PAPER *Special Section on Concurrent/Hybrid Systems: Theory and Applications*

Integration between Scheduling and Design of Batch Systems Based on Petri Net Models

Takashi ITO[†], ***Nonmember***, **Susumu HASHIZUME**[†a)], ***Member***, **Tomoyuki YAJIMA**[†b)], ***Nonmember***, ***and*** **Katsuaki ONOGI**[†c)], ***Member***

SUMMARY A batch process is a discontinuous and concurrent process which is suitable for multi-product, small-sized production. The distinctive feature of a batch process is that various decision making processes, such as scheduling, design, operation, etc. are strongly connected with each other. Interaction among these processes is necessary to dynamically and flexibly cope with a variety of unplanned events. This paper aims at presenting a batch scheduling technique based on Petri net models and showing the possibilities of integration between scheduling and design of batch processes. For this purpose, it first views the behavior of a batch operating system as a discrete event system and presents a Petri net model to be used for scheduling, design and operation. It next formulates batch scheduling problems based on Petri net partial languages, proposes their solution technique and last discusses the integration between scheduling and design of batch systems.

key words: *scheduling, batch process, concurrent system, Petri net, integration*

1. Introduction

A batch process is a discontinuous process from which the output product appears in discrete batches or quantities. It is also a concurrent process which consists of many interacting units to carry out tasks in parallel on several batches at the same time. A batch operating system provides for batch production by controlling variables in the regulatory control, discrete control and sequential control domains [1]. Since multiple processing functions are often performed in one batch unit, batch processes are suitable for multi-product, small-sized production. To fully exhibit the ability of batch processes, a batch operating system must also be flexible enough to cope with a variety of unplanned events, such as arrival of new orders, change of production plans, alteration of control strategies, etc. To develop consistent and flexible batch operating systems, it is desired that scheduling, design and operation are mutually performed in close cooperation. Integration between scheduling, design and operation makes the dynamic and flexible batch production possible. However, integration is not only achieved through sharing of information, but also through utilization of common models used for scheduling, design and operation. Such models must, at least, clearly and correctly describe the behavior of a batch process. The final goal of our study on batch systems is to develop a methodology for integration between scheduling, design and operation based on common models.

Systematic approaches to design of batch systems have not made the remarkable progress. A major difficulty in treating batch processes is the lack of modeling techniques. Although ISA's SP88 Committee leads a significant amount of work around standards for batch control in the international arena, it intends only to establish the frameworks called S88 standards for batch control [2], [3]. Recently, some work on batch controller synthesis based on *Discrete Event System Theory* was done. Sanchez applied the *Supervisory Control Theory* to batch control. He did not, however, deal with concurrent nature in batch processes, because his model was based on formal languages and automata [4]. Tittus also applied the *Supervisory Control Theory* to synthesis of hierarchical batch controllers, but the problem how to generate a control specification was not solved [5]. Although Viswanathan et al. developed a framework for synthesizing control procedures for batch processes, their model, Grafchart, could not explicitly represent the states of batch processes [6], [7]. Owing to both modeling power and decision power, a Petri net is a candidate for a common model of batch system. Wang et al. developed a method of generating batch control procedures on the basis of Petri nets. They, however, separated batch control problem from batch scheduling problem [8]. We formulated a discrete event system design problem based on Petri net partial languages and applied it to batch system design problem [9]–[12]. We also proposed a fault detection technique for batch systems described by Petri nets [13], [14].

The large number of raw materials and products, the short lead times of product orders, and the constraints imposed on production equipment and quality considerations make batch scheduling difficult. Batch production requires greater scheduling opportunities than does continuous production. In addition, it creates the need to reschedule because of various unplanned events [1]. Much work has been done on batch scheduling and various techniques, mathematical programming based, simulation based, knowledge based techniques, have been applied [15], [16]. In order to dynamically and flexibly cope with unplanned events, a batch scheduling system is directly connected with other systems including process control, process monitoring, fault

Manuscript received March 30, 2005.
Manuscript revised June 6, 2005.
Final manuscript received July 11, 2005.

[†]The authors are with the Department of Chemical Engineering, Nagoya University, Nagoya-shi, 464-8603 Japan.
a) E-mail: hashi@nuce.nagoya-u.ac.jp
b) E-mail: yajima@nuce.nagoya-u.ac.jp
c) E-mail: onogi@nuce.nagoya-u.ac.jp

DOI: 10.1093/ietfec/e88–a.11.2989

detecting systems, etc. Utilization of common models makes this possible. In this paper we present a batch scheduling technique based on Petri net common models and show the possibilities of integration between scheduling and design of batch systems.

First, we present a common model of batch system in Sect. 2 and show batch operation using it in Sect. 3. Next, we formulate batch scheduling problems based on the model in Sect. 4 and propose their solution technique in Sect. 5. Last, we show an example and discuss the possibilities of integration in Sect. 6 and in Sect. 7.

2. Modeling of Batch Operating System

2.1 Specification of Batch System

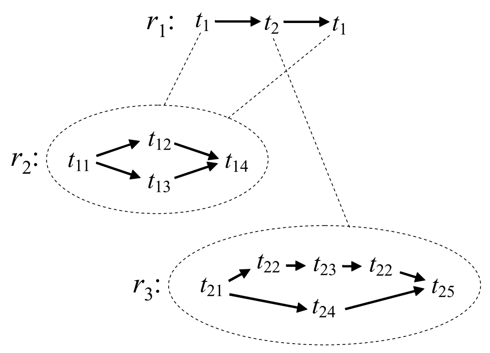
To easily cope with the frequent changes of product orders, production plans, control strategies, etc. information necessary to design of batch operating systems must be managed in a systematized manner. Being conformable to the S88 standards for batch control stated in Sect. 1, we classified the information into the following three categories [12]:

(1) Process information
The process information called a **recipe** contains the knowledge about process-related tasks to define the production requirements for a specific product, but not the knowledge about a plant to be used.

