Computation in Network Reliability

Shin-Guang Chen

Abstract This chapter presents the primitive steps in computation of network reliability, and some of the popular examples by applying network reliability. Network reliability is one of the most interesting topics in network theory. It has been applied to many real-world applications such as traffic planning, computer network planning, power transmission planning, etc. The planners usually want to know how reliable the systems they planned. Network reliability helps them to get the answer of such questions. Due to its NP-hard nature, many approaches have been developed to tackle the efficiency problems in computations. One of the most promising methods is the three-stage approach, which divides the problem into three smaller problems and conquers them, making the resolution process more efficient than others. Several examples are presented in the chapter for illustrative purposes.

1 Introduction

Network reliability is one of the most interesting topics in network theory. It has been applied to many real-world applications such as traffic planning, computer network planning, power transmission planning, etc. The planners usually want to know how reliable the systems they planned. Network reliability helps them to get the answer of such questions. Since 1954, the maximal flow problems have gained much attention in the world [23]. They are also extended to many other fields for applications [14]. Aggarwal et al. firstly discussed the reliability problem without flow in a binary-state network [3]. Lee [17] extended it to cover the flow cases. Aggarwal et al. [2] presented the minimal path (MP) method to solve the network reliability in a binary-state flow network. Xue [26] began to discuss the reliability analysis in a multistate network (MSN), which is also referred to the stochastic-flow network (SFN). A SFN is a network whose flow has stochastic states. Lin et al. [18] illustrated the reliability calculation of a SFN in terms of MPs or minimal cuts (MC)s [15]. They also setup

S.-G. Chen (✉)
Tungnan University, New Taipei City, Taiwan
e-mail: bobchen@mail.tnu.edu.tw

H. Pham (ed.), *Reliability and Statistical Computing*, Springer Series in Reliability Engineering, https://doi.org/10.1007/978-3-030-43412-0_7

the three-stages in these calculations: a. Searching for all MPs [10, 12, 24]/MCs [1, 7, 16, 30]; b. Searching for all $d-$MPs [20, 27]/$d-$MCs [28] from these MPs/MCs; c. Calculating union probability from these $d-$MPs [5, 6, 32]/$d-$MCs [12, 19]. Lin [20] greatly simplified the three-stages method (TSM) and developed more simple and efficient algorithm for the reliability evaluation of a general SFN.

Doulliez and Jamoulle [13] also proposed a novel method to evaluate network reliability, namely state space decomposition (SSD). However, their algorithm has some flaw, and cannot obtain the correct value. Aven [4] presented the correct one for state space decomposition approach, and till now still keeps excellent efficiency in SSD-based approach. The complexity of the worst cases for SSD is $O(N^d)$, where N is the number of arces, and d is the demand. TSM divided the reliability evaluation process into three stages. The first stage—finding MPs has the complexity $O(W + S)$ [10], where W is the number of MPs, and S is the number of cycles. The second stage—searching for d-MPs has the complexity $O(\prod_{k=1}^{u} r_k + |B| \binom{z+d}{d-1})$ [21], where r_k is the kth segment of u flow constraints, $|B|$ is the number of nodes, z is the number of MPs, and d is the demand. The third stage—calculating the union probability of d-MPs has the complexity $O(2^Q)$ [5, 6], where Q is the number of d-MPs. Therefore, when N is large, the evaluation efficiency of TSM is undoubtedly superior to SSD.

The remainder of the chapter is organized as follows. The mathematical preliminaries for the network reliability are presented in Sect. 2. The network representation method [9] is given in Sect. 3. The MP searching method [10] is given in Sect. 4. The d-MP searching method [21] is given in Sect. 5. The union probability calculation method [5] is given in Sect. 6. Finally, Sect. 7 draws the conclusion of this chapter.

2 Preliminaries

Let (A, B) be a SFN, where $A = \{a_i | 1 \le i \le N\}$ is the set of edges, B is the set of nodes. Let $M = (m_1, m_2, \ldots, m_N)$ be a vector with m_i (an integer) being the maximal capacity of a_i. The network is assumed to satisfy the following assumptions.

1. Flow in the network satisfies the flow-conservation law [14].
2. The nodes are perfect.
3. The capacity of a_i is an integer-valued random variable which takes values from the set $\{0, 1, 2, \ldots, m_i\}$ according to a given distribution μ_i.
4. The states in edges are statistically independent from each other.

2.1 Modeling of Networks

Assume that ρ_1, ρ_2, …, ρ_z are totally the MPs from the source to the sink. Thus, the network is modeled by two vectors: the state vector $X = (x_1, x_2, \ldots, x_N)$ and the flow vector $Y = (y_1, y_2, \ldots, y_z)$, where x_i denotes the current state of a_i and y_j

denotes the current flow on ρ_j. Then, Y is feasible if and only if

$$\sum_{j=1}^{z}\{y_j|a_i \in \rho_j\} \leq m_i, \text{ for } i = 1, 2, \ldots, N. \tag{1}$$

This constraint describes that the total flow through a_i can not exceed the maximal capacity of a_i.Then, the set of Y denoted $\kappa_M \equiv \{Y|Y \text{ is feasible under } M\}$.

Similarly, Y is feasible under $X = (x_1, x_2, \ldots, x_N)$ if and only if

$$\sum_{j=1}^{z}\{y_j|a_i \in \rho_j\} \leq x_i, \text{ for } i = 1, 2, \ldots, N. \tag{2}$$

For clarity, let $\kappa_X = \{Y|Y \text{ is feasible under } X\}$. The maximal flow under X is defined as $\varpi_X \equiv \max\{\sum_{j=1}^{z} y_j|Y \in \kappa_X\}$.

Then, the flow vector $Y \in \kappa_M$ could be found such that the total flow of Y equals d. It is defined in the following constraint,

$$\sum_{j=1}^{z} y_j = d. \tag{3}$$

Let $\mathbf{Y}$=$\{Y|Y \in \kappa_M$ and satisfies Eq. (3)$\}$. We have the following lemmas [20].

