Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers and Chemical Engineering 29 (2004) 1822–1836
Computers & Chemical Engineering
www.elsevier.com/locate/compchemeng

# Generation of batch operating procedures for multiple material-transfer tasks with Petri nets

Yi-Feng Wang, Hung-Hsiang Chou, Chuei-Tin Chang*

*Department of Chemical Engineering, National Cheng Kung University, 1 Ta Hseuh Road, Tainan 70101, Taiwan, Republic of China*

Received 2 April 2004; received in revised form 28 February 2005; accepted 7 March 2005
Available online 19 April 2005

**Abstract**

The aim of this study is to develop a systematic method on the basis of Petri net models to automatically generate operating procedures for multiple material transfer tasks in any batch process. The system model of the pipeline network used for such tasks is built by assembling the component nets representing piping elements, i.e., fragments, valves, pumps and/or compressors, according to the network configuration. All possible material-transfer routes and the corresponding operating procedures are then identified in a constrained system net, which can be constructed by imposing a set of control rules to the system model. A realistic example is presented at the end of this paper to demonstrate the effectiveness and correctness of the proposed approach.


*Keywords:* Batch process; Operating procedure; Petri net; Reachability tree

## 1. Introduction

Most low-volume high-value commercial products, such as the pharmaceuticals, foods, specialty chemicals and certain types of polymers, are usually manufactured in batch processes. Due to the need to save capital costs, it is a common practice to share the existing equipments in the plant for manufacturing various different products. As a result, it is often necessary to move the process materials from one unit to another through a complex pipeline network. In addition, every energy transfer duty required in these manufacturing processes may also demand the transportation of a heating (or cooling) medium within another pipeline network to facility this utility function. Traditionally, the tasks of finding possible material transfer routes and then synthesizing the corresponding operating procedures are performed manually on an ad hoc basis. For a complex batch chemical process, the demand for time and effort may be overwhelming and, further, the resulting recipe is often error-prone. In order to relieve work load and also to improve operation safety, it is highly desirable to develop computer algorithms for automatic synthesis of the needed operation procedures.

The research on recipe generation has already advanced significantly in the last two decades. For the sake of brevity, a review of studies concerning the *continuous* processes, e.g., Fusillo and Powers (1987) and Lakshmanan and Stephanopoulos, 1988a, 1988b, 1990, is omitted in this paper. For the batch processes, two basic types of synthesis problems have been addressed in the literature, i.e., (1) the generation of plant-wide operating procedures involving a full spectrum of unit operations, and (2) the sequencing of valve-switching steps for fluid movements in plants.

The former problem is essentially concerned with the issues involved in integrating production planning/scheduling considerations into recipe management. Crooks and Macchietto (1992) has carried out such a study. In their paper, the batch processes were modeled with the State-Task Network (STN) representation (Kondili, Pantelides, & Sargent, 1988). The overall operation goal was achieved by performing a series of sub-goaling steps. These steps were obtained with a logic-based Mixed Integer Linear Programming (MILP) technique. Viswanathan, Johnsson, Venkatasubramanian, and Ärzen (1998a, 1998b) developed a hierarchical planning framework on the basis of the ISA standard S88.01

* Corresponding author. Tel.: +886 6 275 7575x62663; fax: +886 6 234 4496.
*E-mail address:* ctchang@mail.ncku.edu.tw (C.-T. Chang).

0098-1354/$ – see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compchemeng.2005.03.001

for the synthesis of batch operating procedures. In their work, a discrete event modeling tool called Grafchart was used to represent the declarative and also procedural knowledge for incrementally inferencing additional knowledge. The detailed operating procedures can then be synthesized accordingly. Kim and Moon (2000) adopted an automatic safety verification system, i.e., Symbolic Model Verifier (SMV), to synthesize a feasible operation sequence and to verify its safety. More specifically, this method can be used to identify the embedded operation error (if any), to find a minimum makespan and to synthesize an error-free operating procedure at the same time. Several examples were presented to illustrate the effectiveness of their approaches. Finally, Hoshi, Nagasawa, Yamashita, and Suzuki (2002) proposed a knowledge-based method on the basis of two separate graph models. One was used to represent the plant structure and the other the material-conversion procedures. A recursive search algorithm was developed accordingly to generate the operation recipe and this approach was successfully tested in a case study.

As mentioned previously, transporting material between source(s) and sink(s) in a pipeline network is a fundamental task to be performed in the batch processes. This is due to the fact that this task is needed to facilitate almost every type of unit operations. In a pioneering work, Rivas and Rudd (1974) proposed a method for the synthesis of failure-safe operations to assist the operators during emergency situations. The valve operation sequence can be quickly determined to reach the given operation objective. O'Shima (1978) handled this problem with a more efficient solution technique. The author developed the algorithms for finding the route 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 operation procedures were then synthesized on the basis of these algorithms. Foulkes, Walton, Andow, and Galluzzo (1988) 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 (1996) 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 results are still not mature enough for practical applications. Generally speaking, the plant-wide operating procedures generated in the first group of studies are often not given in sufficient detail for actual implementation, e.g., see Kim and Moon (2000) and Hoshi et al. (2002), and are not suitable for concurrent operations, e.g., see Crooks and Macchietto (1992). On the other hand, it is difficult to apply the specific valve-sequencing steps obtained in the second group to achieve a multi-task schedule which are quite common in an industrial plant. The focus of present study is thus concerned with the generation of operating procedures to achieved multiple material-transfer tasks defined by a given Gantt Chart. To this end, the Petri net is used as a modeling tool for the material transfer operations in batch plants. The mathematical representation of the *ordinary* Petri net is provided by Peterson (1981). 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 with a bar. A normal arc is represented with a directed solid line. It connects either a place to a transition or vice versa. In order 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 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 (David & Alla, 1994; Wang & Chang, 2004).

There have been a few related studies in the past to model and design the hierarchical supervisory control system in batch processes with Petri nets, e.g., see Tittus and Lennartson (1999) and Ferrarini and Piroddi (2003). However, none of them are suitable for the synthesis of material-transfer procedures in a complex pipeline network. In the present study, each valve, pump and piping fragment in the given process is first modeled with a component Petri net. These component models are then connected according to the P&ID. Next, in order to identify all reachable states from a chosen initial system state and their relationships, the reachability tree of the Petri net model is developed with a standard algorithm (Murata, 1989). From this reachability tree, *all* possible operating procedures (to achieve the same operation goal) can be easily identified. To describe this procedure in detail, the rest of this paper is organized as follows. The standard component models are described in Section 2. The method for generating operation steps to facilitate a specific material-transfer task is presented in Section 3. Due to the need to ensure operation safety, a unique route must be secured for each individual task. Hence, a set of equipment operation rules are incorporated in the Petri net to produce the required operation steps. A detailed description of these rules can also be found in Section 3. Since the reachability tree is used in this work as a tool for identifying operation steps, its implementation approach is outlined in Section 4. Additional control mechanisms are introduced in the Petri net model to achieve a given multi-task schedule. These features are outlined in Section 5. Finally, the results of applying the proposed synthesis procedure to an industrial-size plant are given at the end of this paper.

## 2. Component models

As mentioned previously, the process materials and heating/cooling mediums can be transferred in a batch chemical

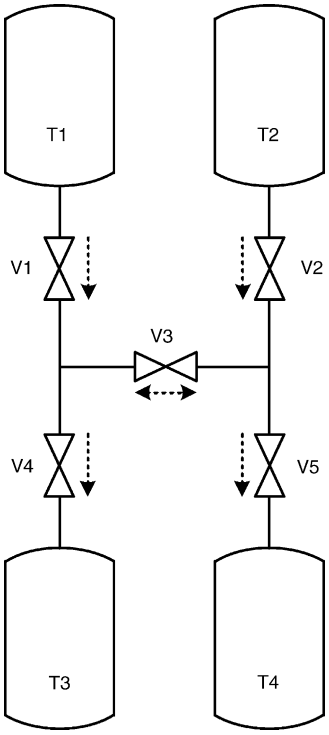


Fig. 1. An example plant.

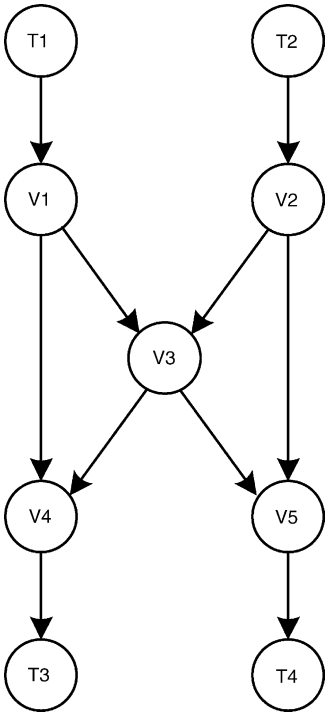


Fig. 2. Digraph model of example plant.

process via various pipeline networks. The first critical issue in modeling any network should be concerned with the division of the system into distinct components. Uthgenannt (1996) constructed the corresponding digraph by treating the process equipment, e.g., valves, pumps and storage tanks, as nodes and the pipeline branches between any two nodes as directed arcs. This approach often produces impractical recipes that may even contain spurious operation actions. Let us consider the network presented in Fig. 1 and its digraph model in Fig. 2 as an example. The material transfer routes from tank T1 to tank T3 can be identified by inspecting Fig. 2, i.e. (1) T1 → V1 → V4 → T3 and (2) T1 → V1 → V3 → V4 → T3. Notice that the opening of V3 in the second procedure is an unnecessary step. This is due to the fact that the material transfer flow through valve V3 is bi-directional in this example and, thus, each of the three pipeline branches between V1, V3 and V4 is associated with two different arcs.