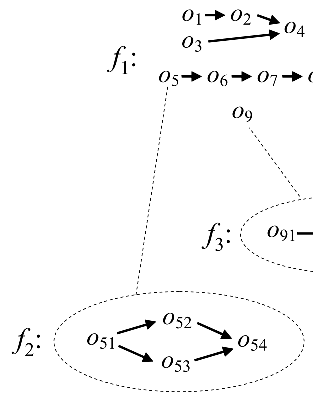
(2) Plant information
The plant information includes a potentially available piece of processing/storage equipment which is used for batch production. It contains the knowledge about operations of processing units, storage vessels and actuators.

(3) Schedule information
The schedule information called a **production plan** contains the long-term schedule information as when or in what order the products are to be produced, and what equipment is to be used. It links the process information to the plant information.

Since batch control handles logical aspects of operations, the behavior of a batch operating system cannot be described by differential equations. If we view that arrival of raw material, start of processing, completion of processing, discharge of product, etc. are events, the evolution of a batch system can be regarded as a discrete event system. In the previous papers we defined **process** and **plant specifications** as partially ordered sets of tasks and operations, respectively [12]–[14]. Each specification is a set of graphs associated with one another in a hierarchical manner as shown in Fig. 1. Every graph consists of nodes representing tasks (operations) and arcs representing precedence relations among the tasks (operations). A graph at higher level defines the rough outline of major tasks (operations), such as reaction, separation, heating, filtration, drying, etc. A high-level task (operation) may be developed into a partially ordered set of low-level tasks (operations). Every low-level


r_1: $t_1 \to t_2 \to t_1$
r_2: t_{11}, t_{12}, t_{13}, t_{14}
r_3: t_{21}, t_{22}, t_{23}, t_{22}, t_{24}, t_{25}

(a) Process specification.


f_1: $o_1 \to o_2 \to o_4$, $o_3 \to o_4$
$o_5 \to o_6 \to o_7 \to o_8$
o_9
f_3: $o_{91} \to o_{92} \to o_{93}$
f_2: o_{51}, o_{52}, o_{53}, o_{54}

(b) Plant specification.

$t_{11} \mapsto o_5$	$t_{21} \mapsto o_3$
$t_{12} \mapsto o_7$	$t_{22} \mapsto o_9$
$t_{13} \mapsto o_6$	$t_{23} \mapsto o_8$
$t_{14} \mapsto o_1$	$t_{24} \mapsto o_2$
	$t_{25} \mapsto o_4$

(c) Schedule specification.

Fig. 1 Specification.

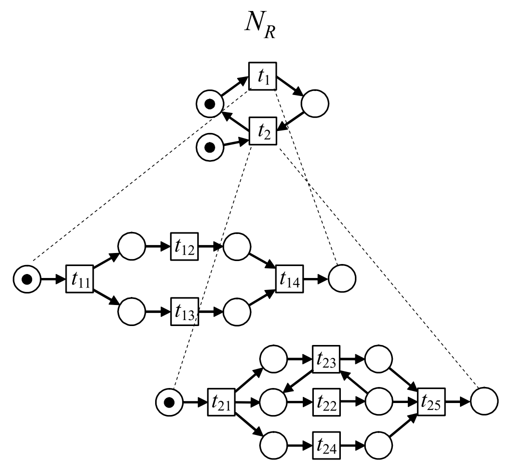
graph defines the refinement of a high-level task (operation).

In Fig. 1(a) the recipe r_1 implies that two tasks t_1 and t_2 are sequentially carried out and again t_1 is carried out. The recipe r_2 implies that after the execution of task t_{11} two tasks t_{12} and t_{13} are concurrently carried out and then t_{14} is carried out. The recipes r_2 and r_3 are the refinements of tasks t_1 and t_2, respectively. Figure 1(b) shows an example of plant specification.

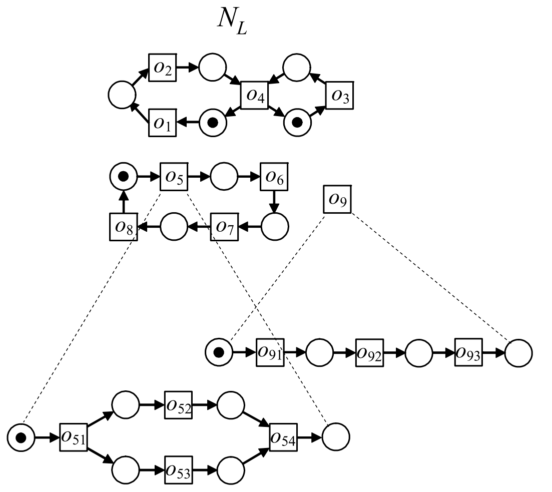
Tasks described in process specifications are carried out by operations described in plant specifications. We defined **schedule specification** as a correspondence from a set of tasks to that of operations. Figure 1(c) shows an example of schedule specification. For example, $t_{11} \mapsto o_5$ indicates that the task t_{11} is carried out by the operation o_5.

2.2 Common Model of Batch System

Since a Petri net is a useful modeling tool to describe a discrete event system with interacting concurrent components, we applied it to representation of the above specifications. Grabowski argued that a partial language could correctly describe the concurrent behavior of a Petri net [17]. A partial



(a) Process model.



(b) Plant model.

Fig. 2 Model.

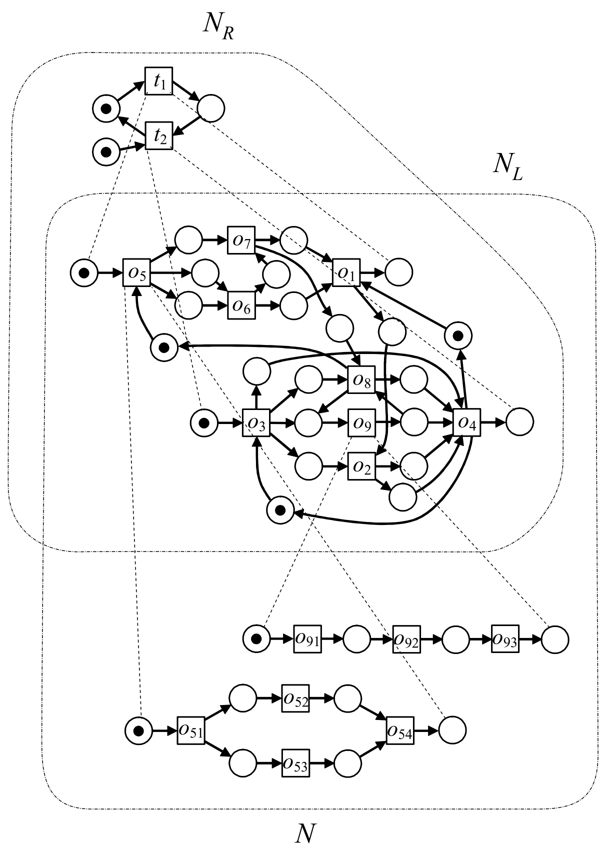


Fig. 3 Common model of batch operating system.

