Multi-tasking multi-machine scheduling system for multi-stage multi-criteria production

Shao-Chen Liu[1], *Yu-Ren* Chen[1], *Cheng-Ju* Kuo[1], *Tzu-Yu* Lin[1], *Kuo-Cheng* Ting[1], *Don-Lin* Yang[1,*], *Hsi-Min* Chen[1], and *Yi-Chung* Chen[2]

[1]Department of Information Engineering and Computer Science, Feng Chia University, Taichung, Taiwan
[2]Department of Industrial Engineering and Management, National Yunlin University of Science and Technology, Yunlin, Taiwan

Abstract. In recent years, many major manufacturers have been incorporating Industry 4.0 technologies such as preventive fault detection, automated scheduling algorithms, and component management to increase productivity and reduce production costs. Achieving this objective requires a substantial amount of working capital to acquire large quantities of new machinery, equipment to extract data from the machinery, and high-priced big data analysis software. However, most factories in the world are small- or medium-sized companies and have not enough capital to replace their machinery or purchase big data analysis software. It is therefore almost impossible for these factories to reach the goal of Industry 4.0. Furthermore, most of the conventional automated production scheduling methods only consider a single criterion in scheduling, which is not applicable for actual situations. This study therefore proposed a multi-tasking multi-machine scheduling system for multi-stage multi-criteria production to address various shortcomings in existing methods. To achieve this goal, we proposed a novel concept based on skyline queries to assist in the scheduling process. Also, a data structure of "heap" is applied in this work to accelerate the scheduling process. The experimental results demonstrated the validity of the proposed approach.

1 Introduction

Put simply, Industry 4.0 involves the use of large quantities of machinery data, sensors, the Internet of Things, big data analysis, and collaboration between man and machine to increase the overall productivity of a factory. To achieve this objective, factories must spend a substantial amount of working capital to install new machinery, extract data from the machinery, and use high-priced big data analysis software for scheduling and production. However, most factories in Taiwan are small- or medium-sized companies and have not enough capital to replace their machinery or purchase big data analysis software. It is therefore almost impossible for these factories to reach the goal of Industry 4.0.

In consideration of this, many researchers have been developing inexpensive ways to help these manufacturers achieve Industry 4.0. For instance, Kuo *et al.* [4][9] proposed

[*]Corresponding author:dlyang.tw@gmail.com

methods that combine machine maintenance personnel experience, add-on triaxial accelerometers, and artificial neural networks for machinery fault detection and successfully applied these methods to spring factories. Kuo *et al.* [5]successfully developed an RFID-based component management system that records what components a machine contains and what processing task it is performing, thereby greatly increasing component management efficiency. However, a comprehensive view of all relevant research revealed that only these two types of cost reduction techniques were discussed. No discussion has been made on automated scheduling under the Industry 4.0 model, so some discussion is warranted to address this issue.

Generally speaking, optimizing the schedules of work orders is one of the most effective ways to increase a factory's production efficiency. A good schedule can efficiently allocate various resources, increase factory productivity, and enable machinery to work continuously in the absence of any major malfunctions. It is because of this advantage that researchers have actively conducted studies on scheduling in the past. Common scheduling methods can be divided into single-machine scheduling and multi-machine scheduling depending on the number of machines. [3][6][8] are examples of single-machine scheduling methods that can design schedules for the shortest work time or shortest lead time. In contrast, multi-machine scheduling methods can make schedules based on two concepts: job shops and flow shops. However, we have found that most of the above methods only considered a single criterion to select suitable scheduling results. In other words, they mixed multiple criteria into a single score and then chose the schedule with the highest score. The major issue is that the results will not be able to adapt to various situations in a factory. Suppose we only consider total work time and delivery delay time, and each of the criteria contributes to 50% of the scheduling score. The total work time of Schedule A is 50 hours (contributing 0.5 points to the scheduling score), and the delivery delay time is 1 hour (contributing 0.5 points to the scheduling score). In contrast, the total work time of Schedule B is 40 hours (contributing 0.7 points to the scheduling score), and the delivery delay time is 5 hours (contributing 0.35 points to the scheduling score). A conventional scheduling algorithm would give Schedule B as the solution because it has a higher total score than Schedule A. However, we must understand that such a solution may be reasonable in some circumstances but unreasonable in others. Even in the same factory, there may be different considerations at different times. If the customer is more easygoing, and a delay of 1 hour or 5 hours makes no difference, then the factory may choose Schedule B because it requires less work time. In contrast, if the customer places a lot of emphasis on the delivery time, then the factory will have to choose Schedule A over Schedule B because they can deliver the order faster. This example makes it clear that even in the same factory, Schedules A and B are both possible choices, so both should be presented to factory personnel for consideration rather than just providing Schedule B. Since the conventional scheduling method has a serious flaw, new methods are needed to address it.

