System Control Planning Technology that Supports Automated Production Site Operations

Authors: *Atsuko Nakai*, Takanobu Yaguchi**

**Advanced Technology Research Center*

Abstract

There is a growing need for automated operations at production sites to compensate for labor shortages. Production schedules at production sites are often created by site supervisors, but this has become an extremely labor-intensive task. To address this issue, we developed a method that, when production capacity drops at production sites where factory workers and production equipment work together, automatically devises and applies the most cost-effective countermeasures by using predefined profile information for factory workers and for production equipment. We also developed an optimization method that combines a formulation part as an integer optimization problem and a rule-based part, in order to rapidly create production schedules that minimize downtime for production equipment.

1. Introduction

In recent years, at production sites in Japan, innovations in production technology have been required, such as a shift to high-mix, low-volume production due to a declining working-age population and the diversification of consumer needs(1). In particular, at sites such as food factories and logistics sites, nighttime and holiday operations are indispensable, making it even more difficult to secure labor. Automation of production equipment is advancing to compensate for labor shortages. Reducing the number of factory workers for labor-saving is expected to ensure a stable workforce and lighten workloads to improve production efficiency. However, a new issue is arisen that the performance difference between production equipment and factory workers have widened, making it difficult to maintain manufacturing line performance. Aiming to improve production efficiency, site supervisors have been creating production schedules based on intuition and experience, but this has become a significant work load at production sites. To overcome these challenges, we devised a method that automatically plans and applies optimal countermeasures when production capacity declines at production sites where factory workers and production equipment coexist. Specifically, we developed an approach that leverages predefined profile information for factory workers and for production equipment to automatically derive the most cost-effective response. Furthermore, to rapidly create production schedules that minimize downtime of production equipment, the many parameters that must be considered are split into a part that performs formulation as an integer optimization and a part that processes them in a rule-based manner and then combined, to develop an optimization method that enables creating a production schedule within tens of seconds. Among the constraints required by production sites, we perform formulation as an integer optimization for the highly variable constraints and use a solver to obtain a solution. Next, the production schedule is created by applying, as rule-based algorithmic processing, the constraints that are generic to production sites to the solution obtained by the solver.

2. Production System Operation Control Method Using Profile Information of Production Equipment and Factory Workers

This chapter describes the method that automatically creates control commands using the profile information defined for production equipment and for factory workers. The proposed method sequentially collects and analyzes the profile information of production equipment—comprising robots, automated transporters, controllers, etc.—and the profile information of factory workers, including status values, sensor readings, operation histories, and so on, and compares them with the profile information from past instances of similar work. This simulates how changing or not changing the profile information of production equipment or factory workers would achieve the desired performance to suit the conditions at the site. It then determines the control details needed to achieve the desired performance and aims to distribute them to the factory workers and production equipment.

Profile information consists of attribute information, which is information specific to production equipment or specific to factory workers, and performance indicator information, which is information about the work capability exhibited by production equipment or factory workers within the production system. That is, profile information is defined as information indicating the overall work capability provided by factory workers and production equipment. Using robots as an example of production equipment, examples of specific profile information items include, as attribute information, not only catalog specification values but also information related to robot operation such as “installed line name.” Performance indicator information includes information calculated from actual operating data, such as “work speed” or “defect incidence rate,” computed for each workpiece or for each set of working conditions. For factory workers, examples of corresponding profile information items include, as attribute information, personnel information and information related to the duties they are engaged in. As with production equipment, performance indicator information includes information calculated from actual operating data, such as “work speed” or “defect incidence rate,” computed for each task or for each set of working conditions. While profile information varies by factory type and by differences in installed equipment, a minimum level of standardization is necessary. Therefore, as a template for profile information, OPC UA*1 information models and companion specifications will be utilized. For example, OPC UA for Robotics has been established for robots. Based on these information models, by adding and extending items related to profile information, standardization of profile information can be achieved, and the costs involved in collecting and evaluating profile information can be reduced.

This method is designed to operate with an edge platform terminal installed in the factory as its core. Figure 1 shows the system configuration. Edgecross, an open software platform in the edge computing domain provided by the Edgecross Consortium, a general incorporated association(2) *2 was adopted within the edge platform terminal, to generically perform data collection, processing, and distribution on the production floor. In this case, we designed a mechanism to share profile information between OPC UA–compliant production equipment and other systems and Edgecross, and to mutually distribute it using the OPC UA API (Application Programming Interface)(3).

