Cao, L. Zhou, B. Hu, and C. Lin, "An adaptive scheduling algorithm for dynamic jobs for dealing with the flexible job shop scheduling problem," *Business & Information Systems Engineering*, vol. 61, no. 3, pp. 299–309, 2019.

[2] P. Pongchairerks, "A two-level metaheuristic algorithm for the job-shop scheduling problem," *Complexity*, vol. 2019, Article ID 8683472, 11 pages, 2019.

[3] A. Jamili, "Job shop scheduling with consideration of floating breaking times under uncertainty," *Engineering Applications of Artificial Intelligence*, vol. 78, pp. 28–36, 2019.

[4] J. Lin, "Backtracking search based hyper-heuristic for the flexible job-shop scheduling problem with fuzzy processing time," *Engineering Applications of Artificial Intelligence*, vol. 77, pp. 186–196, 2019.

[5] Z. C. Li, B. Qian, R. Hu, L. L. Chang, and J. B. Yang, "An elitist nondominated sorting hybrid algorithm for multi-objective flexible job-shop scheduling problem with sequence-dependent setups," *Knowledge-Based Systems*, vol. 173, pp. 83–112, 2019.

[6] K. Gao, F. Yang, M. Zhou, Q. Pan, and P. N. Suganthan, "Flexible job-shop rescheduling for new job insertion by using discrete jaya algorithm," *IEEE Transactions on Cybernetics*, vol. 49, no. 5, pp. 1944–1955, 2019.

[7] L. Sun, L. Lin, M. Gen et al., "A hybrid cooperative coevolution algorithm for fuzzy flexible job shop scheduling," *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 5, p. 1, 2019.

[8] B. Wang, H. Xie, X. Xia et al., "A NSGA-II algorithm hybridizing local simulated-annealing operators for a Bi-criteria robust job-shop scheduling problem under scenarios," *IEEE Transactions on Fuzzy Systems*, vol. 27, 2018.

[9] M. M. Ahmadian, A. Salehipour, and T. C. E. Cheng, "A meta-heuristic to solve the just-in-time job-shop scheduling problem," *European Journal of Operational Research*, vol. 288, 2020.

[10] M. Abedi, R. Chiong, N. Noman et al., "A multi-population, multi-objective memetic algorithm for energy-efficient job-shop scheduling with deteriorating machines," *Expert Systems with Applications*, vol. 157, Article ID 113348, 2020.

[11] M. Qin, R. Wang, Z. Shi et al., "A genetic programming-based scheduling approach for hybrid flow shop with a batch processor and waiting time constraint," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 99, pp. 1–12, 2019.

[12] D. Ferone, S. Hatami, E. M. GonzálezkMeira et al., "A biased-randomized iterated local search for the distributed assembly permutation flow hop problem," *International Transactions in Operational Research*, vol. 27, no. 3, 2020.

[13] S. Aqil and K. Allali, "Local search metaheuristic for solving hybrid flow shop problem in slabs and beams manufacturing," *Expert Systems with Applications*, vol. 162, Article ID 113716, 2020.

[14] P. Valledor, A. Gomez, P. Priore et al., "Modelling and solving rescheduling problems in dynamic permutation flow shop environments," *Complexity*, vol. 2020, Article ID 2862186, 17 pages, 2020.

[15] M. Fazayeli, M. R. Aleagha, R. Bashirzadeh et al., "A hybrid meta-heuristic algorithm for flowshop robust scheduling under machine breakdown uncertainty," *International Journal of Computer Integrated Manufacturing*, vol. 29, pp. 1–11, 2016.

[16] A. Hasani and S. M. H. Hosseini, "A bi-objective flexible flow shop scheduling problem with machine-dependent processing stages: trade-off between production costs and energy consumption," *Applied Mathematics and Computation*, vol. 2020, p. 386, Article ID 125533, 2020.

[17] K. Geng, L. Liu, C. Ye et al., "Bi-objective re-entrant hybrid flow shop scheduling considering energy consumption cost under time-of-use electricity tariffs," *Complexity*, vol. 2020, Article ID 8565921, 17 pages, 2020.

[18] A. Costa, F. V. Cappadonna, and S. Fichera, "Minimizing makespan in a flow shop sequence dependent group scheduling problem with blocking constraint," *Engineering Applications of Artificial Intelligence*, vol. 89, Article ID 103413, 2020.

[19] M. S. Nagano, J. V. S. Robazzi, and C. P. Tomazella, "An improved lower bound for the blocking permutation flow shop with total completion time criterion," *Computers & Industrial Engineering*, vol. 146, Article ID 106511, 2020.

[20] X. Zhi-Qiang, L. Sheng-Hui, and P.-L. Qiao, "Dynamic job-shop scheduling algorithm based on ACPM and BFSM," *Journal of Computer Research and Development*, vol. 40, no. 7, pp. 977–983, 2003.

[21] Z. Xie, J. Yang, G. Yang et al., "Dynamic job-shop scheduling algorithm with dynamic set of operation having priority," *Chinese Journal of Computers*, vol. 31, no. 3, pp. 502–508, 2008.

[22] Z.-Q. Xie, J. Yang, Y. Zhou, D.-L. Zhang, and G.-Y. Tan, "Dynamic critical paths multi-product manufacturing scheduling algorithm based on operation set," *Chinese Journal of Computers*, vol. 34, no. 2, pp. 406–412, 2011.

[23] Z. Xie, Y. Xin, and J. Yang, "Integrated scheduling algorithm based on event-driven by machines' idle," *Journal of Mechanical Engineering*, vol. 47, no. 11, pp. 139–147, 2011.

[24] Z. Xie, Y. Xin, and J. Yang, "Machine-driven integrated scheduling algorithm with rollback- preemptive," *Automatica Sinica*, vol. 37, no. 11, pp. 1332–1343, 2011.

[25] Z. Xie, X. Zhang, Y. Gao, and Y. Xin, "Time-selective integrated scheduling algorithm considering the compactness of serial processes," *Journal of Mechanical Engineering*, vol. 54, no. 6, pp. 191–202, 2018.