Petri-Net-Based Strategy To Synthesize the Operating Procedures for Cleaning Pipeline Networks

Hung-Hsiang Chou and Chuei-Tin Chang*

Department of Chemical Engineering, National Cheng Kung University, Tainan, Taiwan 70101, Republic of China

Cleaning the pipeline network is a routine operation in almost every chemical plant. Traditionally, the tasks for synthesizing the needed operation steps are carried out manually on an ad hoc basis. This approach is often time-consuming for a complex industrial process and, furthermore, the resulting recipe may be error-prone. The aim of this paper is thus to develop a systematic strategy to generate correct cleaning procedures efficiently. Specifically, the proper material-transfer routes are selected on the basis of the Petri-net representation of all possible paths in the pipeline network. The operation steps for transporting detergent along a designated route can be identified from the simulation results obtained with the Petri-net model of the entire system. By connection of this system model with a net representing the schedule manager, a recipe can be produced accordingly to achieve a multiroute sequential/concurrent schedule. A realistic example is presented at the end of this paper to demonstrate the effectiveness and correctness of the proposed strategy.

1. Introduction

Cleaning the pipelines is one of the routine operations that has to be performed in any chemical plant. In continuous processes, it is basically a periodic housekeeping practice for removing scales and/or sediments accumulated during normal operation. On the other hand, the cleaning routines can play a more critical role in keeping a batch process operable. Notice that, in a multiproduct batch plant, the existing equipments are often shared by various different processes and the raw materials, intermediates, and products of these processes are required to be moved from one unit to another through a common pipeline network. To avoid contamination of foreign substances, it is necessary to clean, disinfect, or purge the pipeline network prior to each batch cycle. The "cleaning" procedure can essentially be viewed as the operation steps to transfer a detergent, a disinfectant, or an inert material from the inlets (sources) to the outlets (sinks) of a pipeline network, and also the transfer routes should cover every part of the system. Traditionally, the tasks of finding all possible cleaning routes and then synthesizing the corresponding operating procedure are carried out manually on an ad hoc basis. For a complex chemical process, the demand of these tasks on time and effort may be overwhelming and the resulting recipe is often error-prone. Thus, to relieve the work load and also to improve the cleaning quality, it is highly desirable to develop a systematic strategy to synthesize the needed operating procedures correctly and efficiently.

As mentioned previously, cleaning a pipeline network can be considered as a special material-transfer operation. Several related studies can be found in the literature. In a pioneering work, Rivas and Rudd[1] proposed a method for the synthesis of failure-safe procedures to help the operators make proper decisions during emergency situations. A valve operation sequence can be quickly determined to reach the given operation objective. O'Shima[2] handled this problem with a more efficient solution technique. The author developed the algorithms for finding the routes between the given starting and terminating points of a material stream and also for evaluating the flow state in each unit along the stream. The operating procedures were then synthesized on the basis of these algorithms. Foulkes et al.[3] represented the states of fragments in a plant structure with a series of condition lists. They utilized a combination of artificial intelligence techniques, pattern matching, and path search algorithms to identify all feasible routes for transferring a designated material from one storage tank to another in the plant. Uthgenannt[4] used digraph models to describe the network of interconnected process equipments. The material-transfer routes and the required operating procedures can be obtained using a graph search method.

It should be noted that the above-mentioned methods are not directly applicable in the present application. First of all, the objective of a cleaning operation is, in general, not the same as that of simple material transfer. It is important in the former case to ensure that all parts in the pipeline network are included in the material-transfer routes, while this constraint is not imposed in the latter. Second, it is difficult to generate valve-sequencing steps with the available methods to achieve a multiroute cleaning schedule. Thus, the focus of the present study is concerned with the generation of operating procedures to perform multiple material-transfer tasks for cleaning the entire network.

A formal definition of the terminology, models, and functionality of batch control systems is available from the ISA standard ISA-S88.01.[5] It was shown that the sequential function chart (SFC) is suitable for representing the hierarchical procedural model specified in this standard.[6] Because SFC is essentially derived from the basic concepts of the Petri net, the latter is used in

* To whom correspondence should be addressed. Tel.: 886-6-275-7575 ext. 62663. Fax: 886-6-234-4496. E-mail: ctchang@mail.ncku.edu.tw.

10.1021/ie049253h CCC: $30.25
Published on Web 12/04/2004

this work as the modeling tool to accurately describe the material-transfer operations in pipeline networks. The mathematical representation of the *ordinary* Petri net is provided by Peterson.[7] As originally designed, it is only composed of three types of elements, i.e., a set of discrete places **P**, a set of discrete transitions **T**, and a set of interconnected normal arcs **A**. A discrete place is graphically expressed in the Petri net with a circle, and a discrete transition is expressed with a bar. A normal arc is represented with a directed solid line. It connects either a place to a transition or vice versa. To facilitate proper characterization of the material-transfer patterns in pipeline networks, additional extensions are also used in this work, i.e., the weighted arcs, the inhibitor arcs, and the static test arcs. The execution of a Petri net is controlled according to the numbers and distribution of *tokens* in the places of the Petri net. The vector of all token numbers at a particular instance is referred to as a *marking*. A more detailed review of these and other Petri-net elements and also the transition enabling and firing rules can be found elsewhere.[8,9]

In this study, an accurate component model is first constructed for each valve, pump, and pipeline fragment in the system. The fragment models are then connected according to the network configuration to build a Petri net in which all material-transfer paths can be embedded. This path model is described in section 2. In principle, all alternatives for cleaning the pipeline network can be enumerated from the reachability trees of this Petri net with a standard algorithm.[10] This practice is illustrated with a simple example in section 3. In the next section, two matrix-based procedures are presented to identify the appropriate material-transfer strategies in sequential and concurrent cleaning operations. The Petri-net representations of the valves, pumps, and compressors are given in section 5. By attachment of these equipment models to a modified version of the path model, the Petri-net model of the entire system can be constructed. This model is then used to generate the operation steps needed to clean any given route. Implementation details of this approach are given in section 6. Also, to be able to achieve a multitask schedule, additional control mechanisms must be incorporated in the Petri-net model. These extra features are explained in section 7. Finally, the results of applying the proposed synthesis procedure to an industrial-size system are presented in the last section.

2. Representation of Material-Transfer Paths

The first critical issue in modeling any network should be concerned with the division of the system into distinct components. The concept of the piping *fragments*[3] is adopted in this work for this purpose. In particular, a fragment is defined as a collection of pipeline branches and/or processing units separated from other fragments (or the environment) by valves, pumps, and other means of flow blockages in the pipeline network. Let us consider Figure 1 as an example. Eight fragments can be identified according to this definition, i.e., FR1−FR8. In this case, every pump and its isolation valves are viewed as one *lumped* power-generating system, and this system is treated as a flow blockage if it is turned off. Notice also that, in many industrial plants, the pipeline networks contain dead branches. These branches are usually separated from the external atmosphere by blanks, slip plates,

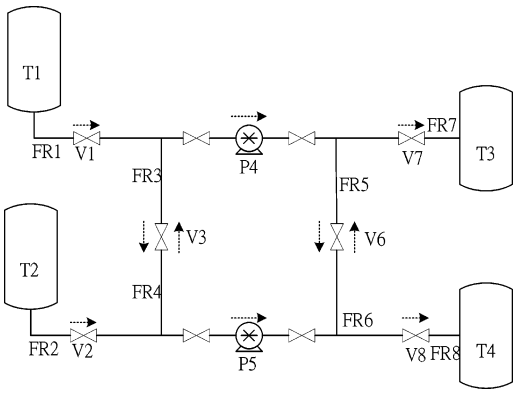

Figure 1. Typical example of a pipeline network.

