Journal of Computational Science 22 (2017) 119–125

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

A maximal flow method to search for *d*-MPs in stochastic-flow networks

Yi-Kuei Lin [a], Shin-Guang Chen [b,c,*]

[a] Department of Industrial Management, National Taiwan University of Science and Technology, Taipei 106, Taiwan
[b] Department of Industrial Management, Tungnan University, New Taipei City 222, Taiwan
[c] Post-Baccalaureate Bachelor Degree Program in Integrated Clinical Interpretation based on Laboratory Data, Kaohsiung Medical University, Kaohsiung, Taiwan

ARTICLE INFO

Article history:
Received 25 April 2017
Received in revised form 9 July 2017
Accepted 14 September 2017
Available online 21 September 2017

Keywords:
Network reliability
d-MPs
Minimal path
Fast enumeration
Maximal flow

ABSTRACT

Since 1954, the maximal flow problems have gained much attention in the world. They are also extended to many other fields for applications. For example, the definition of a network reliability is just to evaluate the probability of a live connection between the source node and the sink node such that the maximal flow of the network is no less than the demand *d*. The popular methods in the literature to evaluate the network reliability are mostly through minimal paths (MP) or minimal cuts (MC) of the network. One of them is the three-stages method: (a) searching for all MPs/MCs; (b) searching for all *d*-MPs (the minimal system states for *d* via MP)/*d*-MCs (the maximal system states for *d* via MC); (c) calculating union probability from these *d*-MPs/*d*-MCs. We found that a creative innovation in solving the maximal flow problems may have benefits in the evaluation of network reliability. Based on this idea, this paper proposes a new approach to tackle the problem of searching for all *d*-MPs by given MPs. The comparisons with the well known algorithm are made for benchmarking. More complicated examples are also examined for illustrative purposes.

1. Introduction

Since 1954, the maximal flow problems have gained much attention in the world [24]. They are also extended to many other fields for applications [14]. For example, the definition of a network reliability is just to evaluate the probability of a live connection between the source node and the sink node such that the maximal flow of the network is no less than the demand *d* [18]. Aggarwal et al. firstly discussed the reliability problem without flow in a binary-state network [3]. Lee [18] 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. [19] illustrated the reliability calculation of a SFN in terms of MPs or minimal cuts (MC)s [16]. They also setup the three-stages in these calculations: (a) searching for all MPs [12,25,10]/MCs [1,17,30,8]; (b) searching for all *d*-MPs [21,27]/*d*-MCs [28] from these MPs/MCs; (c) calculating union probability from these *d*-MPs [31,7,6]/*d*-MCs [12,20]. Lin [21] greatly simplified the three-stages algorithm and developed more simple and efficient algorithm for the reliability evaluation of a general SFN.

There have been many innovative or for special cases algorithms developed for *d*-MP search since then [27,29,22,23]. Chen and Lin [11] started to improve the efficiency of the three-stages algorithm with the technology of fast enumeration [5]. By inspecting the definition of network reliability, there are some potential benefits in tackling *d*-MP searching problems if we can find an innovative way to improve the efficiency of solving maximal flow problems in a SFN. Forghani-elahabad and Mahdavi-Amiri [15] presented a textbook approach [4] of maximal flow algorithm, and shown the benefits in solving the *d*-MPs searching problems by following the steps of Lin's algorithm [21].

This paper proposes a novel algorithm to solve the maximal flow problems, and combines with fast enumeration approach to tackle the *d*-MPs searching problems. This paper also makes comparisons with the well known algorithms for benchmarking. More complicated examples are also examined for illustrative purposes. The remainder of the work is described as follows. The mathematic pre-

* Corresponding author at: Department of Industrial Management, Tungnan University, New Taipei City 222, Taiwan.
E-mail addresses: yklin@mail.ntust.edu.tw (Y.-K. Lin), bobchen@mail.tnu.edu.tw (S.-G. Chen).

http://dx.doi.org/10.1016/j.jocs.2017.09.009
1877-7503/

liminaries for the approach are presented in Section 2. The proposed algorithm is given in Section 3. Some benchmarks are examined and compared in Section 4. Finally, Section 5 draws the conclusion of this paper.