language is a set of partial words each of which is a partially ordered multiset over a set of transitions. Based on partial languages, we investigated the following Petri net construction problem [9], [10]:

***Petri net construction problem*:**
Given a partial language as a specification, construct a Petri net whose partial language is consistent with the specification. ∎

Process and plant models are obtained from process and plant specifications, respectively, by solving *Petri net construction problems*. The hierarchical Petri nets N_R and N_L in Fig. 2 guarantee the desired behavior described in the process and plant specifications in Fig. 1. Schedule specification is also transformed into a correspondence Γ which maps the transitions of N_R to those of N_L, namely

$$\Gamma : T(N_R) \to T(N_L)$$

where $T(N_R)$ and $T(N_L)$ are sets of transitions of N_R and that of N_L, respectively.

A common model of batch operating system is constructed from process and plant models and a schedule specification, according to the following procedure. Figure 3 shows a common model N constructed from N_R and N_L in Fig. 2 and Γ in Fig. 1(c).

***Common model construction procedure*:**
First, a hierarchical Petri net based process model N_R and a plant model N_L are constructed by solving *Petri net construction problems*. Second, the correspondence $\Gamma : T(N_R) \to T(N_L)$ is determined using schedule specification. Then, a Petri net $N_R \oplus N_L$ is constructed by piling the corresponding transitions in $T(N_R)$ and those in $T(N_L)$ according to Γ. Last, unnecessary transitions, places and arcs are eliminated from $N_R \oplus N_L$, in order to obtain a common model N of batch operating system. ∎

3. Batch Operation Based on Common Model

3.1 Verification of Control Procedure

To transfer the state of a batch process from a specified initial state to a desired final state, control procedures must be consistent. We applied the reachability analysis to verification of validity of batch system models. If a desired final marking $\boldsymbol{M}_f$ is reachable from a specified initial marking $\boldsymbol{M}_0$, then

$$\boldsymbol{M}_f = \boldsymbol{M}_0 + \boldsymbol{D}\boldsymbol{x} \tag{1}$$

where $\boldsymbol{D}$ is the incidence matrix of a Petri net common model N whose entry d_{ij} denotes the change of the number of tokens in the i-th place as the result of firing j-th transition, and $\boldsymbol{x}$ is a nonnegative integer vector whose entry x_j denotes the number of times that j-th transition must fire to transform $\boldsymbol{M}_0$ to $\boldsymbol{M}_f$. Necessary condition for control procedures to be consistent is that Eq. (1) has solutions, in nonnegative integers.

Since all process variables are not always observable, the final state is not completely specified in advance of batch production, that is, all the entries of $\boldsymbol{M}_f$ are not necessarily known. We presented an approach for verifying validity of common models of batch systems with unobservable process variables [12].

3.2 Detection of Process Fault

A Petri net based common model of batch system shown in Sect. 2 is not only used for design of batch control, but also for fault detection of batch processes. For any marking $\boldsymbol{M}$ which is reachable from $\boldsymbol{M}_0$,

$$ {}^t\boldsymbol{y}\boldsymbol{M} = {}^t\boldsymbol{y}\boldsymbol{M}_0 \tag{2}$$

where $\boldsymbol{y}$ is a place-invariant (p-invariant) and ${}^t\boldsymbol{y}$ denotes the transpose of $\boldsymbol{y}$. Equation (2) shows a restriction on control actions. We call this type of constraint expressed by Eq. (2) an **operational constraint**. An operational constraint is a kind of safety interlocks to prevent unsafe operations of units. As an example of operational constraint, a feed valve to a reactor must not open when the level in the reactor is above the specified height. Operational constraints require process variable monitoring. Every time any process variable is measured, the marking $\boldsymbol{M}$ in Eq. (2) is renewed. Basic idea of fault detection is to verify if Eq. (2) holds or not with every renewal of marking. If Eq. (2) does not hold then a fault may have occurred. We also presented an approach to fault detection when there existed unobservable process variables [14].

The set of places corresponding to nonzero entries in a p-invariant $\boldsymbol{y}$ is called the **support of $\boldsymbol{y}$** and denoted by $\|\boldsymbol{y}\|$. If k operational constraints do not hold, we can limit the causes of fault to the process variables relative to $\bigcap_{l=1}^{k} \|\boldsymbol{y}_l\|$, where $\boldsymbol{y}_l$ is the p-invariant associated with the l-th operational constraint.

4. Batch Scheduling Based on Common Model

4.1 Formulation of Scheduling Problem

Scheduling is needed at different levels. Scheduling at higher level is often called production planning. Various mathematical programming techniques can be used for production planning, where the horizons may be measured in weeks or months. On the other hand, at lower level, dynamic and flexible scheduling is necessary to cope with unplanned events. In this paper we investigate the scheduling problem at lower level, based on the common model used for operation in Sect. 3.