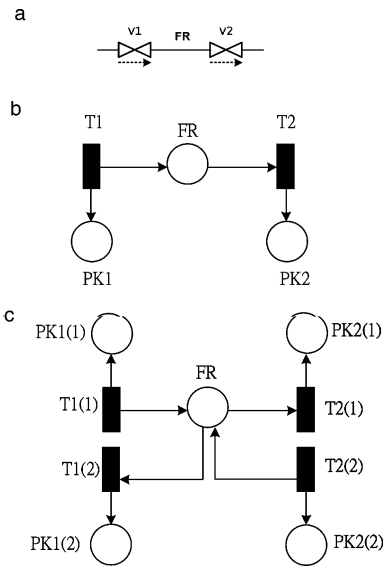

Figure 2. (a) Basic structure of a piping fragment with two single-direction valves. (b) Petri-net model of a basic fragment with two single-direction valves. (c) Basic structure of a basic fragment with two bidirectional valves.

and/or closed and locked valves. According to the definition given above, every dead branch and its connecting branches can still be viewed as a single fragment as long as no flow blockages can be found inside this fragment.

For illustration convenience, let us first examine the most basic structure of the fragment, i.e., a pipe branch isolated by an inlet valve and an outlet valve (see Figure 2a). Notice that, in this case, the flow in either valve is allowed only in one direction. The corresponding Petri-net model is presented in Figure 2b. The place FR in this model is used to reflect the fragment state. More specifically, a token entering such a place denotes the condition that the detergent is delivered to the corresponding fragment from an upstream source fragment. The place PK1 is used to keep a record of the connection status of FR with its upstream fragment, and PK2 is used for the same purpose concerning the downstream fragment of FR. The transitions T1 and T2 can be considered as the operator/controller actions to open valves V1 and V2, respectively. On the other hand, if both valves permit bidirectional material transfer, the fragment model depicted in Figure 2b should be changed to the one shown in Figure 2c. Notice that each transition in the former Petri net is now replaced with two transitions to denote the material-transfer actions to and from the fragments FR via the corresponding valve.

In principle, all mass-transfer paths can be found in

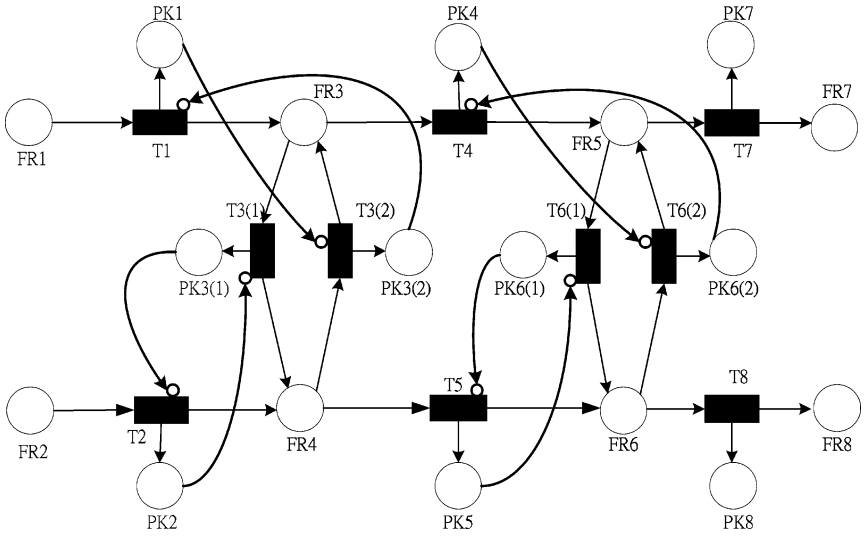

Figure 3. Path model of the example network in Figure 1.

a Petri net assembled by connecting fragment models according to the network configuration. However, if the given system contains bidirectional valves, one or more infinite loops may be identified. These looping paths obviously cannot be adopted as the candidate routes for cleaning operations. Additional constraints are thus imposed in the Petri-net model to eliminate such possibilities. In particular, every record-keeping place, i.e., a place labeled with "PK", is connected with inhibitor arcs to the input transitions of the place representing its downstream fragment. Let us use the pipeline network in Figure 1 to illustrate this model-building practice. The corresponding path model can be found in Figure 3. Notice that the inhibitor arcs in this net are introduced solely for the purpose of imposing the proposed constraints. Specifically, let us consider transition T1 as an example. If it is fired, a token will then be introduced in place FR3 and also another token will be deposited in PK1. As a result, transition T3(2) will be inhibited and remained so even after a token entered place FR4. Notice that the firing of every other transition in this model causes the same effects on its downstream places. Thus, the token movements generated in any simulation run are bound to follow a loop-free route in this Petri net. If the inhibitor arcs in Figure 3 are all removed, a token may travel endlessly in one of the following two loops: (1) FR3 → FR4 → FR3 → FR4 ... and (2) FR5 → FR6 → FR5 → FR6 ....

3. Enumeration of Possible Routes

Because there may be more than one route emanating from a particular source fragment to the sink fragments of a pipeline network, it is desirable to first identify all of them to ensure thoroughness of the cleaning operation. This task can be achieved by constructing a *reachability tree* from a given initial condition on the basis of the Petri-net model. Specifically, by firing the transitions enabled initially, one can first obtain as many "new" markings as the number of the fired transitions. From each new marking, one can then again generate more markings with the same approach. Repeating this procedure over and over results in a reachability tree. This tree consists of nodes and arcs. Other than the node representing the initial state, each node is associated with a generated marking and its input arc denotes the corresponding fired transition. All nodes in the tree can be classified into four different types: (1) frontier nodes, (2) interior nodes, (3) duplicate nodes, and (4) terminal nodes. The frontier nodes are nodes that have not yet been developed by the tree-building algorithm, whereas the interior nodes are processed nodes. The duplicate nodes are the ones that have appeared more than once in the tree. The terminal nodes are nodes that cannot lead to any enabled transition. It should be noted that the construction of a reachability tree should continue as long as the frontier nodes still exist. In other words, every frontier node must eventually be converted to one of the other nodes.

The tree construction process begins by defining the initial marking to be the root node of the tree and also a frontier node. On the basis of the breadth-first strategy, the reachability tree of a given Petri net can be constructed according to the following algorithm:[10,11]

1. Label the initial marking $\mathbf{M}_0$ as the root node of the tree and, initially, tag it as a frontier node.

2. If the frontier nodes exist, do the following:

(a) Select a frontier node. Let the marking of this node be $\mathbf{M}$.

(b) If the marking $\mathbf{M}$ is identical with that of an existing node in the constructed tree, then convert the frontier node to a duplicate node and go to step 2a.

