Pergamon

0098-1354(95)00256-1

Computers chem. Engng Vol. 21, No. 2, pp. 223–239, 1997
Copyright © 1996 Elsevier Science Ltd
Printed in Great Britain. All rights reserved
0098-1354/97 $15.00+0.00

THE LOOP IDENTIFICATION AND CLASSIFICATION ALGORITHMS FOR DIGRAPH-BASED SAFETY ANALYSIS

Chuei-Tin Chang, Ho-Chuan Hwang, Kuo-Shu Hwang and Ding-Shan Hsu

Department of Chemical Engineering, National Cheng Kung University, Tainan, Taiwan 70101, P.R. China

(Received 13 January 1994; revised 18 August 1994)

Abstract—The task of identifying and classifying feedforward and feedback loops is an indispensable step in a digraph-based safety analysis. To address the problems inherited from the traditional approach, i.e. manual inspection, it is highly desirable to develop a generic software for automating the loop-searching process. Computer algorithms that facilitate its realization are presented in this paper. In particular, a simple search and delete strategy is proposed to identify every loop path corresponding to a given ending node. Techniques for assembling these paths to produce a comprehensive list of loops are also described in detail. Finally, several practical examples are provided at the end of this paper. The results show that the proposed algorithms are indeed quite reliable and efficient for the purpose intended. Copyright © 1996 Elsevier Science Ltd

INTRODUCTION

The digraph-based analysis has been adopted extensively in many chemical engineering applications, e.g., data reconciliation and coaptation (Mah *et al.*, 1976), process flowsheeting computation (Steward, 1962; Lee *et al.*, 1966; Christensen, 1970; Mah, 1990), fault detection and diagnosis (Iri *et al.*, 1979; Shiozaki *et al.*, 1985; Kramer and Palowitch, 1987; Oyeleye and Kramer, 1988) and fault tree synthesis (Lapp and Powers, 1977; Allen and Rao, 1980; Andrews and Brennan, 1990). The present study is mainly concerned with the need to identify and also *classify* loops in the safety-related applications. In these applications, the digraph is used as a model for representing the qualitative relationship among process variables, equipment malfunctions and human errors. The system digraph is obtained by connecting component digraphs corresponding to all units in the system. Owing to the interaction between units, complex loops are often formed in these system models.

According to Lapp and Powers (1977), two types of "loops" are important for implementing the digraph-based techniques, i.e. feedforward loops (FFLs—two or more paths from one node to another different node in a digraph) and feedback loops (FBLs—a path through the nodes in a digraph which starts and terminates at the same node). These loops can also be classified according to their functions, i.e. control loops and process loops. The former can be identified by searching for the nodes that represent the variables in control systems. The process loops, however, are caused by the complex relations among system variables. The corresponding loop variables can be operating parameters (e.g., pressure and temperature) in the process and also signals in the control systems (e.g., air pressure to a control valve). In order to ensure the validity of the digraph model, it is obvious that the effects of every external disturbance to these control and process FFLs and FBLs must be evaluated correctly. As a result, the task of identifying and classifying all embedded loops becomes an indispensable step of any credible digraph-based safety analysis.

In general, the FFLs and FBLs can be easily identified by inspection if the digraph is associated with a simple stand-alone unit. However, in practice, a large number of tangled loops may be caused by the interaction among connected units even in a moderately busy P&ID (Chang and Hwang, 1992). Manual enumeration of all of them in a complex digraph is often laborious and error-prone. Thus, there is a definite need to make use of computer algorithms for automating the loop-searching process.

There are a number of techniques available in the literature which can be implemented for similar purposes. For example, Tarjan (1972) developed methods for identifying strongly-connected components in directed graphs and biconnected components in undirected graphs. These methods are adequate for establishing the precedence order in flowsheeting computation. However, in most of the safety-related applications in chemical engineering, the FFLs and FBLs must be handled differently, e.g., Lapp and Powers (1977) and Oyeleye and Kramer (1988). As a result, every loop in a digraph not only needs to be identified but also classified according to its structural characteristics. If the conventional approach are taken to search for the strongly-connected components in a digraph, only the FBLs can be found. This is not useful for our applications since, in many realistic digraph models with *coupled* feedforward

* To whom all correspondence should be addressed.

and feedback loops, some of the FFLs are not identifiable from digraphs obtained by removing the FBLs from the original systems. On the other hand, if the biconnected components are located by treating the directed graph as an undirected one, some of the components may not even satisfy the criteria of loops defined previously. Since none of the available approaches are suitable for the above tasks, algorithms have been developed in our study to meet this demand.

THE BASIC SEARCH STRATEGY

From the definition of FBL, it is clear that every node on the loop must possess at least one input and one output. From the definition of FFL, the nodes on the paths of a FFL (except the starting and ending nodes) also satisfy this same requirement. Notice that the digraph configuration surrounding the starting or ending node of a FFL is different, i.e. the number of outputs from the former and that of the inputs to the latter must exceed one. Based upon these observations, a simple search strategy has been developed in this study.

For illustration purpose, let us consider the digraph presented in Fig. 1a. Notice that virtually every node in this system possesses at least one input and one output.

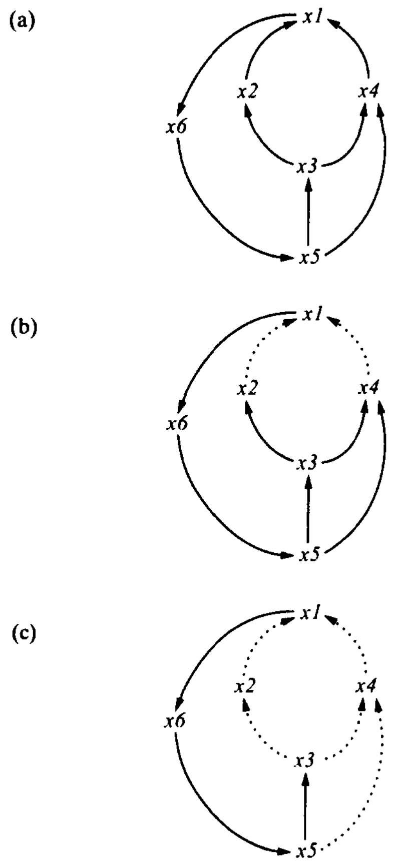


Fig. 1. A fictitious example for illustrating the basic search strategy: i (a), ii (b) and iii (c).

Thus, each node may be on a FBL and/or a path of FFL. In fact, this situation can be found in almost every application, i.e. a large number of nodes satisfy the above mentioned condition. To reduce the degree of arbitrariness in initiating the search process, the nodes with two or more inputs were selected in the proposed algorithm. From Fig. 1a, one can see that each of the two nodes $x1$ and $x4$ has two inputs and, thus, may be chosen as our candidates. Starting from these two nodes respectively, a search and delete procedure can be carried out to obtain lists of loop paths on all FFLs and possibly some FBLs. Specifically, the result produced by a one-step backward (upstream) search from $x1$ can be stored in a list **T**, i.e.

$$\mathbf{T} = \begin{bmatrix} x1 & x2 \\ x1 & x4 \end{bmatrix} \tag{1}$$

Notice that the two rows in this list represent two distinct search paths and they are associated with the two local inputs $x2$ and $x4$, respectively. Also note that, after adding the last entry to each row of **T**, the searched edge must be immediately deleted from the digraph to avoid repetition. Thus, the digraph in Fig. 1a now becomes the one presented in Fig. 1b after the completion of list **T** in equation (1). Here, the deleted edges are denoted by dotted lines. The same backward search and delete techniques are then applied, one by one, on nodes corresponding to the last row members in **T**, i.e. $x2$ and $x4$. The result obtained after this second step is

$$\mathbf{T} = \begin{bmatrix} x1 & x2 & x3 \\ x1 & x4 & x3 \\ x1 & x4 & x5 \end{bmatrix} \tag{2}$$

Notice that there are two inputs to $x4$, i.e. $x3$ and $x5$. Since it is necessary to delineate the fact that two paths are merged at $x4$, the members $x1$ and $x4$ must be repeated in a second row [see row 2 and row 3 in equation (2)]. The corresponding digraph is presented in Fig. 1c.

The same procedure can be applied again to obtain the following result:

$$\mathbf{T} = \begin{bmatrix} x1 & x2 & x3 & x5 \\ x1 & x4 & x3 & \\ x1 & x4 & x5 & x6 \end{bmatrix} \tag{3}$$

Notice that the fourth entry of the second row is void. This is due to the special search and delete strategy adopted in this study. Although the input to $x3$ is $x5$, the edge $x5 \rightarrow x3$ should be removed immediately after the completion of row 1. As a result, $x3$ becomes a node without inputs and the search process corresponding to the second row must be terminated. The final result after one more step is:

$$\mathbf{T} = \begin{bmatrix} x1\ x2\ x3\ x5 \\ x1\ x4\ x3 \\ x1\ x4\ x5\ x6\ x1 \end{bmatrix} \tag{4}$$

From the above discussion, one can observe that the last row members in list **T** may be the starting nodes of FFLs (e.g., $x3$ and $x5$) or nodes on FBLs (e.g., $x1$). However, there are other possibilities. Specifically, the search process associated with each row is terminated under either of the following two conditions:

1. All edges between the node corresponding to the last row member and its inputs have already been removed in a prior search and delete step; or
2. The last row member is associated with a node without inputs in the original digraph.