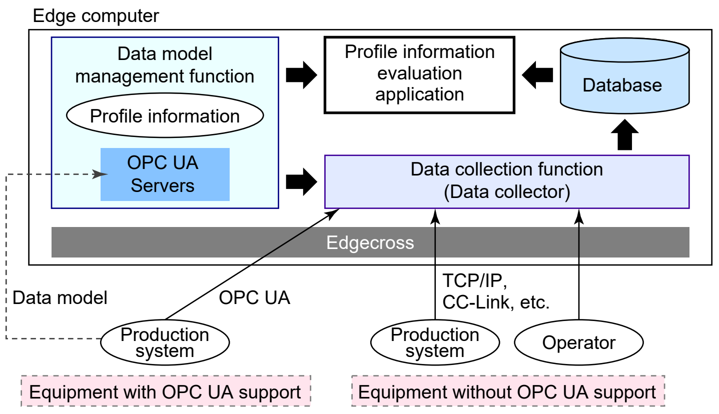
Edge computer
Data model management function
Profile information
OPC UA Servers
Profile information evaluation application
Database
Data collection function (Data collector)
Edgecross
Data model
OPC UA
TCP/IP, CC-Link, etc.
Production system
Production system
Operator
Equipment with OPC UA support
Equipment without OPC UA support

Fig. 1 System configuration

In this work, we created a profile information evaluation application within the edge platform terminal. This application sequentially evaluates the profile information collected by Edgecross and, when a drop in production system performance is anticipated, simulates the effect of issuing control commands to change the profile information items with a high improvement effect among the evaluation indicators preset by the administrator. Based on virtually generated profile information values for workers and production equipment, we ran a prototype of an application that performs sequential analysis to verify whether performance

*1 An international standard in the industrial automation field for enabling equipment and systems from different vendors to exchange data safely and reliably. OPC is a registered trademark of the OPC Foundation.

*2 Edgecross is a registered trademark of the Edgecross Consortium.

drops can be predicted and whether control commands for performance improvement can be created, and conducted verification on a commercially available edge computer. Figure 2 shows an example of the prototype verification screen.

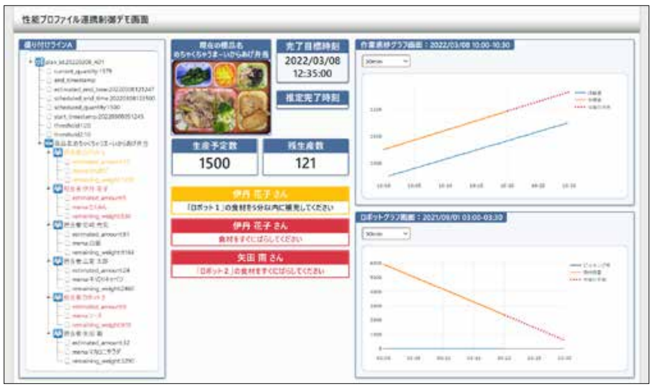
性能プロファイル連携制御デモ画面
完了目標時刻
2022/03/08 12:35:00
推定完了時刻
生産予定数
1500
残生産数
121
伊丹 花子 さん
「ロボット1」の食材を5分以内に補充してください
伊丹 花子 さん
食材をすぐにばらしてください
矢田 雨 さん
「ロボット2」の食材をすぐにばらしてください

Fig. 2 Example of verification screen for the production system operation control method

In this example, a production line in which two picking robots and multiple workers pack lunch box ingredients is simulated, with Edgecross collecting the remaining quantities of ingredients and the operating status of the robots. We verified whether control commands could be prepared in advance so that the line would not stop when the remaining quantity reaches zero while predicting the remaining quantities of ingredients. As a result, we confirmed that the evaluation of profile information and the creation of control commands could be performed with almost no delay(4).

3. Parallel Machine Scheduling Method with Processing Priorities

In this chapter, we propose a method for rapidly creating production schedules for multiple pieces of production equipment at production sites engaged in high-mix, low-volume manufacturing.

We describe the assumptions of the scheduling problem under consideration. At a production site where m machines are operating, n operations called 'jobs' are waiting to be executed. Each job j may be executed in any order, and may be processed on any production equipment. For each job j, the processing time l_j is constant regardless of which machine is used. Furthermore, we define that the maximum processing time is at most 23 hours. Each job j has a due date d_j set as the date by which completion of the job is desired. Each job j is constructed as a unit that can be processed on a single piece of production equipment. Two types of jobs are available—in addition to "regular jobs" created in advance, there are "interrupt jobs" created urgently in cases such as processing failures or insufficient processing, and each job falls into one of these. Based on these assumptions, we establish the following constraints.