(c) Use the revised Murata state equation to obtain all possible enabled transitions. If no transitions are enabled for the marking $\mathbf{M}$, then convert the selected frontier node to a terminal node and go to step 2a. If the enabled transitions can be identified, then select each enabled transition t_j as a firing transition t_f and carry out the following tasks repeatedly:

(i) Obtain the marking $\mathbf{M}_k$ by firing t_f.

(ii) Include $\mathbf{M}_k$ as a node, draw a directed arc with label t_f from $\mathbf{M}$ to $\mathbf{M}_k$, and tag $\mathbf{M}_k$ as a frontier node.

(iii) Remove the original tag from $\mathbf{M}$, and tag it as an interior node.

Let us now turn to the Petri-net model presented in Figure 3. By assuming that the detergent is stored in tank T1 initially, the corresponding reachability tree (shown in Figure 4) can be generated according to the above algorithm. The markings associated with the nodes in this tree can be found in Table 1. To conveniently identify the elements in a marking, the token numbers are classified into two subsets and arranged sequentially in a vector, i.e.

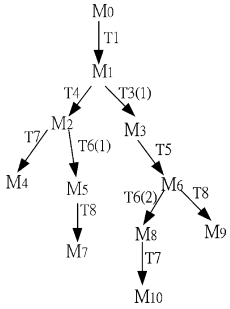

Figure 4. Reachability tree of the path model in Figure 3: starting nonempty FR1.

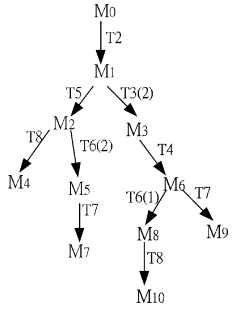

Figure 5. Another reachability tree of the path model in Figure 3: starting nonempty FR2.

Table 1. Markings M_k's (=[FR_k|PK_k]) of the Reachability Tree in Figure 4

k	$\mathbf{FR}_k$	$\mathbf{PK}_k$
0	{1 0 0 0 0 0 0 0}	{0 0 0 0 0 0 0 0 0 0}
1	{0 0 1 0 0 0 0 0}	{1 0 0 0 0 0 0 0 0 0}
2	{0 0 0 0 1 0 0 0}	{1 0 0 0 1 0 0 0 0 0}
3	{0 0 0 1 0 0 0 0}	{1 0 1 0 0 0 0 0 0 0}
4	{0 0 0 0 0 0 1 0}	{1 0 0 0 1 0 0 0 1 0}
5	{0 0 0 0 0 1 0 0}	{1 0 0 0 1 0 1 0 0 0}
6	{0 0 0 0 0 1 0 0}	{1 0 1 0 0 1 0 0 0 0}
7	{0 0 0 0 0 0 0 1}	{1 0 0 0 1 0 1 0 0 1}
8	{0 0 0 0 1 0 0 0}	{1 0 1 0 0 1 0 1 0 0}
9	{0 0 0 0 0 0 0 1}	{1 0 1 0 0 1 0 0 0 1}
10	{0 0 0 0 0 0 1 0}	{1 0 1 0 0 1 0 1 1 0}

$$\mathbf{M}_k = [\mathbf{FR}_k|\mathbf{PK}_k]$$

and $k = 0, 1, 2, ..., 10$. Here, each subset label is identical with the place labels of its elements. For example, the elements stored in $\mathbf{FR}_k$ are the token number of places representing the fragment states. On the basis of this convention, useful information can be directly acquired from the marking of each terminal node in the reachability tree. Specifically, the sink fragment of a material-transfer route should be associated with 1 in subset $\mathbf{FR}_k$ of the terminal node. The corresponding connection status among various fragments in the network can be identified from $\mathbf{PK}_k$. From the reachability tree given in Figure 4 and its markings in Table 1, it can be seen that there are four terminal nodes, i.e., $\mathbf{M}_4$, $\mathbf{M}_7$, $\mathbf{M}_9$, and $\mathbf{M}_{10}$. Four material-transfer routes can be identified accordingly, i.e.

$$\text{FR1} \rightarrow \text{FR3} \rightarrow \text{FR5} \rightarrow \text{FR7} \quad (1)$$

$$\text{FR1} \rightarrow \text{FR3} \rightarrow \text{FR5} \rightarrow \text{FR6} \rightarrow \text{FR8} \quad (2)$$

$$\text{FR1} \rightarrow \text{FR3} \rightarrow \text{FR4} \rightarrow \text{FR6} \rightarrow \text{FR5} \rightarrow \text{FR7} \quad (3)$$

$$\text{FR1} \rightarrow \text{FR3} \rightarrow \text{FR4} \rightarrow \text{FR6} \rightarrow \text{FR8} \quad (4)$$

Similarly, another reachability tree can be built from the second source fragment FR2 of the Petri net given in Figure 3. This tree is presented in Figure 5, and the markings of its nodes can be found in Table 2. Four more material-transfer routes can be found in this tree, i.e.

$$\text{FR2} \rightarrow \text{FR4} \rightarrow \text{FR6} \rightarrow \text{FR8} \quad (5)$$

$$\text{FR2} \rightarrow \text{FR4} \rightarrow \text{FR6} \rightarrow \text{FR5} \rightarrow \text{FR7} \quad (6)$$

$$\text{FR2} \rightarrow \text{FR4} \rightarrow \text{FR3} \rightarrow \text{FR5} \rightarrow \text{FR6} \rightarrow \text{FR8} \quad (7)$$

$$\text{FR2} \rightarrow \text{FR4} \rightarrow \text{FR3} \rightarrow \text{FR5} \rightarrow \text{FR7} \quad (8)$$

4. Route Selection Procedures

Although every material-transfer route identified from the reachability trees of a Petri-net model can be adopted to clean a portion of the given pipeline network, it may not be necessary to include all of them to achieve the operation objective. In this study, the task of cleaning a pipeline network is considered to be accomplished if the detergent is transported either (1) through every fragment at least once or (2) across every blockage at least once. Notice that the former criterion may result in a less rigorous operating procedure than the latter. This is because the requirement of moving material through a pipeline fragment *only* guarantees continuous fluid flows in its inlet and outlet branches. On the other hand, because by definition there is always a blockage on every branch of a fragment, the quality of the cleaning operation can be better ensured with the second criterion. However, it should be noted that the dead branches in the pipeline network are always not cleanable according to this blockage-based criterion. In this case, the blanks, slip plates, or valves on all dead branches may have to be removed/opened to allow for the needed detergent flows.

