Automatic production systems without prior user programming:
Sequencing problem and Navigation during traffic congestion

Shinichi Funase (Faculty of Production Systems Engineering and Sciences, Komatsu University, shinichi.funase@komatsu-u.ac.jp, Japan)
Toshihiko Shimauchi (Faculty of Intercultural Communication, Komatsu University, toshihiko.shimauchi@komatsu-u.ac.jp, Japan)
Haruhiko Kimura (Faculty of Production Systems Engineering and Sciences, Komatsu University, haruhiko.kimura@komatsu-u.ac.jp, Japan)

Abstract
Today, CAD/CAM is increasingly gaining its importance in the field of mechanical design. However, these systems are not operated easily by those without proper computer programming skills. This paper aims to allow these non-programming professionals to automatically perform flexible production process simply by expressing in a certain manner what they intend to manufacture. In this paper, flexible production is defined as a process in which a certain knowledge can be easily extended to a similar problem. The paper investigated two systems to solve "sequencing problems" and "navigation during traffic congestion". The former is a system with problem solving programs developed by a manufacturer in advance, allowing users to operate the system without programming. In the latter system, both a manufacturer and a user do not need to develop a program and the user can solve the problem simply by inputting a knowledge base, which is a set of problems and status altering operations. Their comparison showed the latter system reduces workload for the staff, allowing wider job opportunity for those without programming skills, specifically elders and challenged. The expansion of this system has the potential for realizing a more inclusive and diverse society.

Key words
automatic production system, flexible manufacturing system, sequencing problem, navigating, maze problem

1. Introduction

Flexible Manufacturing System (FMS) is an automatic production system that can cope with high-mix low-volume production. By linking processing machines such as NC machine tools and machining centers, unmanned transport vehicles and industrial robots, all machines can share data with each other, resulting in a unified management of installation, processing, removal, and transportation of products. In addition to processing and transportation, it is also used in all operations at manufacturing and distribution sites, such as assembly and inspection (Chryssolouris, 2005; Flexible Manufacturing System, 2021; Murakami, 2015; Tolio, 2009). The automatic production systems can be divided into following four types: (1) The manufacturer needs to create a program related to the problem in advance and the user also needs to create one. (2) The manufacturer needs to create a program related to the problem in advance but the user has no need to create one. (3) The manufacturer does not need to create a program in advance but the user needs to create one. (4) Both the manufacturer and the user do not need to create a program related to problems. In this paper, a type (2) system is proposed for solving "sequencing problem" and (4) is proposed for finding a solution to the "navigation during traffic congestion". By comparing the two systems, we will investigate which one fits better for social inclusive criteria, that is, more friendly to the elder or disabled.

The paper also aims to identify as many cases as possible in which a type (4) system can be used. The system solves problems simply by the user inputting a knowledge base, which is a set of problems and status altering operations. This would expand the number of fields where automatic production can be implemented without users' programming skills. After the user express the problems, a versatile system, such as production system or AND/OR, is allowed to be utilized. The investigation will be a groundwork for improving the working conditions and enabling older workers as well as younger workers to work as desired. It will also support people with disabilities by expanding job opportunities.

2. Sequencing problem

2.1 Problem to be addressed in the paper

A meeting problem is a problem related to the waiting time of both the arriving customer (job) and the clerk (machine), and probabilistically studies what would happen to the waiting time and the number of waiting lots when the number of machines is increased or decreased. However, in actual settings, companies are required to fully utilize the capabilities of machines without changing their number. A sequencing problem is a problem under these actual settings. In this problem, the number of machines is fixed, while other factors are adjusted: by controlling the processing sequence of arriving or waiting lots, the operation rate of machines can be increased; processing sequence is controlled to minimize the cost (Ibaraki et al., 2011; Sakuraba, 2010).

While a sequencing problem frequently occurs in manufacturing sites, especially in machine assembly factories and semi-process factories, most models do not have an algorithm for finding the optimum solution. The model for which the algorithm for finding the optimum solution is known is 2-machine *n*-job model, to which an algorithm called Johnson's rule (Johnson's Rule, 2021; Johnson, 1954) is applied. Others have known algorithms that can only be used when certain conditions are met (Moriya 1973). This paper covers *m*-machine *n*-job sequencing problem. This model has *n* types of jobs performed using *m* machines and the sequence of the jobs may be changed for each machine. The model tries to find a job execution sequence of each machine to minimize the total elapsed time (time when all jobs is completed). There is no general solution for this kind of problem. Applying the total enumeration method (or brute force enumeration) produces $(n!)^m$ combinations. Therefore, when the number of machines (*m*) and the number of jobs (*n*) increase, the number of combinations increases exponentially, making the method impractical to be applied. For example, even if $n = 5$ and $m = 5$, $(5!)^5$ equals approximately 25 billion combinations, making it impossible to experiment all the combinations even with a computer. Additionally, the sequence of jobs for each machine is the same in many cases, such as 2-machine *n*-job and 3-machine *n*-job. Therefore, this paper also assumes the sequence of job for each machine to be the same. This assumption result in the combination on *n!* This is still equivalent to an exponential function, and the amount of time calculation is $O(n!{\cdot}n{\cdot}m)$, making the processing time enormous as the number of jobs increases. The total elapsed time can be calculated using the Gantt chart (Gantt Chart, 2021) once the job execution sequence is determined. The time complexity required for creating the chart is $O(n{\cdot}m)$.