The existing method that is closest to the scheduling approach which we propose is the multi-tasking multi-machine scheduling algorithm for single-stage multi-criteria production presented by Kuo *et al.* [9].The method that they proposed can efficiently assist factories in scheduling single-stage work orders and is particularly helpful for processes with only a single stage, such as the manufacturing of springs or screws. However, it cannot be applied to factories with multi-stage manufacturing processes, such as household appliance factories or semiconductor fabs. This study therefore developed a multi-tasking multi-machine scheduling algorithm for multi-stage multi-criteria production to resolve this issue.

When developing the target algorithm, we also found some parts of conventional scheduling methods that were unsuitable for Industry 4.0 and needed to be overcome. During computation, most conventional scheduling algorithms assume that the execution time for each work order is precise and known. For instance, the execution time can be obtained from the SOP operation manual of each component or the past experience of the scheduling personnel. However, these SOP operation manuals or experiences came from the Industry 3.0 era or older, when all components followed standard specifications and could only be used in mass production. The Industry 4.0 era emphasizes component specialization and small-scale production, so many components are only manufactured in several batches. We therefore cannot accurately estimate the production time of each component based on previous statistics. In response to this, we used add-on triaxial accelerometers to understand machine conditions at each time point and then determine the amount of production time that is needed for each component based on said conditions. Ultimately, we can optimize scheduling based on the time. Experiments demonstrated the validity of the proposed approach.

The remainder of this paper is arranged as follows. Section 2 introduces related works, Section 3 presents the methodology used in this study, Section 4 explains the experiment results, and Section 5 contains the conclusions of this study and suggestions for future research.

2 Related works

2.1 Sensors commonly used in machinery

Previous research [9] shows that the most common physical features in machinery are motion, vibration, noise, and temperature. To detect these features, we can use triaxial accelerometers (for motion and vibration), microphones (for noise), and thermocouples (for temperature). To simplify the problem, we used only triaxial accelerometers to detect the features in machinery, but the approach that we applied to triaxial accelerometers can be easily applied to other types of sensors.

2.2 Scheduling algorithms

A good scheduling algorithm can efficiently allocate resources, arrange work orders, and increase the production efficiency of a factory. Previous research [9] mentioned that scheduling algorithms can be viewed from three aspects. First, in terms of the objective, scheduling algorithms can be divided into process types and production types. Second, in terms of product structure, they can be divided into one-piece products or assembly products. Finally, in terms of the number of machines, they can be divided into single-machine operations or multi-machine operations [9].

2.3 Skyline queries

In recent years, skyline queries [1][2][7] have become a part of decision-making processes in database systems. Aside from identifying data points with better performance in only a few dimensions out of many, skyline queries can also make queries based on the needs and preferences of different users. Say for example that a user wants to take a trip to the beach and wants to stay at a hotel that is closer to the beach or has lower room rates. The available hotel information is as shown in Table 1. Skyline queries can be used to find the hotels that best fit the user's needs and preferences. Let us map the data in Table 1 onto a graph with the distance from the beach as the X axis and the room rate as the Y axis, as shown in Fig. 1. If we compare Hotel A and Hotel B, the former is closer to the beach and less expensive than the latter (which we simply put as the former dominating the latter), so the user will choose Hotel A rather than Hotel B. In this case, the information of Hotel B

Table 1. An example of hotel database.