After this, we are interested in only task routings which transfer the state of a batch system from a specified initial state to a desired final state. We call such a task routing a **feasible task routing**. A class of batch scheduling problems we consider is as follows:

***Optimal task routing search problem P*:**
Given an initial state and a set of desired final states of a batch system, determine optimal feasible task routings and values of process variables which attain an objective. ▮

Since the Petri net N in Sect. 2 models a batch operating system which produces specified products according to process, plant and schedule specifications, its partial language represents the task (operation) routings potentially available to batch production. We call a partial word which transfers the marking of the net N from a specified initial marking to a desired final making, a **feasible partial word of N**. Then *Problem P* is transformed into the following problem:

***Optimal partial word search problem Q*:**
Given an initial marking and a set of desired final markings of a Petri net common model, determine optimal feasible partial words and values of decision variables which attain an objective. ▮

Problem Q is mathematically reformulated as the following optimization problem:

***Optimal partial word search problem Q′*:**

$$\min_{p_w,\boldsymbol{u}} f(p_w, \boldsymbol{u})$$
$$\text{subject to} \quad p_w \in \Pi(N) \text{ and } \boldsymbol{u} \in \Omega$$

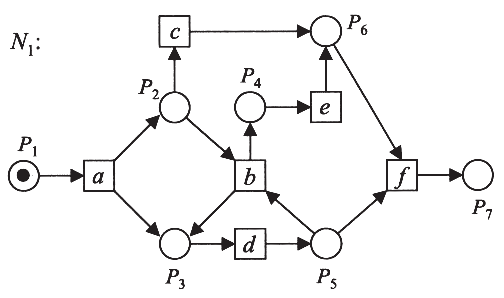
where f is an objective function, p_w a feasible partial word of a Petri net N, $\Pi(N)$ a set of feasible partial words of N, $\boldsymbol{u}$ a decision vector, and Ω a feasible region of $\boldsymbol{u}$. ▮

It should be noted that p_w and $\boldsymbol{u}$ are not, in general, independent to each other. It assumes that decision variables are associated with transitions of the net N. This implies that only executions of tasks (operations) affect the objective of the system. If a decision variable u is associated with a transition t, an objective function f is evaluated every time t fires. For example, let t, u and f be a heating task, heating temperature and the total amount of heat energy consumption, respectively. Then the heat quantity obtained from the current value of u is evaluated, every time the heating task is carried out.

4.2 Search of Optimal Task Routing

Problem Q′ can be solved, in principle, by finding all feasible partial words. Various graph search algorithms, such as the shortest directed path algorithm, etc. are applied to *Problem Q′*, according to the types of problems.

To show how to solve *Problem Q′*, we consider the Petri net N_1 shown in Fig. 4. The transitions of the net N_1 represent the tasks. The processing times of each task are



$M_0 = {}^t(1\,0\,0\,0\,0\,0\,0)$, $M_f = {}^t(0\,0\,0\,0\,0\,0\,1)$
$F(N_1) = \{M_f\}$

Fig. 4 Petri net model.

Table 1 Task and its processing time.

Task	Processing time
a	2
b	1
c	5
d	$u_1 (\geq 3)$
e	$u_2 (\geq 2)$
f	4

p_{w1} : $$\begin{tikzcd} \overset{2}{a} \arrow[r] & \overset{5}{c} \arrow[r] & \overset{u_1}{d} \arrow[r] & \overset{4}{f} \end{tikzcd}$$

p_{w2} : $$\begin{tikzcd} \overset{2}{a} \arrow[r] & \overset{u_1}{d} \arrow[r] & \overset{5}{c} \arrow[r] & \overset{4}{f} \end{tikzcd}$$

p_{w3} : $$\begin{tikzcd} & \overset{5}{c} \arrow[dr] & \\ \overset{2}{a} \arrow[ur] \arrow[dr] & & \overset{4}{f} \\ & \underset{u_1}{d} \arrow[ur] & \end{tikzcd}$$

p_{w4} : $$\begin{tikzcd} \overset{2}{a} \arrow[r] & \overset{u_1}{d} \arrow[r] & \overset{1}{b} \arrow[r] & \overset{u_1}{d} \arrow[r] & \overset{u_2}{e} \arrow[r] & \overset{4}{f} \end{tikzcd}$$

p_{w5} : $$\begin{tikzcd} \overset{2}{a} \arrow[r] & \overset{u_1}{d} \arrow[r] & \overset{1}{b} \arrow[r] & \overset{u_2}{e} \arrow[r] & \overset{u_1}{d} \arrow[r] & \overset{4}{f} \end{tikzcd}$$

p_{w6} : $$\begin{tikzcd} & & & \overset{u_1}{d} \arrow[dr] & \\ \overset{2}{a} \arrow[r] & \overset{u_1}{d} \arrow[r] & \overset{1}{b} \arrow[ur] \arrow[dr] & & \overset{4}{f} \\ & & & \underset{u_2}{e} \arrow[ur] & \end{tikzcd}$$

Fig. 5 Feasible partial words of N_1.

given as shown in Table 1, where two processing times, u_1 and u_2 are variable. The problem to be solved is as follows:

Example 1:
Given an initial marking $\boldsymbol{M}_0$ and a set of desired final markings, $F(N_1) = \{\boldsymbol{M}_f\}$ as shown in Fig. 4, find optimal feasible partial words and values of u_1 and u_2 which minimize the duration time required to transform $\boldsymbol{M}_0$ to $\boldsymbol{M}_f$. ∎

Figure 5 shows a set of feasible partial words of N_1, $\Pi(N_1)$. Every feasible partial word is a directed acyclic graph with the source node a and the sink node f. The real number representing processing times is assigned to each node. The total processing time associated with a directed path is defined by the completion time of the last task in the path. We call a directed path from a source node to a sink node with the maximum total processing time, a **critical path**. The duration time associated with p_w is determined by searching the critical path of p_w. Therefore, the optimal solution of *Example 1* can be obtained by seeking the critical paths of each feasible partial word. Table 2 shows the duration times of task routings denoted by each feasible partial word in Fig. 5. The optimal solution is given by p_{w3} and $3 \leq u_1 \leq 5$, regardless of u_2.

Table 2 Feasible task routing and its duration time.

Feasible task routing	Duration time
p_{w1}	14
p_{w2}	14
p_{w3}	11
p_{w4}	15
p_{w5}	15
p_{w6}	13

The solution technique stated above is based on the enumeration of feasible partial words. The number of feasible partial words, however, increases as the scale of a batch system increases. To overcome these difficulties, an efficient solution technique is necessary. We will present a method of solving *Problem Q′* in Sect. 5.