2.2 Proposed method

In the existing studies, various methods have been utilized for the sequence problem, such as Monte Carlo methods by importance sampling and simulation methods in which goal or criteria is predetermined and multiple processing sequence priority rules are applied (Mohammed, 2016; Moriya, 1973).

This paper proposes a method which collects "job execution sequence" with a short total elapsed time from the existing studies to learn the optimum "job execution sequence" based on the collected sequences. The specific procedure is outlined below.

2.2.1 Outline of proposed method

(1) If the number of machines is 2, Johnson's rule is used to find the "job execution sequence" that minimizes the total elapsed time.

(2) When the number of machines is 3 and at least one of the following conditions is satisfied, Johnson's rule is used to solve the problem of hypothetical 2-machine *n*-job problem. The operation time of the machine to be used first in the *i*-th job is set to be $a_i + b_i$, and that of the machine to be used second is set to be $c_i + b_i$. This job execution sequence is known to be the same as the optimum solution (minimum total elapsed time) of 3-machines *n*-job that satisfy the above conditions (Moriya, 1973). Here, a_i, b_i, and c_i are the operation time of the machine to be executed first, second, and third, respectively.

- Condition 1:
  $\text{Min}\{a_1, a_2, \cdots, a_n\} \geqq \text{Max}\{b_1, b_2, \cdots, b_n\}$
- Condition 2:
  $\text{Min}\{c_1, c_2, \cdots, c_n\} \geqq \text{Max}\{b_1, b_2, \cdots, b_n\}$

(3) In the remaining cases, the Monte Carlo method by importance sampling is applied. Here, the "job execution sequence" is randomly selected and repeated several times to obtain the "job execution sequence" of *M* routes. Then, the "job execution sequence" of the top *N* routes ($N << M$), arranged in ascending order of the total elapsed time, is extracted.

The method of randomly selecting the "job execution sequence" is as follows. The result goes into $B(1), B(2), \cdots, B(n)$. INT (*X*) is a function that truncates the decimal point of *X*. RND (1) is a real random number between 0 and 1 ($0 < \text{RND}(1) < 1$).

```
For i = 1 to n
  A(i) = i
  Next i
For i = n to 1 step –1
  t = INT(RND(1) * i) + 1
  B(i) = A(t)
  A(t) = A(i)
Next i
```

(4) The "job execution sequence" is obtained for each of the processing priority rules in the existing methods. These are arranged in ascending order of total elapsed time and top *L*-routes are extracted.

(5) Learning is conducted based on the "job execution sequence" of $(N + L)$ routes extracted in (3) and (4). Then, *K* candidate routes for the optimum solution are obtained using quantification theory class 2 and 3. From these *K* candidate routes, the "job execution sequence" that minimizes the total elapsed time is generated. *K* routes of "job execution sequence" includes those with the minimum total elapsed time within the ranges extracted in (3) and (4), respectively.

2.2.2 Example 1

There are five jobs and two machines, which are processed in the sequence of *M*1 and *M*2, respectively, to form a product. Each processing time is as shown in Table 1. Here, the question is in what sequence should the five jobs be performed to minimize total elapsed time. Each job cannot be subdivided.

Since this is a 2-machine n-job problem, Johnson's rule is applied. The "job execution sequence" with the minimum total elapsed time is J_2, J_4, J_3, J_5, J_1.

2.2.3 Example 2

There are 5 job (J_1, J_2, J_3, J_4, and J_5) and 3 machines (Machine *A*, *B* and *C*). Each job is processed in the sequence of machine *A*, *B*, and *C*. Table 2 shows the processing time. Find the "job execution sequence" that minimizes the total elapsed time, and find the total elapsed time using the Gantt chart.

Since the condition 1 of the operation (2) of the proposed method is satisfied, a hypothetical 2-machine *n*-job table is created (Table 3). Johnson's rule is applied to obtain the "job execution sequence" that minimizes the total elapsed time, which is J_2, J_1, J_3, J_4, J_5.

The total elapsed time is 62 minutes according to the Gantt chart in Figure 1.

2.2.4 Example 3

The number of execution sequence of *n*-job is *n!* routes.