Name	Cost	Distance
A	10	1
B	12	1.7
C	10	2.5
D	4	3.5
E	8	3.2
F	6	2
G	14	3

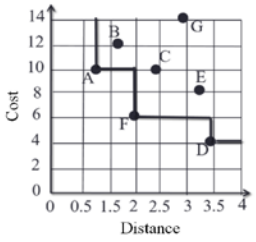

Fig. 1. A graph representation of Table 1.

does not need to be provided to the user as an option. In contrast, if we compare Hotel A and Hotel D, we will find that the two are incomparable; Hotel A is closer to the beach, but Hotel D has a lower room rate. Either is a possible choice for the user, so both should be recommended to the user. The key of skyline queries is to find data points that are not dominated by any other data points in the dataset. In Fig. 1, for example Hotels A, D, and F are the skyline query results because they are not dominated by any other points.

3 Methodology

This section introduces the methodology of this study, including data collection, time analysis, and the scheduling approach.

3.1 Data collection

This study mainly used triaxial accelerometers. During the study, we collected acceleration values along the X, Y, and Z axes and used them to measure the motions and vibrations of the machinery. Figure 2 presents an actual example of the collected data along the X axis. The horizontal axis presents the number of data items, while the vertical axis presents acceleration in m/s^2. Note that the values may contain positive as well as negative values because the sensors have fronts and backs; accelerations in different directions are opposite to each other in terms of positivity and negativity. We can also see from Fig. 2 that aside from the motion information of the machine, the data collected from a triaxial accelerometer also contains a lot of noise. We therefore employed a band pass filter to filter out the high-frequency and low-frequency noise.

3.2 Time analysis

After collecting acceleration signals from the machines, we can then use the signals to determine the precise amount of time that the machine needs to manufacture a piece of each type of component. The operation method is divided into two parts. The first part is windowing the acceleration signals and extracting the feature values of each windowed signal. The second part is finding the cycle of the feature values of the windowed signals and using the cycle to calculate the precise amount of time needed for each workpiece. We explain these two parts in more detail below.

The signal windowing approach in the first part is slightly different from conventional methods, which basically overlap signals in different windows before analysis. In contrast, the focus of this paper is to conduct cycle analysis, so the signals in different windows do not have to overlap. Figure 3, for example, displays the windowed results of the X axis in Fig. 2. Once the signals have been windowed and segmented, we extract five feature values from each windowed signal: (1) maximum value: the maximum value of all of the data

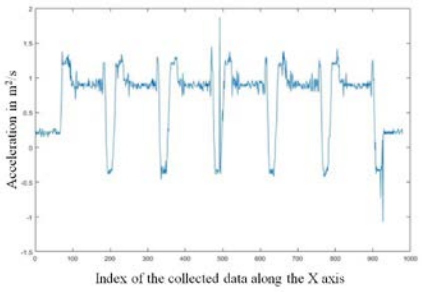

Fig. 2. The collected data along the X axis.

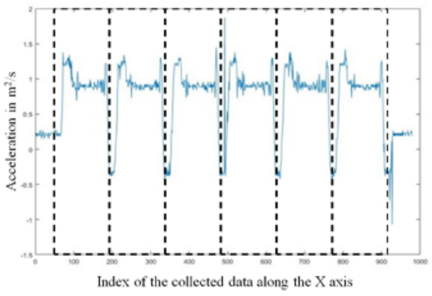

Fig. 3. The windowed result of the collected data along the X axis.

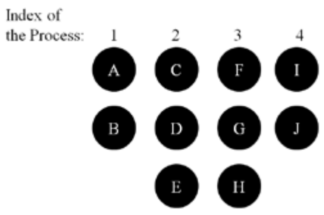

Fig. 4. An example of manufacturing processes.

points in each window, (2) minimum value: the minimum value of all of the data points in each window, (3) mean value: the mean value of all of the data points in each window, (4) energy value: the sum of squares of all of the data in each window divided by the length of the window, (5) standard deviation: the standard deviation of all of the data points in each window.