Table 2. Markings M_k's (=[FR_k|PK_k]) of the Reachability Tree in Figure 5

k	$\mathbf{FR}_k$	$\mathbf{PK}_k$
0	{0 1 0 0 0 0 0 0}	{0 0 0 0 0 0 0 0 0 0}
1	{0 0 0 1 0 0 0 0}	{0 1 0 0 0 0 0 0 0 0}
2	{0 0 0 0 0 1 0 0}	{0 1 0 0 0 1 0 0 0 0}
3	{0 0 1 0 0 0 0 0}	{0 1 0 1 0 0 0 0 0 0}
4	{0 0 0 0 0 0 0 1}	{0 1 0 0 0 1 0 0 0 1}
5	{0 0 0 0 1 0 0 0}	{0 1 0 0 0 1 0 1 0 0}
6	{0 0 0 0 1 0 0 0}	{0 1 0 1 1 0 0 0 0 0}
7	{0 0 0 0 0 0 1 0}	{0 1 0 0 0 1 0 1 1 0}
8	{0 0 0 0 0 1 0 0}	{0 1 0 1 1 0 1 0 0 0}
9	{0 0 0 0 0 0 1 0}	{0 1 0 1 1 0 0 0 1 0}
10	{0 0 0 0 0 0 0 1}	{0 1 0 1 1 0 1 0 0 1}

Table 3. Fragment-Based Path Matrix

route no.	FR1	FR2	FR3	FR4	FR5	FR6	FR7	FR8
1	O		O		O		O	
2	O		O		O	O		O
3	O		O	O	O	O	O	
4	O		O	O		O		O
5		O		O		O		O
6		O		O	O	O	O	
7		O	O	O	O	O		O
8		O	O	O	O		O	

4.1. Sequential Operations. A systematic procedure has been developed in this study to select the appropriate material-transfer routes so that either one of the above two criteria can be satisfied with sequential cleaning operations. For illustration convenience, let us consider all possible cleaning routes of the pipeline network given in Figure 1, i.e., routes 1−8. These routes can be arranged in a matrix form as shown in Table 3. Notice that its columns are associated with the fragments in a pipeline network and each row represents a

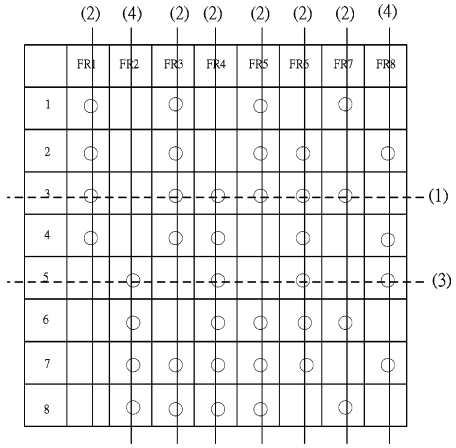

Figure 6. Fragment-based implementation steps for selecting sequential cleaning routes.

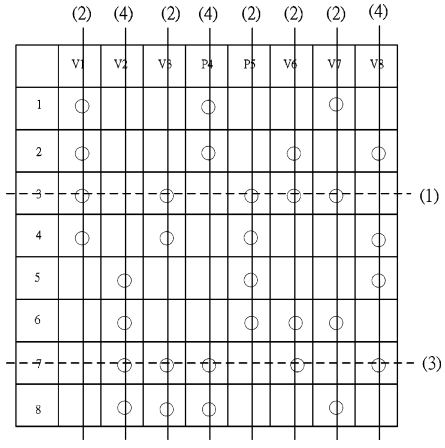

Figure 7. Blockage-based implementation steps for selecting sequential cleaning routes.

route. This matrix is referred to as a *path matrix* in this paper. The route selection procedure can be viewed conceptually as the symbolic manipulation steps to identify the independent vectors that can "span" the row space of the path matrix. To reduce the number of cleaning routes as much as possible, a simple heuristical rule is used in the proposed selection procedure; i.e., the route having the largest number of fragments should be chosen first. Although the resulting routes may not be optimal, this approach is taken because of the relative easiness in implementation. The specific selection steps are presented in the sequel:

1. Select a row in the path matrix with the largest number of elements.
2. Delete the row identified in step 1.
3. Delete all columns in which the elements of the row identified in step 1 are located.
4. Repeat steps 1−3 until all columns are deleted.

Let us now apply this procedure to Table 3. The detailed implementation steps are marked in Figure 6. A brief description of these steps is provided below: (1) Select and then delete the third row because it contains the maximum number of elements. (2) Delete columns 1 and 3−7 because row 3 contains elements in these columns. (3) Select and then delete row 5 because it contains the most elements in the remaining matrix obtained after carrying out the previous two steps. (4) Delete columns 2 and 8 because row 5 contains elements in these two columns and all other columns have already been deleted before. Notice that the selection process is terminated after all columns have been eliminated in step 4. From Figure 6, it is clear that the first cleaning criterion can be satisfied by selecting routes 3 and 5.

It should be noted that a similar path matrix can be constructed if routes 1−8 are expressed in terms of the removed blockages. As mentioned before, a feasible route can be identified from the marking of a terminal node in the reachability tree. Specifically, the elements of subset $\mathbf{PK}_k$ in this marking reflect the states of valves, pumps, and compressors in the pipeline network to facilitate material transfer via the corresponding route. For example, route 1 can also be written as

$$\mathrm{V1} \rightarrow \mathrm{P4} \rightarrow \mathrm{V7} \tag{9}$$

Consequently, the same procedure can be followed to select the routes that satisfy the second criterion. From the implementation steps presented in Figure 7, one can see that routes 3 and 7 should be chosen in this case.

4.2. Concurrent Operations. Notice that the material-transfer routes selected with the above procedure may be partially overlapped. For example, if the fragment-based criterion is used, the chosen routes 3 and 5 in the pipeline network in Figure 1 share two common elements, i.e., FR4 and FR6. As another example, notice that valves V3 and V5 are both required to be open to facilitate material transfer in the same pipeline network via routes 3 and 7, respectively. Because of the possibility of overlapping routes, the cleaning operations of different routes have to be carried out sequentially. Therefore, to save operation time, there are incentives to identify nonoverlapping routes so that the concurrent cleaning strategies can be devised accordingly. A modified version of the above route selection procedure has thus been developed for this purpose. Its implementation steps are presented as follows:

1. Select a row in the path matrix with the largest number of elements.
2. Identify and temporarily remove another row with at least one element in the column where the row selected previously in step 1 also has an element. Repeat this step until no more rows can be identified.
3. Repeat steps 1 and 2 until all rows are exhausted.
4. Recover the temporarily removed rows.
5. Delete all of the rows selected in steps 1−3.
6. Delete all columns in which the elements of the rows selected in steps 1−3 are located.
7. Repeat the above steps until all columns are deleted.

Because the routes selected in steps 1−3 cannot be overlapping, it is possible to execute the material-transfer operations along these routes simultaneously. For illustration purposes, let us consider the fictitious path matrix presented in Figure 8. On the basis of the selection steps marked in this figure, one can easily see that cleaning of routes 1 and 2 should first be carried out concurrently and, after completion of these two tasks, the material-transfer operation along route 4 can then take place. A more realistic example of the concurrent cleaning operations will be presented later in the case studies.

5. Equipment Models

To generate the specific operation steps to realize the cleaning tasks of the selected material-transfer routes, the Petri-net model of a pipeline network must contain not only the component models of the fragments but also

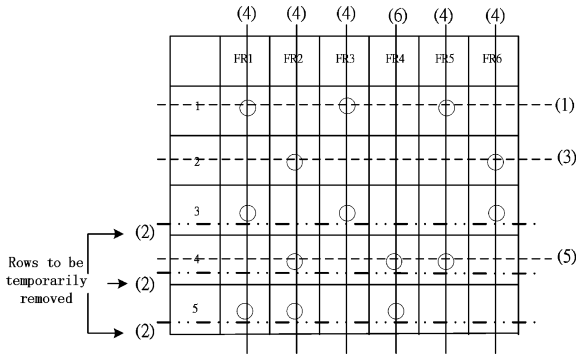

Figure 8. Implementation steps for selecting concurrent cleaning routes.

