A FLEXIBLE SIMULATION SYSTEM FOR SCHEDULING WITH VARIOUS CONSTRAINTS

SHINJI HASEBE, YOSHIHIRO MURAKAMI*² AND IORI HASHIMOTO

Department of Chemical Engineering, Kyoto University, Kyoto 606-01, JAPAN

Key Words: Scheduling, Simulation, Discrete Event System, Optimization, Resource Constraint

A simulation system, which can be widely used to determine the starting times of operations for general batch plants, is proposed. The operations of a batch plant are first expressed by a sequence of basic operations such as filling, processing, discharging and cleaning. By using these basic operations, the simulation problem can be reduced to that of determining the starting times of the basic operations.

A variety of constraints are grouped into four types, and it is demonstrated that the constraints on finite intermediate storage can be converted into the constraints on precedence relationships of basic operations. Then, for each of the three types of constraints, simulation modules to calculate the earliest and latest starting times of basic operations are developed. A simulation program to calculate the starting times of basic operations which satisfy all types of constraints can be generated by combining these simulation modules.

The major feature of the proposed method is that the simulation program can be easily modified by recombining the simulation modules. Therefore, a simulation program with sufficient flexibility can be developed by using the proposed method.

Introduction

The objectives of a scheduling system are to determine the sequence in which the products should be produced (the sequencing problem), and to determine the starting times of various operations such as charging, processing and discharging required for each unit (the simulation problem).

One of the predominant characteristics of batch plants is that the material leaving a batch unit is fluid, and it is sometimes chemically unstable. Therefore, the starting times of operations must be determined by taking into account the storage policy between two operations. Furthermore, the operations which can be carried out at night or over the weekend are limited, and the simultaneous execution of some operations may be prohibited. So, even if the processing order of jobs for each batch unit is fixed, it is still very difficult to determine the optimal starting times of the jobs for each unit in such a way that all the given constraints are simultaneously satisfied. The aim of this paper is to propose a simulation algorithm which can be widely used for scheduling of many different kinds of batch plants.

Much research effort has been devoted to developing algorithms to determine the starting times of operations by taking into account several types of interstage storage policies. For a serial multiproduct batch plant, Suhami and Mah (1981), Wiede and Reklaitis (1987), and Rajagopalan and Karimi (1989) proposed calculation algorithms in

which the completion time of the final job (makespan) is minimized. Egri and Rippin (1986) proposed a simulation algorithm in which they considered the due date of each job, utility constraints, restriction of the batch unit during the weekend and several types of storage policies. The State Task Network representation is used by Kondili *et al.* (1988) to calculate the starting time of each operation.

In this paper, a variety of constraints arising in chemical plants are first classified into several groups in order to make the characteristics of the constraints clearer. For each type of constraint, simulation modules to calculate the starting times of jobs were developed. Then, a method of developing a simulation program, which can be widely used in determining the starting times of operations for general batch plants, is proposed.

1. Modeling of the Operation

The production steps in a batch unit consist of operations such as filling the unit, processing the materials, discharging and cleaning. In many cases, it is possible to insert a waiting period between two successive operations. The duration times for the filling, processing, and discharging operations can be regarded as constant, because the amount of product for each job has been determined before the simulation. The duration time for the cleaning operation can also be regarded as constant, because the production order of jobs for every unit has already been determined by the sequencing program. By taking these facts into account, in the simulation algorithm each of these

* Received on October 13, 1995.

*2) Currently at Dept. of Mechanical Systems Eng., Kansai University

Fig. 1 Schedules for various types of working patterns

operations is treated as the smallest element, and is hereafter called a “basic operation.” The operations in a semicontinuous unit are also expressed by using basic operations. That is, a semicontinuous unit needs to be filled, material processed, and discharged, and the filling and processing operations start simultaneously.

The procedure required to produce a batch of product can be expressed as a sequence of basic operations, the order of which is given a priori, and the production order of products for every batch unit has been determined by the sequencing program. From these data, the precedence relationship among all of the basic operations is thus determined and the simulation problem is reduced to a problem of determining the optimal starting times of the basic operations for every unit so as to satisfy many kinds of constraints.

2. Classification of the Constraints

In this section, the various constraints are classified into four types, and the factors which should be considered in defining each type of constraint are amplified.