To avoid the above problem, the concept of the piping *fragments* (Foulkes et al., 1988) is adopted in this work for the development of Petri net models. In particular, a fragment is defined as a collection of pipeline branches and/or processing units isolated by the valves, pumps and other means of flow blockage in the pipeline network. Let us again consider Fig. 1. Six fragments can be identified with this criterion, i.e., the shaded pipeline branches in Fig. 3. For illustration convenience, let us first consider the most basic structure of fragment, i.e., a section of pipeline FR isolated by an inlet valve VI and also an outlet valve VO [see Fig. 4(a)]. Notice that, in this case, the flow in either valve is allowed only in one direction. The corresponding Petri net model is presented in Fig. 4(b). 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 process material (or the heating/cooling medium) is delivered to the corresponding fragment from an upstream source fragment. The place PKI reflects the connection status of FR with its upstream fragment, and PKO denotes the same with a downstream fragment. On the other hand, if the outlet valve VO of the fragment in Fig. 4(a) permits bi-directional material transfer, the fragment model depicted in Fig. 4(b) should be changed to the one shown in Fig. 4(c). Notice that the transitions TXO(1) and TXO(2) are used to denote respectively the material-transfer actions from FR to the downstream fragment and vice versa. It should also be noted that, although both transfer actions are allowed, only one can be taken at a time. Thus, a token is placed in PXO initially in every simulation run. On the basis of the above model-building convention, all mass-transfer paths in Fig. 3 can be described with the Petri net shown in Fig. 5. By inserting a token in place FR1, it can be observed that there is only one possible route for the token to flow from place FR1 to place FR5, i.e., FR → FR3 → FR5.

Other than the piping fragments, it is also necessary to develop Petri net models for the valves, pumps and compressors. The valve model is presented in Fig. 6. 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

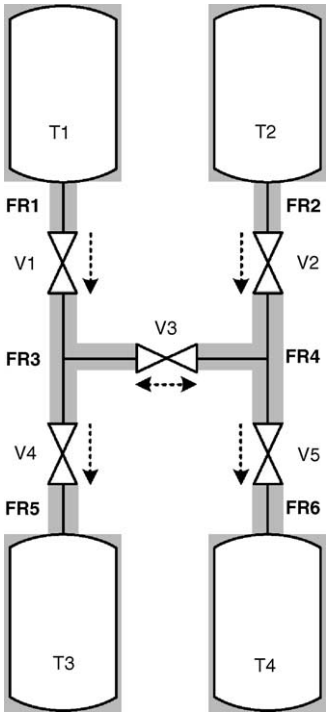


Fig. 3. Piping fragments of example plant.

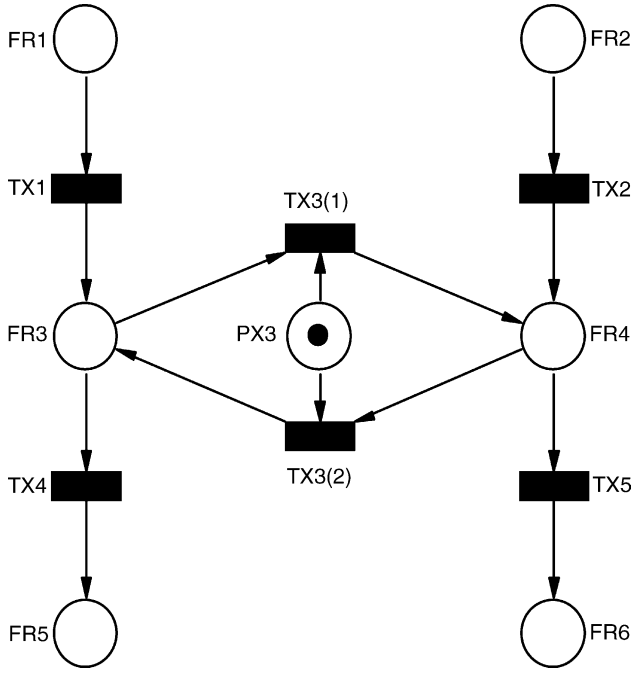


Fig. 5. Petri net model of material-transfer paths in the example plant.

the transitions 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 valve-closing operation. The output places PA(O) and PA(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. Since 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.

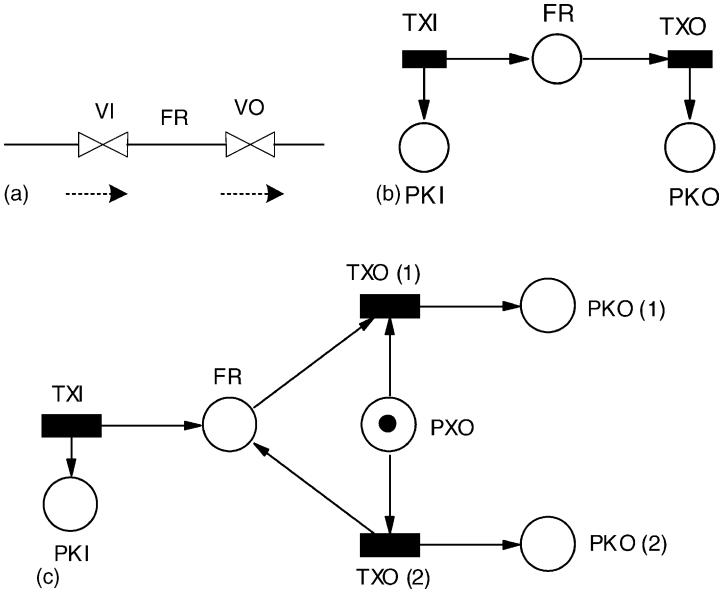


Fig. 4. (a) Basic structure of a piping fragment with two single-direction valves. (b) Petri net model of a basic fragment structure (two single-direction valves). (c) Petri net model of a basic fragment structure (one single-direction valve and one bi-direction valve).

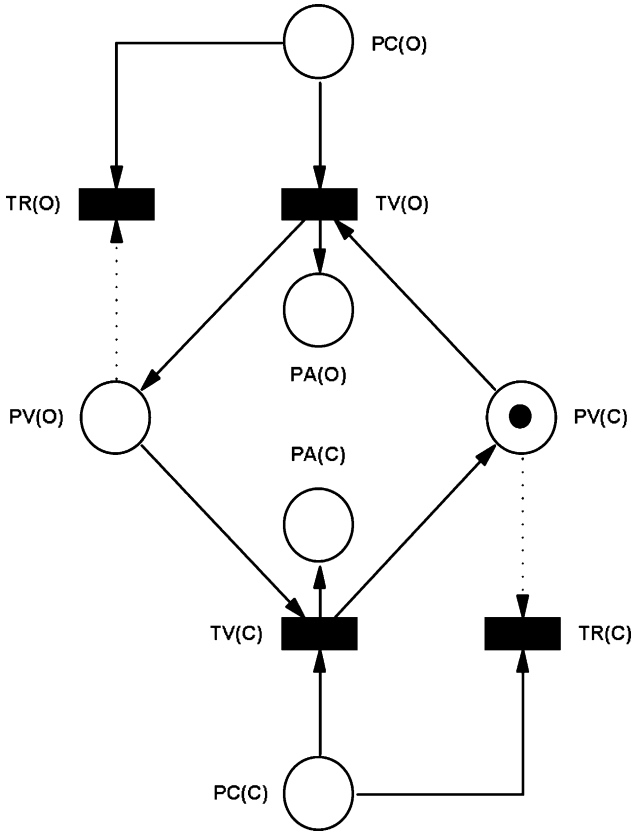


Fig. 6. Petri net model of a valve.

Since the operating procedures of pumps and compressors can be considered as well-established industrial practices, e.g., see Karassik and McGuire (1998), their detailed steps are not described in their component models for the sake of simplicity. Specifically, the valve model presented in Fig. 6 is also used for representing a power-generating device in this study. In this case, the places PV(O) and PV(C) represent two opposite states, i.e., on and off, of the device 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.

## 3. Generation of operation steps

The operating procedure can be obtained based on a system model. To create such a model, the component models should be assembled one-by-one according to the network configuration. For example, the Petri net model of the plant described in Fig. 1 can be produced easily using this approach (see Fig. 7). Notice that the transitions TXs are connected to the places PC(O)s with normal arcs. This practice is meant to reflect the relationship between each material-transfer action and the need to open the corresponding valve. Given a route and a set of initial valve states, a specific operating procedure can then be synthesized by performing simulation with the Petri net. Let us consider the example in Fig. 1 again. Under the condition that all valves are closed initially, valves V1 and V4 should be switched to the open position for transferring material through the previously-identified route FR1 → FR3 → FR5. Notice that the initial system condition of the Petri net model can be set by introducing a token in FR1 and also in the place representing the close position of each valve. The above operation steps can then be identified with the Petri net based simulation. However, it should also be noted that, if a different set of initial valve states is adopted in the simulation run, the resulting operating procedure still remains the same. This is clearly undesirable. Thus, in order 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 upstream fragment* (*except the one used for facilitating the present material transfer*) *should be closed/switched off.*

To realize this requirement of blocking all the entrances and exits not located on the selected route, additional normal output arcs should be introduced to connect the transition representing the given action, i.e., TX, with the places PC(C)s in the component models of the corresponding equipments. Furthermore, to maintain the equipment state needed to execute the given action, an extra inhibitor arc should also be inserted between place PK and the transition TV(C) in the component model of this equipment. As a result of applying these control rules, additional normal and inhibitor arcs are incorporated in the Petri net presented in Fig. 7. This modified version is shown in Fig. 8 and, for better readability, the added arcs are drawn with bold solid lines. Let us once again consider the material-transfer route FR1 → FR3 → FR5 in Fig. 3. The correct operating procedures for two different sets of initial valve states can now be generated with the modified net (see Table 1).

Table 1
The operating steps for two sets of initial valve states in the example plant

| | Case 1 | | | | | Case 2 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | V1 | V2 | V3 | V4 | V5 | V1 | V2 | V3 | V4 | V5 |
| Initial position[a] | C | C | C | C | C | C | O | O | O | C |
| Operation steps | open V1, open V4 | | | | | close V3, open V1 | | | | |

[a] C denotes “close”; O denotes “open”.