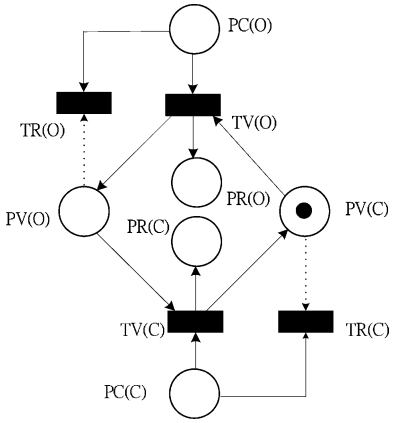

Figure 9. Standard valve model.

those of the installed equipments, such as the valves, pumps, compressors, etc. The valve model is presented in Figure 9. Here, the places PV(O) and PV(C) denote two opposite valve positions respectively, i.e., open and close. The transitions TV(O) and TV(C) represent the valve-switching actions from PV(C) to PV(O) and vice versa. Notice that the input place PC(O) of the transition TV(O) can be interpreted as the valve-opening requirement once a particular material-transfer action is selected in a route. Similarly, the place PC(C) can be considered as the demand for the valve-closing operation. The output places PR(O) and PR(C) of the two transitions TV(O) and TV(C) can be used to record the actual number of times that the corresponding valve-switching actions have been carried out. Because it is possible to call for a material-transfer action when the corresponding valve is already open, transition TR(O) is introduced as the output of both PC(O) and PV(O) in this model. A normal arc is adopted in the former case to avoid a token being permanently kept in place PC(O), while a test arc is used in the latter to prevent loss of the tokens in PV(O). Finally, note that the transition TR(C) is adopted for the same reason.

Because the operating procedures of pumps, compressors, and their isolation valves can be considered as well-established industrial practices, e.g., see Karassik and McGuire,[12] their detailed steps are not described in the equipment models for the sake of simplicity. Specifically, the Petri net presented in Figure 9 is also used to model a power-generating system in this study. In this case, the places PV(O) and PV(C) represent two opposite states, i.e., on and off, of the system, respectively. The transitions TV(O) and TV(C) can be regarded as a series of standard operation actions to turn on and off the pump/compressor system.

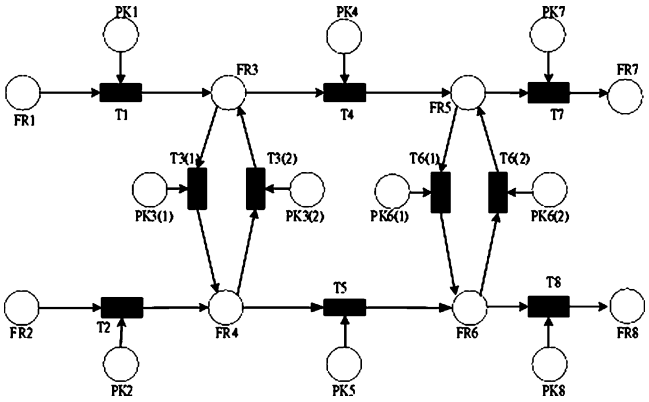

Figure 10. Modified path model of the example network.

Table 4. Normal Arc Connections between Transitions in a Modified Path Model (Figure 10) and Places PC(O)s in Different Equipment Models (Figure 9)

transition	equipment	transition	equipment
T1	V1	T5	P5
T2	V2	T6(1)	V6
T3(1)	V3	T6(2)	V6
T3(2)	V3	T7	V7
T4	P4	T8	V8

6. Generation of Operation Steps

The operating procedure for cleaning a selected route in the pipeline network can be obtained based on a system model. To create such a model, the equipment models should be attached one by one to a modified Petri-net representation of the material-transfer paths. This modified path model can be transformed from its original version by removing all inhibitor arcs and then reversing the directions of all connecting arcs between the record-keeping places (i.e., the places labeled with PK) and their input transitions. The above-mentioned places in the modified net can now be interpreted as the demands to connect the corresponding upstream and downstream fragments. Let us again consider the system in Figure 1 as an example. The path model in Figure 3 can be converted to the Petri net presented in Figure 10 by introducing the proposed modifications. This net is then expanded by connecting its transitions respectively to the places PC(O)'s in the corresponding equipment models with normal arcs. A detailed listing of these connections can be found in Table 4. This practice is meant to reflect the relationship between each material-transfer action and the need to open the corresponding valve or to turn on the corresponding pump/compressor.

Given a route and a set of initial valve states, the required operation steps can be synthesized by performing a simulation with the system model. Under the condition that all valves are closed and all pumps are switched off initially, it can be determined easily by inspecting the pipeline network in Figure 1 that valves V2 and V8 should be open and also pump P5 must be turned on for transporting material through route 5. On the other hand, notice that the initial system condition of the Petri-net model can be set by introducing a token in FR2 and also in the place representing the close position of every valve in the system. Notice also that the selected route 5 can be stipulated by providing a token in every place representing the demand to connect a pair of fragments in route 5, i.e., PK2, PK5, and PK8. A collection of operation steps can then be identified by

Table 5. Normal Arc Connections between Transitions in a Modified Path Model (Figure 10) and Places PC(C)s in Different Equipment Models (Figure 9)

transition	equipment	transition	equipment
T1	V3, P4	T5	V6, V8
T2	V3, P5	T6(1)	P5, V8
T3(1)	V2, P5	T6(2)	P4, V7
T3(2)	V1, P4	T7	
T4	V6, V7	T8	

executing the Petri-net-based simulation accordingly. It has been verified that the simulation results are the same as those obtained by inspection.

However, if a different set of initial equipment states is adopted in the simulation run, it can be shown that the resulting operating procedure still remains unchanged for route 5. In the network presented in Figure 1, if the cleaning of route 3 is carried out before that of route 5 and only pump P5 and valve V1 are switched off to terminate the former task, the operating procedure identified above is clearly insufficient for accomplishing the latter. Thus, to guarantee the feasibility and safety of the material-transfer steps through a selected route, it is necessary to impose additional auxiliary control rules in operating the related valves, pumps, and/or compressors. These *equipment operation rules* are summarized below:

Equipment Operation Rules. *Given a specific material-transfer action, all valves and/or pumps surrounding its downstream fragment (except the one used for facilitating the present material transfer) should be closed/switched off.*

To realize this requirement of blocking all of the entrances and exits not located on the selected route, additional normal arcs should be introduced to connect the transition representing the given action in the modified path model to the places PC(C)'s in Petri-net models of the equipments surrounding its downstream fragment. In the case of our example system in Figure 1, these additional connections are shown in Table 5. As a result, the cleaning procedure of route 5 can be correctly generated from any given initial condition with the proposed simulation approach. To be specific, let us assume that the valves V3, V6, and V7 are left open after cleaning route 3 in our previous example, while in the meantime, the other valves are closed and both pumps are off. The operating procedure to clean route 5 in this situation can be found to be as follows: close V3 and V6, open V2 and V8, and then turn on P5.