Type I) Constraint on precedence relationship

The relationship between the starting times of two basic operations i and j can be expressed by the following inequalities:

$$t_i + h_{ij} \leq t_j \leq t_i + h'_{ij} \quad (1)$$

where t_i is starting time of basic operation i , and h_{ij} and h'_{ij} are time determined as a function of basic operations i and j respectively.

The value of h'_{ij} depends on the chemical stability of the materials being processed. If two basic operations must be executed successively without any waiting time, h_{ij} is equal to the processing time of basic operation i and $h'_{ij} = h_{ij}$. If two basic operations must be started simultaneously, then $h_{ij} = h'_{ij} = 0$.

By assigning proper values to h_{ij} and h'_{ij} in Eq. (1), it is possible to express many different types of storage policies such that material can be held in a batch unit before the processing is started.

Type II) Constraint on working patterns

The second type of constraint is the restriction with respect to the processing of particular basic operations during a fixed time period. This type of constraint is hereafter called “the constraint on working patterns.” In order to better understand the meaning of this constraint, several examples will be discussed.

(a) The discharging operation cannot be executed between T_1 and T_2 .

(b) Every kind of operation cannot be executed between T_1 and T_2 . It is possible to interrupt the processing temporarily, and the remaining part of the processing can be restarted at T_2 .

(c) Every kind of operation cannot be executed between T_1 and T_2 . But it is possible to hold the material in a batch unit.

(d) Unit i cannot be used between T_1 and T_2 because of maintenance.

Figure 1 presents a schedule which satisfies each of the above constraints. In this figure, the available starting times for the filling operation are all the same, but the simulation results are completely different from each other.

Type III) Utility constraint

The third type of constraint is the restriction on simultaneous processing of several basic operations. If the maximum level of utilization of any utility or manpower is limited, the basic operations which use a large amount of utility cannot be executed simultaneously. The feature of this constraint is that the restricted period is not fixed but depends on the starting times of basic operations which are processed simultaneously. This type of constraint is hereafter called “the utility constraint.”

Type IV) FIS constraint

If the number and the capacities of storage tanks between two batch units are finite, the starting times of some basic operations must be adjusted so that each storage tank does not overflow. This type of constraint is called “the finite intermediate storage (FIS) constraint.”

3. Simulation Modules

The simulation algorithm which is appropriate to a scheduling problem depends on the performance index used in the problem. In some cases, the performance index is optimized by executing basic operations as early as possible, and in other cases, it is optimized by executing basic operations as late as possible. Minimization of the completion time is a typical example of the former cases, and minimization of the storage cost of products is that of the latter cases. Therefore, it is difficult to develop a general simulation program which suits various kinds of scheduling problems. By taking this fact into account, in this paper, we propose a module type simulation program, where many kinds of simulation algorithms are prepared in advance as simulation modules (subroutines), and the total simulation program is generated by combining these modules, taking into account the characteristics of the particular scheduling problem.

3.1 Modules which satisfy Type I constraints

In this group of simulation modules, the starting times of the basic operations are calculated by taking into account the constraints on the precedence relationship among the basic operations only. Each constraint expressed by Eq. (1) has only two variables, t_i and t_j . This feature is effectively used to calculate the earliest and latest starting

Fig. 2 Steps to calculate starting times which satisfy constraint of working patterns

times of the basic operations which satisfy the constraints on the precedence relationship. The algorithms consist of the following steps:

<Module Ia : earliest starting time calculation>

- (1) $t_i \leftarrow E_i$ for every basic operation i .
where E_i : temporary earliest starting time of basic operation i , B.O. i , which is given as input data.
 $a \leftarrow b$: b is substituted for a .
- (2) Repeat steps (3) to (7) for every basic operation.
- (3) Select a basic operation (B.O. J), and repeat steps (4) to (6) for all of the basic operations which precedes B.O. J .
- (4) Select a basic operation (B.O. M) which precedes B.O. J .
- (5) Change the starting times of B.O. J and B.O. M if they satisfy the following conditions:
If $t_M + h_{MJ} > t_J$, then $t_J \leftarrow t_M + h_{MJ}$.
If $t_J - h'_{MJ} > t_M$, then $t_M \leftarrow t_J - h'_{MJ}$.
- (6) If $t_J > T_{max}$, then go to step (9).
where T_{max} : ending time of the scheduling period given as input data.
- (7) If the starting times of some basic operations are changed during steps (2) to (6), return to step (2).
- (8) Stop the calculation. The derived starting times are the earliest starting times which satisfy all of the constraints expressed in Eq. (1).
- (9) Stop the calculation. In this case, there is no feasible solution.