Constraint.1 Minimization of total tardiness:	For each job j, the due date d_j is specified on a daily basis. The difference between the due date d_j and the processing date after scheduling is defined as "tardiness"; if the processing date is before the due date, the tardiness is 0. A schedule that minimizes the total tardiness across all jobs is created.
Constraint.2 Priority constraint:	Each interrupt job is assigned a priority p, and higher priority jobs are scheduled earlier time. Regular jobs are scheduled after the date and time when all interrupt jobs have completed processing.
Constraint.3 Operating time constraint:	For each production equipment i, an upper limit on operating time t_i can be set. In that case, for each production equipment i, the total processing time of the jobs it processes does not exceed t_i.
Constraint.4 Load balancing constraint:	A schedule that considers load balancing is created, so that, for each production equipment i, the total processing time of the jobs processed is approximately uniform.

Constraint.5 Sequential execution constraint: On each production equipment i, jobs are executed sequentially.

Here, under the assumptions stated earlier, as a method to obtain a schedule that satisfies the constraints, we adopt the policy of dividing it into a part handled by integer optimization formulation and a part processed by a rule-based algorithm.

The integer optimization processing part formulates the assumptions and the constraints from Constraint.1-4 as an integer optimization. Because Constraint.2-3 are conditions that any desired schedule must always satisfy, they are incorporated as hard constraints into the constraint equations of the integer optimization. Because Constraints.1,4 are desirable conditions to obtain a better solution, they are incorporated as soft constraints into the objective function of the optimization problem. The formulation is as follows.

$$x_{ij} = \begin{cases} 1 \text{ (assign job } j \text{ to production equipment } i\text{)} \\ 0 \text{ (otherwise)} \end{cases} \qquad (1)$$

$l_j \in \mathbb{Z}$: Processing time for job j on the production equipment (2)

J : Collection of all jobs (3)

$d_j \in \{0, 1, 2, \cdots\}$: Number of days until due date of job j (4)

t_i : Maximum operating time of production equipment i (5)

$p \in \{1, 2, \cdots\}$: Priority assigned to each job j (6)

$J_p \subseteq J$: Collection jobs with priority p (7)

$b_p \in \{0, 1\}$: Auxiliary variable for priority constraints (8)

$e_j = f(d_j)$: Benefit from on-time delivery (9)

minimize $-P_1 - P_2$ (10) Objective function for Constraints.1-2

subject to $P_1 = \Sigma_i \Sigma_j x_{ij} e_j$ (11) Constraint.1 Minimize total tardiness

$P_2 = y$ (12) Constraint.4 Load balancing constraint

$y \leq \Sigma_{j \in J} l_j x_{ij} (i = 1, 2, \cdots, m)$ (13) Constraint.4 Load balancing constraint

$\Sigma_{j \in J} l_j x_{ij} \leq t_i (i = 1, 2, \cdots, m)$ (14) Constraint.3 Operating time constraint

$\Sigma_i x_{ij} \leq 1 (j = 1, 2, \cdots, n)$ (15)

$b_p \geq b_{p+1} (p = 1, 2, \cdots)$ (16) Constraint.2 Priority constraint

$\Sigma_i x_{ij} \leq b_p (j \in J_p, p = 1, 2, \cdots)$ (17) Constraint.2 Priority constraint

$b_{p+1} \leq \Sigma_i x_{ij} (j \in J_p, p = 1, 2, \cdots)$ (18) Constraint.2 Priority constraint

Next, this outlines the rule-based processing part. Among the constraints, Constraint.5 is handled by a rule-based algorithm. Specifically, by solving the integer optimization expressed by formulas (1)–(18), we obtain the relationship between each job and the production equipment that processes it, and using this, we create a time-sequenced schedule for each production equipment with a rule-based algorithm. The algorithm used is shown below.

Step 1: Gather the jobs assigned to each production equipment.

Step 2: Among the gathered jobs, sort the interrupt jobs in order of higher priority. If priorities are the same, order them starting from the jobs with the shortest margin for the due date.

Step 3: After all interrupt jobs, arrange the regular jobs in order starting from those with the tightest due dates.

Step 4: Align the ordered jobs with the time axis to create a chronological production schedule.

We prepared jobs that simulate a typical electrical discharge machining (EDM) process on the production site, and conducted numerical experiments to derive production schedules by applying the proposed method to them. We describe the experimental method. The prepared jobs were created to apply several test cases, and for each we predetermined an example production schedule. For each test case, we applied the proposed method, compared the resulting schedule with the example schedule, and verified the effectiveness of the proposed method.

The numerical experiments were conducted under the following settings.