7. Execution of Multiple Tasks

In this study, it is assumed that two separate cleaning tasks can be scheduled sequentially according to the Gantt chart shown in Figure 11a and concurrently according to Figure 11b. It is required in the former case that

$$t_0 < t_1 \leq t_2 < t_3 \tag{10}$$

while in the latter case, the constraint is either

$$t_0 \leq t_1 < t_2 \leq t_3 \tag{11}$$

or

$$t_0 \leq t_1 < t_3 \leq t_2 \tag{12}$$

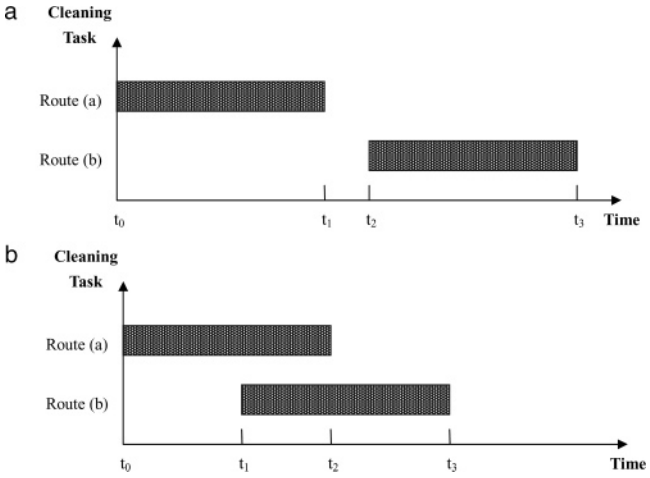

Figure 11. (a) Typical sequential schedule. (b) Typical concurrent schedule.

The time needed to accomplish a particular task should be determined on a case-by-case basis.

The systematic approach for generating the operation steps of a single-route cleaning task has already been presented in the previous sections. Here, the proposed Petri-net-based techniques are extended to synthesize the operating procedure for executing a multitask schedule. To coordinate the implementation times of various different tasks according to the given schedule, the Petri-net model of a so-called *schedule manager* is attached to the system model. If the tasks of cleaning routes 3 and 5 in the example system are to be carried out according to the schedule given in Figure 11a, this schedule manager can be modeled on the basis of the Petri net given in Figure 12. Notice first that, instead of a single event, transition TX1 should be interpreted as a collection of operation steps (events) to establish route 3 for cleaning purposes and, similarly, TX6 represents another set of operation steps to establish route 5. To activate the operation steps associated with route 3, transition TX1 is connected to the place representing the source fragment, i.e., FR1, and also to those representing the operation demands for establishing fragment connections, i.e., PK1, PK3(1), PK5, PK6-(2), and PK7. Similarly, TX6 is connected to FR2, PK2, PK5, and PK8 to trigger the operation actions for cleaning route 5. On the other hand, transitions TX4 and TX9 represent the actions to terminate the cleaning operations of routes 3 and 5, respectively. It is assumed in the present study that each termination procedure consists of only two steps, i.e., switching off the pump on the cleaning route and then closing the exit valve of the source fragment. The connections from TX4 and TX9 to the places in the system model reflect the demands for these actions. Notice also that the delay times of TX1, TX4, TX6, and TX9 are assigned to meet the given schedule exactly. Other types of connections between the Petri-net model of the schedule manager and the system model are concerned with places P1, P4, P6, and P9. They are used simply for maintenance purposes. Places P1 and P6 mark the initialization phases of the operations to establish routes 3 and 5, respectively, while P4 and P9 represent the preparation stages prior to the termination steps for these two routes, respectively. It should be noted that every such place and all of the places labeled with PR(O) and PR(C) in the system model are connected to a common output transition. Because the operation records of pumps and valves

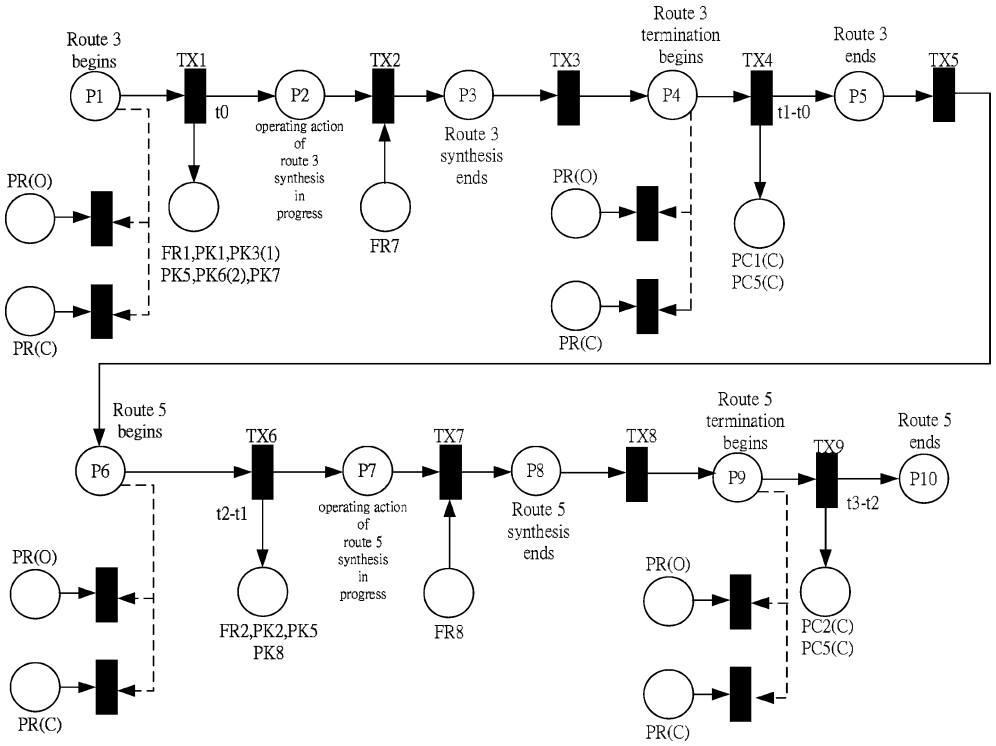

Figure 12. Petri-net model of a sequential schedule manager.

Table 6. Fragment-Based Operation Steps for Cleaning the Pipeline Network in Figure 1

time	operation step
t_0	open valves V1, V3, V6, and V7 switch on pump P5
t_1	switch off pump P5 close valve V1
t_2	close valves V3 and V6 open valves V2 and V8 switch on pump P5
t_3	switch off pump P5 close valve V2

Table 7. Blockage-Based Operation Steps for Cleaning the Pipeline Network in Figure 1

time	operation step
t_0	open valves V1, V3, V6, and V7 switch on pump P5
t_1	switch off pump P5 close valve V1
t_2	close valve V7 open valves V2 and V8 switch on pump P4
t_3	switch off pump P4 close valve V2

are stored in the latter places, this practice is in essence to reset these records before carrying out each of the above four distinct sets of operation steps.

For illustration convenience, let us assume that all valves are closed and all pumps are off initially in the example system. To realize the schedule in Figure 11a, an operating procedure can be generated by placing a token in the place P1 and then executing the simulation run. The resulting operation steps are presented in Table 6. Notice that the operating procedure obtained on the basis of the blockage-based criterion can also be produced with the same approach (see Table 7). These operation steps can be followed to clean routes 3 and 7 in sequence. Finally, although a specific example of the concurrent operations cannot be given at this point, it should be emphasized that the Petri-net model of the corresponding schedule manager can be built easily in a similar fashion. The simulation results of a sequential/concurrent hybrid operation are presented later in the next section.