<Note> If there are no feasible solutions for the given production sequence, the starting times of some jobs become infinity by continuing the execution of Module Ia. Therefore, we can judge the unfeasibility of the given production sequence at step (6) by assigning the proper value to T_{max} .

The latest starting times of basic operations can be calculated by using almost the same algorithm as that of

Fig. 3 Schedule which does not satisfy the utility constraint

the earliest starting time calculation, and is given in Appendix.

3.2 Modules which satisfy Type II constraints

The algorithm which calculates the earliest starting times of basic operations by taking into account the constraints on working patterns consists of the following steps:

<Module IIa : earliest starting time calculation>

- (1) $t_i \leftarrow E_i$ for every basic operation i .
- (2) Repeat steps (3) to (5) for every constraint on working patterns.
- (3) Choose a constraint on working patterns.
- (4) Find basic operations which do not satisfy the constraint selected at step (3).
- (5) For each of basic operations selected at step (4), delay the starting time so that the basic operation satisfies the constraint.
- (6) Stop the calculation.

Note, that only the starting time of the selected basic operation is changed at step (5). The modification of the starting time of a basic operation affects the starting times of other basic operations. Therefore, the derived schedule may not satisfy the constraints for the precedence relationship. So, after the modification of the starting time of the basic operation, all steps of Module Ia must be executed again to derive the starting times which satisfy the constraint for precedence relationship. The schedule which satisfies all of the constraints of Types I and II can be derived by the following algorithm.

<Combined Module I-IIa>

- (1) Execute Module Ia.
- (2) $E_i \leftarrow t_i$ for every basic operation i .
- (3) Execute Module IIa.
- (4) $E_i \leftarrow t_i$ for every basic operation i .
- (5) If the starting times of some basic operations are changed at step (3), return to step (1). Otherwise, stop the calculation.

The starting times obtained by Combined Module I-IIa are strictly the earliest starting times of basic operations which satisfies all of the constraints of Types I and II. **Figure 2** shows how the starting times of two basic operations are changed so as to satisfy the constraints. In this example, it is assumed that two basic operations must be executed without any waiting period and that both of these basic operations cannot be executed between times, T_1 and T_2 . By executing steps (1) to (4) in Combined Module I-IIa three times, the earliest starting times of both basic

Fig. 4 Processing order of jobs on two batch units

operations which satisfy the constraints are derived.

The algorithm for calculating the latest starting times of the basic operations which satisfy constraints of Types I and II (Combined Module I-IIb) is similar to that for calculating the earliest starting times.

3.3 Modules which satisfy Type III constraints

Figure 3 shows a case where three basic operations use the same utility and where the utility constraint is not satisfied during the time period t_2 and e_j . Note that utility constraints can be satisfied in several alternative ways. In this example, the starting time of one of the three basic operations must be delayed or advanced to satisfy the constraint. The number of time intervals during which some utility constraints are not satisfied can be reduced by the following algorithm:

<Module IIIa: earliest starting time calculation>

- (1) For each type of utility, calculate the consumption rate at each time period.
- (2) Find utility constraints which are not satisfied during some time periods, and select one which has the earliest time period. If every constraint is satisfied, go to step (5).
- (3) Enumerate the basic operations which are used during the selected period, and select one using predetermined rules.
- (4) Delay the starting time of the basic operation selected at step (3) so as to satisfy the utility constraint.
- (5) Stop the calculation.

Many heuristics are available for the selection order of a basic operation to be delayed at step (3). Even if the initial schedule satisfies the constraints on the precedence relationship and working patterns, the derived schedule may not satisfy these constraints. So, Combined Module I-IIa and Module IIIa must be executed repeatedly to derive the starting times which satisfy all of the three types of constraints.

The algorithm for calculating the latest starting times of the basic operations which satisfy constraints on working patterns (Module IIIb) is similar to that for calculating the earliest starting times.