4.3 Execution of Optimal Task Routing

Once the optimal task routing is determined, it next must be carried out. One of the aims of this paper is to present the possibilities of integration between scheduling and design based on Petri net common models. To perform the optimal task routing, we must avoid the executions of undesired task routings by use of logic controllers. Since places of a Petri net represent conditions necessary for firings, an addition of new places, in general, restricts the behavior of the net. Therefore, the executions of undesired task routings may be avoided by suitably introducing additional places into the original net. The problem how to introduce additional places is formulated as the following problem:

***Control logic synthesis problem S*:**
Given a Petri net N and its optimal feasible partial word p_w^*, construct a Petri net N^* which satisfies the following conditions:

(1) N^* generates p_w^*, but not undesired feasible partial words in $\Pi(N)$.
(2) N is a subnet of N^*. ∎

Let N_C be a net satisfying

$$N^* = N \oplus N_C \tag{3}$$

then the subnet N_C represents the synthesized control logics. The solution techniques for *Petri net construction problem* in Sect. 2 can be used for the solution to *Problem S* [11].

For example, in order to perform the optimal task routing described by p_{w3} in Fig. 5, we must prohibit firing b. The sequence of transitions firings ad transforms the initial marking $\boldsymbol{M}_0$ of the net N_1 to $\boldsymbol{M}_1$ shown in Fig. 6. Two transitions c and b are enabled at $\boldsymbol{M}_1$. The prohibition of firing b is achieved by introducing a new place P_c connected with b and c into the net N_1 as shown in Fig. 6. The part of the net

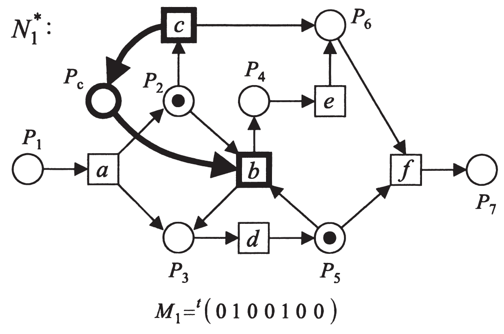


Fig. 6 Control logic synthesis.

N_1^* depicted by a heavy line represents a new control logic. The transition b never fires due to the place P_c. In this way, design of control systems is performed based on the same model used for scheduling.

5. Batch Scheduling Algorithm

As stated above, the number of feasible task routings may become large even a simple system. It is, however, seriously reduced with a few devices. A bounding process has the effect of reducing a search space. It eliminates the unpromising solutions from further consideration but still preserves all possible solutions.

An example of search process is illustrated in Fig. 7. A partial word generated by a Petri net model is assigned to each node of the search tree. We denote the partial word assigned to node i by σ_i. The node label σ_i represents a set of the partial words PW_i such that

$$PW_i = \{p \mid \sigma_i \in p^{PREF}\}$$

where p^{PREF} is a set of prefixes of partial word p. For example,

$$PW_4 = \left\{ \begin{array}{l} t_1 \to t_4,\ t_1 \to t_4 \to t_3,\ t_1 \to t_4 \to t_5, \\ t_1 \genfrac{}{}{0pt}{}{\nearrow t_4}{\searrow t_5},\ \cdots \end{array} \right\}$$

The proposed algorithm for finding optimal solutions of *Problem Q′* is as follows:

***Optimal partial word search algorithm*:**
Branching continues until a feasible partial word is obtained. If, in the process, a partial word assigned to a node yields a deadlock marking, branching stops at the node. If a new feasible partial word is found having a value of the objective function f less than the current best value, then this value is stored as the new best value. To avoid the excessive branching, a lower bound of f is estimated at every node. The lower bound may be obtained by relaxing the constraints of *Problem Q′* and solving the resulting relaxed problem. If the lower bound is smaller than the current best value of f, branching continues. Otherwise, the further branching from the current node is not performed and backtracking to one of the promising nodes is carried out. When the backtracking takes us no promising nodes, the search is

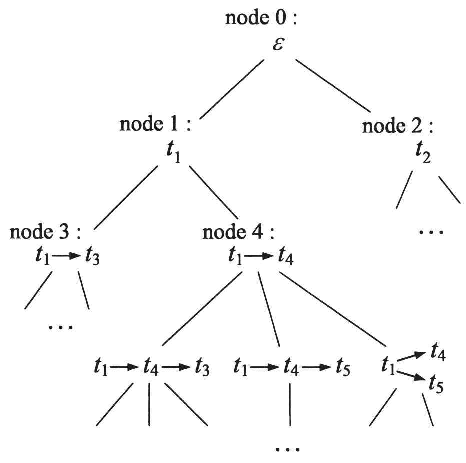


Fig. 7 Search tree.

complete. The strategy for estimation of lower bounds must be worked out according to the types of the problems. ∎

6. Integration between Scheduling and Design

In this section we show an example of integration between scheduling and design.

Problem description:

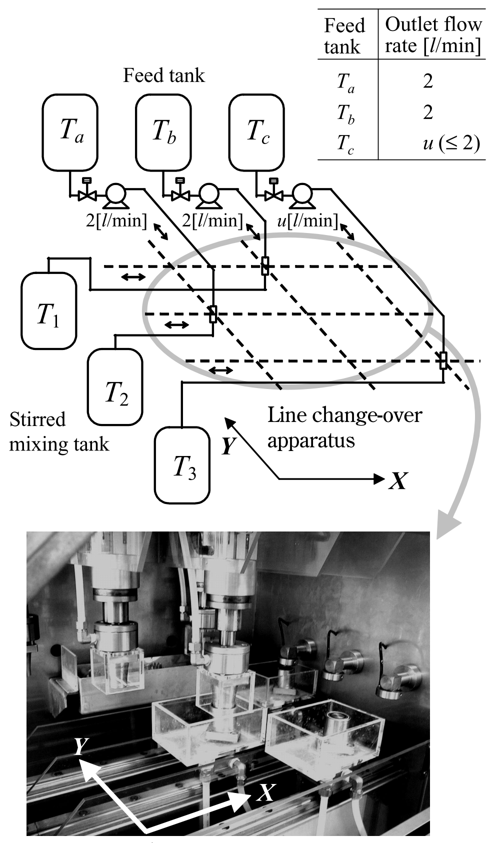
We consider the mixing equipment illustrated in Fig. 8 which is a custom-designed experimental plant installed in our laboratory. It consists of three feed tanks, three stirred mixing tanks and an automated line change-over apparatus. The line change-over apparatus used there is the Toyo Engineering Corporation's original patented 'XY Router.' It is composed of two sub-blocks, each of which incorporates a hose, traveling head and automatic positioning driving mechanism. Each of the traveling heads contains a coupling and docking device. The device of the lower sub-blocks moves in direction-X, and the device of the upper sub-blocks moves in direction-Y. These two devices are positioned to face each other at a crossing point and then interlocked to build a flow passage. Simultaneous supply on two or more mixing tanks from one feed tank, or simultaneous supply on one mixing tank from two or more feed tanks cannot be performed. Multiple flow passages can, however, be built in parallel. For example, three flow passages, T_a–T_2, T_b–T_1 and T_c–T_3 are simultaneously built as shown in Fig. 8.