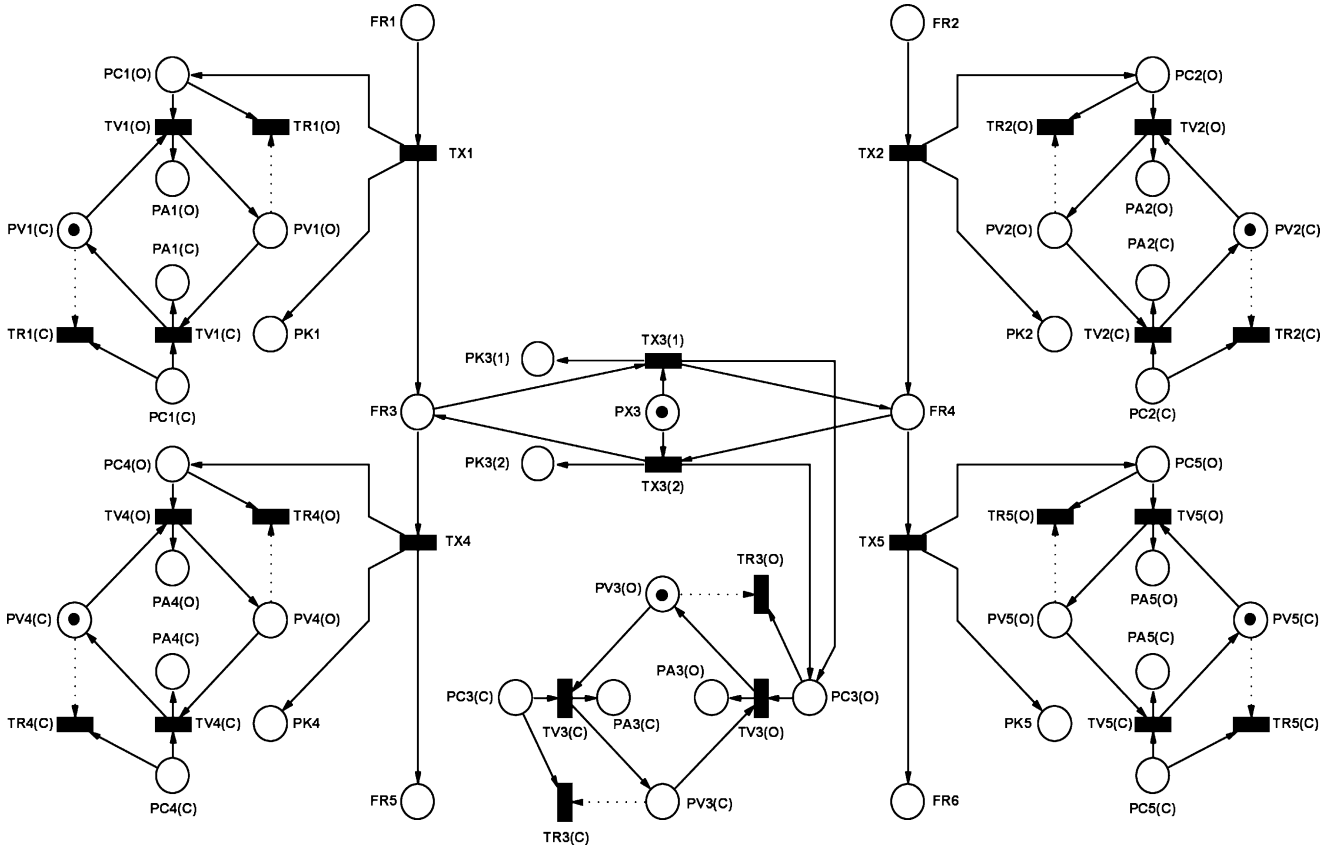


Fig. 7. The system model of example plant.

## 4. Identification of feasible routes

In the above discussions, the material-transfer route is assumed to be available before the required operation steps can be generated. Since there may be more than one route between a source fragment and a sink fragment, it is often desirable to identify all possible routes for the sake of operation flexibility. This task can be achieved by constructing a reachability tree on the basis of the corresponding Petri net model. Specifically, one can first obtain as many "new" markings as the number of the enabled transitions according to the initial marking. From each new marking, one can then again generate more markings. Repeating this procedure over and over results in a reachability tree. The 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 reachability tree should be carried out 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 breadth-first strategy, the reachability tree of a given Petri net can be constructed according to the following algorithm (Murata, 1989; Wang, Wu, & Chang, 2002):

1. Label the initial marking $M_0$ 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 $M$.
   (b) If the marking $M$ is identical to that of an existing node in the constructed tree, then convert the frontier node to a duplicate node and then go to step 2(a).
   (c) Use the revised Murata's state equation to obtain all possible enabled transitions. If no transitions are enabled for the marking $M$, then convert the selected frontier node to a terminal node and then go to step 2(a). 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:
      - Obtain the marking $M_k$ by firing $t_f$.
      - Include $M_k$ as a node, draw a directed arc with label $t_f$ from $M$ to $M_k$, and tag $M_k$ as a frontier node.

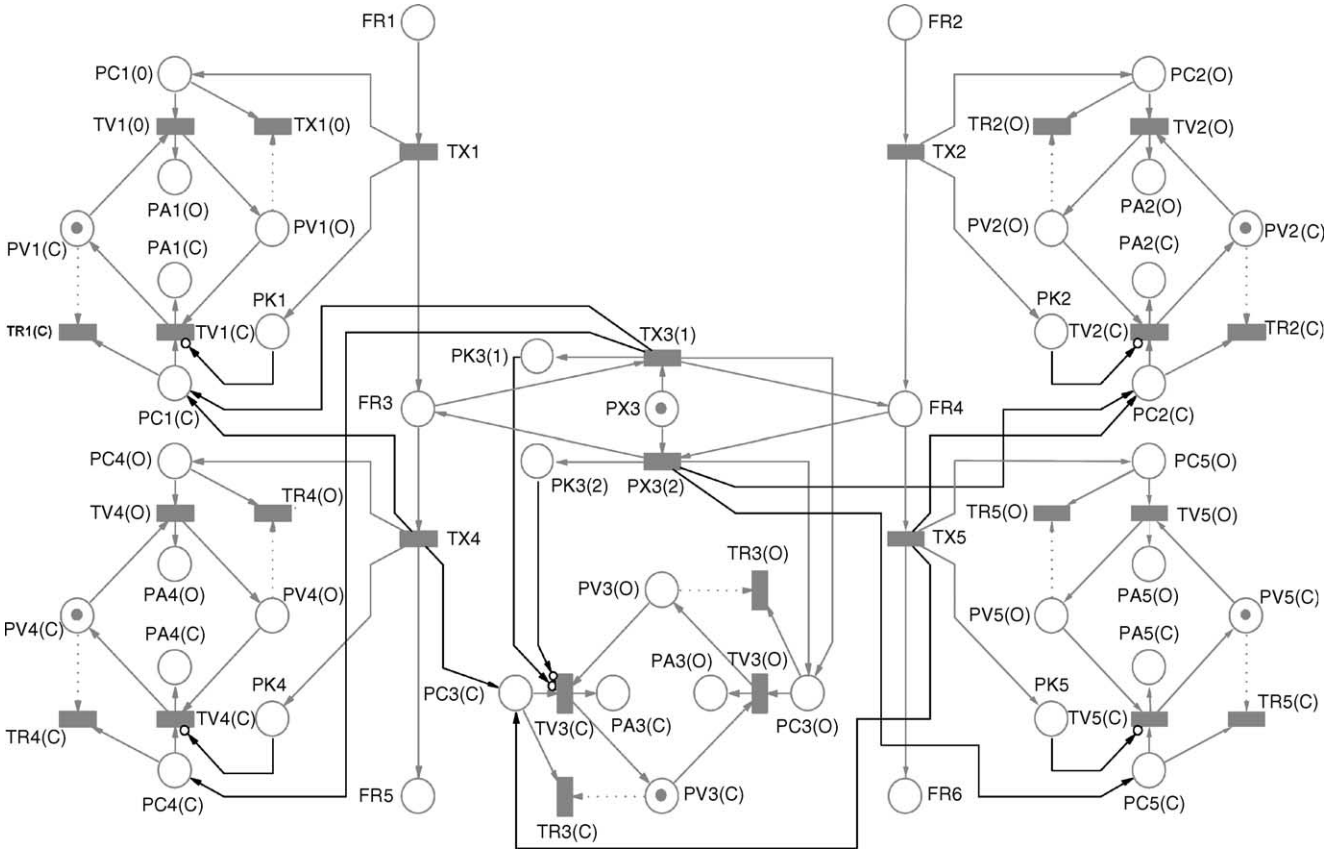


Fig. 8. The system model with equipment operation rules.

- Remove the original tag from $M$ and tag it as an interior node.

In developing the reachability tree for our purpose, all enabled transitions associated with the valve-switching actions, i.e., those labeled as "TR" and "TV", must be fired *before* triggering any transition representing material movement, i.e., the one labeled as "TX", to avoid generating duplicate branches. This constraint can be easily imposed by modifying the firing sequence of enabled transitions in step 2(c) of above algorithm.

Let us turn to the system described by Fig. 3. By assuming that the raw material is stored in tank T1 initially and all valves are closed except for valve V3, the corresponding reachability tree (shown in Fig. 9) can be generated according to the proposed algorithm. The markings associated with the nodes in this tree can be found in Table 2. In order to clearly identify the elements in a marking, the token numbers are grouped into six subsets and arranged sequentially in a vector, i.e.,

$$M_k = [\mathbf{FR}_k, \mathbf{PA(O)}_k, \mathbf{PA(C)}_k, \mathbf{PV(O)}_k, \mathbf{PV(C)}_k, \mathbf{PK}_k]$$

and $k = 0, 1, 2, \ldots, 8$. Here, each subset label is identical to the place labels of its elements. For example, the elements stored in $\mathbf{FR}_k$ are the token numbers of places representing the fragment states. Based on 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 one in the first subset $\mathbf{FR}_k$ of the terminal node. The actual valve-opening and valve-closing actions that make the transfer in each route possible can be found from the elements in the subsets $\mathbf{PA(O)}_k$ and $\mathbf{PA(C)}_k$, respectively. The final valve states can be obtained from the subsets $\mathbf{PV(O)}_k$ and $\mathbf{PV(C)}_k$. Finally, the connection status among various fragments in the network can be identified from $\mathbf{PK}_k$. Notice that the subsets $\mathbf{PC(O)}_k$, $\mathbf{PC(C)}_k$ and $\mathbf{PX}_k$ are not included in Table 2 since these data are not useful for the identification of the feasible operation steps. According to the reachability tree given in Fig. 9, it can be concluded that two material-transfer routes are originated from fragment FR1, i.e., FR1 → FR3 → FR5 (see $M_6$ in Table 2) and FR1 → FR3 → FR4 → FR6 (see $M_8$ in Table 2). From the subsets $\mathbf{PA(O)}_6$ and $\mathbf{PA(C)}_6$, it is clear that the former material transfer task can be accomplished by closing V3 and then opening both V1 and V4. Similarly, the operation steps for the latter can be found in $\mathbf{PA(O)}_8$ and $\mathbf{PA(C)}_8$, i.e., opening V1 and V5. Notice that the valve-closing actions should be implemented at instances *earlier* than those of the valve-opening steps in the above procedures. This is due to the need to eliminate the possibility of transferring material to fragments that are not located