To clarify the first condition, let us consider the digraph configuration presented in Fig. 2. After applying the search and delete procedure to $x4$, $x5$ and $x6$, the result should be:

$$\mathbf{T} = \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & x4 & x \\ \cdots & x5 & x \\ \cdots & x6 & x \end{bmatrix} \tag{5}$$

The same procedure is then repeated. The list **T** becomes:

$$\mathbf{T} = \begin{bmatrix} \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & x4 & x & x1 & \cdots \\ \cdots & x4 & x & x2 & \cdots \\ \cdots & x4 & x & x3 & \cdots \\ \cdots & x5 & x & & \\ \cdots & x6 & x & & \end{bmatrix} \tag{6}$$

There are only two possibilities that can be characterized by node x in such a digraph configuration. The first type (TYPE I) is concerned with a starting node of a FFL having at least one input. This is because of the fact that there are at least two rows in **T** with the same first member and also the member x. The second possibility is concerned with a FBL node having at least one off-FBL output (TYPE II). In such a case, this node (x) must either appear twice in the same row of **T** or appear in two different rows.

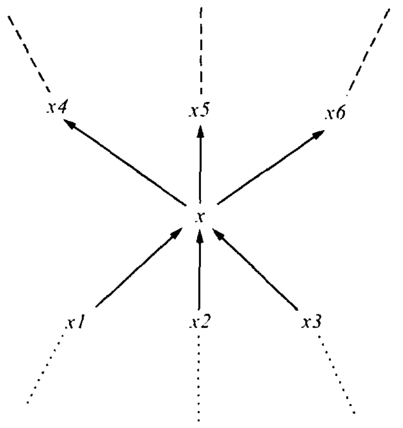


Fig. 2. The digraph configuration associated with TYPE I and TYPE II nodes.

The second condition mentioned above can be attributed to three other types of digraph configuration. The first possibility is that the node is indeed a starting node of FFL without inputs (TYPE III). Secondly, the node may be one with only a single output (TYPE IV), i.e. a pendent node. Note that these type of nodes cannot be the starting nodes of FFLs or nodes on FBLs. Thus, to avoid including TYPE IV nodes in list **T** during the search process, it is necessary to remove them in advance. Finally, it is also possible that the last row member is associated with the starting node of several (more than one) paths and, also, only one of them ends at the node corresponding to the first row member in **T** (TYPE V). Thus, they cannot be the loop nodes of FFLs or FBLs either. Again, it is desirable to exclude them before the search.

Therefore, in conclusion, if the pendent nodes and TYPE V nodes (and their attached edges) are removed, as many as possible in advance, the basic search strategy can be considered as an efficient way for identifying all loop paths corresponding to a given ending node.

DIGRAPH REPRESENTATIONS

Although a digraph model can be easily understood and analyzed by human beings, it is somewhat awkward for computer storage and manipulation. Thus, there is a need to represent its embedded informations with a more suitable format. One of the most popular tools is the *incidence matrix*. In such a matrix, the rows and columns represent the nodes and edges of a digraph respectively. If the jth edge enters the ith node, the value of the corresponding entry (i,j) in the matrix equals "+1." Similarly, if the jth edge leaves the ith node, a value of "−1" should be assigned to this entry. Let us consider example 1.

Example 1. A simple heat exchange system (Fig. 3a) is used in this example. Here, TT represents a temperature sensor-transmitter; TIC represents a temperature controller; EX represents a heat exchanger; the label "CV A/O" denotes an air-to-open control valve. The corresponding digraph model can be found in Fig. 3b. The physical meanings of the symbols used in the digraph can be found in the Nomenclature section at the end of this paper. On the basis of Fig. 3b, the incidence matrix can then be constructed accordingly (Table 1a).

Although the incidence matrix is quite suitable for computer manipulation, the main drawback of such an approach is that the demand for computer storage space is often unnecessarily large. This is due to the fact that the matrix contains a large number of void entries. In realistic applications, this may become a problem that cannot be overcome. There are actually many other alternative representations available in the literature. However, since a backward-search strategy is adopted in this study, a modified version of the successor list (Doe, 1974), i.e. the *predecessor list*, has been used in our computer program to capture the implied informations in a digraph model. In such a list, the first member in each row is the output node (rather than the input node in a successor list) of the other members in the same row.

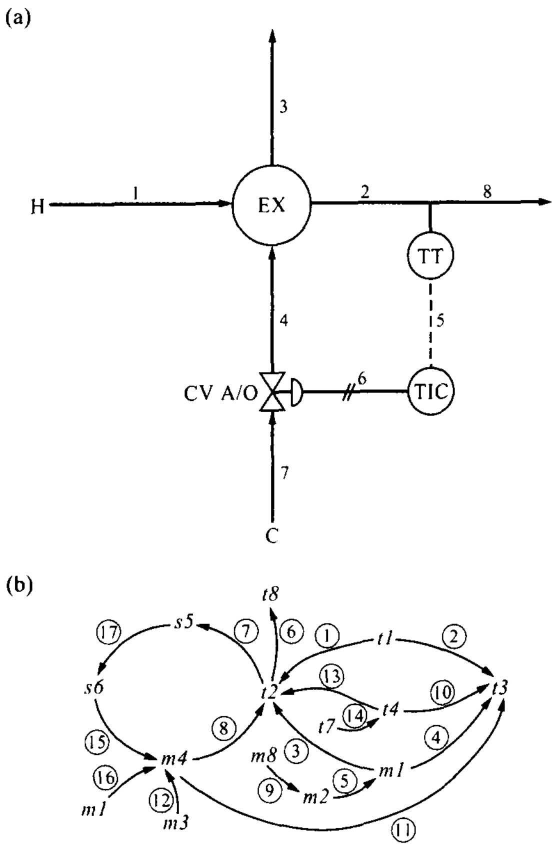


Fig. 3. (a) The process flow diagram of Example 1. (b) The system digraph of Example 1.

The predecessor list associated with Fig. 3b is presented in Table Ib.

It is plain that a predecessor list is essentially equivalent to an incidence matrix for the same digraph. However, since the incidence matrix is somewhat easier for visual inspection, it is still adopted as a tool only for *illustrating* our algorithm throughout this paper. It should be understood that equivalent yet more efficient operations are actually performed on the predecessor lists in our code.

IDENTIFICATION AND CLASSIFICATION ALGORITHMS

Based on the search strategy described previously, algorithms have been developed for identifying and classifying all FFLs and FBLs in a given digraph. The proposed procedure is outlined below.

1. Translate the system digraph into a format suitable for computer manipulation. The formats used in the computer code and in this paper have already been described in the previous section. Let the resulting incidence matrix be **A**.
2. Remove all pendent nodes and their attached edges from the system digraph.The degree d of a node is defined as the number of edges attached to it. In other words, the value of d equals to the number of nonzero elements in the corresponding row of the incidence matrix, e.g., the degree of node $t1$ in Example 1 is 2. A pendent node is a node with degree one. The removal of such nodes can be carried out with the algorithm described below.
   (a) Select a row from the incidence matrix which has only one nonzero element. Then, remove the row and column in which the element is located.
   (b) Repeat the above steps until every row in the resulting matrix contains two or more nonzero elements. Let this matrix be **B**.

   As an example, the incidence matrix for the system in Example 1 can be reduced to form the matrix **B** presented in Table IIa.
3. Remove part of TYPE V nodes and their attached edges from the digraph obtained in Step 2.

Table Ia. The incidence matrix **A** for Example 1

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
*t*1	−1	−1															
*m*1			−1	−1	+1												
*t*2	+1		+1			−1	−1	+1					+1				
*m*2					−1				+1								
*t*3		+1		+1						+1	+1						
*m*3												−1					
*t*4										−1			−1	+1			
*m*4								−1			−1	+1			+1	+1	
*s*5							+1										−1
*s*6															−1		+1
*t*7														−1			
*m*7																−1	
*t*8						+1											
*m*8									−1								

Table Ib. The predecessor list for Example 1

$m1$	$m2$			
$t2$	$t1$	$m1$	$m4$	$t4$
$m2$	$m8$			
$t3$	$t1$	$m1$	$t4$	$m4$
$t4$	$t7$			
$m4$	$m3$	$s6$	$m7$	
$s5$	$t2$			
$s6$	$s5$			
$t8$	$t2$			

This task can be accomplished by making use of a standard algorithm for determining the components in a graph and the nodes included in each component, i.e. Algorithm 1 in Doe (1974). For The sake of conciseness, this algorithm is not repeated here. Following are the proposed steps.

(a) Identify and remove the candidate nodes and their attached edges. Search for a row in **B** which does not contain elements with value "+1". Remove this candidate row and all the columns corresponding to its nonzero elements, i.e. the ones with value "−1". Save the labels of the candidate node and its outputs in a list **L**. Repeat this process until all such rows are eliminated. Let the resulting matrix be **C**.

(b) Apply Algorithm 1 (Doe, 1974) to determine the components and the nodes included in each component.

(c) If two or more outputs of a candidate nodes belong to the same component, then it may not be a TYPE V node. Add the corresponding row and columns back to **C**. Repeat this process until all candidates in **L** are checked against the above criterion.

(d) Repeat Step 2.

(e) Let $\mathbf{C}_1 = \mathbf{C}_2 = \mathbf{C}$.

Since the original digraph in Example 1 does not contain TYPE V nodes, the matrices obtained after this step are the same as **B**.

4. Determine the initiating nodes of the search process. Identify all the rows in matrix **C** that contain two or more "+1". Store the corresponding node variable n_m in a vector **n** and $\mathbf{n} = [n_1, n_2,..., n_M]^T$. For example, the initiating nodes in Table IIa can be stored in:

$$\mathbf{n} = [t2\ t3]^T \tag{7}$$

5. Trace and record all the loop paths from the initiating nodes in **n**.

Here, path lists $\mathbf{T}_m$ ($m = 1, 2,..., M$) are used to record the paths which can be traced from node n_m using the *basic search strategy*. The member $t_{i,j}$ in the list represents the name of the jth node in the ith path which terminates at n_m. The detailed algorithm is presented below.