3.4 Modules which satisfy Type IV constraints

In this paper, we treat the case where intermediate products of different jobs can be stored in a tank by sharing the period, though they cannot be mixed in a tank. In this case, the maximum number of jobs which are stored in the tanks must be equal to the number of storage tanks installed between two batch units. Wiede and Reklaitis (1987) proposed a calculation method of the starting times

of jobs for the case where all products are produced in the same order on each unit (permutation schedules). In this section, their method is expanded so that the starting times of jobs can be calculated even for the case where the processing orders of jobs on two batch units are not the same.

Here, we first explain the method of calculating the minimum number of storage tanks required for a given schedule. Figure 4 shows an example of the processing order of seven jobs on two batch units. If some jobs executed after a job (job i) at unit 4 are executed before job i at unit 5, job i must be stored in a tank. For the case of Fig. 4, jobs B, E, and F must be stored in tanks. Thus, a set of jobs stored in storage tanks is given by the following equation:

(A set of jobs stored in tanks at the time when job i is executed at unit 5)

= (A set of jobs executed after job i at unit 5)

∩ (A set of jobs executed before job i at unit 4) (2)

The number of jobs calculated by Eq. (2) is shown under the circles expressing the operations of jobs at unit 5. The maximum value in these numbers is the minimum number of storage tanks required by this processing order of jobs. Thus, if this maximum value is greater than the number of storage tanks actually installed, the given processing order is decided to be infeasible. Hereafter, it is assumed that the given processing order satisfies this constraint.

Suppose the case where the number of storage tanks between unit i and unit j is S_{ij} , and the number of jobs obtained by Eq. (2) for job k is H^k . Let the starting time of the discharging operation of job k from unit i be t_{id}^k . The number of jobs stored in tanks at t_{id}^k is equal to the number of jobs which satisfy such condition that the discharging operation from unit i has already finished before t_{id}^k , but the filling operation to unit j has not finished before that time. Therefore, Type IV constraints are always satisfied if t_{id}^k for every job k is decided so that the number of jobs stored at that time is equal to or less than the number of storage tanks. By using this relationship, the condition such that t_{id}^k satisfies Type IV constraints is derived as follows:

case 1) $S_{ij} = H^k$

In this case, t_{id}^k must be equal to the starting time of the filling operation to unit j , t_{jf}^k , i.e.,

$$t_{id}^k = t_{jf}^k \quad (3)$$

case 2) $S_{ij} > H^k$

Let G^k be a set of jobs executed before job k at both units i and j . When we consider jobs belonging to G^k , let job J_k be a job which has been executed ($S_{ij} - H^k$) jobs prior to job k at unit j . In order to satisfy Type IV constraints, t_{id}^k must be later than the ending time of the filling operation of job J_k at unit j . This condition is expressed as follows:

$$t_{id}^k \geq t_{jf}^{J_k} + P_{jf}^{J_k} \quad (4)$$

where

Fig. 5 Gantt chart for schedule shown in Fig. 4

Fig. 6 Batch plant consisting of two production lines

p_{jj}^k : time required for the filling operation of job k to unit j .

Figure 5 shows the starting times of jobs on units 4 and 5 for the processing order given by Fig. 4. In this schedule, the discharging operation of job D from unit 4 cannot start till time T indicated in the figure, if the number of storage tanks is restricted to two.

The expressions of Eqs. (3) and (4) are the same as that of the constraint of precedence relationship (Type I constraint), and these expressions are generated uniquely when a processing order of jobs is fixed. Therefore, by generating these inequality constraints when the processing order of jobs is given, and by treating these constraints and Type I constraints simultaneously, a schedule which satisfies both Type I and Type IV constraints can be obtained.

By combining the above modules, one can derive a schedule which satisfies every kind of constraint discussed in the previous sections of this paper.