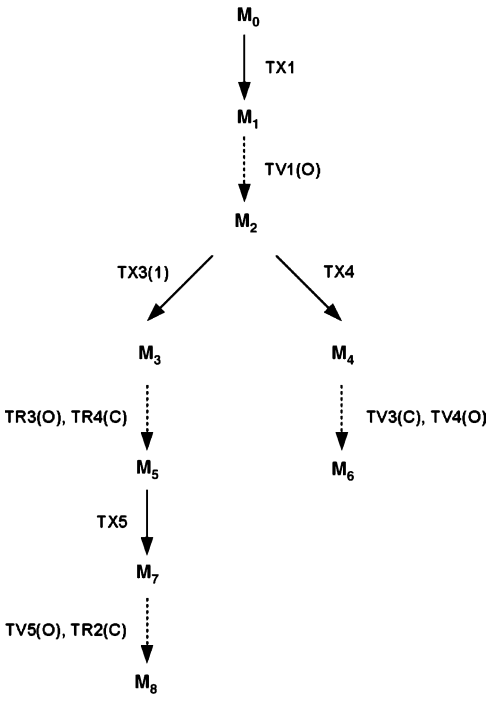


Fig. 9. A reachability tree of the Petri net in Fig. 8.

on the selected route. Notice also that these operation steps can only be used to *facilitate* material transfer. The operating procedures to *terminate* the transfer tasks have not yet been discussed at this point.

Finally, it should be pointed out that the proposed route-searching algorithm may fail if the Petri net contains loops. For example, let us apply the above procedure to the network presented in Fig. 10. It can be found that there are five possible routes emanating from the fragment FR1. However, one of them forms an infinite loop, i.e.

$$\mathrm{FR1} \rightarrow \mathrm{FR3} \rightarrow \mathrm{FR4} \rightarrow \mathrm{FR6} \rightarrow \mathrm{FR5} \rightarrow \mathrm{FR3} \rightarrow \mathrm{FR4} \rightarrow \mathrm{FR6} \rightarrow \mathrm{FR5} \rightarrow \cdots$$

This loop is caused basically by the bi-directional valves V3, V4, V5 and V6. The effort to assemble these looping routes is obviously futile and, thus, they should be avoided in the search process. To this end, it is necessary to impose another set of control rules on the system model. These *connection enforcement rules* are summarized below.

Table 2
The markings associated with the nodes in Fig. 9

| Markings | Subsets |
|---|---|
| $M_0$ | {100000}{00000}{00000}{00100}{11011}{000000} |
| $M_1$ | {001000}{00000}{00000}{00100}{11011}{100000} |
| $M_2$ | {001000}{10000}{00000}{10100}{01011}{100000} |
| $M_3$ | {000100}{10000}{00000}{10100}{01011}{101000} |
| $M_4$ | {000010}{10000}{00000}{10100}{01011}{100010} |
| $M_5$ | {000100}{10000}{00000}{10100}{01011}{101000} |
| $M_6$ | {000010}{10010}{00100}{10010}{01101}{100010} |
| $M_7$ | {000001}{10000}{00000}{10100}{01011}{101001} |
| $M_8$ | {000001}{10001}{00000}{10101}{01010}{101001} |

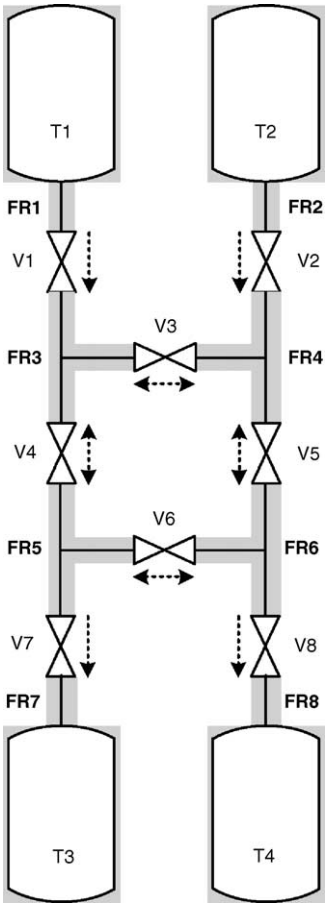


Fig. 10. Another fictitious example plant.

**Connection Enforcement Rules:** *Given a specific material-transfer action, any additional transport into the upstream fragment of this action should be prohibited to avoid development of looping routes.*

Specifically, an inhibitor arc should be added between the place representing the connection status resulting from the transfer action under consideration and each transition representing an inlet connection of its upstream fragment.

## 5. Execution of multiple tasks

In this study, it is assumed that an appropriate schedule for carrying out all material-transfer tasks is already available before the synthesis of specific operating procedures. For illustration convenience, let us consider two tasks scheduled according to the Gantt charts shown in Fig. 11(a) and (b). They are referred to as *schedule A* and *schedule B*, respectively in this paper. In essence, the key decision in generating the operating procedures to implement these two schedules is concerned with the issue of *fragment sharing*. In the former case, the two material-transfer routes adopted to accomplish the given tasks are allowed to be overlapping. To provide such opportunities, the token number in every place with ini-

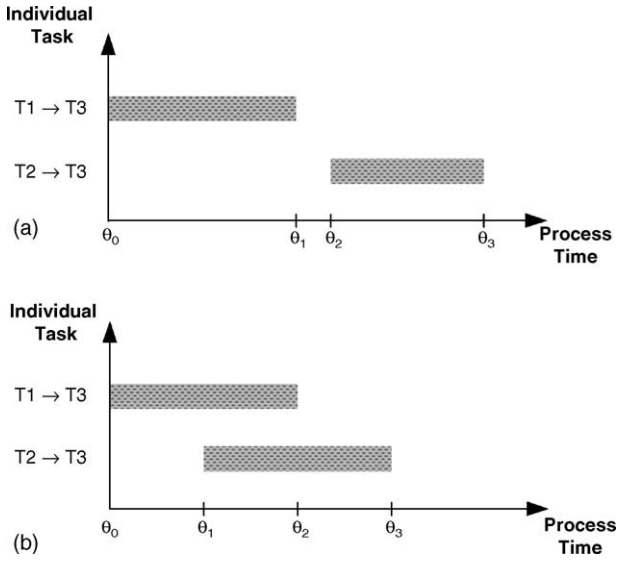


Fig. 11. (a) Schedule A. (b) Schedule B.

tial label "PK" should be reset to zero before initiating the search process to create the operation steps of the second task. On the other hand, all place-resetting possibilities must be eliminated in schedule B.

The above requirements can be viewed as the final set of control rules in Petri net model. For illustration purpose, let us consider a mixing operation in the pipeline network given in Fig. 3. It is assumed that the raw materials of this operation (say X and Y) are stored in tank T1 and tank T2, respectively and tank T3 is used as a mixer. Thus, the needed material-transfer tasks should be: (1) transporting material X from tank T1 to tank T3 and (2) transporting material Y from tank T2 to tank T3. The corresponding control rules can be imposed by attaching the component model of a fictitious *schedule manager* to the system model. This component in present example consists of a *timer* model [see Fig. 12(a)] and the models of two *task coordinators* [see Fig. 12(b)].

The timer model can be constructed according to a Gantt chart. On the basis of the delay times assigned to the transitions TC(1)–TC(4) in Fig. 12(a), it is clear that the places P(1), P(2) and P(3) can be used to reflect the time intervals $(\theta_0, \theta_1)$, $(\theta_1, \theta_2)$ and $(\theta_2, \theta_3)$, respectively for both schedules. The place PS($i$) ($i = 1, 2, 3, 4$) marks the instance that a task begins or ends. If schedule A is adopted, PS(1) and PS(3) denote the initiation commands of the first and second tasks, respectively and PS(2) and PS(4) the corresponding termination commands. On the other hand, PS(1) and PS(2) represent the initiation signals and PS(3) and PS(4) the termination signals in schedule B.

The Petri net model of a task coordinator can be divided into three smaller nets. They are referred to in this paper as the models of *route synthesizer*, *reset processor* and *task terminator* according to their respective functionalities. Notice that, once a task initiation signal is generated by the timer, i.e., a token is introduced in place PS($i$), a token should flow pass the place "reset" and then "start" in the route synthesizer. The former is the entry point of reset processor, which performs basically two types of routine maintenance works on the system model:

1. clearing out all tokens in PA(O)s, PA(C)s, PC(O)s and PC(C)s, and
2. feeding a token to each empty PX.

These practices are taken to avoid confusion in repeating the route synthesis process for multiple tasks.

On the other hand, the route synthesis mechanism described in the previous section is activated after a token is given in the place "start" and, subsequently, in the place representing source fragment of the designated material-transfer. Note that the route synthesis process ends if a token reaches the place denoting the status of sink fragment. At this point, the task terminator is called for to perform two types of bookkeeping duties, i.e.,

1. storing the token number in every PK with the attached place representing a *recorder*, and
2. recording the token numbers in PV(O)s for the outlet valves of the source fragment and in PV(O)s for all pumps/compressors.

The operation steps to end a task can also be generated in the task terminator after a termination signal is issued by the timer. In particular, a valve (or pump/compressor) shall be closed (or switched off) if a token is residing in the corresponding *recorder* place. The tokens in PKs are also removed under the same condition. This mechanism is installed to al-

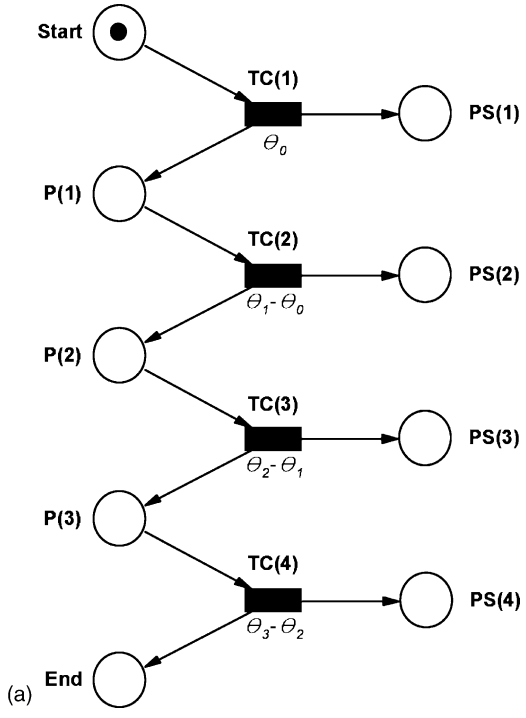


Fig. 12. (a) Petri net model of a timer. (b) Petri net model of a task coordinator.