(a) Let $i = 1$ and $m = 1$.

(b) Let $j = 1$ and $t_{i,j} = n_m$.

(c) Identify the row in $\mathbf{C}_1$ which is associated with $t_{i,j}$. Let this row be R_0. Look for an element "−1" in a column C_i which contains an element "+1" in R_o. Let the row which contains the above element "−1" be R_i. The variable corresponding to R_i represents the input node of $t_{i,j}$. Store it in $t_{i,j+l}$ and eliminate column C_i from $\mathbf{C}_1$. If $\mathbf{C}_2$ contains the same column, remove it also.

(d) Let $i = i + 1$ and $t_{i,k} = t_{i-1,k}$ ($k = 1,2,..., j$). Repeat Steps (c) and (d) until row R_o does not have any element with the value "+1".

(e) Identify and remove all the rows in $\mathbf{C}_1$ and $\mathbf{C}_2$ which contain only zeros.

(f) Examine the last member of every row in list $\mathbf{T}_m$. If the corresponding row in $\mathbf{C}_1$ contains elements that equal "+1", then interchange this row and the ith row in $\mathbf{T}_m$. Let $j = j + 1$ and repeat Steps (c) to (f). This process is continued until none of the rows in $\mathbf{C}_1$, which are associated with the last members of the paths in $\mathbf{T}_m$, contain the element "+1". Thus, after this step, each row in list $\mathbf{T}_m$ represents a path which terminates at node n_m.

(g) If $m < M$, then let $\mathbf{C}_1 = \mathbf{C}$, $m = m + 1$, $i = 1$ and repeat Steps (b) to (g). If $m = M$, then go to Step 6.

As an example, let us consider the first member of **n** in Example 1, i.e. $t2$. In this case, $t_{1,1} = t2$ for $m = 1$. When $j = 1$, the path list $\mathbf{T}_1$ becomes

$$\mathbf{T}_1 = \begin{bmatrix} t2 & t1 \\ t2 & m1 \\ t2 & t4 \\ t2 & m4 \end{bmatrix} \tag{8}$$

The corresponding matrix $\mathbf{C}_1$ is presented in Table IIb. When $j = 2$, the path list is

$$\mathbf{T}_1 = \begin{bmatrix} t2 & t1 \\ t2 & m1 \\ t2 & t4 \\ t2 & m4 \end{bmatrix} s6 \tag{9}$$

Table IIa. The reduced incidence matrix **B** for Example 1

	1	2	3	4	7	8	10	11	13	15	17
$t1$	−1	−1									
$m1$			−1	−1							
$t2$	+1		+1		−1	+1			+1		
$t3$		+1		+1			+1	+1			
$t4$							−1		−1		
$m4$						−1		−1		+1	
$s5$					+1						−1
$s6$									−1		+1

The corresponding matrix is presented in Table IIc. The final path list can be obtained when $j = 4$ for this example, i.e.

$$\mathbf{T}_1 = \begin{bmatrix} t2 & t1 & & & \\ t2 & m1 & & & \\ t2 & t4 & & & \\ t2 & m4 & s6 & s5 & t2 \end{bmatrix} \tag{10}$$

The second path list can be obtained with the same algorithm:

$$\mathbf{T}_2 = \begin{bmatrix} t3 & t1 & & & & \\ t3 & m1 & & & & \\ t3 & t4 & & & & \\ t3 & m4 & s6 & s5 & t2 & m4 \\ t3 & m4 & s6 & s5 & t2 & t1 \\ t3 & m4 & s6 & s5 & t2 & m1 \\ t3 & m4 & s6 & s5 & t2 & t4 \end{bmatrix} \tag{11}$$

6. If the matrix $\mathbf{C}_2$ is not empty, then trace a path

Table IIb. The matrix $\mathbf{C}_1$ obtained in Step 5 ($j = 1$) for Example 1

	2	4	7	10	11	15	17
*t*1	−1						
*m*1		−1					
*t*2			−1				
*t*3	+1	+1		+1	+1		
*t*4				−1			
*m*4					−1	+1	
*s*5			+1				−1
*s*6						−1	+1

Table IIc. The matrix $\mathbf{C}_1$ obtained in Step 5 ($j = 2$) for Example 1

	2	4	7	10	11	17
*t*1	−1					
*m*1		−1				
*t*2			−1			
*t*3	+1	+1		+1	+1	
*t*4				−1		
*m*4					−1	
*s*5			+1			−1
*s*6						+1

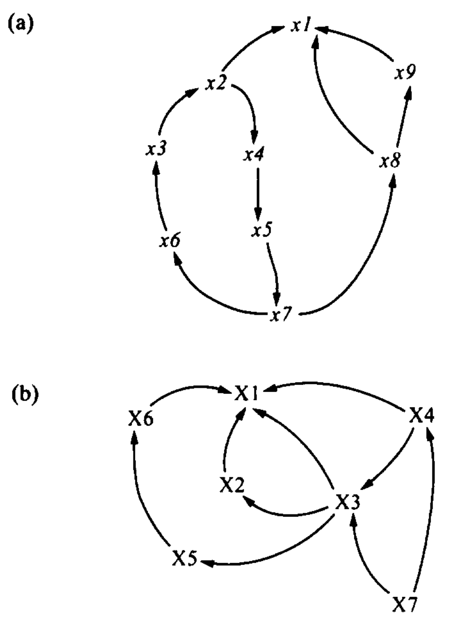


Fig. 4. (a) A fictitious example for illustrating the loop construction algorithm. (b) Another fictitious example for illustrating the loop construction algorithm.

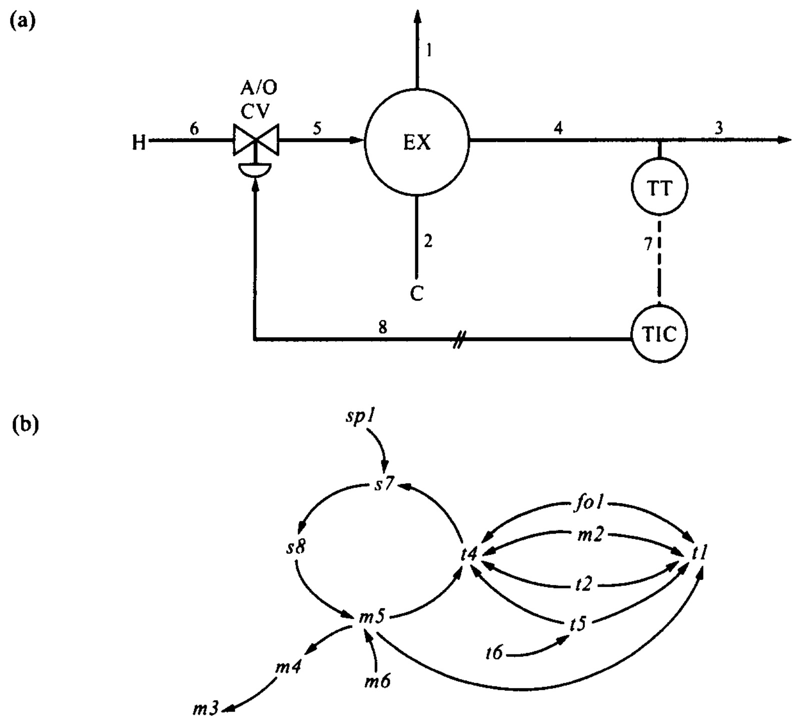


Fig. 5. (a) The process flow diagram of Example 2. (b) The system digraph of Example 2.

Table III. The FFLs and FBLs identified in Example 2

```
[FFLP LIST]

1.
t1 t5
t1 m5 s8 s7 t4 t5

2.
t1 t2
t1 m5 s8 s7 t4 t2

3.
t1 m2
t1 m5 s8 s7 t4 m2

4.
t1 fo1
t1 m5 s8 s7 t4 fo1

[END FFLP LIST]

[FBLP LIST]
t4 m5 s8 s7 t4
[END FBLP LIST]
```

backward from any node corresponding to a row in C_2 and record the result in list T_{M+1}. Repeat this process until all the edges in C_2 have been searched.

7. Assemble the paths in T_m ($m = 1, 2,..., M$) to form FFLs and FBLs.

Other than very few exceptions (i.e. the TYPE V nodes which have not been removed in Step 3), the last row members of T_m should be the starting nodes of FFLs or nodes on FBLs. Thus, a row in T_m may contain a path in a FFL which ends at the node corresponding to the first row member, or a portion of a FBL. The algorithm used in this study for assembling them to form various feedforward and feedback loops is described below.

(a) Let $i = 1$. Select one of the last row members.
(b) Store the rows corresponding to the chosen member in a list **LIST** and the rest in list **OTHER**. Let **OTHER0** = **OTHER**.
(c) Look for the feedback loops embedded in the rows added to **LIST** in the previous step. If a member appears twice in the same row, then a FBL is identified. Store the corresponding path in list **FBLP** and remove all but one of the repeated members in this path from **LIST**.
(d) Check the rows in **LIST**. Remove the rows with only one member. Also, if there are two or more identical rows, remove all but one of them from **LIST**.
(e) Select a row in **OTHER** and compare its last member with all members except the last row members in **LIST**. If an identical member is identified in one of the rows in **LIST**, add all the members after it to the end of the selected row in **OTHER** and, then, move this extended row into **LIST**. Repeat the present step for all the rows in **OTHER**.
(f) Let **OTHER** = **OTHER0**.
(g) Repeat Steps (c)–(f) until the number of rows in **LIST** does not increase after successive execution of these steps.