(1) Schedule creation period:	7 days
(2) Number of operating machines:	3 units
(3) Working hours per machine per day:	23 hours
(4) Number of experiment runs per test case:	20
(5) Scheduling execution unit:	1 day
(6) Experimental PC CPU:	11th Gen Intel Core*3 i5-1145G7@2.60GHz
(7) Experimental PC memory:	16GB

The scheduling execution unit is the length of the schedule obtained by applying the proposed method once. In these numerical experiments, we set the scheduling execution unit to one day to prevent jobs from spanning across dates.

We prepared three test-case examples as follows and created an example schedule for each. The test cases were designed to increase in difficulty in the order of Case 1, Case 2, and Case 3.

Case 1: Create jobs such that the total processing time for one day is 23 hours. Prepare these for each day in the scheduling period and for each operating machine. At this stage, all jobs have the same priority. In the example, the jobs scheduled on each day are due on that day. Each job's processing time shall also be a multiple of 60 minutes.

Case 2: Assign priorities to each job created in Case 1.

Case 3: For the jobs created in Case 2, add, in a number equal to the operating machines, "500-minute" jobs whose processing time is not divisible by 60 minutes. With the due date of the "500-minute" jobs set to day 7, scheduling even one of them makes it impossible to plan 23 hours of jobs per day without gaps, due-date violations are expected to occur.

As an example for each test case, a one-day schedule example is shown in Fig. 3.

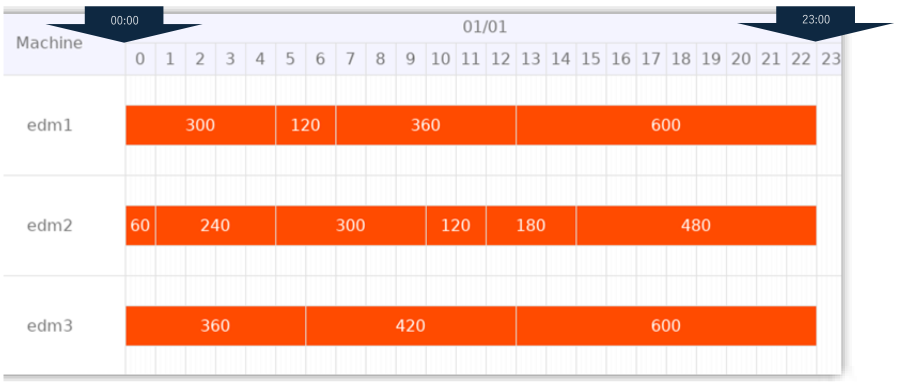

Fig. 3 Ideal example of parallel machine scheduling method

Figure 3 shows a one-day schedule example for three machines. The horizontal axis is the time in one-hour increments, and the vertical axis shows the scheduling results for each machine. From top to bottom: Machine 1, Machine 2, and Machine 3, interrupt jobs are shown in red, and regular jobs in blue. The number inside each job's rectangle also indicates the processing time (minutes). Thus, in the example, a schedule with no idle time from 0:00 to 23:00 can be created. Therefore, if the schedule obtained by the proposed method has no idle time, it can be considered that an optimal solution equivalent to the example has been achieved. If the schedule obtained by the proposed method contains idle time, we evaluate how close it is to the optimal solution by the degree of deviation from the example and define the following evaluation metrics. For every metric, the evaluation indicator is that it is equal to or less than the example.

*3 Intel Core is a registered trademark of Intel Corp.

1) Number of due-date violations:	As a result of scheduling each job, if the planned day for processing a job is later than its due date, it is a due-date violation. We use as an evaluation indicator the number of due-date violations obtained by checking this condition for each job.
2) Due-date violation amount:	As a result of scheduling each job, if the planned day for processing a job is later than its due date, the job's processing time is counted as the due-date violation amount. However, if the planned day is earlier than the due date, the due-date violation amount is 0. In this case, the sum of the due-date violation amounts across all jobs is used as an evaluation indicator.
3) Computation time:	We set an upper limit of 60 seconds and use as an evaluation indicator whether the computation finishes within this limit because the time allowed for scheduling is often limited at production sites.

In formula (11), the total tardiness being minimized is evaluated—among the above evaluation indices—by the number of due-date violations and the due-date violation amount. This is to prevent assigning the same evaluation when either a job with a large processing time or a job with a small processing time becomes tardy, because tardiness is calculated in units of whole days. For both the schedule obtained by the proposed method and the schedule of the assumed example, we evaluate them using these indices and, based on the differences, verify whether the schedule obtained by the proposed method is valid.