The problem to be considered is as follows:

Example 2:
Using the equipment shown in Fig. 8, three products are produced according to the recipe in Table 3. The problem is to find the optimal task routings and outlet flow rate from the feed tank T_c which minimize the completion time of the last task, provided that every product is produced without an interruption. ∎

Common model:

The process model representing the recipe in Table 3 is shown in Fig. 9(a). It implies that all tasks can be carried out



Feed tank	Outlet flow rate [l/min]
T_a	2
T_b	2
T_c	u (≤ 2)

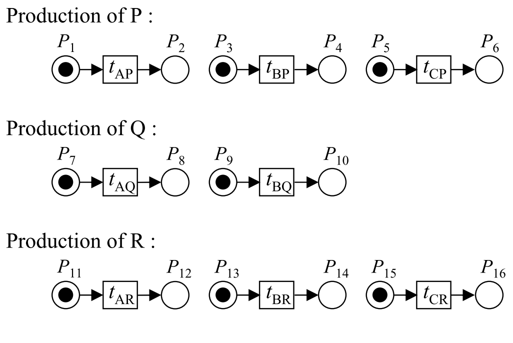
Line change-over apparatus

Fig. 8 Mixing equipment.

Table 3 Recipe.

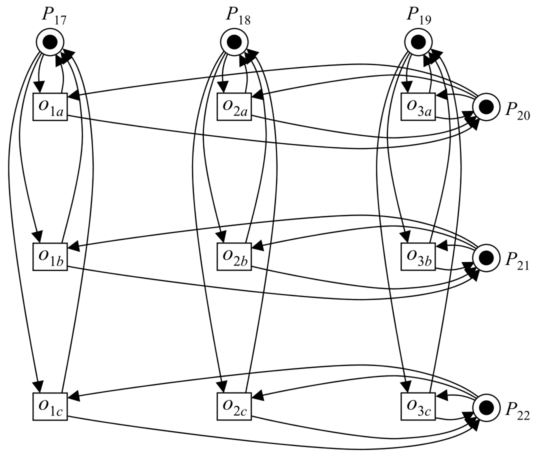
		Raw material		
		A	B	C
Product	P	4	2	6
	Q	6	2	–
	R	3	8	3

The numbers represent the amount of each row material. [l]



t_{XY} : Supply of X to produce Y

(a) Process model.



o_{ij} : Supply on stirred mixing tank T_i from feed tank T_j

(b) Plant model.

$$
\begin{array}{lll}
t_{AP} \mapsto o_{1a} & t_{BP} \mapsto o_{1b} & t_{CP} \mapsto o_{1c} \\
t_{AQ} \mapsto o_{2a} & t_{BQ} \mapsto o_{2b} & \\
t_{AR} \mapsto o_{3a} & t_{BR} \mapsto o_{3b} & t_{CR} \mapsto o_{3c}
\end{array}
$$

(c) Schedule specification.

Fig. 9 Model and specification.

independently to each other. The plant model representing the equipment in Fig. 8 is also shown in Fig. 9(b). It implies that some sets of flow passages can be built in parallel and some not. If the raw materials, A, B and C are stored in the tanks, T_a, T_b and T_c, respectively, and the products, P, Q and R are produced in the tanks, T_1, T_2, and T_3, respectively, then the schedule specification is given by the correspondence in Fig. 9(c). Figure 10 shows the common model obtained from such models and specification. The initial marking $\boldsymbol{M}_0$ represents that the production of P, Q and R can start, and the final marking $\boldsymbol{M}_f$ that the production of all products is complete.

Optimal solution:

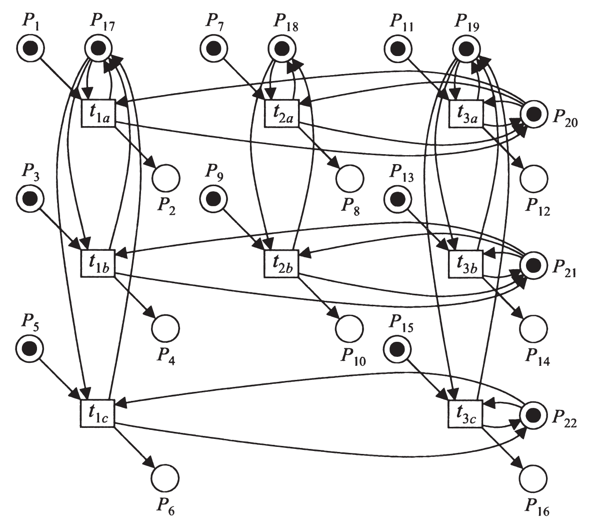
The Petri net shown in Fig. 10 generates about 7300 feasible partial words. Several feasible partial words are presented in Fig. 11. By applying the optimal partial word search algorithm proposed in Sect. 5 to the net, the number of feasible partial word evaluations can be reduced to one-sixtieth. The optimal task routing is shown in Fig. 12 by the way of a Gantt chart. The chart suggests that two or three tasks are carried out in parallel to produce all products more efficiently. The value of u is adjusted not to interrupt continuous productions.