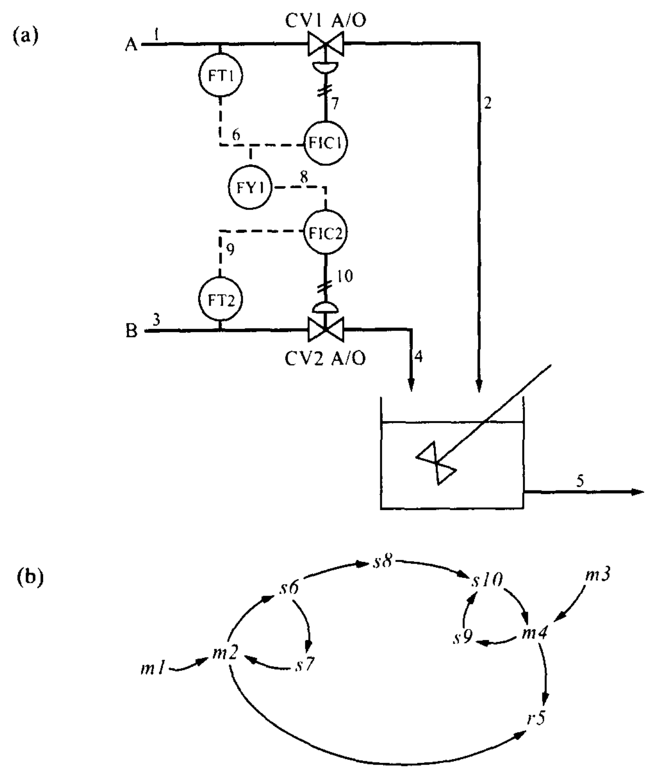


Fig. 6. (a) The process flow diagram of Example 3. (b) The system digraph of Example 3.

Table IV. The FFLs and FBLs identified in Example 3

[FFLP LIST]

1.
r5 m4 s10 s8 s6
r5 m2 s7 s6

2.
r5 m2
r5 m4 s10 s8 s6 m2

[END FFLP LIST]

[FBLP LIST]
s10 s9 m4 s10
s6 m2 s7 s6
[END FBLP LIST]

(h) Check the intermediate members (i.e. member which is not the first or last) in each row of **OTHER**. If any of them is identical to the last row member in **LIST**, then delete all members after it and copy the remaining row into **LIST**.

(i) Determine whether the paths in **LIST** form a FFL which starts at the node corresponding to the last row member. This decision can be made by searching for an intermediate member which is common to all rows. If none can be found, then store them in a list $\mathbf{FFLP}_{mi}$. Otherwise, these paths should be excluded.

(j) Let $i = i + 1$. Select another last row member and repeat Steps (b)–(j) until all such members are exhausted.

Applying the above algorithm to Example 1, one can construct three FFLs, i.e.

$$\mathbf{FFLP}_{21} = \begin{bmatrix} t3\ t1 \\ t3\ m4\ s6\ s5\ t2\ t1 \end{bmatrix} \tag{12}$$

$$\mathbf{FFLP}_{22} = \begin{bmatrix} t3\ m1 \\ t3\ m4\ s6\ s5\ t2\ m1 \end{bmatrix} \tag{13}$$

$$\mathbf{FFLP}_{23} = \begin{bmatrix} t3\ t4 \\ t3\ m4\ s6\ s5\ t2\ t4 \end{bmatrix} \tag{14}$$

with the paths in $\mathbf{T}_2$, but none can be obtained with the paths in $\mathbf{T}_1$.

Since the digraph configuration in Example 1 is really quite simple, three additional examples are also presented here to demonstrate the capability of the proposed algorithm. Let us first consider the fictitious system digraph presented in Fig. 4a. The path list corresponding to the node x_1 is

$$\mathbf{T}_1 = \begin{bmatrix} x1\ x9\ x8 \\ x1\ x2\ x3\ x6\ x7 \\ x1\ x8\ x7\ x5\ x4\ x2 \end{bmatrix} \tag{15}$$

After completing Step (b), the lists corresponding to the last row member $x7$ are:

$$\mathbf{LIST} = [x1\ x2\ x3\ x6\ x7] \tag{16}$$

$$\mathbf{OTHER} = \begin{bmatrix} x1\ x9\ x8 \\ x1\ x8\ x7\ x5\ x4\ x2 \end{bmatrix} \tag{17}$$

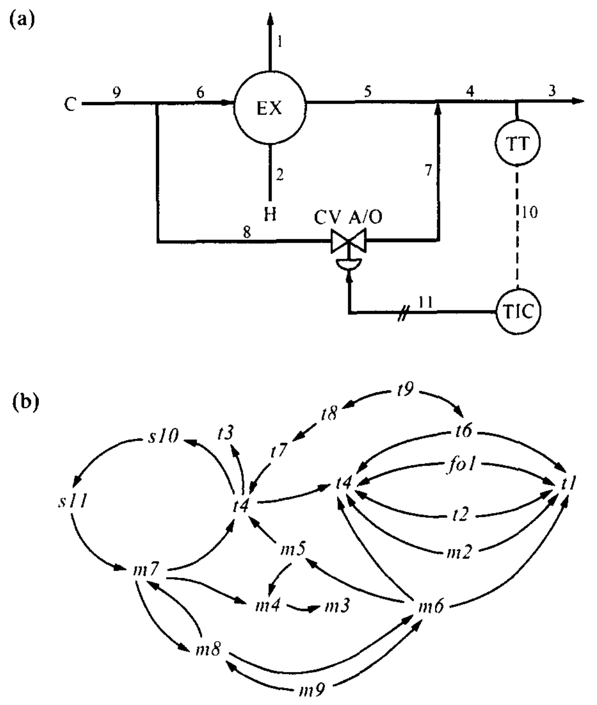


Fig. 7. (a) The process flow diagram of Example 4. (b) The system digraph of Example 4.

Table V. The FFLs and FBLs identified in Example 4

[FFLP LIST]

1.
t1 t2
t1 m6 m8 m7 s11 s10 t4 t5 t2

2.
t1 m2
t1 m6 m8 m7 s11 s10 t4 t5 m2

3.
t1 fo1
t1 m6 m8 m7 s11 s10 t4 t5 fo1

4.
t1 t6 t9
t1 m6 m8 m7 s11 s10 t4 t7 t8 t9
t1 m6 m8 m7 s11 s10 t4 t5 t6 t9

5.
t1 m6 m8 m7 s11 s10 t4 t5 t6
t1 t6

6.
t4 t5 m6
t4 m5 m6

7.
t4 t7 t8 t9
t4 t5 t6 t9

8.
t4 m7
t4 m5 m6 m8 m7
t4 t5 m6 m8 m7

9.
t4 m7 m8 m9
t4 m5 m6 m9
t4 t5 m6 m9
t4 m5 m6 m8 m9
t4 t5 m6 m8 m9

10.
t4 m5 m6 m8
t4 t5 m6 m8
t4 m7 m8

11.
m7 m8 m9
m7 s11 s10 t4 m5 m6 m9
m7 s11 s10 t4 t5 m6 m9
m7 s11 s10 t4 m5 m6 m8 m9
m7 s11 s10 t4 t5 m6 m8 m9

12.
m7 s11 s10 t4 m5 m6 m8
m7 s11 s10 t4 t5 m6 m8
m7 m8

13.
m8 m9
m8 m7 s11 s10 t4 m5 m6 m9
m8 m7 s11 s10 t4 t5 m6 m9

14.
t5 t6 t9
t5 m6 m8 m7 s11 s10 t4 t7 t8 t9

15.
m4 m7
m4 m5 m6 m8 m7

Table V. Continued

16.
m4 m7 m8 m9
m4 m5 m6 m9
m4 m5 m6 m8 m9
m4 m7 s11 s10 t4 m5 m6 m9
m4 m7 s11 s10 t4 t5 m6 m9
m4 m7 s11 s10 t4 m5 m6 m8 m9
m4 m7 s11 s10 t4 t5 m6 m8 m9

17.
m4 m5 m6 m8
m4 m7 s11 s10 t4 m5 m6 m8
m4 m7 s11 s10 t4 t5 m6 m8
m4 m7 m8

18.
m4 m7 s11 s10 t4 m5
m4 m5

19.
m4 m7 s11 s10 t4 t5 m6
m4 m5 m6

20.
m6 m9
m6 m8 m9

[END FFLP LIST]

[FBLP LIST]
m8 m7 m8
m7 s11 s10 t4 m7
m6 m8 m7 s11 s10 t4 m5 m6
m6 m8 m7 s11 s10 t4 t5 m6
[END FBLP LIST]

Applying Steps (c)–(f) once, one can obtain

$$\mathbf{LIST} = \begin{bmatrix} x1\ x2\ x3\ x6\ x7 \\ x1\ x8\ x7 \end{bmatrix} \tag{18}$$

After applying Steps (c)–(f) twice, the following results can be obtained:

$$\mathbf{LIST} = \begin{bmatrix} x1\ x2\ x3\ x6\ x7 \\ x1\ x8\ x7 \\ x1\ x9\ x8\ x7 \end{bmatrix} \tag{19}$$

The corresponding list **FBLP** is:

$$\mathbf{FBLP} = [x7\ x5\ x4\ x2\ x3\ x6\ x7] \tag{20}$$

At this point, the number of rows in **LIST** cannot be increased with Steps (c)–(f). Further, since $x7$ is not one of the intermediate members in **OTHER**, no more rows can be added in **LIST** after Step (h). Notice that Step (h) is still necessary in many other instances, e.g., when the present algorithm is applied on the last row member $x8$.