4. Simulation Algorithm for a Batch Plant

In order to show how the simulation algorithm is developed by combining simulation modules, a scheduling problem for a batch plant consisting of two production lines as shown in Fig. 6 is presented. There are intermediate storage tanks between units 4 and 5, and units 12 and 17, and one batch of intermediate product can be stored in each tank. In this batch plant, the production path of each product is given a priori. Each product takes one of the following 5 production paths:

Fig. 7 Processing order of jobs on five unit groups

Fig. 8 Processing order of jobs on batch units

- Units 1 → 2 → 3 → 4 → (storage tank) → 5 → 6 → 7
- Units 8 → 2 → 3 → 4 → (storage tank) → 5 → 6 → 7
- Units 8 → 10 → 11 → 12 → 13 → 14 → 15 → 16
- Units 9 → 10 → 11 → 12 → 13 → 14 → 15 → 16
- Units 8 → 10 → 11 → 12 → (storage tank) → 17 → 16

From the viewpoint of sequencing of jobs, the batch units can be classified into the groups (encircled by the broken lines) shown on Fig. 6. Each group of units is hereafter called a "unit group." By determining the processing order of jobs for each of the five unit groups, the processing order of jobs for each unit is automatically determined, and furthermore the processing order of all of the basic operations is also determined.

Figure 7 shows an example of the processing order of 13 jobs for every unit group. In this graph, the circles represent the processing of a job assigned to a unit group. The arrows in a unit group express the processing order of jobs, and the broken arrows between two unit groups denote the production path of each job. Figure 8 shows the processing order of three jobs, Jobs J , K , and L , for batch units which belong to unit groups 2, 3 and 5. The schedule shown in Fig. 8 is further decomposed into the schedule using the basic operations, and a part of the

Fig. 9 Processing order of basic operations of two jobs

Fig. 10 Part of a schedule derived by using algorithm A

decomposed schedule is shown in Fig. 9. The processing order of jobs on unit groups such as shown in Fig. 7 is given to the proposed simulation system, and in the simulation system, it is decomposed to the processing order of basic operations such as shown in Fig. 9.

This scheduling problem has the following constraints to calculate the starting times of the jobs.

- (1) The material processed at unit 2 must be transferred to unit 3 as soon as the processing is completed, and the processing operation at unit 3 must be started as soon as the filling operation is finished. For other units, however, a job can be held in a unit before and after processing.
- (2) The start of an operation which cannot be finished by Saturday evening is delayed till Monday morning. It is possible to hold the material in the batch unit over the weekend.
- (3) Unit 4 is closed every other Saturday for maintenance. The material cannot be held in the unit during that period.
- (4) A unit amount of manpower is required for each of the filling operations. During the day time, up to three units of manpower can be used, and only a unit of manpower on Saturday and during the night.
- (5) The performance index is a monotonically increasing function of the completion time of each job.

By taking into account the above constraints, the following simulation algorithm was developed:

<Simulation Algorithm A>

- (1) FIS constraints are changed to the constraints of precedence relationship.

Fig. 11 Part of a schedule derived by using algorithm B

- (2) Execute Combined Module I-IIa.
- (3) $E_i \leftarrow t_i$ for every basic operation i .
- (4) Execute Module IIIa.
- (5) $E_i \leftarrow t_i$ for every basic operation i .
- (6) If the starting times of some basic operations are changed at step (4), return to step (2).
- (7) Stop the calculation.

Figure 10 shows a part of the schedule which satisfies all of the constraints. Note in this simulation that the upper bound of the holding time in a batch unit is not considered. Therefore, some jobs are held in a batch unit for a long time period without executing any basic operations (operations circled in Fig. 10). In order to reduce idle holding time in batch units, simulation algorithm A is revised.

<Simulation Algorithm B>

- (1) to (6) Same as Algorithm A.
- (7) $L_i \leftarrow t_i$ for basic operation which corresponds to the last basic operation of each job.
- (8) Execute Combined Module I-IIb.
- (9) $L_i \leftarrow t_i$ for every basic operation i .
- (10) Execute Module IIIb.
- (11) $L_i \leftarrow t_i$ for every basic operation i .
- (12) If the starting times of some basic operations are changed at step (10), then return to step (8). Otherwise, stop the calculation.

Figure 11 shows a schedule derived by executing Algorithm B. The completion time of each job calculated by Algorithm B is the same as that of Algorithm A. However, the starting times of some basic operations are delayed so as to reduce the holding period in the batch unit. The upper bound of the manpower available at each time and the values actually required are also shown in Fig. 11.