To solve this problem, we estimated a lower bound LB_i at node i by the equation

$$LB_i = \min_{u} f(\sigma_i, u) + \min_{\tau_j, u} f(\tau_j, u) \tag{4}$$

The concatenation $\sigma_i \cdot \tau_j$ is a partial word σ_i followed by τ_j. To efficiently exhibit the effect of bounding, the feasibility of the partial word τ_j is not necessarily needed. The first term in the right hand side of Eq. (4) is estimated by use of linear programming approach. On the other hand, the second term is estimated by assuming that the remaining tasks in τ_j are to be carried out in parallel as soon as possible.

Control logic synthesis:



t_{ij} : Task to be carried out by operation o_{ij} (Supply on T_i from T_j)

$$M_0 = {}^t(1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 1\ 1\ 1)$$
$$M_f = {}^t(0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1)$$

Fig. 10 Petri net common model.

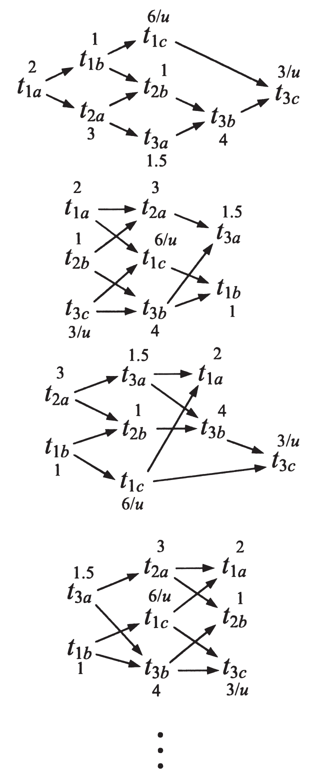


Fig. 11 Feasible partial word.

We realize the optimal task routing obtained above by solving *Control logic synthesis problem S* defined in Sect. 4. The only part of the net depicted by a heavy line in Fig. 13 represents additional logics to control the equipment according to the optimal task routing. There is no measure of time in a Petri net but the tasks (operations) represented by transitions take variable amounts of time in real-world.

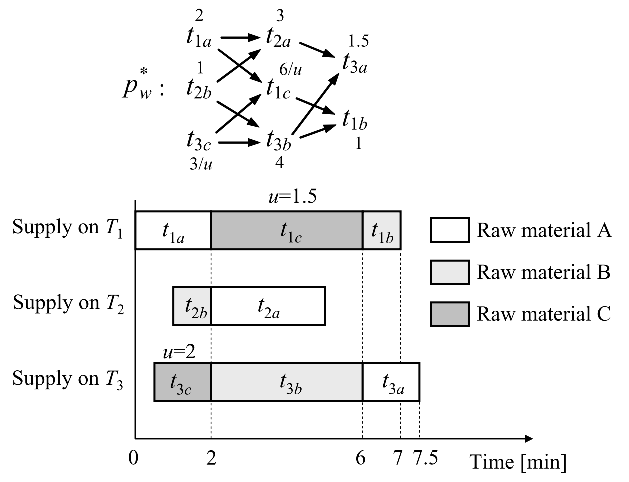


Fig. 12 Optimal task routing.

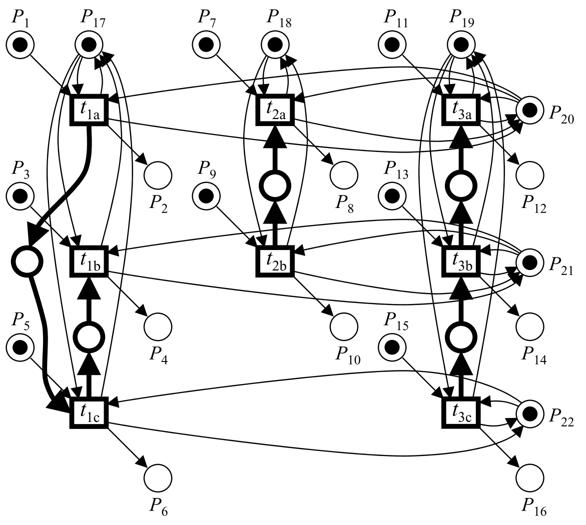


Fig. 13 Controller synthesis.

Therefore, lapse of time is necessary for controller synthesis. The actual controllers are constructed using the Condition/Event net based Sequence Controller Synthesis Support Tool, CESeC which we had developed [11].

7. Discussion

Scheduling of batch processes needs various information flows. The production planning process attempts to optimize the overall long-term resource allocation based on orders, due dates, resource requirements, etc. Decisions made at this higher planning level affect the lower level scheduling process. After a production plan has been generated, it is necessary to determine short-term schedules including production orders, lot sizing, control strategies, etc. This lower level scheduling process must interact with other decision making processes, such as control, operation, etc. A distinctive feature of batch processes is that each decision making process is strongly connected with other processes. Interaction among these decision making processes is necessary to dynamically and flexibly cope with unplanned events.

The design of a large system proceeds to successively lower levels of the system. A hierarchical design approach involves first specifying a rough outline of a system and then successively refining the outline. In this case, scheduling is also carried out in a hierarchical manner. The master problem at higher level generates a solution corresponding to a rough overall schedule. The subproblems at lower level receive the overall schedule from the master problem, generate their solutions corresponding to refined subschedules and send them to the master problem. The master problem again generates a solution by considering the sent subschedules. The iteration proceeds until a satisfactory solution is obtained.

The algorithm for searching optimal partial words in Sect. 5 is a kind of branch and bound algorithm. To efficiently shrink a search space we must appropriately estimate a lower bound of an objective function at every stage. A lower bound is often obtained from a solution to a relaxed problem without imposing that solution candidates be feasible. Equation (4) in Sect. 6 is such an example. The second term in the right hand side of Eq. (4) ignores the feasibility of a solution.

Empirical knowledge is also useful for the efficient reduction of a search space. For actual batch processes, flexibility of schedules may be important at the cost of optimality to some extent. In such cases, heuristics about understanding of bottlenecks, due dates, etc. may restrict the possible solutions to those that are not only possible but reasonable.

8. Conclusion

Scheduling, design, and operation of batch processes are strongly connected with each other. Integration among these decision making processes is important to manage batch production effectively. Integration for batch production is not only achieved through sharing of information, but also through utilization of common models used for various decision making processes. In this paper, we have formulated batch scheduling problems based on Petri net based common models and presented their solution technique. We have also shown the possibilities of integration for batch production. We conclude that the approach stated in this paper will provide a framework for dynamic and flexible batch production.