Finally, after Step (i), one should be certain that

$$\mathbf{FFLP}_{12} = \begin{bmatrix} x1\ x2\ x3\ x6\ x7 \\ x1\ x8\ x7 \\ x1\ x9\ x8\ x7 \end{bmatrix} \tag{21}$$

is a FFL in the digraph presented in Fig. 4a. The same steps can be performed on the other two last row members $x8$ and $x2$. Two additional FFLs can then be constructed:

$$\mathbf{FFLP}_{11} = \begin{bmatrix} x1\ x9\ x8 \\ x1\ x8 \end{bmatrix} \tag{22}$$

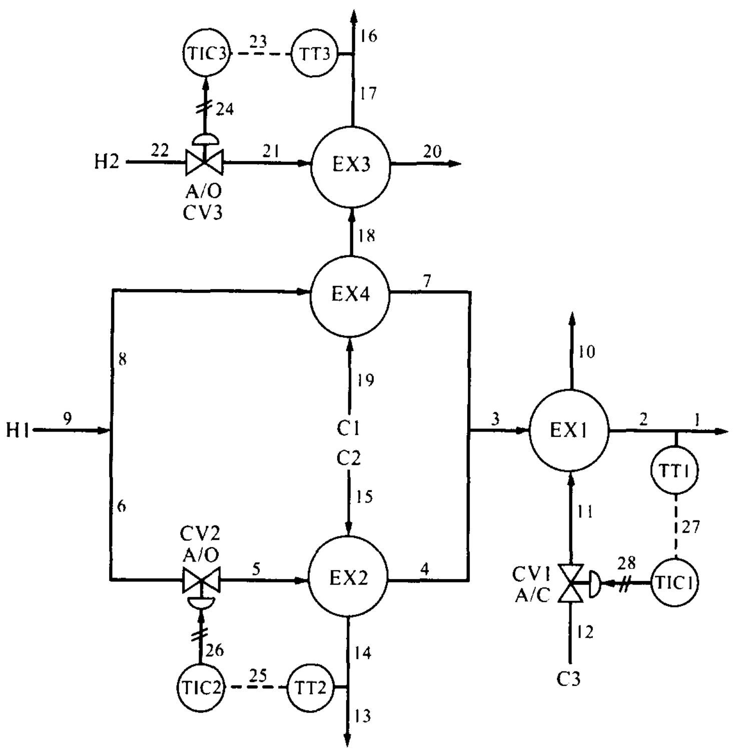


Fig. 8. (a)The process flow diagram of Example 5.

$$\mathbf{FFLP}_{13} = \begin{bmatrix} x1\ x8\ x7\ x5\ x4\ x2 \\ x1\ x9\ x8\ x7\ x5\ x4\ x2 \\ x1\ x2 \end{bmatrix} \tag{23}$$

Next, let us use the paths obtained in equation (4) as our second example. Two feedforward loops can be found with the present algorithm, i.e.

$$\mathbf{FFLP}_{11} = \begin{bmatrix} x1\ x4\ x3 \\ x1\ x2\ x3 \end{bmatrix} \tag{24}$$

$$\mathbf{FFLP}_{12} = \begin{bmatrix} x1\ x2\ x3\ x5 \\ x1\ x4\ x3\ x5 \\ x1\ x4\ x5 \end{bmatrix} \tag{25}$$

The corresponding list for storing feedback loops is:

$$\mathbf{FBLP} = \begin{bmatrix} x5 & x6 & x1 & x2 & x3 & x5 \\ x5 & x6 & x1 & x4 & x3 & x5 \\ x5 & x6 & x1 & x4 & x5 & \\ x4 & x5 & x6 & x1 & x4 & \end{bmatrix} \tag{26}$$

Note that the last two rows are actually the same FBL path. Notice also that, although the first three rows in **FBLP** do represent distinct loop paths, they form one FBL only. This is due to the fact that the two FFLs in equations (24) and (25) are embedded in the same FBL. Thus, it is better to eliminate the intermediate nodes of the FFLs from the list **FBLP** since they can be recovered from the lists $\mathbf{FFLP}_{mi}$. This task is accomplished later in Step 8.

Finally, the last example is designed to demonstrate the need for recovering **OTHER** in Step (f). Let us consider the digraph presented in Fig. 4b. The path list corresponding the node $x1$ is:

$$\mathbf{T}_1 = \begin{bmatrix} x1 & x2 & x3 & \\ x1 & x3 & x4 & \\ x1 & x3 & x7 & \\ x1 & x4 & x7 & \\ x1 & x6 & x5 & x3 \end{bmatrix} \tag{27}$$

The FFL corresponding to the last row member $x7$ can be obtained using the proposed algorithm:

$$\mathbf{FFLP}_{11} = \begin{bmatrix} x1 & x3 & x7 & & & \\ x1 & x4 & x7 & & & \\ x1 & x3 & x4 & x7 & & \\ x1 & x2 & x3 & x7 & & \\ x1 & x6 & x5 & x3 & x7 & \\ x1 & x2 & x3 & x4 & x7 & \\ x1 & x6 & x5 & x3 & x4 & x7 \end{bmatrix} \tag{28}$$

It should be emphasized that the last two rows cannot be obtained without Step (f).

8. Identify additional FBLs from $\mathbf{T}_{M+1}$ and condense the list **FBLP**.

(a) If the path list $\mathbf{T}_{M+1}$ is not empty, then search for the FBLs embedded in its rows. In other words, if a member appears twice in the same row, then a FBL is identified. Add all such paths into **FBLP**.

(b) Check each row in **FBLP**. If the starting and ending nodes of any of the FFLs identified in Step 7 can be found in the same row, delete the members corresponding to the intermediate nodes of the FFL from **FBLP**.

(c) Since identical FBLs may appear in **FBLP**, it is necessary to identify and delete all but one of them from the list.

As an example, the result of implementing the present algorithm to Example 1 is

$$\mathbf{FBLP} = [t2\ m4\ s6\ s5\ t2] \tag{29}$$

APPLICATION EXAMPLES

On the basis of the algorithms proposed in this study, a generic computer program IDLP has been developed accordingly. To demonstrate its effectiveness, four additional application examples are presented in this section.

Example 2. The heat exchange between a hot stream (H) and a cold stream (C) in an exchanger (EX) is considered in this example. In order to maintain the exit temperature of H at a constant level, the flow rate of H is adjusted via a negative feedback control system. A simplified flow diagram is presented in Fig. 5a. In this figure, TT represents a temperature sensor-transmitter, TIC is a reverse-action temperature controller, and the control valve (CV) is air to open (A/O). The system digraph has been drawn accordingly in Fig. 5b. The physical meanings of the symbols used in this digraph can be found in the Nomenclature section at the end of this paper. The FFLs and FBLs in this system were identified almost instantaneously on a IBM PC 486 with IDLP. They are presented in Table III. It is interesting to observe that the path $t4 \rightarrow s7 \rightarrow s8 \rightarrow m5$ exists in all four feedforward loops of the system, i.e. these process FFLs are actually caused by the feedback control

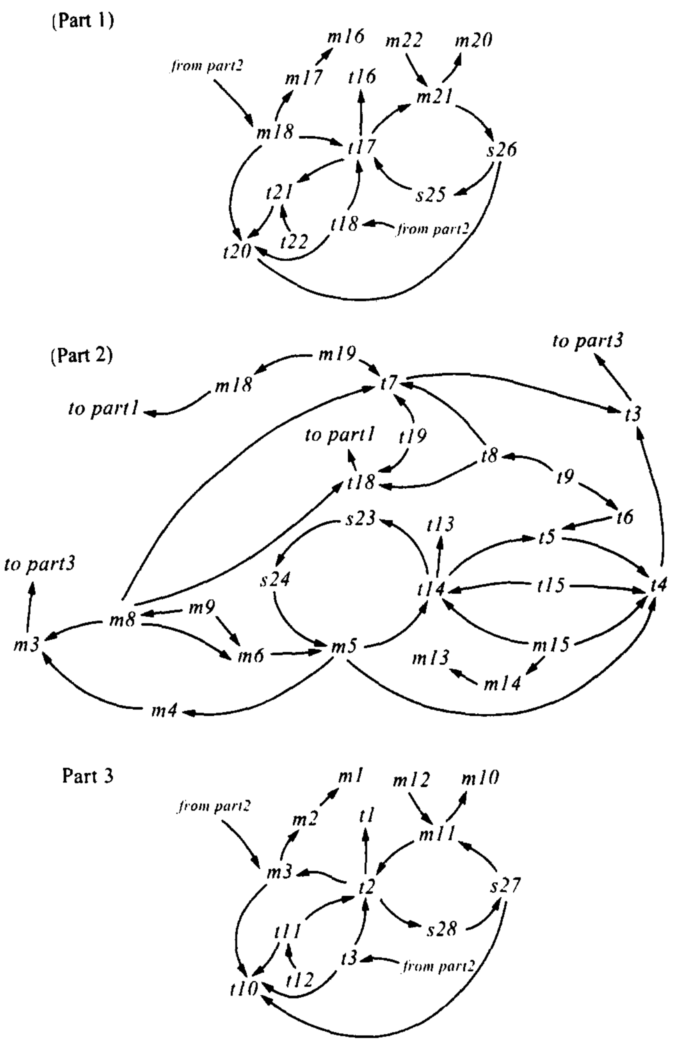


Fig. 8. (b) The system digraph of Example 5.