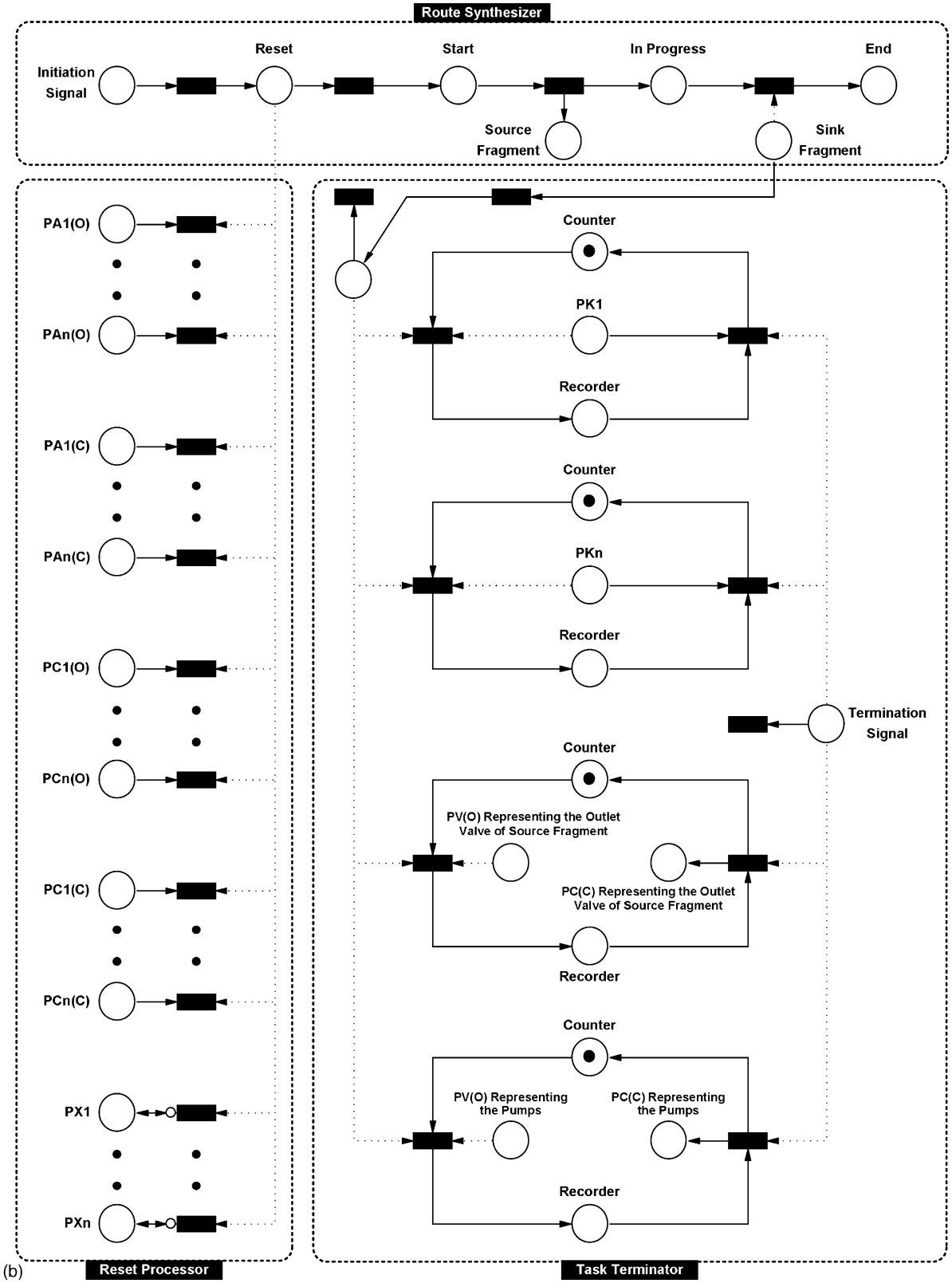


Fig. 12. (*Continued*).

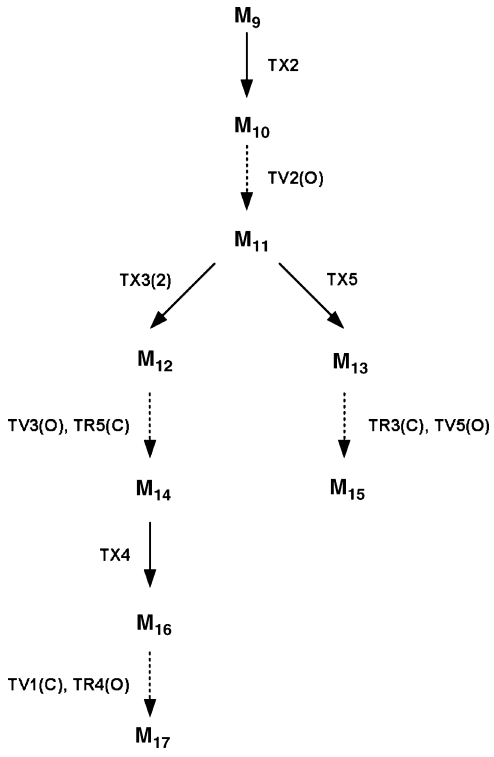


Fig. 13. The reachability tree associated with the second task in schedule A.

low overlapping routes in schedule A and also to produce non-overlapping ones in schedule B. Notice also that, since the second task begins at a time when the first is still in progress in schedule B, two material-transfer routes should have already been created before task 1 can be put to an end. It is thus necessary to provide a means to distinguish the two in determining the termination steps. This is achieved with the places denoted as the *counters*. Since only one set of counters are used in our model and these places are meant to be shared by the two task coordinators, the operation steps required to stop material transfer in the first route cannot be adopted in the second.

Let us again assume that all valves in Fig. 3 are closed initially except V3 and try to generate the operating procedures to achieve schedule A. The corresponding reachability tree can be constructed with the proposed algorithm. Notice that the first part of this tree, which is associated with the first material-transfer task, has already been produced previously in Fig. 9. The remainder is presented in Fig. 13. It should be noted that this subtree is developed from the terminal node $M_6$ in Fig. 9 since its marking reflects the system state reached after the material-transfer routes of the first task have been successfully synthesized. The markings of the nodes in Fig. 13 are given in Table 3. From the information embedded in the entire reachability tree, one can deduce the operating procedures presented in Table 4.

Let us next consider the operating procedures to execute schedule B. The corresponding subtree and the markings of its nodes can be found in Fig. 14 and Table 5, respectively.

Table 3
The markings associated with the nodes in Fig. 13

| Markings | Subsets |
|---|---|
| $M_9$ | {010000}{00000}{00000}{00010}{11101}{000000} |
| $M_{10}$ | {000100}{00000}{00000}{00010}{11101}{010000} |
| $M_{11}$ | {000100}{01000}{00000}{01010}{10101}{010000} |
| $M_{12}$ | {001000}{01000}{00000}{01010}{10101}{010100} |
| $M_{13}$ | {000001}{01000}{00000}{01010}{10101}{010001} |
| $M_{14}$ | {001000}{01100}{00000}{01110}{10001}{010100} |
| $M_{15}$ | {000001}{01001}{00000}{01011}{10100}{010001} |
| $M_{16}$ | {000010}{01100}{00000}{01110}{10001}{010110} |
| $M_{17}$ | {000010}{01100}{10000}{00110}{11001}{010110} |

Table 4
The operating steps for executing schedule a in the example plant

| Task | Route | Operation steps |
|---|---|---|
| 1 | FR1 → FR3 → FR5 | (1) Close V3 and open V1, V4 at $\theta_0$; (2) close V1 at $\theta_1$ |
| 2 | FR2 → FR4 → FR3 → FR5 | (1) Open V2, V3 at $\theta_2$; (2) close V2 at $\theta_3$ |

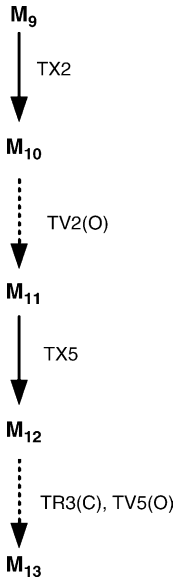


Fig. 14. The reachability tree associated with the second task in schedule B.

Notice that only one material-transfer route can be identified, i.e., FR2 → FR4 → FR6, and its sink fragment does not contain tank T3. Thus, one can see that it is really not possible to carry out the second task from the system state attained after activating the first.

Table 5
The markings associated with the nodes in Fig. 14

| Markings | Subsets |
|---|---|
| $M_9$ | {010000}{00000}{00000}{10010}{01101}{100010} |
| $M_{10}$ | {000100}{00000}{00000}{10010}{01101}{110010} |
| $M_{11}$ | {000100}{01000}{00000}{11010}{00101}{110010} |
| $M_{12}$ | {000001}{01000}{00000}{11010}{00101}{110011} |
| $M_{13}$ | {000001}{01001}{00000}{11011}{00100}{110011} |

## 6. Case study

The pipeline network described by Foulkes et al. (1988) is adopted in the present work as a realistic example to demonstrate the capability of the proposed method. The network contains eight storage tanks, 40 valves and four pumps (see Fig. 15). A total of 36 pipeline fragments, i.e., FR1–FR36, can be defined in this system. Here, five different raw materials are stored in five separate tanks, i.e., T1–T5, respectively. It is assumed that all valves are closed and all pumps are turned off initially, and also that the schedule shown in Fig. 16 is adopted in the present case study.