2. Preliminaries

Let $G = (E, V)$ be a SFN, where $E = \{e_i | 1 \le i \le n\}$ is the set of edges, V is the set of nodes. Let $T = (t_1, t_2, \ldots, t_n)$ be a vector with t_i (an integer) being the maximal capacity of e_i. The network G 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 e_i is an integer-valued random variable which takes values from the set $\{0, 1, 2, \ldots, t_i\}$ 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 $p_1, p_2, \ldots, p_z$ are totally the MPs from the source to the sink. Thus, the network model 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 e_i and y_j denotes the current flow on p_j. Then, Y is feasible if and only if

$$\sum_{j=1}^{z} \{y_j | e_i \in p_j\} \le t_i, \quad \text{for} \quad i = 1, 2, \ldots, n. \tag{1}$$

This constraint describes that the total flow through e_i cannot exceed the maximal capacity of e_i. Then, the set of Y as $\kappa_T \equiv \{Y | Y$ is feasible under $T\}$, and Ψ_Y as the set of formulas in Eq. (1).

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

$$\sum_{j=1}^{z} \{y_j | e_i \in p_j\} \le x_i, \quad \text{for} \quad i = 1, 2, \ldots, n. \tag{2}$$

For clarity, let $\kappa_X = \{Y | Y$ 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 $Y \in \kappa_T$ 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 $\mathbf{Y} = \{Y | Y \in \kappa_T$ and satisfies Eq. (3)$\}$. We have the following lemmas [21].

Lemma 2.1. *If X is a d-MP for d, then there is an* $Y \in \mathbf{Y}$ *such that*

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

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

Lemma 2.2. *Λ_{min} is the set of d-MPs for d.*

2.2. Evaluation of reliability

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

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

which can be calculated by reduced recursive inclusion-exclusion principle (RRIEP) [6].

3. Searching for *d*-MPs

Let $L_i = (l_1, l_2, \ldots, l_n)$ be a unit vector with $l_i = 1$ and $l_j = 0$ for all $j \ne i$. We proof the following lemmas.

Lemma 3.1. *If X is a d-MP, then $\varpi_{X-L_i} < d$ for all i with $x_i > 0$.*

Proof. Let W be a d-MP and $W = X - L_k$ for some k. Since $\varpi_W = \varpi_{X-L_k} = d$ implies that $X < W$, this conflicts that W is a d-MP. □

3.1. Evaluation of maximal flow

The maximal flow problem is a classical optimization problem since 1954 [24]. The famous Ford–Fulkerson method [14] was a popular method to evaluate the maximal flow of a network. The complexity of Ford–Fulkerson method is $O(n\lambda\left(d - 1_{z+d}\right))$ where λ is the maximal value of flows. However, their method cannot converge when the flow is not integer. Edmonds–Karp algorithm [13] specialized Ford–Fulkerson method to maximal integer-flow problems, and gets a similar complexity. We developed a more efficient algorithm than the above ones and can evaluate either integer or real flows. Let $R = (r_1, r_2, \ldots, r_n)$ be the vector of the assigned capacities for edges in G, and $\delta_j = \min\{x_i - r_i | e_i \in p_j\}$ be the maximal flow in p_j under X. We have the following lemmas.

Lemma 3.2. *Let $\delta_{(j)}$ be the descending ordered series such that*

$$\delta_{(1)} \ge \delta_{(2)} \ge \cdots \ge \delta_{(\beta)}. \tag{6}$$

Then, the flows assigned to R under such sequence are maximal in total.

Proof. Suppose $k < z$ and only $\delta_{(1)}$ to $\delta_{(k)}$ are assigned to R. If we have another assignment: $\delta_{(1)}, \delta_{(2)}, \ldots, \delta_{(k-1)}, \delta_{(k+1)}$, which are assigned to R', then $R' < R$ because $\delta_{(k+1)} < \delta_{(k)}$ by definition. □

3.2. Fast enumeration of flows

We recalled that $\mathbf{Y}$ can be enumerated by fitting the formulas in Ψ_Y and Eq. (3). Then, X_Y can be built by $Y \in \mathbf{Y}$ via Eq. (4). Next, $\Lambda = \{X_Y | Y \in \mathbf{Y}\}$ is constructed for all X_Y. Finally, $\Lambda_{min} = \{X | X$ is a lower boundary vector in $\Lambda\}$ is built. Then, Λ_{min} is the set of d-MPs for d. Let Ξ_{Ψ_Y} be the fast enumeration form (FE-form) of formulas in Ψ_Y [11]. We can get the fast enumeration of flows in these formulas.