Table VI. The FFLs and FBLs identified in Example 5

[FFLP LIST]

1.
t2 m3 m4
t2 t3 m4

2.
t2 m3 m7
t2 t3 m7

3.
t2 t3 t4 t15
t2 t3 t4 m5 s24 s23 t14 t15
t2 t3 m4 m5 s24 s23 t14 t15
t2 t3 t7 m8 m5 s24 s23 t14 t15
t2 m3 m4 m5 s24 s23 t14 t15
t2 t3 m7 m8 m5 s24 s23 t14 t15
t2 m3 m7 m8 m5 s24 s23 t14 t15

4.
t2 t3 t4 m15
t2 t3 t4 m5 s24 s23 t14 m15
t2 t3 m4 m5 s24 s23 t14 m15
t2 t3 t7 m8 m5 s24 s23 t14 m15
t2 m3 m4 m5 s24 s23 t14 m15
t2 t3 m7 m8 m5 s24 s23 t14 m15
t2 m3 m7 m8 m5 s24 s23 t14 m15

5.
t2 t3 m4 m5
t2 t3 t7 m8 m5
t2 t3 t4 m5
t2 m3 m4 m5
t2 t3 m7 m8 m5
t2 m3 m7 m8 m5

6.
t2 t3 m7 m8
t2 m3 m7 m8
t2 t3 t7 m8

7.
t2 t3 t4 m5 m9
t2 t3 t7 m8 m9
t2 t3 m4 m5 m9
t2 t3 m7 m8 m9
t2 t3 t7 m8 m5 m9
t2 m3 m4 m5 m9
t2 m3 m7 m8 m9
t2 t3 m7 m8 m5 m9
t2 m3 m7 m8 m5 m9

8.
t2 t3 t7 t8 t9
t2 t3 t4 t5 t6 t9
t2 t3 t4 m5 s24 s23 t14 t5 t6 t9
t2 t3 m4 m5 s24 s23 t14 t5 t6 t9
t2 t3 t7 m8 m5 s24 s23 t14 t5 t6 t9
t2 m3 m4 m5 s24 s23 t14 t5 t6 t9
t2 t3 m7 m8 m5 s24 s23 t14 t5 t6 t9
t2 m3 m7 m8 m5 s24 s23 t14 t5 t6 t9

9.
t2 t3 t4 m5 s24 s23 t14 t5
t2 t3 m4 m5 s24 s23 t14 t5
t2 t3 t7 m8 m5 s24 s23 t14 t5
t2 m3 m4 m5 s24 s23 t14 t5
t2 t3 m7 m8 m5 s24 s23 t14 t5
t2 m3 m7 m8 m5 s24 s23 t14 t5
t2 t3 t4 t5

10.
t10 t11
t10 m11 s27 s28 t2 t11

Table VI. Continued

11.
t10 m3 m4
t10 m11 s27 s28 t2 m3 m4
t10 t3 m4

12.
t10 m3 m7
t10 m11 s27 s28 t2 m3 m7
t10 t3 m7

13.
t10 t3 t4 t15
t10 t3 t4 m5 s24 s23 t14 t15
t10 t3 m4 m5 s24 s23 t14 t15
t10 t3 t7 m8 m5 s24 s23 t14 t15
t10 m11 s27 s28 t2 t3 t4 t15
t10 m11 s27 s28 t2 t3 t4 m5 s24 s23 t14 t15
t10 m3 m4 m5 s24 s23 t14 t15
t10 t3 m7 m8 m5 s24 s23 t14 t15
t10 m11 s27 s28 t2 t3 m4 m5 s24 s23 t14 t15
t10 m11 s27 s28 t2 t3 t7 m8 m5 s24 s23 t14 t15
t10 m3 m7 m8 m5 s24 s23 t14 t15
t10 m11 s27 s28 t2 t3 m7 m8 m5 s24 s23 t14 t15
t10 m11 s27 s28 t2 m3 m4 m5 s24 s23 t14 t15
t10 m11 s27 s28 t2 m3 m7 m8 m5 s24 s23 t14 t15

14.
t10 t3 t4 m15
t10 t3 t4 m5 s24 s23 t14 m15
t10 t3 m4 m5 s24 s23 t14 m15
t10 t3 t7 m8 m5 s24 s23 t14 m15
t10 m11 s27 s28 t2 t3 t4 m15
t10 m11 s27 s28 t2 t3 t4 m5 s24 s23 t14 m15
t10 m3 m4 m5 s24 s23 t14 m15
t10 t3 m7 m8 m5 s24 s23 t14 m15
t10 m11 s27 s28 t2 t3 m4 m5 s24 s23 t14 m15
t10 m11 s27 s28 t2 t3 t7 m8 m5 s24 s23 t14 m15
t10 m3 m7 m8 m5 s24 s23 t14 m15
t10 m11 s27 s28 t2 t3 m7 m8 m5 s24 s23 t14 m15
t10 m11 s27 s28 t2 m3 m4 m5 s24 s23 t14 m15
t10 m11 s27 s28 t2 m3 m7 m8 m5 s24 s23 t14 m15

15.
t10 t3 m4 m5
t10 t3 t7 m8 m5
t10 t3 t4 m5
t10 m3 m4 m5
t10 t3 m7 m8 m5
t10 m11 s27 s28 t2 t3 m4 m5
t10 m11 s27 s28 t2 t3 t7 m8 m5
t10 m11 s27 s28 t2 t3 t4 m5
t10 m3 m7 m8 m5
t10 m11 s27 s28 t2 t3 m7 m8 m5
t10 m11 s27 s28 t2 m3 m4 m5
t10 m11 s27 s28 t2 m3 m7 m8 m5

16.
t10 t3 m7 m8
t10 m3 m7 m8
t10 m11 s27 s28 t2 t3 m7 m8
t10 m11 s27 s28 t2 m3 m7 m8
t10 t3 t7 m8

17.
t10 t3 t4 m5 m9
t10 t3 t7 m8 m9
t10 t3 m4 m5 m9
t10 t3 m7 m8 m9
t10 t3 t7 m8 m5 m9
t10 m11 s27 s28 t2 t3 t4 m5 m9
t10 m11 s27 s28 t2 t3 t7 m8 m9
t10 m3 m4 m5 m9
t10 m3 m7 m8 m9
t10 t3 m7 m8 m5 m9
t10 m11 s27 s28 t2 t3 m4 m5 m9
t10 m11 s27 s28 t2 t3 m7 m8 m9
t10 m11 s27 s28 t2 t3 t7 m8 m5 m9
t10 m3 m7 m8 m5 m9
t10 m11 s27 s28 t2 t3 m7 m8 m5 m9
t10 m11 s27 s28 t2 m3 m4 m5 m9
t10 m11 s27 s28 t2 m3 m7 m8 m9
t10 m11 s27 s28 t2 m3 m7 m8 m5 m9

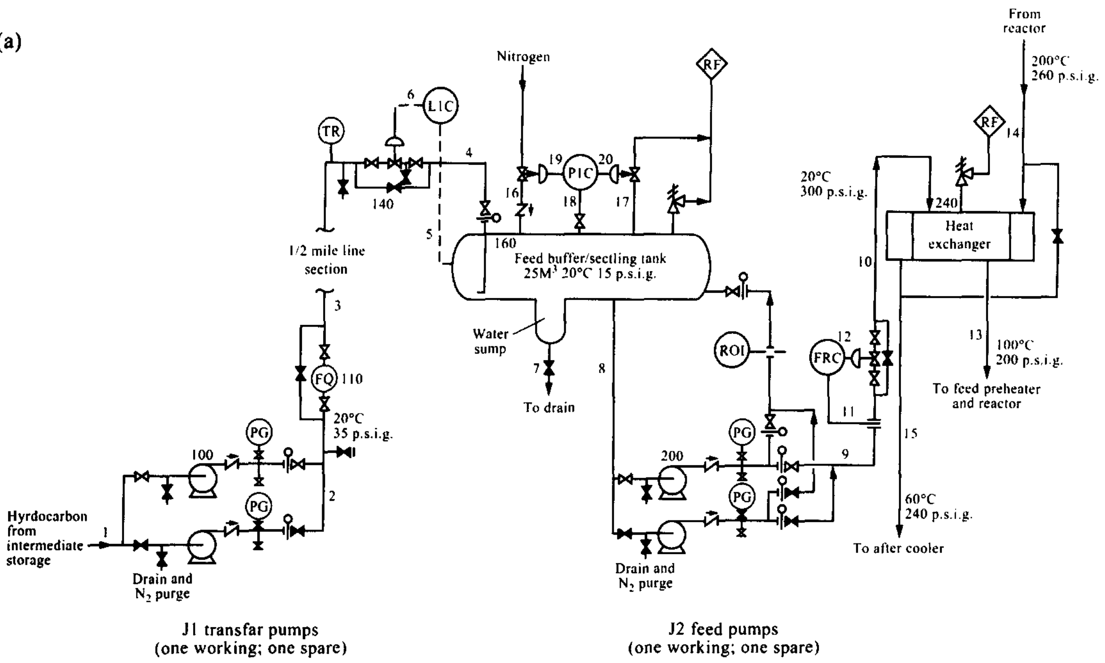
Table VI. Continued

```
18.
t10 t3 t7 t8 t9
t10 t3 t4 t5 t6 t9
t10 m11 s27 s28 t2 t3 t7 t8 t9
t10 m11 s27 s28 t2 t3 t4 t5 t6 t9
t10 t3 t4 m5 s24 s23 t14 t5 t6 t9
t10 t3 m4 m5 s24 s23 t14 t5 t6 t9
t10 t3 t7 m8 m5 s24 s23 t14 t5 t6 t9
t10 m11 s27 s28 t2 t3 t4 m5 s24 s23 t14 t5 t6 t9
t10 m3 m4 m5 s24 s23 t14 t5 t6 t9
t10 t3 m7 m8 m5 s24 s23 t14 t5 t6 t9
t10 m11 s27 s28 t2 t3 m4 m5 s24 s23 t14 t5 t6 t9
t10 m11 s27 s28 t2 t3 t7 m8 m5 s24 s23 t14 t5 t6 t9
t10 m3 m7 m8 m5 s24 s23 t14 t5 t6 t9
t10 m11 s27 s28 t2 t3 m7 m8 m5 s24 s23 t14 t5 t6 t9
t10 m11 s27 s28 t2 m3 m4 m5 s24 s23 t14 t5 t6 t9
t10 m11 s27 s28 t2 m3 m7 m8 m5 s24 s23 t14 t5 t6 t9

19.
t10 m11 s27 s28 t2 t3
t10 t3

20.
t10 m11 s27 s28 t2 m3
t10 m3

21.
t10 t3 t4 m5 s24 s23 t14 t5
t10 t3 m4 m5 s24 s23 t14 t5
t10 t3 t7 m8 m5 s24 s23 t14 t5
t10 m11 s27 s28 t2 t3 t4 m5 s24 s23 t14 t5
t10 m3 m4 m5 s24 s23 t14 t5
t10 t3 m7 m8 m5 s24 s23 t14 t5
t10 m11 s27 s28 t2 t3 m4 m5 s24 s23 t14 t5
t10 m11 s27 s28 t2 t3 t7 m8 m5 s24 s23 t14 t5
t10 m3 m7 m8 m5 s24 s23 t14 t5
t10 m11 s27 s28 t2 t3 m7 m8 m5 s24 s23 t14 t5
t10 m11 s27 s28 t2 m3 m4 m5 s24 s23 t14 t5
t10 m11 s27 s28 t2 m3 m7 m8 m5 s24 s23 t14 t5
t10 t3 t4 t5

22.
t3 t4 t15
t3 t4 m5 s24 s23 t14 t15
t3 m4 m5 s24 s23 t14 t15
t3 t7 m8 m5 s24 s23 t14 t15
t3 m7 m8 m5 s24 s23 t14 t15

23.
t3 t4 m15
t3 t4 m5 s24 s23 t14 m15
t3 m4 m5 s24 s23 t14 m15
t3 t7 m8 m5 s24 s23 t14 m15
t3 m7 m8 m5 s24 s23 t14 m15

24.
t3 m4 m5
t3 t7 m8 m5
t3 t4 m5
t3 m7 m8 m5

25.
t3 m7 m8
t3 t7 m8

26.
t3 t4 m5 m9
t3 t7 m8 m9
t3 m4 m5 m9
t3 m7 m8 m9
t3 t7 m8 m5 m9
t3 m7 m8 m5 m9

27.
t3 t7 t8 t9
t3 t4 t5 t6 t9
t3 t4 m5 s24 s23 t14 t5 t6 t9
t3 m4 m5 s24 s23 t14 t5 t6 t9
t3 t7 m8 m5 s24 s23 t14 t5 t6 t9
t3 m7 m8 m5 s24 s23 t14 t5 t6 t9
```