For the sake of brevity, the complete system model and the corresponding reachability tree are not included in this paper. In principle, the proposed approach can be applied to generate all feasible material-transfer routes to meet the given schedule. A total of 52 routes have been identified to accomplish task 1 (see Table 6). From Fig. 16, it is clear that the first two tasks should be executed according to schedule B and it may not be possible to find any feasible route to facilitate task 2 at time $\theta_1$ after a particular selection has already been made for task 1 at time $\theta_0$. In this example, 31 different feasible combinations have been identified (see Table 7). Finally, the material-transfer routes of task 3 should be synthesized on the basis of the feasible route combinations of the first two tasks. The first and third tasks in this case must

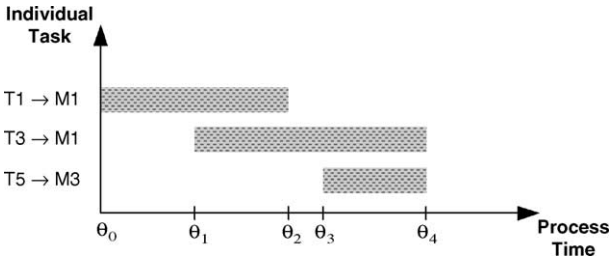


Fig. 16. Task schedule of case study.

be carried out on the basis of schedule A and, on the other hand, the second and third should follow schedule B. The total number of resulting feasible combinations for the complete schedule is 79 and, for the sake of brevity, only four of them are listed in Table 8. The corresponding operating procedures can be found in Table 9. As mentioned previously, in the procedures to facilitate a material-transfer task, there is a preferred order in operating the valves, i.e., the closing steps should be implemented before the opening steps can be applied. In addition, the pump/compressor must be turned on after all valve-switching actions are completed. This practice is taken as a measure of safety precaution. On the other hand, the preferred order should be reversed in the operation procedure to terminate a task, i.e., the pump/compressor should be switched off before any valve can be closed.

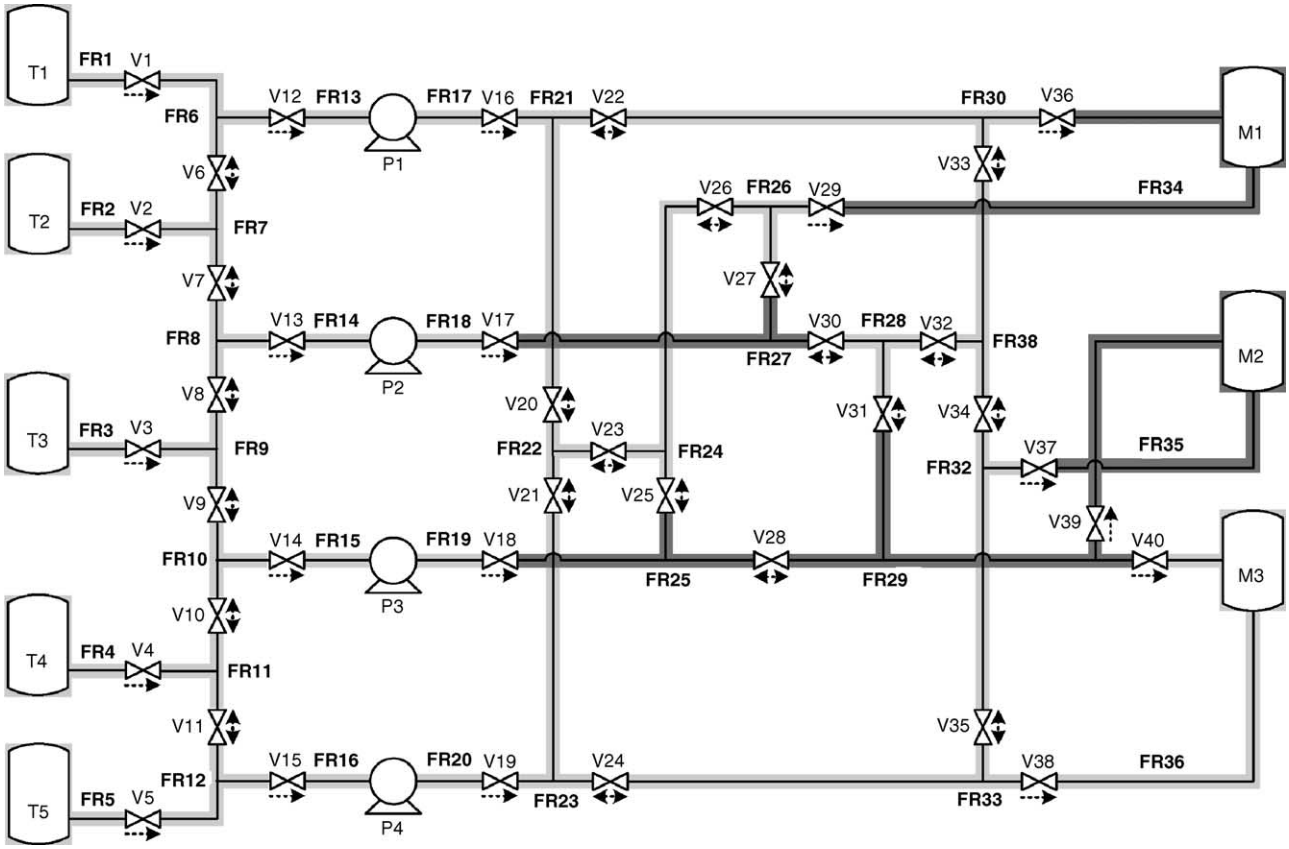


Fig. 15. The pipeline network in a batch plant.

Table 6
The feasible material-transfer routes from tank T1 to tank $M_1$

| Route no. | Material-transfer route |
|---|---|
| 1 | 1 → 6 → 13 → 17 → 21 → 22 → 24 → 26 → 34 |
| 2 | 1 → 6 → 13 → 17 → 21 → 30 → 31 → 28 → 27 → 26 → 34 |
| 3 | 1 → 6 → 13 → 17 → 21 → 30 → 31 → 28 → 29 → 25 → 24 → 26 → 34 |
| 4 | 1 → 6 → 13 → 17 → 21 → 22 → 24 → 25 → 29 → 28 → 27 → 26 → 34 |
| 5 | 1 → 6 → 13 → 17 → 21 → 22 → 23 → 33 → 32 → 31 → 28 → 27 → 26 → 34 |
| 6 | 1 → 6 → 13 → 17 → 21 → 30 → 31 → 32 → 33 → 23 → 22 → 24 → 26 → 34 |
| 7 | 1 → 6 → 13 → 17 → 21 → 22 → 23 → 33 → 32 → 31 → 28 → 29 → 25 → 24 → 26 → 34 |
| 8 | 1 → 6 → 13 → 17 → 21 → 30 → 31 → 32 → 33 → 23 → 22 → 24 → 25 → 29 → 28 → 27 → 26 → 34 |
| 9 | 1 → 6 → 7 → 8 → 14 → 18 → 27 → 26 → 34 |
| 10 | 1 → 6 → 7 → 8 → 14 → 18 → 27 → 28 → 29 → 25 → 24 → 26 → 34 |
| 11 | 1 → 6 → 7 → 8 → 14 → 18 → 27 → 28 → 31 → 30 → 21 → 22 → 24 → 26 → 34 |
| 12 | 1 → 6 → 7 → 8 → 14 → 18 → 27 → 28 → 31 → 32 → 33 → 23 → 22 → 24 → 26 → 34 |
| 13 | 1 → 6 → 7 → 8 → 9 → 10 → 15 → 19 → 25 → 24 → 26 → 34 |
| 14 | 1 → 6 → 7 → 8 → 9 → 10 → 15 → 19 → 25 → 29 → 28 → 27 → 26 → 34 |
| 15 | 1 → 6 → 7 → 8 → 9 → 10 → 15 → 19 → 25 → 24 → 22 → 21 → 30 → 31 → 28 → 27 → 26 → 34 |
| 16 | 1 → 6 → 7 → 8 → 9 → 10 → 15 → 19 → 25 → 29 → 28 → 31 → 30 → 21 → 22 → 24 → 26 → 34 |
| 17 | 1 → 6 → 7 → 8 → 9 → 10 → 15 → 19 → 25 → 29 → 28 → 31 → 32 → 33 → 23 → 22 → 24 → 26 → 34 |
| 18 | 1 → 6 → 7 → 8 → 9 → 10 → 15 → 19 → 25 → 24 → 22 → 23 → 33 → 32 → 31 → 28 → 27 → 26 → 34 |
| 19 | 1 → 6 → 7 → 8 → 9 → 10 → 11 → 12 → 16 → 20 → 23 → 22 → 24 → 26 → 34 |
| 20 | 1 → 6 → 7 → 8 → 9 → 10 → 11 → 12 → 16 → 20 → 23 → 33 → 32 → 31 → 28 → 27 → 26 → 34 |
| 21 | 1 → 6 → 7 → 8 → 9 → 10 → 11 → 12 → 16 → 20 → 23 → 22 → 21 → 30 → 31 → 28 → 27 → 26 → 34 |
| 22 | 1 → 6 → 7 → 8 → 9 → 10 → 11 → 12 → 16 → 20 → 23 → 22 → 24 → 25 → 29 → 28 → 27 → 26 → 34 |
| 23 | 1 → 6 → 7 → 8 → 9 → 10 → 11 → 12 → 16 → 20 → 23 → 33 → 32 → 31 → 28 → 29 → 25 → 24 → 26 → 34 |
| 24 | 1 → 6 → 7 → 8 → 9 → 10 → 11 → 12 → 16 → 20 → 23 → 33 → 32 → 31 → 30 → 21 → 22 → 24 → 26 → 34 |
| 25 | 1 → 6 → 7 → 8 → 9 → 10 → 11 → 12 → 16 → 20 → 23 → 22 → 21 → 30 → 31 → 28 → 29 → 25 → 24 → 26 → 34 |
| 26 | 1 → 6 → 7 → 8 → 9 → 10 → 11 → 12 → 16 → 20 → 23 → 33 → 32 → 31 → 30 → 21 → 22 → 24 → 25 → 29 → 28 → 27 → 26 → 34 |
| 27 | 1 → 6 → 13 → 17 → 21 → 30 → 34 |
| 28 | 1 → 6 → 13 → 17 → 21 → 22 → 23 → 33 → 32 → 31 → 30 → 34 |
| 29 | 1 → 6 → 13 → 17 → 21 → 22 → 24 → 26 → 27 → 28 → 31 → 30 → 34 |
| 30 | 1 → 6 → 13 → 17 → 21 → 22 → 24 → 25 → 29 → 28 → 31 → 30 → 34 |
| 31 | 1 → 6 → 7 → 8 → 14 → 18 → 27 → 28 → 31 → 30 → 34 |
| 32 | 1 → 6 → 7 → 8 → 14 → 18 → 27 → 26 → 24 → 22 → 21 → 30 → 34 |
| 33 | 1 → 6 → 7 → 8 → 14 → 18 → 27 → 28 → 29 → 25 → 24 → 22 → 21 → 30 → 34 |
| 34 | 1 → 6 → 7 → 8 → 14 → 18 → 27 → 26 → 24 → 25 → 29 → 28 → 31 → 30 → 34 |
| 35 | 1 → 6 → 7 → 8 → 14 → 18 → 27 → 26 → 24 → 22 → 23 → 33 → 32 → 31 → 30 → 34 |
| 36 | 1 → 6 → 7 → 8 → 14 → 18 → 27 → 28 → 31 → 32 → 33 → 23 → 22 → 21 → 30 → 34 |
| 37 | 1 → 6 → 7 → 8 → 14 → 18 → 27 → 28 → 29 → 25 → 24 → 22 → 23 → 33 → 32 → 31 → 30 → 34 |
| 38 | 1 → 6 → 7 → 8 → 14 → 18 → 27 → 26 → 24 → 25 → 29 → 28 → 31 → 32 → 33 → 23 → 22 → 21 → 30 → 34 |
| 39 | 1 → 6 → 7 → 8 → 9 → 10 → 15 → 19 → 25 → 24 → 22 → 21 → 30 → 34 |
| 40 | 1 → 6 → 7 → 8 → 9 → 10 → 15 → 19 → 25 → 29 → 28 → 31 → 30 → 34 |
| 41 | 1 → 6 → 7 → 8 → 9 → 10 → 15 → 19 → 25 → 24 → 26 → 27 → 28 → 31 → 30 → 34 |
| 42 | 1 → 6 → 7 → 8 → 9 → 10 → 15 → 19 → 25 → 24 → 22 → 23 → 33 → 32 → 31 → 30 → 34 |
| 43 | 1 → 6 → 7 → 8 → 9 → 10 → 15 → 19 → 25 → 29 → 28 → 27 → 26 → 24 → 22 → 21 → 30 → 34 |
| 44 | 1 → 6 → 7 → 8 → 9 → 10 → 15 → 19 → 25 → 29 → 28 → 31 → 32 → 33 → 23 → 22 → 21 → 30 → 34 |
| 45 | 1 → 6 → 7 → 8 → 9 → 10 → 15 → 19 → 25 → 24 → 26 → 27 → 28 → 31 → 32 → 33 → 23 → 22 → 21 → 30 → 34 |
| 46 | 1 → 6 → 7 → 8 → 9 → 10 → 15 → 19 → 25 → 29 → 28 → 27 → 26 → 24 → 22 → 23 → 33 → 32 → 31 → 30 → 34 |
| 47 | 1 → 6 → 7 → 8 → 9 → 10 → 11 → 12 → 16 → 20 → 23 → 22 → 21 → 30 → 34 |
| 48 | 1 → 6 → 7 → 8 → 9 → 10 → 11 → 12 → 16 → 20 → 23 → 33 → 32 → 31 → 30 → 34 |
| 49 | 1 → 6 → 7 → 8 → 9 → 10 → 11 → 12 → 16 → 20 → 23 → 22 → 24 → 25 → 29 → 28 → 31 → 30 → 34 |
| 50 | 1 → 6 → 7 → 8 → 9 → 10 → 11 → 12 → 16 → 20 → 23 → 22 → 24 → 26 → 27 → 28 → 31 → 30 → 34 |
| 51 | 1 → 6 → 7 → 8 → 9 → 10 → 11 → 12 → 16 → 20 → 23 → 33 → 32 → 31 → 28 → 29 → 25 → 24 → 22 → 21 → 30 → 34 |
| 52 | 1 → 6 → 7 → 8 → 9 → 10 → 11 → 12 → 16 → 20 → 23 → 33 → 32 → 31 → 28 → 27 → 26 → 24 → 22 → 21 → 30 → 34 |