Lemma 1 *If X is a d-MP for d, then there is an $Y \in \mathbf{Y}$ such that*

$$x_i = \sum_{j=1}^{z}\{y_j|a_i \in \rho_j\}, \quad \text{for each } i = 1, 2, \ldots, N. \tag{4}$$

Given $Y \in \mathbf{Y}$, a state vector $X_Y = (x_1, x_2, \ldots, x_N)$ can be built by Eq. (4). The set $\Lambda = \{X_Y|Y \in \mathbf{Y}\}$ is created. Let $\Lambda_{\min} = \{X|X \text{ is a lower boundary vector in } \Lambda\}$. Then,

Lemma 2 *Λ_{min} is the set of d-MPs for d.*

2.2 Evaluation of Network Reliability

Given a demand d, the reliability denoted by ω_d is the probability at sink node that the maximal flow in the network is no less than d, i.e., $\omega_d \equiv \Pr\{X|\varpi_X \geq d\}$. To calculate ω_d, find the lower boundary vectors directly in the set $\{X|\varpi_X \geq d\}$. A lower boundary vector X is said to be a d-MP for d if and only if (i) $\varpi_X \geq d$ and (ii) $\varpi_W < d$ for any other vector W such that $W < X$, in which $W \leq X$ if and only if $w_j \leq x_j$ for each $j = 1, 2, \ldots, n$ and $W < X$ if and only if $W \leq X$ and $w_j < x_j$

for at least one j. Suppose there are totally q d-MPs for d: $X_1, X_2, \ldots, X_q$. The reliability is equal to

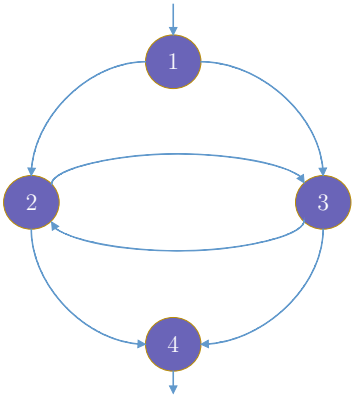
$$\omega_d = \Pr\left\{\bigcup_{k=1}^{q}\{X|X \geq X_k\}\right\}, \tag{5}$$

which can be calculated by reduced recursive inclusion-exclusion principle (RRIEP) [5].

3 Network Representation

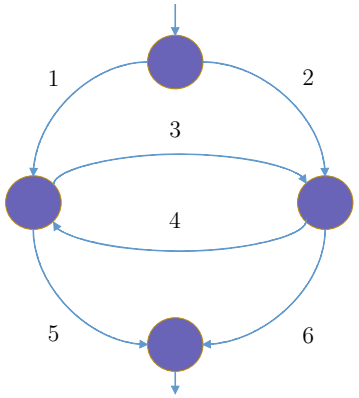
In investigating network problems, the method for expressing network structure is important. Usually, the adjacency matrix [25] (AM) is employed. An adjacency matrix is a square matrix used to represent a finite graph. The elements of the matrix indicate whether pairs of vertices (arcs) are adjacent or not in the graph. Figure 1 gives an example of node-oriented AM expression, where the entry in the matrix indicates the number of links between 'from' node and 'to' node. Such entry implicitly indicates the direction of the links between two nodes. Thus, multiple same direction links between two nodes are allowed in such AM expression. Figure 2 gives another arc-oriented AM expression. However, only 1 or 0 are allowed in this expression. Since both nodes-oriented and arc-oriented AM are equivalent in nature, we will take node-oriented AM in the following discussion.

A network is a graph except that the network has source nodes and sink nodes. Therefore, using AM to solve network problems normally faces difficulties in handling source nodes and sink nodes. Furthermore, in the view points of computer science, AM is not the best choice for program implementation. Specially, in solving network reliability, the linked path structure (LPS) [10] is now more and more



Fm\To	1	2	3	4
1	0	1	1	0
2	0	0	1	1
3	0	1	0	1
4	0	0	0	0

Fig. 1 A graph and its node-oriented adjacency matrix expression



Fm\To	1	2	3	4	5	6
1	0	0	1	0	1	0
2	0	0	0	1	0	1
3	0	0	0	1	0	1
4	0	0	1	0	1	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0

Fig. 2 A graph and its arc-oriented adjacency matrix expression

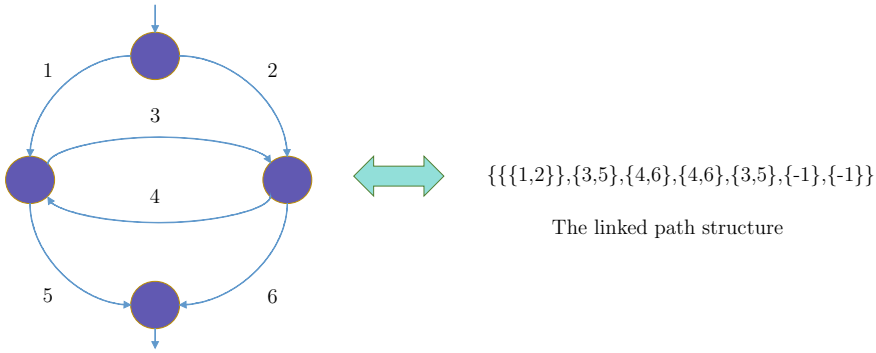


Fig. 3 A network and its linked path structure

popular in the literature. A LPS is a hybrid form of node-oriented and arc-oriented representation of a network. In LPS, entry 0 indicates the source nodes, and the negative entry values denote the sink nodes. The other entry values in LPS are vectors representing nodes with outbound arcs as their values in the vector. The index of LPS is the arc no. pointing the node (i.e. the entry value). Therefore, a LPS gives all the necessary information for networks in the applications. Figure 3 shows a LPS example. This LPS has a source node {1,2} with outbound arc 1 and 2. Arc 1 is connected with node {3,5} with outbound arc 3 and 5. Arc 2 is connected with node {4,6} with outbound arc 4 and 6. Arc 3 is connected with node {4,6} with outbound arc 4 and 6. Arc 4 is connected with node {3,5} with outbound arc 3 and 5. Arc 5 is connected with sink node $\{-1\}$. Arc 6 is connected with sink node $\{-1\}$. Thus, a tool for transforming AM to LPS is required.

An AM is a $N \times N$ square matrix V, where v_{ij} represents the number of arcs from node i to node j. We have the following lemas hold.

Lemma 3 $\exists i$, $\sum_{j=1}^{N} v_{ij} = 0$, *then, node* i *is a sink node.*

Lemma 4 $\exists j$, $\sum_{i=1}^{N} v_{ij} = 0$, *then, node* j *is a source node.*

Lemma 5 $\exists i$, j *and* $i \neq j$, $v_{ij} = v_{ji}$, *then, node* i *and* j *have undirected links.*

By above lemas, we can derive the following novel approaches for transforming AM to LPS.

3.1 The Proposed Approach

Let L be a LPS for the network. Then, we have the following properties.