Table VI. Continued

```
28.
t3 t4 m5 s24 s23 t14 t5
t3 m4 m5 s24 s23 t14 t5
t3 t7 m8 m5 s24 s23 t14 t5
t3 m7 m8 m5 s24 s23 t14 t5
t3 t4 t5

29.
m3 m4 m5 m9
m3 m7 m8 m9
m3 m7 m8 m5 m9

30.
m3 m7 m8 m5
m3 m4 m5

31.
t4 t15
t4 m5 s24 s23 t14 t15

32.
t4 m15
t4 m5 s24 s23 t14 m15

33.
t4 m5 s24 s23 t14 t5
t4 t5

34.
t7 t8 t9
t7 m8 m5 s24 s23 t14 t5 t6 t9

35.
t18 t8 t9
t18 m8 m5 s24 s23 t14 t5 t6 t9

36.
m8 m9
m8 m5 m9

37.
t20 t21
t20 m21 s26 s25 t17 t21

38.
t20 t18 m19
t20 m18 m19
t20 m21 s26 s25 t17 t18 m19
t20 m21 s26 s25 t17 m18 m19

39.
t20 m21 s26 s25 t17 t18
t20 t18

40.
t20 m21 s26 s25 t17 m18
t20 m18

[END FFLP LIST]

[FBLP LIST]
t2 m11 s27 s28 t2
m5 s24 s23 t14 m5
m21 s26 s25 t17 m21
[END FBLP LIST]
```

system. Without the controller TIC, an increase in the inlet temperature of the cold stream should create the result of an increase in the exit temperature of the same stream. However, this effect is uncertain in the present example. This is due to the fact that a process feedforward loop with the starting node *t*2 exists in the digraph. A change in *t*2 produces opposite effects on *t*1 along the two paths of FFL. Thus, the net result of such



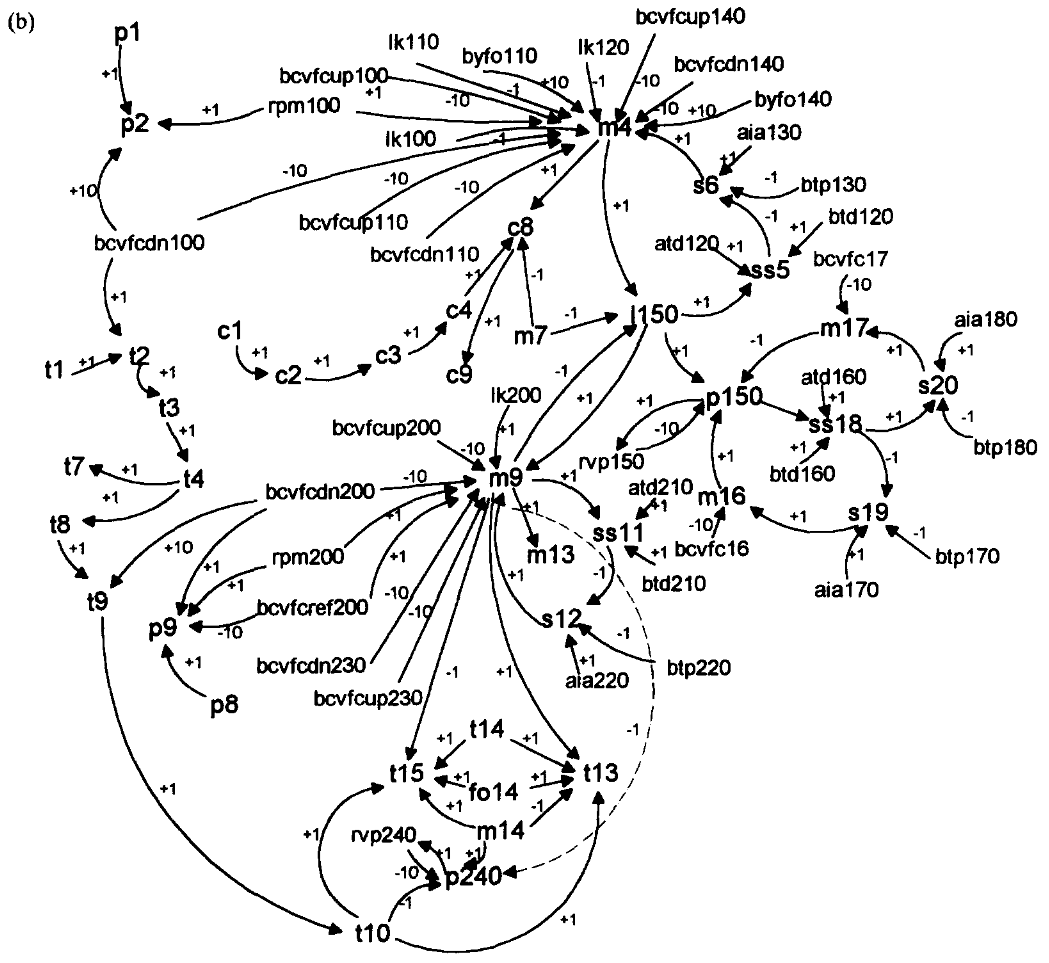


Fig. 9. (a) The process flow diagram of the feed section of an olefine dimerization plant. (b) The system digraph of (a).

a change in *t2* cannot really be determined without additional quantitative analysis of the system behavior.

Example 3. Ratio-control is a standard control technique implemented routinely in numerous chemical processes. In this example, let us consider a continuous reactor operated with such a control system. Two reactants, A and B, are fed to this reactor to produce a chemical C. The feed ratio of B to A is required to maintain at some desired value. The simplified flow diagram is presented in Fig. 6a. In this system, the flow of stream A is controlled by a feedback control system. The measurement signal obtained from the flow sensor-transmitter (FT1) is used in the controller FIC1 for manipulating the opening of control valve CV1. The