Table 7
The feasible route combinations for performing tasks 1 and 2

| Route no. for $T1 \rightarrow M_1$ | Route no. for $T3 \rightarrow M_1$ | Material-transfer route for $T3 \rightarrow M_1$ |
|---|---|---|
| 1 | 1 | $3 \rightarrow 9 \rightarrow 8 \rightarrow 14 \rightarrow 18 \rightarrow 27 \rightarrow 28 \rightarrow 31 \rightarrow 30 \rightarrow 34$ |
| | 2 | $3 \rightarrow 9 \rightarrow 10 \rightarrow 15 \rightarrow 19 \rightarrow 25 \rightarrow 29 \rightarrow 28 \rightarrow 31 \rightarrow 30 \rightarrow 34$ |
| | 3 | $3 \rightarrow 9 \rightarrow 10 \rightarrow 11 \rightarrow 12 \rightarrow 16 \rightarrow 20 \rightarrow 23 \rightarrow 33 \rightarrow 32 \rightarrow 31 \rightarrow 30 \rightarrow 34$ |
| 4 | 3 | $3 \rightarrow 9 \rightarrow 10 \rightarrow 11 \rightarrow 12 \rightarrow 16 \rightarrow 20 \rightarrow 23 \rightarrow 33 \rightarrow 32 \rightarrow 31 \rightarrow 30 \rightarrow 34$ |
| 9 | 2 | $3 \rightarrow 9 \rightarrow 10 \rightarrow 15 \rightarrow 19 \rightarrow 25 \rightarrow 29 \rightarrow 28 \rightarrow 31 \rightarrow 30 \rightarrow 34$ |
| | 4 | $3 \rightarrow 9 \rightarrow 10 \rightarrow 15 \rightarrow 19 \rightarrow 25 \rightarrow 24 \rightarrow 22 \rightarrow 21 \rightarrow 30 \rightarrow 34$ |
| | 5 | $3 \rightarrow 9 \rightarrow 10 \rightarrow 15 \rightarrow 19 \rightarrow 25 \rightarrow 24 \rightarrow 22 \rightarrow 23 \rightarrow 33 \rightarrow 32 \rightarrow 31 \rightarrow 30 \rightarrow 34$ |
| | 6 | $3 \rightarrow 9 \rightarrow 10 \rightarrow 15 \rightarrow 19 \rightarrow 25 \rightarrow 29 \rightarrow 28 \rightarrow 31 \rightarrow 32 \rightarrow 33 \rightarrow 23 \rightarrow 22 \rightarrow 21 \rightarrow 30 \rightarrow 34$ |
| | 7 | $3 \rightarrow 9 \rightarrow 10 \rightarrow 11 \rightarrow 12 \rightarrow 16 \rightarrow 20 \rightarrow 23 \rightarrow 22 \rightarrow 21 \rightarrow 30 \rightarrow 34$ |
| | 3 | $3 \rightarrow 9 \rightarrow 10 \rightarrow 11 \rightarrow 12 \rightarrow 16 \rightarrow 20 \rightarrow 23 \rightarrow 33 \rightarrow 32 \rightarrow 31 \rightarrow 30 \rightarrow 34$ |
| | 8 | $3 \rightarrow 9 \rightarrow 10 \rightarrow 11 \rightarrow 12 \rightarrow 16 \rightarrow 20 \rightarrow 23 \rightarrow 22 \rightarrow 24 \rightarrow 25 \rightarrow 29 \rightarrow 28 \rightarrow 31 \rightarrow 30 \rightarrow 34$ |
| | 9 | $3 \rightarrow 9 \rightarrow 10 \rightarrow 11 \rightarrow 12 \rightarrow 16 \rightarrow 20 \rightarrow 23 \rightarrow 33 \rightarrow 32 \rightarrow 31 \rightarrow 28 \rightarrow 29 \rightarrow 25 \rightarrow 24 \rightarrow 22 \rightarrow 21 \rightarrow 30 \rightarrow 34$ |
| 27 | 10 | $3 \rightarrow 9 \rightarrow 8 \rightarrow 14 \rightarrow 18 \rightarrow 27 \rightarrow 26 \rightarrow 34$ |
| | 11 | $3 \rightarrow 9 \rightarrow 8 \rightarrow 14 \rightarrow 18 \rightarrow 27 \rightarrow 28 \rightarrow 29 \rightarrow 25 \rightarrow 24 \rightarrow 26 \rightarrow 34$ |
| | 12 | $3 \rightarrow 9 \rightarrow 8 \rightarrow 14 \rightarrow 18 \rightarrow 27 \rightarrow 28 \rightarrow 31 \rightarrow 32 \rightarrow 33 \rightarrow 23 \rightarrow 22 \rightarrow 24 \rightarrow 26 \rightarrow 34$ |
| | 13 | $3 \rightarrow 9 \rightarrow 10 \rightarrow 15 \rightarrow 19 \rightarrow 25 \rightarrow 24 \rightarrow 26 \rightarrow 34$ |
| | 14 | $3 \rightarrow 9 \rightarrow 10 \rightarrow 15 \rightarrow 19 \rightarrow 25 \rightarrow 29 \rightarrow 28 \rightarrow 27 \rightarrow 26 \rightarrow 34$ |
| | 15 | $3 \rightarrow 9 \rightarrow 10 \rightarrow 15 \rightarrow 19 \rightarrow 25 \rightarrow 24 \rightarrow 22 \rightarrow 23 \rightarrow 33 \rightarrow 32 \rightarrow 31 \rightarrow 28 \rightarrow 27 \rightarrow 26 \rightarrow 34$ |
| | 16 | $3 \rightarrow 9 \rightarrow 10 \rightarrow 15 \rightarrow 19 \rightarrow 25 \rightarrow 29 \rightarrow 28 \rightarrow 31 \rightarrow 32 \rightarrow 33 \rightarrow 23 \rightarrow 22 \rightarrow 24 \rightarrow 26 \rightarrow 34$ |
| | 17 | $3 \rightarrow 9 \rightarrow 10 \rightarrow 11 \rightarrow 12 \rightarrow 16 \rightarrow 20 \rightarrow 23 \rightarrow 22 \rightarrow 24 \rightarrow 26 \rightarrow 34$ |
| | 18 | $3 \rightarrow 9 \rightarrow 10 \rightarrow 11 \rightarrow 12 \rightarrow 16 \rightarrow 20 \rightarrow 23 \rightarrow 33 \rightarrow 32 \rightarrow 31 \rightarrow 28 \rightarrow 27 \rightarrow 26 \rightarrow 34$ |
| | 19 | $3 \rightarrow 9 \rightarrow 10 \rightarrow 11 \rightarrow 12 \rightarrow 16 \rightarrow 20 \rightarrow 23 \rightarrow 22 \rightarrow 24 \rightarrow 25 \rightarrow 29 \rightarrow 28 \rightarrow 27 \rightarrow 26 \rightarrow 34$ |
| | 20 | $3 \rightarrow 9 \rightarrow 10 \rightarrow 11 \rightarrow 12 \rightarrow 16 \rightarrow 20 \rightarrow 23 \rightarrow 33 \rightarrow 32 \rightarrow 31 \rightarrow 28 \rightarrow 29 \rightarrow 25 \rightarrow 24 \rightarrow 26 \rightarrow 34$ |
| 28 | 10 | $3 \rightarrow 9 \rightarrow 8 \rightarrow 14 \rightarrow 18 \rightarrow 27 \rightarrow 26 \rightarrow 34$ |
| | 11 | $3 \rightarrow 9 \rightarrow 8 \rightarrow 14 \rightarrow 18 \rightarrow 27 \rightarrow 28 \rightarrow 29 \rightarrow 25 \rightarrow 24 \rightarrow 26 \rightarrow 34$ |
| | 13 | $3 \rightarrow 9 \rightarrow 10 \rightarrow 15 \rightarrow 19 \rightarrow 25 \rightarrow 24 \rightarrow 26 \rightarrow 34$ |
| | 14 | $3 \rightarrow 9 \rightarrow 10 \rightarrow 15 \rightarrow 19 \rightarrow 25 \rightarrow 29 \rightarrow 28 \rightarrow 27 \rightarrow 26 \rightarrow 34$ |
| 30 | 10 | $3 \rightarrow 9 \rightarrow 8 \rightarrow 14 \rightarrow 18 \rightarrow 27 \rightarrow 26 \rightarrow 34$ |
| 31 | 13 | $3 \rightarrow 9 \rightarrow 10 \rightarrow 15 \rightarrow 19 \rightarrow 25 \rightarrow 24 \rightarrow 26 \rightarrow 34$ |
| | 17 | $3 \rightarrow 9 \rightarrow 10 \rightarrow 11 \rightarrow 12 \rightarrow 16 \rightarrow 20 \rightarrow 23 \rightarrow 22 \rightarrow 24 \rightarrow 26 \rightarrow 34$ |
| 36 | 13 | $3 \rightarrow 9 \rightarrow 10 \rightarrow 15 \rightarrow 19 \rightarrow 25 \rightarrow 24 \rightarrow 26 \rightarrow 34$ |