Property 1 $\exists l_k \in L$, then, l_0 is the set of source nodes.

Property 2 $\exists l_k \in L$ and $k > 0$, then, l_k is a vector of arcs outbound from the node connected by arc k.

Property 3 $\exists l_k \in L$ and $k > 0$, if l_k is a vector of one negative integer, then, arc k connected with a sink node.

3.2 Algorithm of Transformation from AM to LPS

This algorithm will create two outputs: the LPS L, and a list of arc-node pairs K for verification purposes.

Algorithm 1

1. Input $N \times N$. // the AM for the underlined network.
2. $k = 1$ and $R = \phi$ // k the index for LPS and R the set of outbound arcs for the nodes.
3. For $i \in \{1, \ldots, N\}$ do // column handling.
4. For $j \in \{1, \ldots, N\}$ do // row handling.
5. if $v_{ij} - 1 < 0$ then continue,
6. else $W = W \cup \{k, \{i, j\}\}$, $R_i = R_i \cup \{k\}$, $k = k + 1$, and $v_{ij} = v_{ij} - 1$, goto step 5.

endfor
endfor
7. $k = 0$
8. For $i \in \{1, \ldots, N\}$ do // search for sink nodes.
9. if $R_i = \phi$, then, $k = k - 1$ and $R_i = \{k\}$.

endfor
10. $k = 0$

11. For $j \in \{1, \ldots, N\}$ do // search for source nodes.
12. if $\sum_{i=1}^{n} v_{ij} = 0$, then, $L_0 = L_0 \cup R_j$.
endfor
13. For $k \in \{1, \ldots, z\}$ do // search for LPS.
14. $L_k = R_{W_k(3)}$. // $W_k(3)$ means the third value in W_k.
endfor
15. Return.

The complexity analysis for this algorithm is as follows. For the AM transformation, it requires $O(N^2)$ to do step 3. Therefore, the total complexity is $O(N^2)$.

3.3 Illustrative Examples

Figure 4 gives five examples for this illustration, where Fig. 4 (a) is a 9×12 (nodes $\times$ arcs) network, (b) is a 16×20 network, (c) is a 25×36 network, (d) is a 32×48 network, and (e) is a 32×48 network. The transformed LPS are listed in Table 1.

Table 1 shows the LPS transformed correspondingly. All the cases are running on a PC with intel CPU of Core i5 6200U 2.30 GHz 2.40 GHz and 8 GB RAM.

4 Searching for MPs

Firstly, we transform the undirected edge to its directed form of arcs, and to search for all MPs in that transformed network. Then, the directed MPs are transformed back to the original form of the MPs. The following gives the details.

4.1 The General Form of a Network Transforming to Its Directed One

The difference between the general form of a network and its directed one is that there is only one way of flow for an arc in the directed form, but its general form may have both directions of flow for an edge. The following properties describe the details.

Property 4 An edge from a source node in a general flow network only has outgoing flow.

Property 5 An edge to a sink node in a general flow network only has ingoing flow.

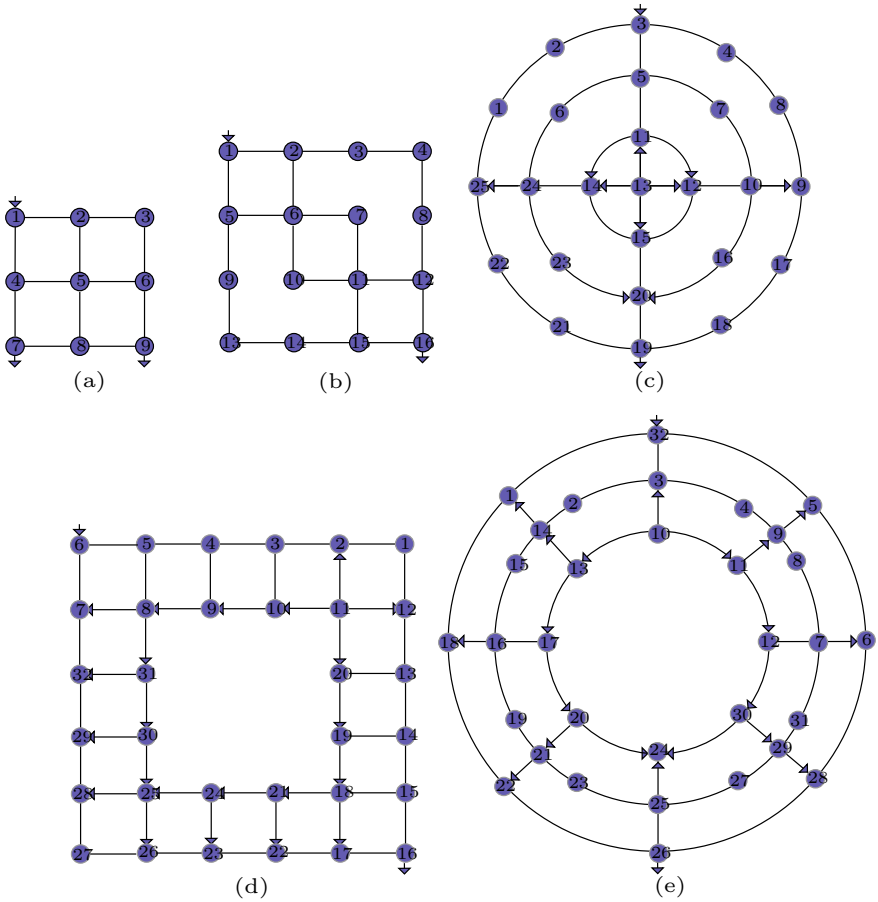


Fig. 4 Five examples for AM transformation

Property 6 A pair of reverse arcs is equivalent to a single undirected edge.

Property 7 The transformed network has an arc number greater than the total number of edges in the original network.

Thus, the edges from the source nodes or to the sink nodes have only one way of flows. The undirected edges on the other part of the network can be transformed to two reverse arcs. Figure 5 illustrates the transformation. Note that the additional reverse arcs are required for modeling the reverse flows. The backward transformation of the directed MP is to replace the added arc number in the MP back to its counterpart.

The following property shows that a cycle of length two for the undirected or hybrid edges is a parallel. Figure 6 illustrates the situation.