Table 1: Processing time (in minute)

Job	Machine 1	Machine 2
J_1	5	2
J_2	1	6
J_3	9	7
J_4	3	8
J_5	10	4

Table 2: Processing time (in minute)

Job	Machine A	Machine B	Machine C
J_1	10	8	9
J_2	9	7	15
J_3	12	4	10
J_4	14	6	7
J_5	8	5	4

Table 3: Processing time (in minute)

Job	Machine AB	Machine BC
J_1	18	17
J_2	16	22
J_3	16	14
J_4	20	13
J_5	13	9

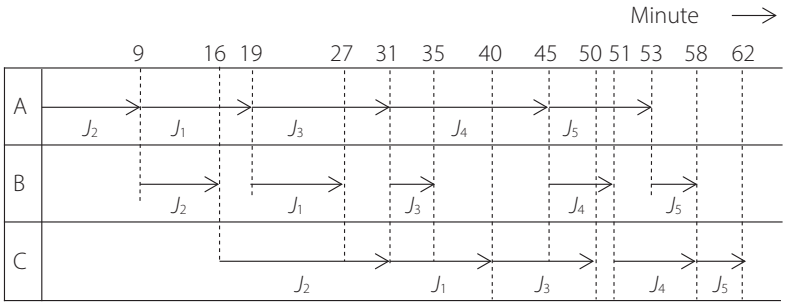

Figure 1: Gant chart to find the optimum sequence

How many samples (k) are required to ensure that the execution sequence of a certain job is within the top P (× 100) % with a probability of α (× 100) % or higher, when the sequences are arranged in ascending order of the total elapsed time until n-job completion?

Since the probability that one sample falls within P (× 100) % is P, the probability that one sample does not fall is $1 - P$. The probability that the number of samples k does not fall within P (× 100) % is $(1 - P)^k$ and the probability that at least one of k falls within P is $1 - (1 - P)^k$. Therefore, since this probability should be set to α or more,

$$\begin{aligned} &1 - (1 - P)^k \geqq \alpha \\ &1 - \alpha \geqq (1 - P)^k \\ &\log (1 - \alpha) \geqq k \log (1 - P) \\ &k \geqq \log (1 - \alpha) / (\log (1 - P) \end{aligned} \quad (1)$$

For example, to find k with a 95 % probability of being in the top 10 %, $\alpha = 0.95$ and $P = 0.1$ are input to equation (1) to yield 28.3. Hence the necessary number of samples is 29 or more.

2.2.5 Example 4

A method using slack is shown as a typical example of a processing priority rule.

$P_{ij} = D(i) - \Sigma T(i, j, j + 1) - k\Sigma P(i, j)$
= (Delivery time) – (Total transportation time)
– kx (Total processing time)
i: sequence, production lot or product number
j: process number
$D(i)$: due time to complete the job
$T(i, j, j + 1)$: Time required for the i-th lot to be transported from process j to process $j + 1$ (tracking time)
$P(i, j)$: Processing time for the i-th lot at process j
k: parameter (determined arbitrarily or empirically)

As shown in the above equation, this priority P_{ij} means the time margin until the delivery date. Job execution sequence starts from the smallest P_{ij}.

3. Navigation during traffic congestion

3.1 Functions

Once the starting point (current location) and the end point (destination) are designated, routes between the two points are proposed. Roads that can be passed through are registered as arcs in advance, and roads that are congested or have traffic accidents can be delisted from the recommendation. The navigation system also shows routes with shortest distance and/or time. Route finding methods include a blind search (vertical and horizontal methods), a heuristic search (best-first search method, A*algorithm, and hill climbing method).

3.2 Basic problem

The basic problem of navigating when the road is congested is the maze problem (Maze-solving Algorithm, 2021).

[Example 5] Consider the maze in Figure 2. The alphabet is a mark on the floor and functions as a two-dimensional coordinate in the navigation system. In this maze problem, the route from A to Y will be searched.

3.3 Search method outline

(1) The problem is expressed in a state-space representation.
(2) A knowledge base is generated from the set of rules in (1).
(3) The production system outputs a solution.

In step (2), the knowledge base outputs rules to avoid an infinite loop and a series of landmarks (coordinates) that specify the route.

[Example 6] Figure 3 shows a state-space representation for the maze problem of Figure 2.