In the second part, we examine the cycle of maximum and minimum values of these five feature values in the series of windowed signals and derive five sets of cycle data. Ultimately, the average of the 30 sets of cycle data serves as the precise amount of time that the machine needs to finish manufacturing a certain component and is used to estimate the amount of time needed to complete work orders for the same type of component but in a different quantity.

3.3 Target scheduling algorithm

The multi-tasking multi-machine scheduling algorithm for multi-stage multi-criteria production developed in this study was designed based on the concept of skyline queries. It mainly assumes that each work order requires multiple stages of manufacturing and that each stage of manufacturing may be performed by multiple parallel machines. However, the machines may vary in efficiency or have different limitations. Take Fig. 4 as an example. This factory has four manufacturing processes, the first of which is performed by two parallel machines. It takes these two machines 1 hour and 2 hours respectively to complete the first manufacturing process (derived from the time analysis in the previous section), and the yield score of both machines is 2. The second manufacturing process is performed by three parallel machines, and it takes them 3 hours, 4 hours, and 1 hour respectively to complete the manufacturing process (derived from the time analysis in the previous section). Their yield scores are 1, 2, and 1, and so on and so forth for the third and fourth manufacturing processes. To not lose generality, we specifically assumed that once a machine begins manufacturing, it does not stop until one type of component is completed. To insert an additional work order, the machine must complete the current component before the proposed algorithm can be applied for rescheduling.

Below we explained the process of the algorithm by using Fig. 5. When starting up the algorithm, we first initialize a heap and a list. The former caches all of the current scheduling results of the algorithm, while the latter is used to find the query results of the currently known multiple criteria. The first step of the algorithm is to calculate all of the possible schedules when only the first work order is considered. Suppose that there are four work orders that need to be scheduled: a, b, c, and d. The results of this step are <a>, <b>, <c>, and <d>, where <•> represents a schedule. The second step is to place the results of the first step into the heap. The schedules are arranged in ascending order based on the total sum of all of their dimensions. The conditions for each dimension are customized by the factory, such as total work hours and delay time. In the third step, the first schedule (i.e., the schedule with the smallest total of all the dimensions) is extracted from the heap and checked. The check is divided into two parts: the first determining whether the scheduling combinations are completed (i.e., all work orders have been scheduled) and the second determining whether the schedule is dominated by known results in the list. Based on the results of the two parts, we can derive the four possible outcomes below.

Case 1: the schedule is not complete and is not dominated by a result in the list. In this case, we cannot be sure whether expansions of the schedule will include other multi-criteria schedule results, so we expand the schedule, place it in the heap, and rearrange the order for the next round of checks. For example, with work orders a, b, c, and d to be scheduled, suppose that Schedule <c, d> is not dominated by any result in the list. Schedule <c, d> is first expanded to <c, d, a> and <c, d, b>, and the expanded results are rearranged into the heap based on the total of the dimension values to await the next round of checks.

Case 2: the schedule is not complete and is dominated by a result in the list. In this case, the schedule in question will inevitably be dominated by a result in the list no matter how the schedule combination is expanded. This is because the conditions of the schedule will only increase with the number of work orders, so it cannot be the final multi-criteria scheduling result. We can therefore directly remove this schedule combination from the heap. The expanded schedule combinations will not be checked, either, which accelerates the algorithm. For example, suppose that with work orders a, b, c, and d to be scheduled, Schedule <a, b> is found to be dominated by a result in the list. Schedule <a, b> is then directly eliminated from the heap, and <a, b, c, d> and <a, b, d, c> will not be checked.

Case 3: the schedule is complete and is not dominated by a result in the list. In this case, the schedule is a result of the target query, so it is directly added to the list.

Case 4: the schedule is complete and is dominated by a result in the list. In this case, the schedule cannot be the result, so it is directly removed from the heap.

The steps above are repeated until there are no more schedules in the heap.