Table 1 The results derived by the algorithm for the five examples

Examples	LPS
(9×12)	{{(1, 2)}, (3, 4), (7, 8), (5, 6), (9, 10, 11, 12), (3, 4), (13, 14, 15), (9, 10, 11, 12), (16, 17), (3, 4), (7, 8), (13, 14, 15), (18, 19, 20), (5, 6), (9, 10, 11, 12), (−1), (7, 8), (18, 19, 20), (9, 10, 11, 12), (16, 17), (−1)}
(16×20)	{{(1, 2)}, (3, 4), (9, 10), (5, 6), (11, 12, 13, 14), (3, 4), (7, 8), (5, 6), (17, 18), (11, 12, 13, 14), (19, 20), (3, 4), (9, 10), (15, 16), (21, 22), (11, 12, 13, 14), (23, 24, 25, 26), (7, 8), (27, 28, 29), (9, 10), (30, 31), (11, 12, 13, 14), (23, 24, 25, 26), (15, 16), (21, 22), (27, 28, 29), (34, 35, 36), (17, 18), (23, 24, 25, 26), (−1), (19, 20), (32, 33), (30, 31), (34, 35, 36), (23, 24, 25, 26), (32, 33), (−1)}
(25×36)	{{(4, 5, 6), (27, 28, 29, 30)}, (3), (54, 55), (1, 2), (3), (7), (8, 9), (14, 15), (10, 11), (12, 13), (8, 9), (50, 51, 52, 53), (8, 9), (18, 19, 20, 21), (7), (16, 17), (14, 15), (38, 39), (12, 13), (16, 17), (25, 26), (36, 37), (8, 9), (25, 26), (31, 32), (18, 19, 20, 21), (33, 34, 35), (22, 23, 24), (25, 26), (31, 32), (33, 34, 35), (33, 34, 35), (50, 51, 52, 53), (25, 26), (31, 32), (42, 43), (18, 19, 20, 21), (42, 43), (16, 17), (40, 41), (38, 39), (−1), (33, 34, 35), (−1), (−1), (46, 47), (44, 45), (54, 55), (42, 43), (50, 51, 52, 53), (10, 11), (31, 32), (48, 49), (54, 55), (1, 2), (46, 47)}
(32×48)	{{(13, 14), (23, 24, 25, 26)}, (3, 4), (27, 28), (1, 2), (5, 6, 7), (3, 4), (8, 9, 10), (21, 22), (5, 6, 7), (11, 12), (19, 20), (8, 9, 10), (16, 17, 18), (11, 12), (15), (68, 69), (11, 12), (15), (66, 67), (8, 9, 10), (16, 17, 18), (5, 6, 7), (19, 20), (3, 4), (21, 22), (27, 28), (44, 45), (1, 2), (29, 30), (27, 28), (44, 45), (29, 30), (34, 35, 36), (42, 43), (31, 32, 33), (−1), (39, 40, 41), (−1), (48, 49), (34, 35, 36), (37, 38), (46, 47), (31, 32, 33), (39, 40, 41), (29, 30), (42, 43), (48, 49), (52, 53), (37, 38), (50, 51), (48, 49), (56, 57), (50, 51), (54, 55), (56, 57), (60, 61), (50, 51), (58, 59), (56, 57), (60, 61), (58, 59), (62, 63), (60, 61), (68, 69), (54, 55), (62, 63), (64, 65), (68, 69), (15), (62, 63)}
(32×48)	{{(20, 21, 22), (68, 69, 70)}, (40, 41), (4, 5), (29, 30, 31), (2, 3), (6, 7), (4, 5), (17, 18, 19), (9, 10), (8), (57, 58), (9, 10), (15, 16), (25, 26), (66, 67), (11, 12, 13, 14), (17, 18, 19), (6, 7), (8), (15, 16), (4, 5), (23, 24), (27, 28), (17, 18, 19), (25, 26), (11, 12, 13, 14), (64, 65), (29, 30, 31), (38, 39), (1), (2, 3), (32, 33), (29, 30, 31), (34, 35, 36, 37), (32, 33), (38, 39), (40, 41), (42, 43), (34, 35, 36, 37), (44, 45), (1), (49, 50), (34, 35, 36, 37), (46, 47, 48), (46, 47, 48), (−1), (42, 43), (49, 50), (51, 52), (40, 41), (−2), (46, 47, 48), (53, 54, 55, 56), (51, 52), (−1), (−2), (57, 58), (53, 54, 55, 56), (61, 62, 63), (9, 10), (−2), (57, 58), (59, 60), (66, 67), (−1), (61, 62, 63), (11, 12, 13, 14), (61, 62, 63), (1), (4, 5), (8)}

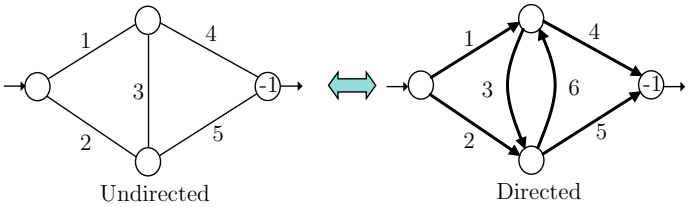
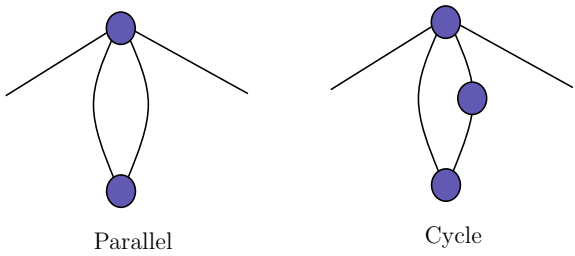


Fig. 5 The transformation between undirected and directed networks

Fig. 6 The parallel and cycle for an undirected network



Property 8 A cycle of length two for the undirected or hybrid edges is a parallel.

So, the minimal cycle length in a general flow network is 3 as we define it. This means that the minimal cycle length in its corresponding directed form should be set to no less than 3.

From the above notations, we have the following lemmas.

Lemma 6 (Cycle detection) *$\rho_j \cup v$ constructs a cycle iff $\exists v_k \in \rho_j$ such that $v_k = v$.*

From Lemma 6, we can search for all cycles in the network. Because a loop is a special case of a cycle, Lemma 6 lets such networks be applicable. Let S and T denote the source nodes and sink ones.

Lemma 7 (Minimal Path detection) *ρ is a MP iff ρ has no cycles, and the corresponding $w = s \cup v_1 \cup v_2 \cup \cdots \cup v_k \cup t$, where $v_k \in B - (S \cup T)$, $s \in S$, and $t \in T$.*

4.2 Algorithm for Searching MPs

Given the reverse arc information for all undirected links, the algorithm is given in the form in which backtracking is indicated at the line of activation.