3.3. Algorithm

Let $J = \{j | X_j \notin \Lambda_{min}\}$ and $s_j = \min\{t_i | e_i \in p_j\}$. The new algorithm via maximal flow evaluation to search for d-MPs by given MPs is as follows.

Algorithm. Searching for d-MPs for d by given MPs.

Step 1. Find the feasible flow vector $Y = (y_1, y_2, \ldots, y_z)$ satisfying both capacity and demand constraints.

(a) **fast enumerate** $0 \le y_j \le s_j$ with Ξ_{Ψ_Y} **do**
(b) **if** y_j satisfies $\sum_{j=1}^{z} y_j = d$,

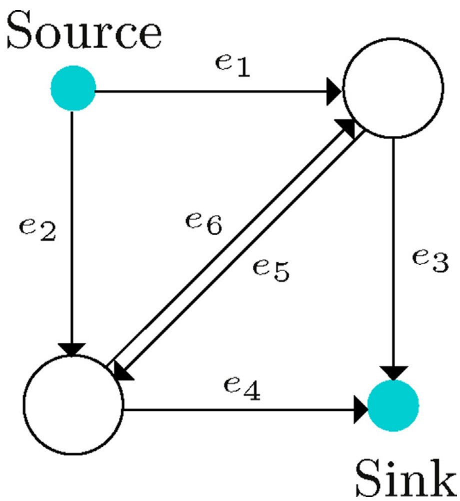

Fig. 1. The white-stone bridge network [12].

then $\mathbf{Y} = \mathbf{Y} \cup \{Y\}$.
end enumerate.

Step 2. Directly generate the set Λ_{min} from $\mathbf{Y}$.

(a) **for** Y in $\mathbf{Y}$ **do**
(b) $x_i = \sum_{j=1}^{z} \{y_j | e_i \in p_j\}$ for $i = 1, 2, \ldots, n$.
(c) **for** $1 \le i \le n$ and $x_i > 0$ **do**
(d) $R = \mathbf{0}$ // Initialize R to zero vector.
(e) $x_i = x_i - 1$.
(f) $\delta_j = \min\{x_k - r_k | e_k \in p_j\}$ for $j = 1, 2, \ldots, z$.
(g) **CallMergeSort** on $\{\delta_j | 1 \le j \le z\}$ and get the sorted set $\{\delta_{(j)} | 1 \le j \le z\}$
(h) **for** $1 \le j \le z$ **do**
(i) $\delta_{(j)} = \min\{x_k - r_k | e_k \in p_(j)\}$
(j) $r_k = r_k + \delta_{(j)}$, for $e_k \in p_(j)$.
endfor.
(k) **if** $\sum_{k=1}^{z} \delta_k \ge d$, **then** go to Step 2a.
endfor.
(l) $\Lambda_{min} = \Lambda_{min} \cup \{X\}$.
endfor.
(m) **Return** Λ_{min}.

The steps from 2c to 2k are the new steps for maximal-flow evaluation to calculate the maximal flow of the network under the current states, where the merge sort in Step 2g can be referred to [9]. It takes a complexity of $O(z\log(z))$.

Step 1 requires $O(\prod_{k=1}^{u} q_k)$ to generate all feasible Y, since FE-form is by re-arranging the original Ψ_F into u groups of alternative orders, and q_k is the total number of enumerations in the kth group [11]. Step 2 including the maximal flow calculation requires $O(nz\log(z)\left(d - 1_{z+d}\right))$.