8. Case Study

The pipeline network described in Foulkes et al.[3] is adopted in the present work as a realistic example to demonstrate the capability of the proposed method. The network contains 8 storage tanks, 36 valves, and 4 pumps (see Figure 13). A total of 5 source fragments, i.e., FR1−FR5, 20 internal fragments, i.e., FR6−FR25, and 6 sink fragments, i.e., FR26(1)−FR28(1) and FR26-(2)−FR28(2), can be defined in this system. It is assumed that there are no upper limits imposed upon the amounts of detergent stored in the source tanks, i.e., T1−T5, and also the capacities of the sink tanks, i.e., M1−M3. In addition, the spent materials gathered from separate cleaning routes are allowed to be stored in the same sink tank. Because each sink tank in the present system has two inlet pipelines, these two inlets are thus treated as two distinct fragments in this example. Finally, it is also assumed that all valves are closed and all pumps are turned off initially.

The path model of the given system can be found in Figure 14. To enhance model readability, the inhibitor arcs are not shown in this figure. As mentioned previously, the cleaning routes can be selected on the basis of this model using either the fragment-based criterion (see Table 8) or the blockage-based criterion (see Table 9). Notice that, in both cases, the required cleaning tasks must be implemented sequentially in three stages. During each stage, multiple material-transfer operations can be executed concurrently via the selected routes. For illustration convenience, let us further assume that the operation periods needed to carry out

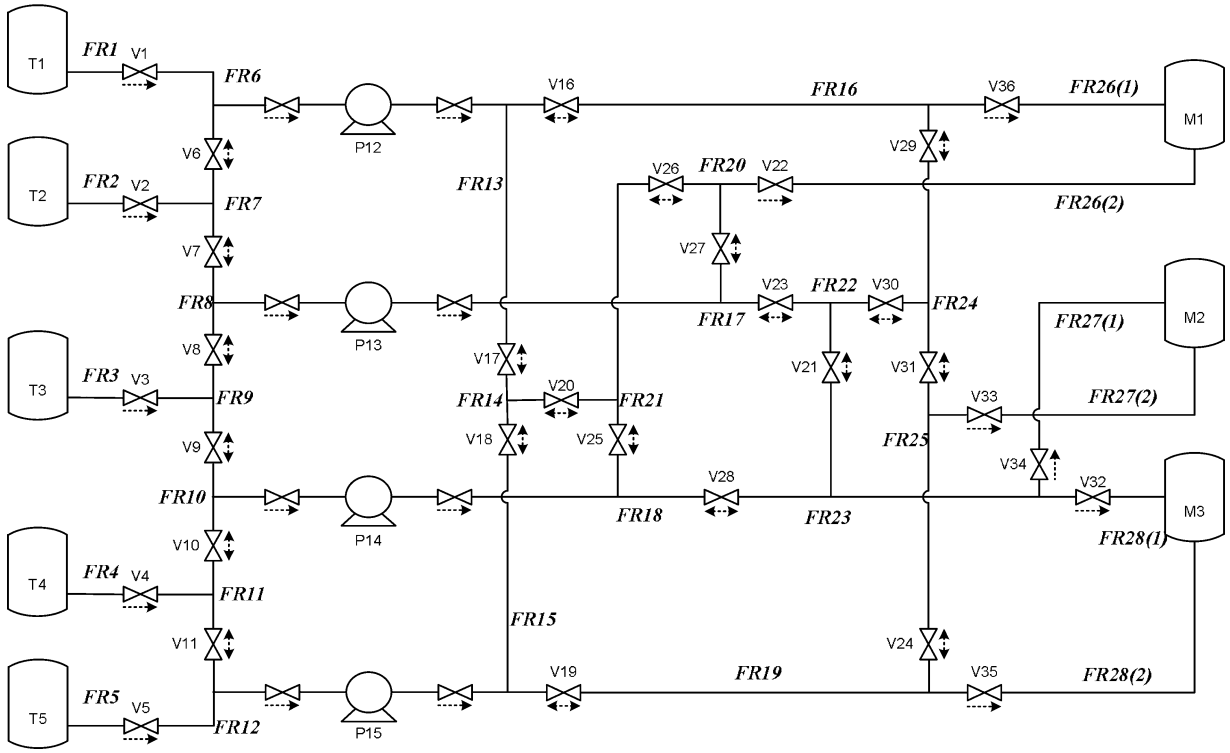

Figure 13. Complex pipeline network.