Algorithm 2 Searching for all MP.

1. Input L. // Input the direct form of network configuration.
2. For $v \in L(0) = S$, and $v \neq \phi$, do the following steps.// Select a candidate.
3. $R = v \cup L - S$. // Construct the working list R with starting index 0.
4. Let $j = 0$, $\rho = w = \phi$. // Initialize the starting point, the current path, node path.
5. If $j < 0$, then $\rho = \cup_{2 \leq l}\{\rho(l)\}$, transform back to undirected form if necessary, output the MP, and backtrack. // Got an MP.
6. Else, begin $v = R(j)$. // Go to the node pointed by j.
7. If $w(k) = v$ is true, then backtrack.
8. Else, if $j \in v$, and $\{j\} \neq \phi$, then $\rho = \{j\} \cup \rho$, $w = \{v\} \cup w$, and call Step 5 and its following. // Recursive call.
9. Else, backtrack. // Exhaust the outgoing branches of the node.

End.
end For. // Searching finished.

4.3 Illustrative Examples

Figure 7 shows the illustrative network in the literature [31]. The results are given in Table 2. Besides, all cycles were found too. No initial network is required for the proposed approach, and the CPU time for this case was only 0.03 s.

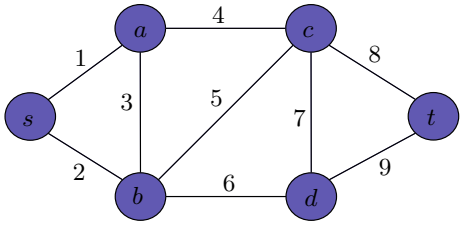


Fig. 7 The ARPA network

Table 2 The MPs searched

	The proposed approach
MPs	{2, 3, 4, 8} {2, 3, 4, 7, 9} {1, 4, 5, 6, 9} {2, 6, 7, 8} {1, 3, 6, 7, 8} {1, 3, 5, 7, 9} {1, 3, 5, 8} {1, 3, 6, 9} {1, 4, 7, 9} {1, 4, 8} {2, 5, 7, 9} {2, 5, 8} {2, 6, 9}
Cycles	{4, 5, 3} {4, 7, 6, 3} {6, 7, 5}

5 Searching for *d*-MPs

Let $X = \{x_1, x_2, \ldots, x_N\}$ be the states of the network. The following property and lemmas hold.

Property 9 (Path existence) If a_i has a path, then, $x_i \geq 0$.

Lemma 8 (Cycle existence) *$\rho_j \cup v$ is a cycle iff $\exists v_k \in \rho_j$ with $v_k = v$.*

By Property 9 and Lemma 8, all cycles or loops in the network can be found. Yeh [29] proposed the following lemma.

Lemma 9 (*d*-MP existence) *X is a d-MP iff no cycle exists in X.*

5.1 Flow Enumerations

Based on the Chen's work [8], $\mathbf{Y}$ can be fast enumerated by optimal re-arrangement of the formulas in Ψ_Y and Eq. (3). Next, we build X_Y by $Y \in \mathbf{Y}$ with Eq. (4). Then, $\Lambda = \{X_Y | Y \in \mathbf{Y}\}$ is created for all X_Y. Finally, $\Lambda_{min} = \{X | X$ is a d-MP in $\Lambda\}$ is created. Let Ξ_{Ψ_Y} be the optimal fast enumeration form (OFE-form) of formulas in Ψ_Y [8]. The fast enumeration of flows are created from these formulas.

5.2 Algorithm for Searching d-MPs

Let $p_j = \min\{x_i | a_i \in \rho_j\}$ and L be the linked path structure for the network [10]. Based on the above subsection discussion, The following algorithm is proposed.

Algorithm 3 Searching for all d-MPs for the network.

1. Enumerate $Y = (y_1, y_2, \ldots, y_z)$ under OFE-form.
   a. **fast enumerate** $0 \leq y_j \leq p_j$ **under** Ξ_{Ψ_Y} **do**
   b. **if** y_j satisfies $\sum_{j=1}^{z} y_j = d$,
      then $\mathbf{Y} = \mathbf{Y} \cup \{Y\}$.
      end enumerate.
2. Generate Λ_{min} via $\mathbf{Y}$.
   a. **for** Y in $\mathbf{Y}$ **do**
   b. $x_i = \sum_{j=1}^{z} \{y_j | a_i \in \rho_j\}$ for $i = 1, 2, \ldots, N$.
   c. **for** $1 \leq i \leq N$ **do**
   d. **if** $x_i \geq 0$, **then** $I = I \cup \{i\}$. //Non-zero columns.
      end for.
   e. Let $\rho = \phi$.

f. **for** $i \in I$ and $i > 0$ **do**
g. **if** not($L_i \cap I \neq \phi$ or $L_i \in \rho$), **then** recursive call Step f by replacing I with L_i and $\rho = \rho \cup L_i$.
h. **else** goto Step a.
end for.
i. $\Lambda_{min} = \Lambda_{min} \cup \{X\}$.
endfor. // Searching finished.
j. **Return** Λ_{min}.

Step 1 requires $O(\prod_{k=1}^{u} r_k)$ to generate all feasible X, since OFE-form is by re-arranging the original Ψ_X into u groups of alternative orders, and r_k is the total number of enumerations in the k^{th} group [11]. Step 2 performs d-MPs searching. It's computation complexity is $O(|B| \binom{z+d}{d-1})$.

5.3 Illustrative Examples

Figure 8 gives several complicated examples for exploration. Figure 8 shows (a) 25×32 (Vertices $\times$ Arcs) network, (b) 25×36 network, (c) 32×48 network, and (d) 32×48 network. Table 3 lists the results derived by the proposed algorithm. There are 26, 28, 28 and 26 6-MPs for Fig. 8a–d respectively, and 49,907, 34,784, 344,624 and 76,713 6-MPs derived from them, respectively. The total CPU seconds for them are 40, 388.002, 45, 900.912, 70, 988.449 and 144, 412.762 s, respectively. The results denotes the efficiency for applying the proposed algorithm.