Acknowledgments

The authors gratefully acknowledge partial financial support from the Japan Society for the Promotion of Science through a Grant-in Aid for Scientific Research (B) (No. 16360394).

References

[1] T.G. Fisher, Batch Control Systems: Design, Application, and Implementation, Instrument Society of America, North Carolina, 1990.

[2] SP88 Committee, Batch Control Part 1: Models and Terminology, ISA—The Instrumentation, Systems, and Automation Society, North Carolina, 1995.

[3] SP88 Committee, Batch Control Part 2: Data Structures and Guidelines for Languages, ISA—The Instrumentation, Systems, and Automation Society, North Carolina, 2001.

[4] A. Sanchez, Lecture Notes in Control and Information Sciences 212, Springer-Verlag, London, 1996.

[5] M. Tittus and B. Lennartson, "Hierarchical supervisory control for batch processes," IEEE Trans. Control Syst. Technol., vol.7, no.5, pp.542–554, 1999.

[6] S. Viswanathan, C. Johnsson, R. Srinivasan, V. Venkatasubramanian, and K.E. Arzen, "Automating operating procedure synthesis for batch processes: Part I. Knowledge representation and planning framework," Computers and Chemical Engineering, vol.22, no.11, pp.1673–1685, 1998.

[7] S. Viswanathan, C. Johnsson, R. Srinivasan, V. Venkatasubramanian, and K.E. Arzen, "Automating operating procedure synthesis for batch processes: Part II. Implementation and application," Computers and Chemical Engineering, vol.22, no.11, pp.1687–1698, 1998.

[8] Y.F. Wang, H.H. Chou, and C.T. Chang, "Generation of batch operating procedures for multiple material-transfer tasks with Petri nets," Computers and Chemical Engineering, vol.29, no.8, pp.1822–1836, 2005.

[9] S. Hashizume, T. Suzuki, K. Onogi, and Y. Nishimura, "A construction problem of condition/event-nets and its solvability," Trans. SICE, vol.28, no.5, pp.632–639, 1992.

[10] S. Hashizume, Y. Mitsuyama, Y. Matsutani, K. Onogi, and Y. Nishimura, "Construction of Petri nets from a given partial language," IEICE Trans. Fundamentals, vol.E79-A, no.12, pp.2192–2195, Dec. 1996.

[11] K. Onogi, S. Hashizume, and Y. Nishimura, "Design of sequential control systems based on condition/event nets," Proc. International Symposium on Design, Operation and Control of Next Generation Chemical Plants (PSE Asia 2000), pp.261–266, Kyoto, Japan, Dec. 2000.

[12] T. Yajima, S. Hashizume, K. Onogi, and Y. Nishimura, "Establishment of operational procedures for batch control," Kagaku Kogaku Ronbunshu, vol.28, no.3, pp.262–267, 2002.

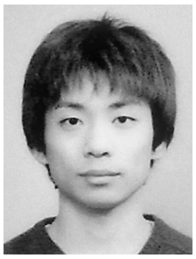
[13] S. Hashizume, T. Yajima, T. Ito, and K. Onogi, "Synthesis of operating procedures and procedural controllers for batch processes based on Petri nets," Proc. International Symposium on Design, Operation and Control of Chemical Processes (PSE Asia 2002), pp.253–258, Taipei, Taiwan, Dec. 2002.

[14] T. Yajima, T. Ito, S. Hashizume, H. Kurimoto, and K. Onogi, "Control of batch processes based on hierarchical Petri nets," IEICE Trans. Fundamentals, vol.E87-A, no.11, pp.2895–2904, Nov. 2004.

[15] A.E. Nisenfeld, Batch Control: Practical Guides for Measurement and Control, Instrument Society of America, North Carolina, 1996.

[16] X. Lin and C.A. Floudas, "Design, synthesis and scheduling of multipurpose batch plants via an effective continuous-time formulation," Computers and Chemical Engineering, vol.25, pp.665–674, 2001.

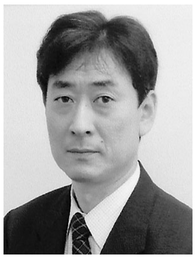
[17] J. Grabowski, "On partial language," Fundamenta Informaticae, vol.4, no.2, pp.427–498, 1981.



Takashi Ito received B.E. and M.E. degrees in chemical engineering from Nagoya University in 2001 and 2003, respectively. He is currently a doctor course student at Graduate School of Engineering, Nagoya University. His research interests include batch control system framework and scheduling of real-time systems. He is a member of IPSJ and SCEJ (Society of Chemical Engineers, Japan).

Susumu Hashizume received B.E., M.E. and Dr.(Eng.) degrees in systems engineering from Toyohashi University of Technology in 1986, 1988 and 1992, respectively. From 1992 to 1998, he was a reseach associate at the Department of Productin System Engineering, Toyohashi University of Technology. He is currently an assistant professor at the Department of Chemical Engineering, Nagoya University. His research interests include design of hybrid systems and control of discrete event systems. He is a member of SICE, IPSJ and SCEJ (Society of Chemical Engineers, Japan).

Tomoyuki Yajima received B.E., M.E. and Dr.(Eng.) degrees in chemical engineering from Nagoya University in 1989, 1991 and 1994, respectively. He is currently a reseach associate at the Department of Chemical Engineering, Nagoya University. His research interests include design of batch control systems and simulation of macroreactor. He is a member of SICE and SCEJ (Society of Chemical Engineers, Japan).



Katsuaki Onogi received B.E., M.E. and Dr.(Eng.) degrees in chemical engineering from Nagoya University in 1973, 1975 and 1980, respectively. From 1978 to 1980, he was a reseach associate at the Department of Chemical Engineering, Nagoya University. From 1980 to 1996, he was an assistant professor, associate professor and professor at the Department of Productin System Engineering, Toyohashi University of Technology. Since 1996, he has been a professor at the Department of Chemical Engineering, Nagoya University. His research interests include control of discrete event systems and design of batch control systems. He is a member of SICE, ISCIE and SCEJ (Society of Chemical Engineers, Japan).