4 Experimental results of our simulation

In the simulation experiment of this study we used synthetic datasets because we wanted to verify whether the proposed approach can be applied to different factories for any given number of manufacturing stages and any given number of machines. To do this, we can only conduct simulations with synthetic data. Table 2 presents the data that we generated, where P1 to P5 represent five processes in a factory. Numbers 1 to 35 under each of P1 to P5 mean the indexes of machines in different process. For example, the machine 5 under P1 means the fifth machine in the first process. Also, each machine has two scores with regard to a single process. To not lose generality, the calculations of the two scores can be set by the user, such as total work time or delay time. In this experiment, we compared the result of the proposed method with that of a brute force approach, which involved finding all of

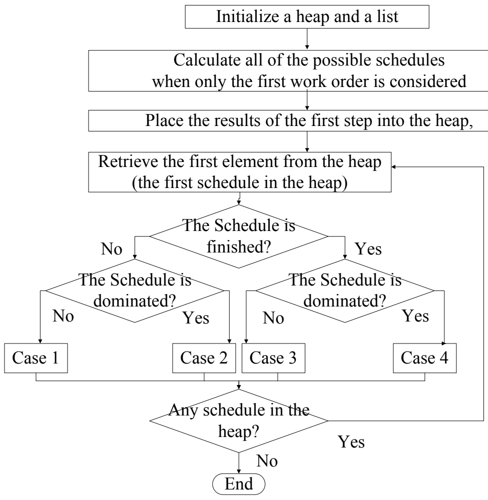
Initialize a heap and a list
Calculate all of the possible schedules when only the first work order is considered
Place the results of the first step into the heap,
Retrieve the first element from the heap (the first schedule in the heap)
The Schedule is finished?
No
Yes
The Schedule is dominated?
No
Yes
The Schedule is dominated?
No
Yes
Case 1
Case 2
Case 3
Case 4
Any schedule in the heap?
Yes
No
End

Fig. 5. The flow chart of the proposed scheduling algorithm.

Table 2. The factory data we used in this work, where 'Pi' represents a process and 'Sj' means a score.

# of machine in P1	S1 of machines in P1	S2 of machines in P1	# of machine in P2	S1 of machines in P2	S2 of machines in P2
1	7	4	8	5	4
2	9	5	9	9	1
3	6	8	10	1	5
4	2	3	11	8	9
5	3	9	12	2	8
6	2	7	13	6	3
7	7	5	14	3	7

# of machine in P3	S1 of machines in P3	S2 of machines in P3	# of machine in P4	S1 of machines in P4	S2 of machines in P4	# of machine in P5	S1 of machines in P5	S2 of machines in P5
15	5	8	22	9	1	29	9	3
16	3	4	23	2	6	30	3	6
17	1	6	24	7	5	31	4	4
18	6	9	25	4	7	32	5	8
19	9	2	26	8	4	33	8	1
20	1	6	27	1	8	34	2	9
21	6	9	28	6	2	35	6	7

the schedule combinations and then applying a skyline query to the schedule combination using a BNL algorithm [2].The experiment was conducted in Windows 10 64-bit with Intel i7-6500U and 16G RAM, and the program was written using Python.

In Figure 6 we compare the results of our proposed approach and the brute force approach. The process of the brute force algorithm is finding all possible schedule combinations in the first step and then using the BNL algorithms to find the skyline results. As can be seen, no matter what parameter combination was used, the brute force approach (top four red lines) required more time than the proposed approach (blue lines at bottom). This is because the proposed approach can effectively reduce the number of schedules that need to be checked, whereas the brute force approach cannot do so. Consequently, the brute force approach required significantly more time than the proposed approach. Furthermore, we can see that the computation time increased with the number of manufacturing stages and the number of machines in both the proposed approach and the brute force method.

This is because increasing any such variable will increase the number of schedule combinations, so the algorithms need more time to finish checking them. Of particular note, the amount of time that the brute force method required increased exponentially because the computation time of the BNL algorithm used in the brute force method increases exponentially with the quantity of data that needs to be checked. In contrast, the amount of time required by the proposed approach only presented linear increases because the number of schedule combinations that need to be checked does not increase too much even if the numbers of manufacturing stages and machines increase.