6 Calculating the Union Probability for d-MPs

The network reliability is also the union probability for d-MPs. The most popular method to calculate the probability of union events is the inclusion-exclusion principle (IEP), which originates from the idea of Abraham de Moivre (1718) [22]. For example, Fig. 9 illustrates the principle of IEP for three events, and Eq. (6) explains how to calculate the probability of $\{A \cup B \cup C\}$.

$$\begin{aligned} \Pr\{A \cup B \cup C\} = & \Pr\{A\} + \Pr\{B\} + \Pr\{C\} \\ & - \Pr\{A \cap B\} - \Pr\{B \cap C\} \\ & - \Pr\{A \cap C\} + \Pr\{A \cap B \cap C\} \end{aligned} \tag{6}$$

According to IEP, it equals to the sum of the probability of all individual events, minus the probability of the intersection of every compound events, and plus the probability of the intersection of all events. In other words, IEP is equivalent to do the power set expansion. So, it is a general principle, and can be applied to any kind of events. However, the power set expansion makes the computation complexity

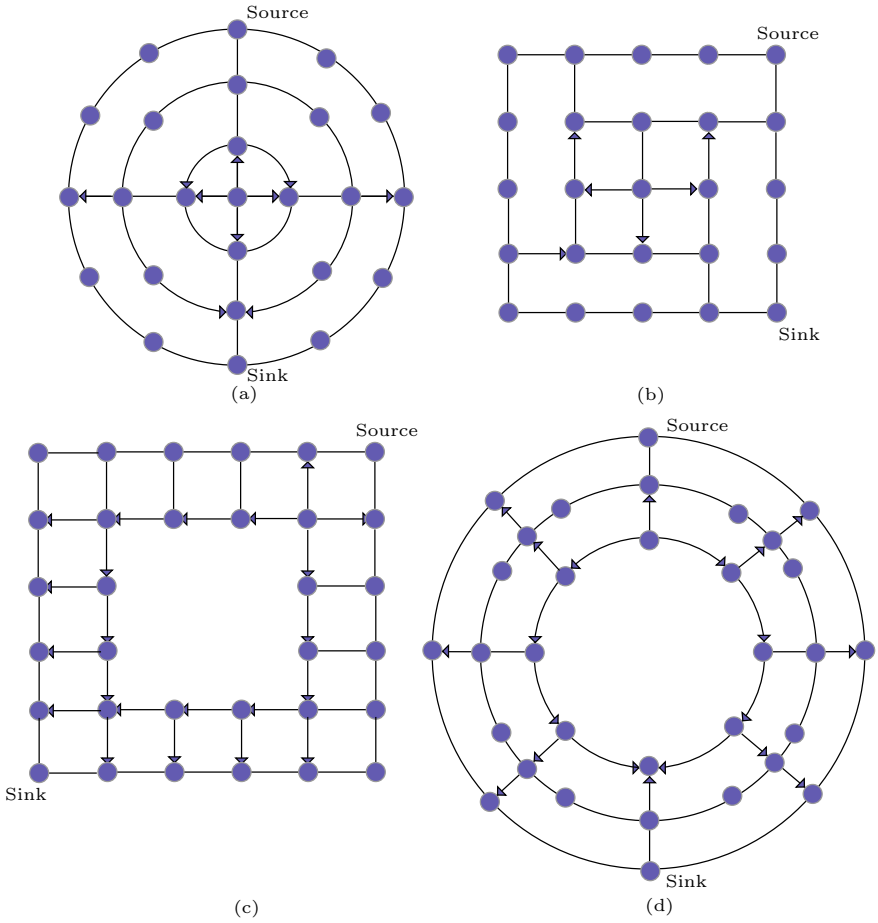
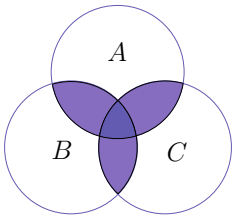


Fig. 8 Several complicated examples

Table 3 The CPU seconds for the complicated examples

(Vertices × Arcs)	# of MPs	# of 6-MPs	CPU (s)
(25 × 32)	26	49,907	40,388.002
(25 × 36)	28	34,784	45,900.912
(32 × 48)	28	344,624	70,988.449
(32 × 48)	26	76,713	144,412.762

Fig. 9 An union of 3 events



of IEP equals exactly to $O(2^Q)$ no matter what the events are. A much efficient method namely Recursive Inclusion-Exclusion Principle (RIEP) was constructed by rearranging Eq. (6) to its recursive form as Eq. (7). The computation complexity is also $O(2^Q)$ in the worse cases, but it usually has 10 times efficiency than IEP in the normal cases.

$$\begin{aligned}\Pr\{A \cup B \cup C\} &= \Pr\{A\} \\ &\quad + \Pr\{B\} - \Pr\{A \cap B\} \\ &\quad + \Pr\{C\} - \Pr\{B \cap C\} \\ &\quad - \Pr\{A \cap C\} + \Pr\{A \cap B \cap C\} \\ &= \Pr\{A\} \\ &\quad + \Pr\{B\} - \Pr\{A \cap B\} \\ &\quad + \Pr\{C\} - \Pr\{B \cap C \cup A \cap C\}\end{aligned} \tag{7}$$

This chapter presents an innovated reduction method for the recursive inclusion-exclusion principle (RRIEP) to calculate the probability of union events such that the calculation terms can be mostly eliminated, especially in the cases of events with monotonic property like network reliability applications. The network reliability is the probability of a live connection between source nodes and sink nodes. Such reliability can be calculated by means of minimal paths (MPs) [10, 12, 24] or minimal cuts (MCs) [1, 16, 30]. When the network is live, there are several minimum path vectors with respect to system state d, called d-MP, can be found. Then, the network reliability is the union probability of all these d-MPs.

6.1 Inclusion-Exclusion Principle

Let $\{A_1, A_2, \ldots, A_n\}$ be n events. Then, the probability of union events in terms of IEP is as follows.

$$\Pr\{\bigcup_{i=1}^{n} A_i\} = \sum_{k=1}^{n} \left((-1)^{k-1} \sum_{\substack{I \subset \{1,\ldots,n\} \\ |I|=k}} \Pr\{\bigcap_{i \in I} A_i\} \right). \tag{8}$$

6.2 Recursive Inclusion-Exclusion Principle

Figure 10 presents the concept of recursive inclusion-exclusion principle. Let $\pi(X_1, X_2, \ldots, X_n) \equiv \Pr\{X_1 \cup X_2 \cup \cdots \cup X_n\}$ be the function of probability of union events. According to the equivalent rearrangement, we have the following definition of recursive inclusion-exclusion principle.

$$\begin{aligned} \pi(A_1) &= \Pr\{A_1\}, \quad \text{for } n = 1, \\ \pi(A_1, A_2, \ldots, A_n) &= \sum_{i=1}^{n} (\Pr\{A_i\} \\ -\pi(A_1 \cap A_i, A_2 \cap A_i, \ldots, A_{i-1} \cap A_i)), & \\ &\text{for } n > 1, \end{aligned} \tag{9}$$

where $\pi(\bullet)$ is a recursive function.