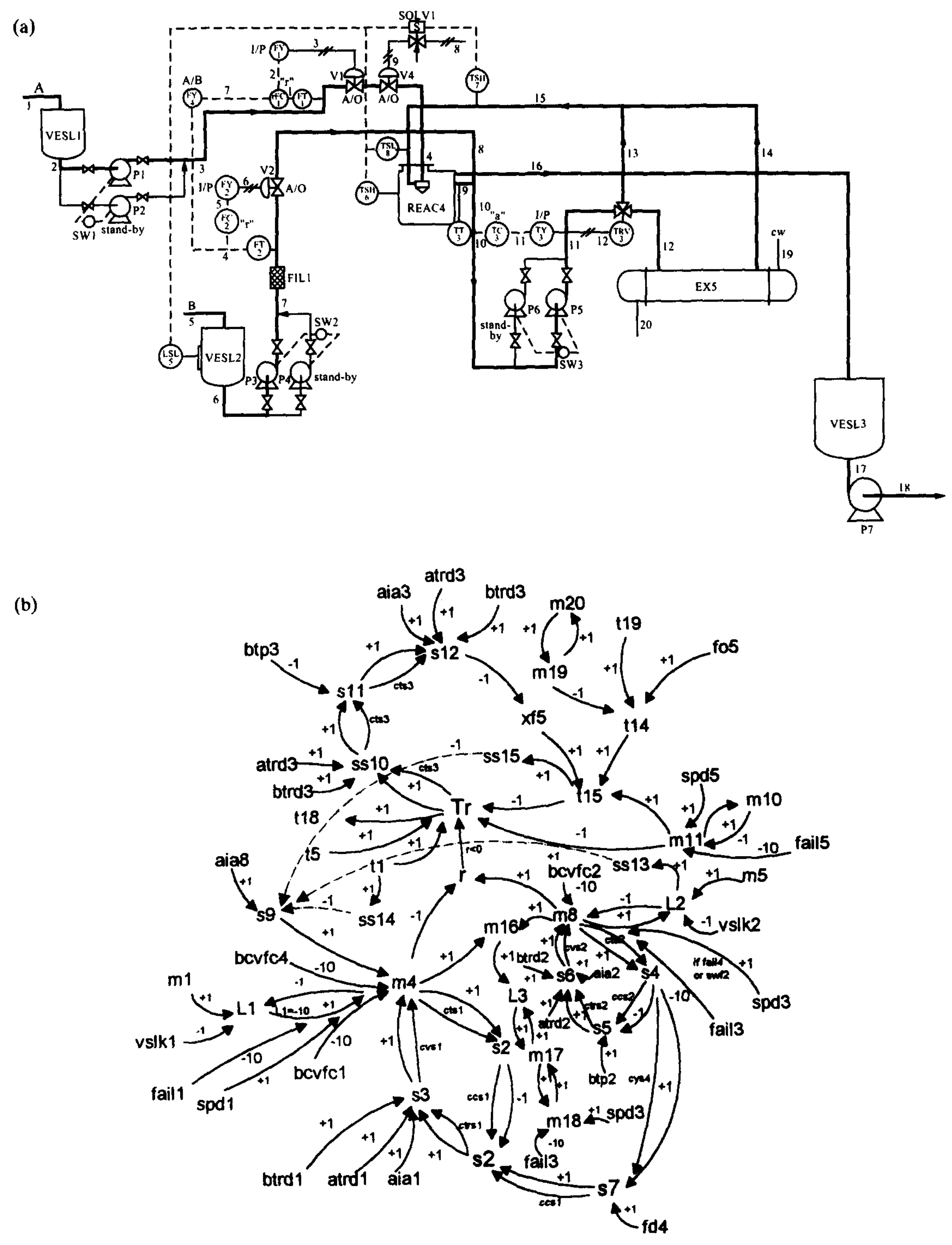


Fig. 10. (a) The process flow diagram of a reaction system. (b) The system digraph of (a).

same signal is also multiplied by a desired value in the ratio station FY1 to calculate the required flow of stream B. The output of FY1 is then used as the set point of the flow controller for stream B, FIC2. The controller FIC2 receives measurement signal of stream B from the sensor-transmitter FT2 and manipulates the control valve CV2. The corresponding system digraph is presented in Fig. 6b. The physical meanings of the symbols used in this digraph is again provided in the Nomenclature section. The FBLs and FFLs of this system are presented in Table IV. Notice that the starting nodes of the two FFLs in this example, i.e. *s*6 and *m*2, are located on the FBL formed in the flow-control system of stream A. In other words, it is this special configuration that cause the simultaneous existence of these two FFLs. Without FIC1 and CV1, the ratio-control system contains only one FFL. In such case, the feed ratio can still be maintained constant, but not the throughput.

Example 4. A heat exchanger with bypass control system is considered in this example. Although its process diagram (Fig. 7a) appears quite simple, the corresponding system diagram is actually very complex (Fig. 7b). Since the symbols used in these figures are similar to those in Example 1 and Example 2, their definitions are thus omitted here for the sake of conciseness. The results obtained with IDLP are presented in Table V. It can be observed that there are in total 4 FBLs and 20 FFLs in this simple process. Notice all of them are process loops except the second FBL. These informations are critical in correctly assessing the effects of any external disturbance to all important state variables in the system.

Example 5. The final example is designed to demonstrate the capability of IDLP in handling extremely complicated digraphs which are often encountered in practical applications. Let us consider the heat exchanger network presented in Fig. 8a. Due to the complexity of the system configuration, the digraph is divided into three parts and drawn separately in Fig. 8b. They represent the models corresponding to the subsystems surrounding exchanger EX3 (Part 1), EX2 and EX4 (Part 2) and EX1 (Part 3), respectively. After executing IDLP, a total of 40 FFLs and 3 FBLs were successfully identified (Table VI). It can be observed from Table VI that, with the help of IDLP, a large quantity of information can be generated within a short period of time. It is also quite obvious that a complete and correct list of these loops cannot be produced easily by manually inspecting the digraph.

Finally, it should be emphasized that the program IDLP has been successfully tested with many other industrial-scale problems. One of them is presented in Fig. 9a. This P&ID represents the feed section of an olefine dimerization plant (Lawley, 1974). In this part of the plant, an alkene/alkane fraction containing small amounts of suspended water is continuously pumped from a bulk intermediate storage tank via a half-mile pipeline into a buffer/settling tank where residual water is settled out prior to passing via a feed/product heat exchanger and preheater to the reaction section. The water in the settling tank is run off manually at intervals. The corresponding digraph model is presented in Fig. 9b. By executing IDLP, it was found that there are two FFLs and nine FBLs in the system. Another somewhat more demanding problem can be found in Fig. 10a. In this process, the reaction between reactants A and B takes place in reactor REAC4. The product is overflowed into storage tank VESL3. Since the reaction is exothermic, the content of the reactor is circulated through the cooler EX5 to remove heat. A ratio control system is equipped to maintain the flow ratio of B to A at 2. If this ratio is significantly lower than this set point, a runaway reaction may occur. Thus, several interlocks have been installed to protect the reactor. Specifically, the instrument air to control valve V4 can be vented via solenoid valve SOLV1. Any of the following conditions can trigger SOLV1: (1) reactor temperature too high (TSH6); (2) flow rate in the circulation line too low (FSL8); (3) temperature in the circulation line too high (TSH7); (4) the level in storage tank VESL2 too low (LSL5). The corresponding digraph is provided in Fig. 10b. The numbers of FFLs and FBLs identified in this case are 21 and 9, respectively. Therefore, from a structural viewpoint, these industrial-scale examples are not more complicated than Example 5. For the sake of brevity, detailed lists of the identified loops are not included in this paper.

CONCLUSIONS

On the basis of a simple search strategy, algorithms for identification and classification of *all* FFLs and FBLs in a digraph have been developed in this study. Also, a generic computer program IDLP has been coded accordingly. The proposed approach has been tested with numerous realistic examples. From the results generated so far, one can conclude that, with the help of IDLP, it is indeed possible to produce a comprehensive and correct list of loop paths in any given digraph within a short period of time.

NOMENCLATURE

*fo*1	=fouling in heat exchanger EX.
mi	=the mass flow rates in pipeline *i* (*i* = 1,2,...).
*r*5	=flow ratio.
si	=the electric or pneumatic signals in line *i* (*i* = 1,2,...).
*sp*1	=the set point of controller *TIC*.
ti	=the temperature in pipeline *i* (*i* = 1,2,...).

Acknowledgements—This work is supported by the National Science Council of the Republic of China under project NSC80-0402-E006-02.

REFERENCES

Allen D.J. and M.S.M. Rao, New algorithms for the synthesis and analysis of fault trees. *I&EC Fundam.* **19**, 79 (1980).

Andrews J. D. and G. Brennan, Application of the digraph method of fault tree construction to a complex control configuration. *Reliability Engineering and System Safety* **28**, 357 (1990).

Chang C. T. and H. C. Hwang, New developments of the digraph-based techniques for fault-tree synthesis. *I&EC Res.* **31**, 1492 (1992).

Christensen J.H., The structuring of process optimization. *AIChE J.* **16**, 177 (1970).

Doe N., Graph Theory with Applications to Engineering and Computer Science, Prentice-Hall Englewood Cliffs, New Jersey (1974)

Iri M., K. Aoki, E. O'Shima and H. Matsuyama, An algorithm for diagnosis of system failures in the chemical process. *Comput. Chem. Eng.* **3**, 489 (1979).

Kramer M. A. and B. L. Palowitch Jr, A rule-based approach to fault diagnosis Using the signed directed graph. *AIChE J.* **33**, 1067 (1987).

Lapp S. A. and G. J. Powers, Computer-aided synthesis of fault trees. *IEEE Trans. Reliab.* **R-26**, 2 (1977).

Lawley H. G., Operability studies and hazard analysis. *Chem. Eng. Prog.* **70**, 105 (1974).

Lee W. Y., J. H. Christensen and D. F. Rudd, Design variable selection to simplify process calculations. *AIChE J.* **12**, 1104 (1966).

Mah R. S. H., Chemical Process Structures and Information Flows, Butterworths Boston, Massachusetts (1990).

Mah R. S. H., G. M. Stanley and D. M. Downing, Reconciliation and rectification of process flow and inventory data. *I&EC Proc. Des. Dev.* **15**, 175 (1976).

Oyeleye O. O. and M. A. Kramer, Qualitative simulation of chemical process systems: steady state analysis. *AIChE J.* **34**, 1441 (1988).

Shiozaki J., H. Matsuyama, E. O'Shima and M. Iri, An improved algorithm for diagnosis of system failures in the chemical process. *Comput. Chem. Eng.* **9**, 285 (1985).

Steward D. V., On an approach to techniques for the analysis of the structure of large systems of equations. *SIAM Rev.* **4**, 321 (1962).

Tarjan R., Depth-first search and linear graph algorithms. *SIAM J. Comput.* **1**, 146 (1972).