Conclusion

In a batch plant, even if the processing order of the jobs in each unit is fixed, it is very difficult to determine the optimal starting times of those jobs for each unit in such a way that all the given constraints are simultaneously satisfied. In this study, a simulation system, which can be widely used in determining the starting times of operations for general batch plants, was proposed.

The operations of a batch plant are first expressed by a sequence of basic operations such as filling, processing, discharging and cleaning. By using these basic operations, it is possible to uniformly handle many different types of storage policies. The constraints on finite intermediate storage were the hardest to handle in the simulation algorithm. In this study, it is shown that those constraints can be converted into much simpler constraints on precedence relationships of basic operations.

A variety of constraints are grouped into several types, and for each type of constraint, two different kinds of simulation module have been prepared: one is the module for calculating the earliest starting times of basic operations and the other for calculating the latest starting times. A simulation program of a resin production plant has been developed by combining these simulation modules. The results show that a similar simulation program to calculate the starting times of jobs which satisfy various types of constraints may be easily developed for other practical problems as well.

Types of constraints and objective functions can be quite different for each plant and they may also change under different economic conditions. Furthermore, a simulation system is required of the scheduling system to generate schedules with varying degrees of precision and speed. In the proposed method, a simulation program consists of several subroutines called simulation modules, and the combination of these modules can be easily modified so as to satisfy the requirements of the scheduling system. Therefore, the proposed simulation method can be widely used to develop scheduling and/or planning systems of general batch plants.

Appendix

<Module 1b : latest starting time calculation>

(1) $t_i \leftarrow L_i$ for every basic operation i .

where L_i : temporary latest starting time of B.O. i which is given as input data.

(2) Repeat steps (3) to (7) for every basic operation.

(3) Select a basic operation (B.O. J), and repeat steps (4) to (6) for all of the basic operations which precede B.O. J .

(4) Select a basic operation (B.O. M) which precedes B.O. J .

(5) Change the starting times of B.O. J and B.O. M if they satisfy the following conditions:

If $t_M + h'_{MJ} < t_J$, then $t_J \leftarrow t_M + h'_{MJ}$.

If $t_J - h_{MJ} < t_M$, then $t_M \leftarrow t_J - h_{MJ}$.

(6) If $t_J < T_{min}$, then go to step (9).

where T_{min} : Starting time of the scheduling period given as input data.

(7) If the starting times of some basic operations are changed during steps (2) to (6), return to step (2).

(8) Stop the calculation.

(9) Stop the calculation. In this case, there is no feasible solution.

Nomenclature

E_i	= temporary earliest starting time of basic operation i	[h]
H^k	= number of jobs obtained by Eq. (2) for job k	[-]
h_{ij}, h'_{ij}	= time determined as a function of basic operations i and j	[h]
L_i	= temporary latest starting time of basic operation i	[h]
p^k_{if}	= time required for filling operation of job k at unit i	[h]
S_{ij}	= number of storage tanks between unit i and unit j	[-]
t_i	= starting time of basic operation i	[h]
t^k_{if}	= starting time of filling operation of job k to unit i	[h]
t^k_{id}	= starting time of discharging operation of job k from unit i	[h]
T_{max}	= ending time of scheduling period given as input data	[h]

Literature Cited

- Egri, U. M. and D. W. T. Rippin; "Short-term Scheduling for Multiproduct Batch Chemical Plants," *Comp. & Chem. Eng.*, **10**, 303-325 (1986)
- Kondili, E., C. C. Pantelides and R. W. H. Sargent; "A General Algorithm for Scheduling Batch Operations," *Proc. of PSE'88*, Sydney, 62-75 (1988)
- Rajagopalan, D. and I. A. Karimi; "Completion Times in Serial Mixed-Storage Multiproduct Processes with Transfer and Set-up Times," *Comp. & Chem. Eng.*, **13**, 175-186 (1989)
- Suhami, I. and R. S. H. Mah; "An Implicit Enumeration Scheme for the Flowshop Problem with No Intermediate Storage," *Comp. & Chem. Eng.*, **5**, 83-91 (1981)
- Wiede Jr, W. and G. V. Reklaitis; "Determination of Completion Times for Serial Multiproduct Processes-3. Mixed Intermediate Storage Systems," *Comp. & Chem. Eng.*, **11**, 357-368 (1987)