For test cases (Cases 1–3), we verify whether the schedules obtained by applying the proposed method satisfy Constraints.1-5. First, by applying the proposed method to each test case, we obtain production schedules. Next, for these schedules, we compute evaluation values based on each evaluation indicator. The evaluation results are shown in Table 1. The values of the evaluation indicators are the averages of data obtained by performing scheduling 20 times.

Table 1 Evaluation results of the proposed method for each index

	Case 1	Case 2	Case 3
Number of due-date violations	0	0	3
Number of correct due-date violations	0	0	3
Due-date violation amount (minutes)	0	0	1500
Due-date correct violation amount (minutes)	0	0	1500
Average computation time (seconds)	1.836	1.970	2.589

Table 1 shows that, across all test cases (Cases 1–3), compared with the assumed example prepared in advance, schedules equivalent in all evaluation indicators were obtained. Also, in the most complex test case, Case 3, Table 1 does not show Constraint.2 Priority constraint —to confirm whether its conditions are satisfied, a diagram of the created schedule is provided in Fig. 4. Figure 4 is an excerpt of the scheduling results by the proposed method, showing Days 7 and 8. If a "500-minute" job is scheduled within the schedule period up to Day 7, the values of the evaluation indicators could be worse than those in the assumed example. However, Fig. 4 results show that they are equivalent to the assumed schedule, and it was confirmed that the values of the evaluation indicators are also equivalent. Moreover, Fig. 4 shows the overall scheduling results for Days 1–8 at the top: red jobs representing interrupt jobs are clustered on Days 1–3, and blue jobs representing regular jobs are clustered from Day 4 onward; therefore, it can be said that Constraint.2 Priority constraint is also satisfied. From these results, we consider that an optimal schedule could be created by formulas (1)–(15). That is, it was confirmed that the proposed method has the basic capability of obtaining an optimal schedule in simple test cases with a small number of jobs(5).

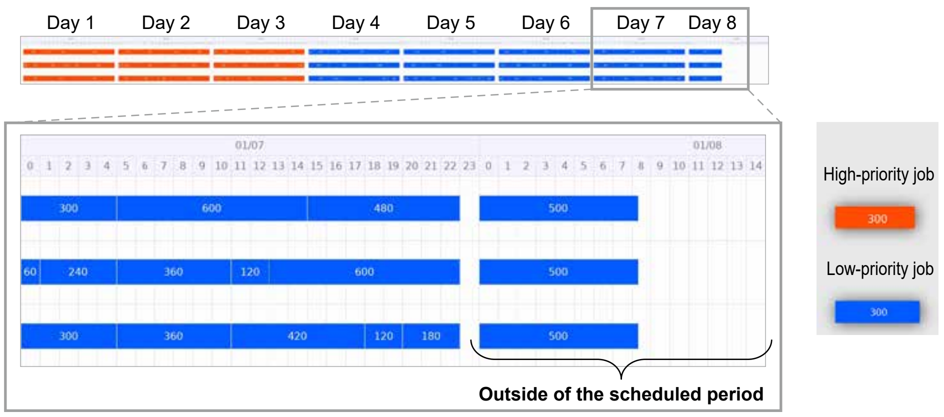

Fig. 4 Result of scheduling at Case 3 (Day 7 to Day 8)

4. Conclusion

At production sites, in situations where factory workers and production equipment coexist, we outlined that: (i) a performance-profile coordinated control method for automating operations by collecting and analyzing line operation data with an edge platform to perform integrated monitoring of the entire line, and (ii) a process scheduling method, characterized by splitting the constraints into an integer-optimization processing part and a rule-based processing part, to rapidly decide the execution order of processes at production sites. Both methods achieved to generate plan in seconds as required in production settings.

References

(1) Council on Competitiveness-Nippon (COCN): New manufacturing with people at the center, Final report of the FY2017 project (2018)
http://www.cocn.jp/report/theme97-L.pdf

(2) Edgecross Consortium
https://www.edgecross.org/ext/ja/index.html

(3) Nakai, A.: A Study on a Method for Linking ROS and Production Systems using an Edge Platform, 2022 Annual Conference on Electronics, Information and Systems, IEEJ, OS1-6 (2022)

(4) Nakai, A.: A Study of Production System Operation Control Method using Profile Information of Workers and Production Equipment, 22nd Forum on Information Technology, J-022 (2023)

(5) Yaguchi, T., et al.: Parallel Machine Scheduling for FA that Considers Processing Priority, 2023 Annual Conference on Electronics, Information and Systems, IEEJ, OS1-2-5 (2023)