5 Conclusion and future research

In recent years, many industries have begun to incorporate Industry 4.0 technologies to increase industry value. However, doing so requires a substantial amount of working capital, which is difficult for many small factories. Especially in Taiwan, most of the manufacturers are small- or medium-sized companies and have not enough capital to incorporate Industry 4.0 technologies. In view of this, our study proposed a novel approach to help these manufacturers achieve automated scheduling at lower costs. Furthermore, the production concept of small quantities and great variety in Industry 4.0 is different from that of large quantities and small variety in Industry 3.0. Since existing scheduling methods may no longer be applicable, we also proposed a new approach in this regard. Accordingly, we developed a multi-tasking multi-machine scheduling system for multi-stage multi-criteria production. In future research, we will further develop a scheduling algorithm that allows for the insertion of additional work orders or impromptu maintenance to better satisfy the actual needs of factories.

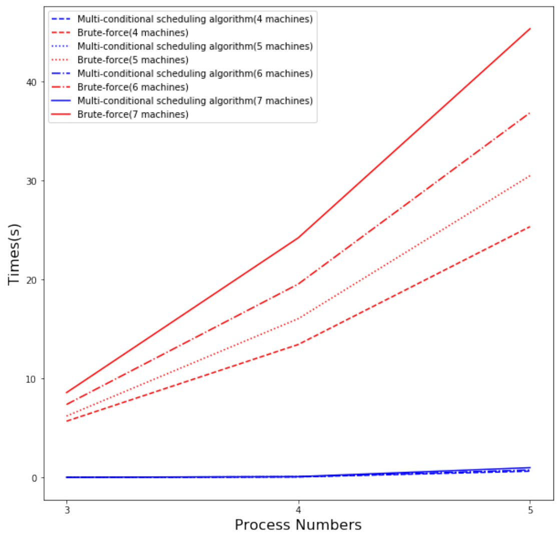

Fig. 6. The comparisons between the proposed method and other methods.

Acknowledgment

This work was supported in part by the Ministry of Science and Technology of Taiwan, under contracts MOST 106-2119-M-224-003 and 106-2218-E-126-001.

References

1. I. Bartolini, P. Ciaccia, and M. Patella, SaLSa: Computing the skyline without scanning the whole sky, *Proceedings of Conference on Information and Knowledge Management*, pp. 405-414, 2006
2. S. Borzsonyi, D. Kossmann, and K. Stocker, The skyline operator, *Proceedings of International Conference on Data Engineering*, pp. 235-254, 2001
3. D. Ivanov, A. Dolgui, B. Sokolov, F. Werner and M. Ivanova, A dynamic model and an algorithm for short-term supply chain scheduling in the smart factory industry 4.0, *International Journal of Production Research*, vol. 54, no. 2, pp. 386-402, 2016
4. C.J. Kuo, K.C. Ting, and Y.C. Chen, State of product detection method applicable to Industry 4.0 manufacturing models with small quantities and great variety: An example with springs, *Proceedings of IEEE International Conference on Applied System Innovation*, pp. 1650-1653, 2017
5. C.J. Kuo, K.C. Ting, J.C. Ying, S.K. Chen, and Y. C. Chen, The RFID-based real-time monitoring system and the management algorithm of RFIDs, *Proceedings of International Conference on Machine Learning and Cybernetics*, pp. 499-503, 2017
6. G.J. Kyparisis and C.P. Koulamas, Flexible Flow Shop Scheduling with Uniform Parallel Machineries, *European Journal of Operational Research*, vol. 168, pp. 985–997, 2006
7. D. Papadias, Y. Tao, G. Fu, and B. Seeger, An optimal and progressive algorithm for skyline queries, *Proceedings of 2003 ACM SIGMOD International Conference on Management of Data*, pp. 467-478, 2003
8. D.N. Tahar, F. Yalaoui, C. Chu, and L. Amodeo, A Linear Programming Approach for Identical Parallel Machinery Scheduling with Job Splitting and Sequence-dependent Setup Times, *International Journal of Production Economics*, vol. 99, no. 1-2, pp. 63-73, 2006
9. C.J. Kuo, K.C. Ting, Y.C. Chen, D.L. Yang, H. M. Chen, Automatic machine status prediction in the era of Industry 4.0: Case study of machines in a spring factory, Journal of Systems Architecture: Embedded Software Design, vol. 81, pp. 44-53, 2017