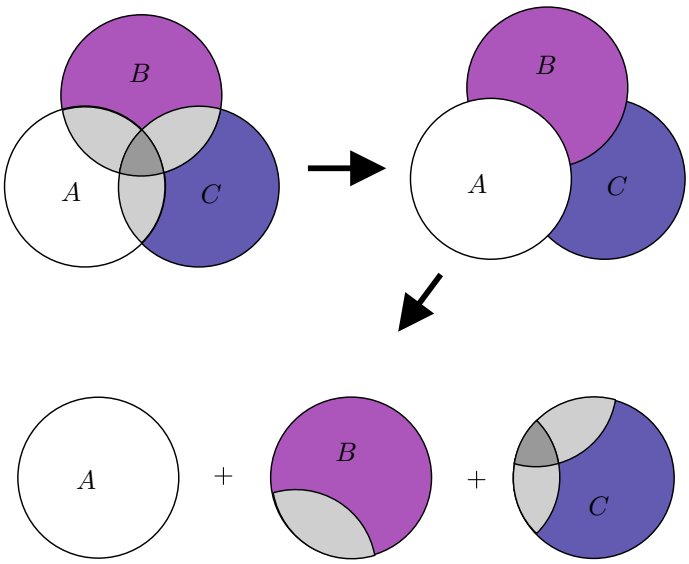


Fig. 10 An illustration of rearrangement

6.3 Reduced Recursive Inclusion-Exclusion Principle

One term is defined first.

Definition 1 (*Reduction*) $\{X_1, X_2, \ldots, X_m\}$ is a reduction of $\{Y_1, Y_2, \ldots, Y_n\}$, if for any $m, n \in \mathbb{N}$ and $m < n$ such that $\{X_1 \cup X_2 \cup \ldots \cup X_m\} = \{Y_1 \cup Y_2 \cup \ldots \cup Y_n\}$.

We have the following lemmas.

Lemma 10 *$\{X_1, X_2, ..., X_m\}$ is a reduction of $\{Y_1, Y_2, ..., Y_n\}$, then $\pi(X_1, X_2, \ldots, X_m) = \pi(Y_1, Y_2, \ldots, Y_n)$.*

Lemma 11 *Along with Lemma 10, the computation complexity of $\pi(X_1, X_2, \ldots, X_m)$ is less than that of $\pi(Y_1, Y_2, \ldots, Y_n)$.*

Let $\{B_1, B_2, \ldots, B_{m_i}\}$ be a reduction of $\{A_1 \cap A_i, A_2 \cap A_i, \ldots, A_{i-1} \cap A_i\}$. From Lemma 10, Eq. (9) can be rewritten as:

$$\begin{aligned} \pi(A_1) &= \Pr\{A_1\}, \quad \text{for } n = 1, \\ \pi(A_1, A_2, \ldots, A_n) &= \sum_{i=1}^{N} (\Pr\{A_i\} \\ -\pi(A_1 \cap A_i, A_2 \cap A_i, \ldots, A_{i-1} \cap A_i)), & \qquad (10) \\ &= \sum_{i=1}^{n} \Pr\{A_i\} \\ -\pi(B_1, B_2, \ldots, B_{m_i})), &\quad \text{for } n > 1, \end{aligned}$$

By Lemma 11, RRIEP is much efficient than RIEP.

In network reliability applications, it is easier to implement the reduction of union events. At first, we have the following property.

Property 10 (*Monotonicity*) Let $\bar{Q} = (q_1, q_2, \ldots, q_h)$ and $\bar{W} = (w_1, w_2, \ldots, w_h)$ be two capacity vectors in a network. Then, $\bar{Q}$ and $\bar{W}$ satisfy the following Monotonicity properties:

1. $\bar{Q} \leq \bar{W}$ iff $q_j \leq w_j$ for each $j = 1, 2, \ldots, h$.
2. $\bar{Q} < \bar{W}$ iff $\bar{Q} \leq \bar{W}$ and $q_j < w_j$ for at least one j.

6.4 Network Reliability Calculation

Given a demand d, the network reliability denoted by ω_d is the probability that the maximal flow is no less than d, i.e., $\omega_d = \Pr\{X | \varpi_X \geq d\}$. To calculate ω_d, it is advantageously to find the minimal vector (namely, d-MP) in the set $\{X | \varpi_X \geq d\}$.

Suppose there are totally h d-MPs for d: $X_1, X_2, \ldots, X_h$, the network reliability is equal to

$$\omega_d = \Pr\left\{\bigcup_{k=1}^{h}\{X|X \geq X_k\}\right\}, \tag{11}$$

Let $\bar{Q} \oplus \bar{W} \equiv \{g_i|g_i = \max(q_i, w_i), q_i \in \bar{Q}, w_i \in \bar{W}, \forall i\}$. The following RRIEP algorithm can be used to calculate ω_d.

Algorithm 4 RRIEP for network reliability calculation.

1. $dMP = \{X_1, X_2,\ldots,X_h\}$.
2. $\omega = 0$.
3. If $h = 0$, then return ω,
4. For $1 \leq i \leq |dMP|$ do
5. $\omega_2 = Pr\{X \geq X_i\}$.
6. For $1 \leq j < i$ do
7. $X_{j,i} = X_j \oplus X_i$.
   End for.
8. Let $E = \{X_{j,i}|1 \leq j < i\}$ and $W_l, W_k \in E$.
9. For $k \notin J$ and $1 \leq k \leq ||E||$ do // Reduction process.
10. For $l \notin J$ and $k < l \leq ||E||$ do
11. If $W_l \leq W_k$, then $J = J \cup \{k\}$ and go to Step 9.
    Else, if $W_l > W_k$, then $J = J \cup \{l\}$.
    End for.
12. $dMP^* = dMP^* \cup \{W_k\}$.
    End for.
13. Recursive call Step 1 with the new dMP^* and get the value ω_3.
14. $\omega = \omega + \omega_2 - \omega_3$.
    End for.
15. return ω.

The reduction implementation is started at Step 6.4, which greatly improves the efficiency of RRIEP algorithm. Since RRIEP can reduce the computation efforts in most of the time, the computation complexity in the worst cases still need $O(2^Q)$ in which no reduction can be performed at all.