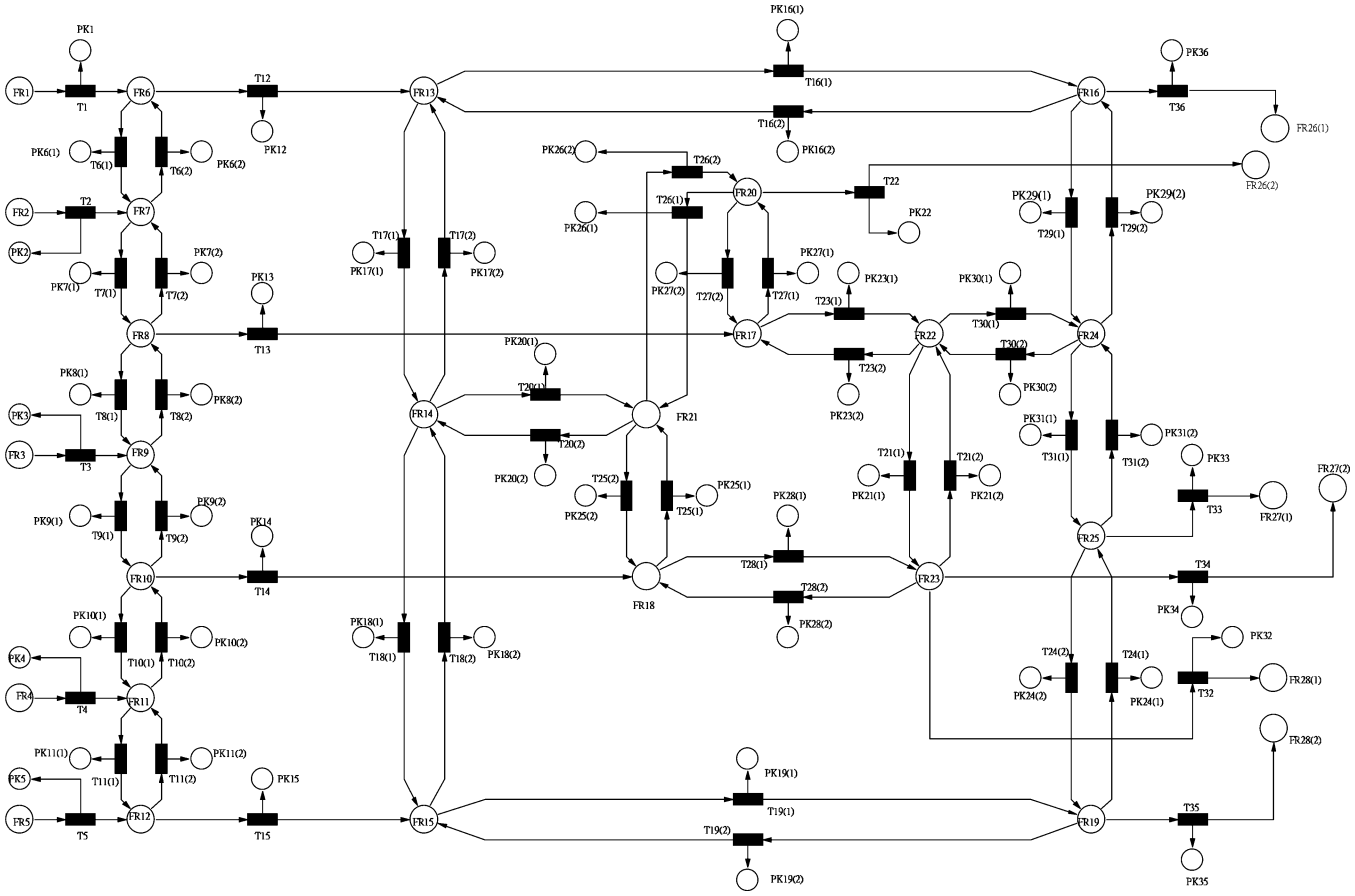

Figure 14. Path model of the complex network in Figure 13.

different operations in the same stage are identical. More specifically, the three distinct operation periods are assumed to be $[t_0, t_1]$, $[t_2, t_3]$, and $[t_4, t_5]$. The complete operating procedures to achieve the corresponding schedules can be generated from the proposed Petri-net-based simulation runs. The results of the two cases considered in this example can be found in Tables 10 and 11.

Table 8. Material-Transfer Routes Selected with Fragment-Based Criterion for Cleaning the Pipeline Network in Figure 13

implementation stage	route(s)
1	FR1 → FR6 → FR7 → FR8 → FR9 → FR10 → FR11 → FR12 → FR15 → FR19 → FR25 → FR24 → FR16 → FR13 → FR14 → FR21 → FR18 → FR23 → FR22 → FR17 → FR20 → FR26(2)
2	FR2 → FR7 → FR6 → FR13 → FR16 → FR26(1)
	FR3 → FR9 → FR10 → FR18 → FR23 → FR28(1)
	FR4 → FR11 → FR12 → FR15 → FR19 → FR28(2)
3	FR5 → FR12 → FR15 → FR19 → FR25 → FR27(2)
	FR1 → FR6 → FR7 → FR8 → FR17 → FR22 → FR23 → FR27(1)

Table 9. Material-Transfer Routes Selected with Blockage-Based Criterion for Cleaning the Pipeline Network in Figure 13

implementation stage	route(s)
1	FR1 → FR6 → FR7 → FR8 → FR9 → FR10 → FR11 → FR12 → FR15 → FR19 → FR25 → FR24 → FR16 → FR13 → FR14 → FR21 → FR18 → FR23 → FR22 → FR17 → FR20 → FR26(2)
2	FR2 → FR7 → FR6 → FR13 → FR16 → FR24 → FR22 → FR17 → FR20 → FR21 → FR14 → FR15 → FR19 → FR28(2)
	FR3 → FR9 → FR10 → FR18 → FR23 → FR28(1)
3	FR4 → FR11 → FR10 → FR9 → FR8 → FR17 → FR22 → FR23 → FR27(1)
	FR5 → FR12 → FR15 → FR19 → FR25 → FR27(2)
	FR1 → FR6 → FR13 → FR16 → FR26(1)

Table 10. Operation Steps To Implement Table 8

time	operation step
t_0	open valves V1, V6−V11, V16, V17, V19−V25, V27−V29, and V31
	switch on pump P15
t_1	switch off pump P15
	close valve V1
t_2	close valves V7, V8, V10, V17, V21, V24, V25, and V29
	open valves V2−V4, V32, V35, and V36
	switch on pumps P12, P14, and P15
t_3	switch off pumps P12, P14, and P15
	close valves V2−V4
t_4	close valves V11, V27, V28, V31, and V35
	open valves V1, V5, V7, V21, V24, V33, and V34
	switch on pumps P13 and P15
t_5	switch off pumps P13 and P15
	close valves V1 and V5

Table 11. Operation Steps To Implement Table 9

time	operation step
	switch on pump P15
t_1	switch off pump P15
	close valve V1
t_2	close valves V7, V8, V10, V17, V21, V22, V24, V25, and V31
	open valves V2, V3, V26, V30, V32, and V35
	switch on pumps P12 and P14
t_3	switch off pumps P12 and P14
	close valves V2 and V3
t_4	close valves V6, V11, V27−V30, V32, and V35
	open valves V1, V4, V5, V8, V10, V21, V24, V33, V34, and V36
	switch on pumps P12, P13, and P15
t_5	switch off pumps P12, P13, and P15
	close valves V1, V4, and V5

9. Conclusion

A systematic strategy is presented in this paper for generating detailed operating procedures to clean any given pipeline network. The cleaning routes are selected on the basis of the Petri-net representation of all material-transfer paths. The operation steps for transporting detergent through a designated route can be identified from the simulation results obtained with the Petri-net model of the entire system. By connection of this net with another one representing the schedule manager, the multiroute cleaning recipes can also be produced with the proposed simulation techniques. The effectiveness of this approach is clearly demonstrated with the realistic example given at the end of this paper.

Literature Cited

(1) Rivas, J. R.; Rudd, D. F. Synthesis of failure-safe operations. *AIChE J.* **1974**, *20*, 320.

(2) O'Shima, E. Safety supervision of valve operation. *J. Chem. Eng. Jpn.* **1978**, *11*, 390.

(3) Foulkes, N. R.; Walton, M. J.; Andow, P. K.; Galluzzo, M. Computer-aided synthesis of complex pump and valve operations. *Comput. Chem. Eng.* **1988**, *12* (9/10), 1035.

(4) Uthgenannt, J. A. Path and equipment allocation for multiple, concurrent processes on networked process plant units. *Comput. Chem. Eng.* **1996**, *20* (9), 1081.

(5) ISA. *S88.01 Batch Control, Part 1: Models and Terminology*; ISA: Research Triangle Park, NC, 1995.

(6) Årzén, K.-E.; Johnsson, C. Object-oriented SFC and ISA-S88.01 recipes. *ISA Trans.* **1996**, *35*, 237.

(7) Peterson, J. L. *Petri Net Theory and the Modeling of Systems*; Prentice Hall: Englewood Cliffs, NJ, 1981.

(8) David, R.; Alla, H. Petri nets for modeling of dynamic systems—a survey. *Automatica* **1994**, *30* (2), 175.

(9) Wang, Y. F.; Chang, C. T. A Petri-net based deductive reasoning strategy for fault identification in batch processes. *Ind. Eng. Chem. Res.* **2004**, *43* (11), 2704.

(10) Murata, T. Petri nets: properties, analysis and applications. *Proc. IEEE* **1989**, *77* (4), 541.

(11) Wang, Y. F.; Wu, J. Y.; Chang, C. T. Automatic hazard analysis of batch operations with Petri nets. *Reliab. Eng. Syst. Saf.* **2002**, *76* (1), 91.

(12) Karassik, I. J.; McGuire, J. T. *Centrifugal Pumps*, 2nd ed.; Chapman & Hall: New York, 1998; pp 885−887.

Received for review August 16, 2004
Revised manuscript received October 10, 2004
Accepted October 11, 2004

IE049253H