Table 8
Four feasible route combinations for the entire schedule

| Route no. for $T1 \rightarrow M_1$ | Route no. for $T3 \rightarrow M_1$ | Route no. for $T5 \rightarrow M_3$ | Material-transfer route for $T5 \rightarrow M_3$ |
|---|---|---|---|
| 1 | 1 | 1 | $5 \rightarrow 12 \rightarrow 11 \rightarrow 10 \rightarrow 15 \rightarrow 19 \rightarrow 25 \rightarrow 24 \rightarrow 22 \rightarrow 23 \rightarrow 33 \rightarrow 36$ |
| | | 2 | $5 \rightarrow 12 \rightarrow 16 \rightarrow 20 \rightarrow 23 \rightarrow 33 \rightarrow 36$ |
| | | 3 | $5 \rightarrow 12 \rightarrow 11 \rightarrow 10 \rightarrow 15 \rightarrow 19 \rightarrow 25 \rightarrow 29 \rightarrow 36$ |
| | | 4 | $5 \rightarrow 12 \rightarrow 16 \rightarrow 20 \rightarrow 23 \rightarrow 22 \rightarrow 24 \rightarrow 25 \rightarrow 29 \rightarrow 36$ |

Table 9
Four feasible operating procedures to achieve the entire schedule

| Route no. for $T5 \rightarrow M_3$ | Execution time | Operation steps |
|---|---|---|
| 1 | $\theta_0$ | Open: V1, V12, V16, V20, V23, V26, V29, P1 |
| | $\theta_1$ | Open: V3, V8, V13, V17, V30, V32, V33, V36, P2 |
| | $\theta_2$ | Close: P1, V1 |
| | $\theta_3$ | Close: V20, V26 |
| | | Open: V5, V11, V10, V14, V18, V25, V23, V21, V24, V38, P3 |
| | $\theta_4$ | Close: P2, P3, V3, V5 |
| 2 | $\theta_0$ | Open: V1, V12, V16, V20, V23, V26, V29, P1 |
| | $\theta_1$ | Open: V3, V8, V13, V17, V30, V32, V33, V36, P2 |
| | $\theta_2$ | Close: P1, V1 |
| | $\theta_3$ | Open: V5, V15, V19, V24, V38, P4 |
| | $\theta_4$ | Close: P2, P4, V3, V5 |
| 3 | $\theta_0$ | Open: V1, V12, V16, V20, V23, V26, V29, P1 |
| | $\theta_1$ | Open: V3, V8, V13, V17, V30, V32, V33, V36, P2 |
| | $\theta_2$ | Close: P1, V1 |
| | $\theta_3$ | Open: V5, V11, V10, V14, V18, V28, V40, P3 |
| | $\theta_4$ | Close: P2, P3, V3, V5 |
| 4 | $\theta_0$ | Open: V1, V12, V16, V20, V23, V26, V29, P1 |
| | $\theta_1$ | Open: V3, V8, V13, V17, V30, V32, V33, V36, P2 |
| | $\theta_2$ | Close: P1, V1 |
| | $\theta_3$ | Close: V20, V26 |
| | | Open: V5, V15, V19, V21, V23, V25, V28, V40, P4 |
| | $\theta_4$ | Close: P2, P4, V3, V5 |

## 7. Conclusion

A systematic procedure is presented in this paper for automatic generation of detailed operating procedures to achieved multiple material-transfer tasks in a batch process according to any given Gantt Chart. The proposed algorithm can be implemented on the basis of the Petri net model of plant structure. By imposing additional control rules, all feasible material-transfer routes and the corresponding operation steps can be identified from the reachability trees of the resulting Petri net. This approach has been successfully applied to a number of moderately complex examples.

## References


Crooks, C. A., & Macchietto, S. (1992). A combined MILP and logic-based approach to the synthesis of operating procedures for batch plants. *Chemical Engineering Communications*, *114*, 117.

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

Ferrarini, L., & Piroddi, L. (2003). Modular design and implementation of a logic control system for a batch process. *Computers and Chemical Engineering*, *27*, 983.

Foulkes, N. R., Walton, M. J., Andow, P. K., & Galluzzo, M. (1988). Computer-aided synthesis of complex pump and valve operations. *Computers and Chemical Engineering*, *12* (9/10), 1035.

Fusillo, R. H., & Powers, G. J. (1987). A synthesis method for chemical plant operating procedures. *Computers and Chemical Engineering*, *11*, 369.

Hoshi, K., Nagasawa, K., Yamashita, Y., & Suzuki, M. (2002). Automatic generation of operating procedures for batch production plants by using graph representations. *Journal of Chemical Engineering of Japan*, *35* (4), 377.

Karassik, I. J., & McGuire, J. T. (1998). *Centrifugal pumps* (2nd ed., pp. 885–887). New York, USA: Chapman & Hall.

Kim, J., & Moon, I. (2000). Synthesis of safety operating procedure for multi-purpose batch processes using SMV. *Computers and Chemical Engineering*, *24*, 385.

Kondili, E., Pantelides, C. C., & Sargent, R. W. H. (1988). A general algorithm for scheduling batch operations. *Proceedings of the Process Systems Engineering '88*, *Sydney, Australia*.

Lakshmanan, R., & Stephanopoulos, G. (1988a). Synthesis of operating procedures for complete chemical plants – I. Hierarchical, structured modelling f or nonlinear planning. *Computers and Chemical Engineering*, *12* (9/10), 985.

Lakshmanan, R., & Stephanopoulos, G. (1988b). Synthesis of operating procedures for complete chemical plants – II. A nonlinear planning methodology. *Computers and Chemical Engineering*, *12* (9/10), 1003.

Lakshmanan, R., & Stephanopoulos, G. (1990). Synthesis of operating procedures for complete chemical plants – III. Planning in the presence of qualitative, mixing constraints. *Computers and Chemical Engineering*, *14* (3), 301.

Murata, T. (1989). Petri nets: properties, analysis and applications. *Proceedings of the IEEE*, *77* (4), 541.

O'Shima, E. (1978). Safety supervision of valve operation. *Journal of Chemical Engineering of Japan*, *11*, 390.

Peterson, J. L. (1981). *Petri net theory and the modeling of systems*. Englewood Cliffs, NJ, USA: Prentice-Hall.

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

Tittus, M., & Lennartson, B. (1999). Hierarchical supervisory control for batch processes. *IEEE Transactions on Control Systems Technology*, *7* (5), 542.

Uthgenannt, J. A. (1996). Path and equipment allocation for multiple, concurrent processes on networked process plant units. *Computers and Chemical Engineering*, *20* (9), 1081.

Viswanathan, S., Johnsson, C., Venkatasubramanian, V., & Ärzen, K. E. (1998a). Automating operating procedure synthesis for batch processes: Part I. Knowledge representation and planning framework. *Computers and Chemical Engineering*, *22* (11), 1673.

Viswanathan, S., Johnsson, C., Venkatasubramanian, V., & Ärzen, K. E. (1998b). Automating operating procedure synthesis for batch processes: Part II. Implementation and application. *Computers and Chemical Engineering*, *22* (11), 1687.

Wang, Y.-F., & Chang, C.-T. (2004). A Petri-net based deductive reasoning strategy for fault identification in batch processes. *Industrial and Engineering Chemistry Research*, *43* (11), 2704.

Wang, Y.-F., Wu, J. Y., & Chang, C.-T. (2002). Automatic hazard analysis of batch operations with Petri nets. *Reliability Engineering and System Safety*, *76* (1), 91.