6.5 *Illustrative Examples*

We randomly generate d-MP vectors of length 20, and compare IEP, RIEP, and RRIEP with efficiency. Table 4 gives the results. All benchmarks were implemented with PROLOG on a personal computer with the CPU of Intel Core i7-2600 CPU@ 3.40 GHz, 3.40 GHz and with 16 GB RAM.

Table 4 The comparison of CPU time (s) for benchmarks

# of d-MPs	IEP	RIEP	RRIEP
15	1.765	0.390	0.0003
18	19.891	1.891	0.0005
20	75.953	2.953	0.016
25	9256.750	9.359	0.016
30	NA*	43.343	0.015
35	NA	106.141	0.031
40	NA	416.579	0.063
45	NA	1112.687	0.078
50	NA	2144.000	0.093
55	NA	5924.610	0.110
59	NA	8362.469	0.109

*Not stopped over 1 day

7 Conclusion

This chapter presents the primitive steps in computation of network reliability, and some of the popular examples of network reliability. Although we present the most efficient method for all stages in the literature, they are subjected to change due to new method emerging. This chapter also give several numerical examples for exploration of their calculations, which help one to follow the algorithms to go through these examples.

Acknowledgements This work was supported in part by the National Science Council, Taiwan, Republic of China, under Grant No. MOST 107-2221-E-236-004-MY3.

References

1. Abel U, Bicker R (1982) Determination of all minimal cut-sets between a vertex pair in an undirected graph. IEEE Trans Reliab 31:167 – 171
2. Aggarwal KK, Chopra YC, Bajwa JS (1982) Capacity consideration in reliability analysis of communication systems. IEEE Trans Reliab 31:177–80
3. Aggarwal KK, Gupta JS, Misra KB (1975) A simple method for reliability evaluation of a communication system. IEEE Trans Commun 23:563–565
4. Aven T (1985) Reliability evaluation of multistate systems with multistate components. IEEE Trans Reliab 34:473–479
5. Chen SG (2014) Reduced recursive inclusion-exclusion principle for the probability of union events. In: 2014 IEEE international conference on industrial engineering and engineering management. Selangor, Malaysia, pp 1–3
6. Chen SG (2014) Reduced recursive sum of disjoint product in network reliability. In: 2014 the 20th ISSAT international conference on reliability and quality in design. Seattle, Washington, U.S.A., pp 170–173
7. Chen SG (2015) Search for all MCs with backtracking. Int J Reliab Qual Perform 6(2):101–106

8. Chen SG (2016) Optimal re-arrangement in fast enumeration for integer programming problems. In: 2016 IEEE international conference on industrial engineering and engineering management, pp 1255–1258
9. Chen SG (2019) A way from adjacency matrix to linked path structure. In: Proceedings of the 25th ISSAT international conference on reliability and quality in design, pp 20–23
10. Chen SG, Lin YK (2012) Search for all minimal paths in a general large flow network. IEEE Trans Reliab 61(4):949–956
11. Chen SG, Lin YK (2016) Searching for d-MPs with fast enumeration. J Comput Sci 17:139–147
12. Colbourn CJ (1987) The combinatorics of network reliability. Oxford University Press, UK
13. Doulliez P, Jamoulle J (1972) Transportation networks with random arc capacities. Recherche Operationnelle 3:45–60
14. Ford LR, Fulkerson DR (1962) Flows in networks. Princeton University Press, NJ
15. Jane CC, Lin JS, Yuan J (1993) On reliability evaluation of a limited-flow network in terms of minimal cutsets. IEEE Trans Reliab 42:354–361, 368
16. Jasmon GB, Foong KW (1987) A method for evaluating all the minimal cuts of a graph. IEEE Trans Reliab 36:538 – 545
17. Lee SH (1980) Reliability evaluation of a flow network. IEEE Trans Reliab 29:24–26
18. Lin JS, Jane CC, Yuan J (1995) On reliability evaluation of a capacitated-flow network in terms of minimal pathsets. Networks 25:131–138
19. Lin YK (2001) On reliability evaluation of a stochastic-flow network in terms of minimal cuts. J Chin Inst Ind Eng 18:49–54
20. Lin YK (2001) A simple algorithm for reliability evaluation of a stochastic-flow network with node failure. Comput Oper Res 28(13):1277–1285
21. Lin YK, Chen SG (2019) An efficient searching method for minimal path vectors in multi-state networks. Ann Oper Res. 1–12. https://doi.org/10.1007/s10479-019-03158-6
22. Roberts FS, Tesman B (2009) Applied Combinatorics, 2nd edn. CRC Press
23. Schrijver A (2002) On the history of the transportation and maximum flow problems. Math Program 91(3):437–445
24. Shen Y (1995) A new simple algorithm for enumerating all minimal paths and cuts of a graph. Microelectron Reliab 35:973–976
25. Wilson RJ (1972) Introduction to graph theory. Prentice-Hall
26. Xue J (1985) On multistate system analysis. IEEE Trans Reliab 34(4):329–337
27. Yeh WC (2001) A simple algorithm to search for all d-MPs with unreliable nodes. Reliab Eng Syst Saf 73:49–54
28. Yeh WC (2001) A simple approach to search for all d-MCs of a limited-flow network. Reliab Eng Syst Saf 71:15–19
29. Yeh WC (2002) A simple method to verify all d-minimal path candidates of a limited-flow network and its reliability. Int J Adv Manuf Technol 20(1):77–81
30. Yeh WC (2006) A simple algorithm to search for all MCs in networks. Eur J Oper Res 174:1694–1705
31. Yeh WC (2007) A simple heuristic algorithm for generating all minimal paths. IEEE Trans Reliab 56(3):488–494
32. Zuo MJ, Tian Z, Huang HZ (2007) An efficient method for reliability evaluation of multistate networks given all minimal path vectors. IIE Trans 39:811–817

Shin-Guang Chen received his B.Sc. in Computer Engineering, NCTU, Taiwan and his Ph.D. in IE, NCTU, Taiwan. He has been with the Institute of Industrial Management at Tungnan University, Taiwan, where he holds the rank of Distinguished Professor. His research interests are in management, reliability, ERPS and sustainable energy technologies. His recent work involves problems related to the analysis of flow networks. He has published papers in IJPR, IJIE, ESWA, JORSJ, CSTM, IJOR, JCIIE, IJRQP, APEN, IJPE, Omega, IEEE T Reliab, APM, JIMO, APJOR, AOR, etc. He is a member of IEEE, IEEE Reliab, POMS, CIIE, CERPS and CSQ.