4. Comparison of the two systems

The "sequencing problem" is a type (2) automatic production system. The manufacturer uses the Monte Carlo method and processing priority rules in advance to find the optimum

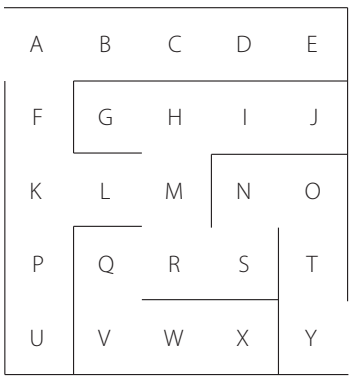

Figure 2: Maze problem

Initial state	A
Goal state	Y
Rules	A→B, B→C, C→D, D→E, A→F, F→K, K→L, L→M, M→H, H→G, H→I, I→J, K→P, P→U, M→R, R→Q, Q→V, V→W, W→X, R→S, S→N, N→O, O→T, T→Y

Figure 3: State-space representation of the maze problem in Figure 2

"job execution sequence" options. The Gantt chart is used to find the total elapsed time of job execution. Users use these systems to narrow down the options for the optimal "job execution sequence". In the combination problem which this paper investigated, the burden on the user increases when narrowing down the options.

"Navigation during traffic congestion" is a type (4) automatic production system in which both the manufacturer and the user are not required to create a program. The basic procedure is a production system that solves the maze problem. The knowledge base of the production system corresponds to a set of rules for state-space representation. The only difference between the production system and state-space representation is that the knowledge base in the production system has a rule to avoid an infinite loop and a rule to output a route on the way to the goal.

In the production system, we only give a rule that shows how to change the state for each phase and the above two types of rules. These rules are familiar to production staff, but creating a general program is time consuming. In the production system, all what the staff needs to do is to create a set of these rules (knowledge base), and there is no need to consider a series of instructions (order of instructions), resulting in a significant reduction in workload. In addition, when the problem expressed by the state space representation is executed by the corresponding PS, almost no logical error (bug) occurs. On the other hand, to create a program with equivalent functions in a compiler language such as C, it is necessary to consider a series of processing procedures (algorithms) so that bugs do not occur, which requires specialized skills and a large amount of time. By using a production system that significantly reduces the workload compared to programming, even employees with insufficient understanding of computers and programs can work on improving the production process.

If the number of automatic production systems of type (4) increases in the future, the working environment for production process improvement offices will be enhanced. As a result, it will lead to an increase in employment opportunities for those who wish to work from various backgrounds, such as the elderly and people with disabilities. It will be a system that greatly contributes to the realization of a more inclusive and diverse society better fitted for 21st century.

5. Conclusion

This paper aims to allow non-programming professionals to automatically perform flexible production process simply by expressing in a certain manner what they intend to manufacture. We investigated two systems to solve "sequencing problems" and "navigation during traffic congestion". The former is a system with problem solving programs developed by a manufacturer in advance, allowing users to use the system without programming. In the latter system, both a manufacturer and a user do not need to develop a program and the user can solve the problem simply by inputting a knowledge base, which is a set of problems and status altering operations. Their comparison showed the latter system reduces workload for the staff, allowing wider job opportunity for those without programming skills, specifically elders and challenged. The expansion of this system has the potential for realizing a more inclusive and diverse society.

References

Chryssolouris, G. (2005). *Manufacturing systems: Theory and Practice*. New York, NY: Springer Verlag.

Flexible Manufacturing System (2021). Wikipedia. Accessed on March 29, 2021. https://en.wikipedia.org/wiki/Flexible_manufacturing.

Gantt Chart (2021). Wikipedia. Accessed on January 10, 2021. https://en.wikipedia.org/wiki/Gantt_chart.

Ibaraki, T., Imamichi, T., Koga, Y., Nagamochi, H., Nonobe, K., and Yagiura, M. (2011). Efficient branch-and-bound algorithms for weighted MAX-2-SAT. *Mathematical Programming*. Vol. 127, 297-343.

Johnson's Rule (2021). Wikipedia. Accessed on January 3, 2021. https://en.wikipedia.org/wiki/Johnson%27s_rule.

Johnson, S. M. (1954). Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, Vol. 1, No. 1, 61-68.

Maze-solving Algorithm (2021). Wikipedia. Accessed on March 13, 2021. https://en.wikipedia.org/wiki/Maze-solv-

ing_algorithm.
Mohammed, G. T. (2016). New algorithm for n-jobs on m-machine flow shop scheduling problems. *Applied and Computational Mathematics*, Vol. 5, No. 1, 1-9.
Moriya, E. (1973). *Operations research*. Ohmsha. (in Japanese)
Murakami, H. (2015). Technology trends and future outlook of automated systems. *Journal of the society of Instrument and Control Engineers*, Vol. 54, No. 12, 889-894. (in Japanese)
Sakuraba, C. S. (2010). Efficient local search algorithms for the linear sequencing problem. *International Transactions in Operational Research*, Vol. 17, 711-737.
Tolio, T. (2009). *Design of flexible production systems: Methodologies and tools*. Berlin: Springer.

(Received: June 10, 2021; Accepted: July 3, 2021)