4. Benchmarking and examples

At first, we give a step-by-step illustration of the proposed algorithm with the example of Fig. 1, which presents a simple white-stone network [12]. 4 MPs are found: $p_1 = \{a_1, a_3\}$, $p_2 = \{a_2, a_4\}$, $p_3 = \{a_1, a_5, a_4\}$, and $p_4 = \{a_2, a_6, a_3\}$. The corresponding flows are $Y = \{y_1, y_2, y_3, y_4\}$, respectively. Table 1 is the capacity distributions for the six edges. The required demand is 5.

From Table 1, we have $T = (6, 6, 5, 6, 5, 6)$, and $s_1 = 5$, $s_2 = 6$, $s_3 = 5$, $s_4 = 6$. Step 1 gives the FE-form of the flow enumerations along with the demand constraint.

Table 1
The capacity distributions for the six edges of Fig. 1.

Arcs	The capacity distributions μ_i						
	0	1	2	3	4	5	6
e_1	0.000	0.000	0.001	0.011	0.083	0.337	0.568
e_2	0.000	0.000	0.000	0.004	0.042	0.264	0.690
e_3	0.000	0.000	0.003	0.039	0.262	0.696	0
e_4	0.000	0.000	0.000	0.006	0.055	0.292	0.647
e_5	0.000	0.001	0.017	0.111	0.372	0.498	0
e_6	0.000	0.000	0.001	0.011	0.083	0.337	0.568

Step 1. Find the feasible flow vector $Y = (y_1, y_2, \ldots, y_4)$ satisfying both capacity and demand constraints.

(a) **for** $0 \le y_3 \le 5$ **do** // The FE-form Ξ_{Ψ_Y}.
(b) **if** $y_3 \le 5$, **then**
(c) **for** $0 \le y_4 \le 6$ **do**
(d) **if** $y_4 \le 6$, **then**
(e) **for** $0 \le y_1 \le 5$ **do**
(f) **if** $y_3 + y_1 \le 6$ and $y_4 + y_1 \le 5$, **then**
(g) **for** $0 \le y_2 \le 6$ **do**
(h) **if** $y_3 + y_2 \le 6$ and $y_4 + y_2 \le 6$ and $y_1 + y_2 + y_3 + y_4 = 5$, **then**
(i) $\mathbf{Y} = \mathbf{Y} \cup \{Y\}$.
endif
endfor
endif
endfor
endif
endfor
endif
endfor

Then, the step-by-step exploration is as follows:
Step 1. Find the feasible flow vector $Y = (y_1, y_2, \ldots, y_4)$ satisfying both capacity and demand constraints.

(a) $y_3 = 0$
(b) $y_3 \le 5$ is true, then
(c) $y_4 = 0$
(d) $y_4 \le 6$ is true, then
(e) $y_1 = 0$
(f) $y_3 + y_1 \le 6$ and $y_4 + y_1 \le 5$ are true, then
(g) $y_2 = 0$
(h) $y_3 + y_2 \le 6$ and $y_4 + y_2 \le 6$ and $y_1 + y_2 + y_3 + y_4 = 5$ are false, go back to Step (g).
(i) $y_2 = 1$

1 · · ·
At the end of the step: $\mathbf{Y}$ = {(0, 5, 0, 0), (1, 4, 0, 0), (2, 3, 0, 0), (3, 2, 0, 0), (4, 1, 0, 0), (5, 0, 0, 0), (0, 4, 0, 1), (1, 3, 0, 1), (2, 2, 0, 1), (3, 1, 0, 1), (4, 0, 0, 1), (0, 3, 0, 2), (1, 2, 0, 2), (2, 1, 0, 2), (3, 0, 0, 2), (0, 2, 0, 3), (1, 1, 0, 3), (2, 0, 0, 3), (0, 1, 0, 4), (1, 0, 0, 4), (0, 0, 0, 5), (0, 4, 1, 0), (1, 3, 1, 0), (2, 2, 1, 0), (3, 1, 1, 0), (4, 0, 1, 0), (0, 3, 1, 1), (1, 2, 1, 1), (2, 1, 1, 1), (3, 0, 1, 1), (0, 2, 1, 2), (1, 1, 1, 2), (2, 0, 1, 2), (0, 1, 1, 3), (1, 0, 1, 3), (0, 0, 1, 4), (0, 3, 2, 0), (1, 2, 2, 0), (2, 1, 2, 0), (3, 0, 2, 0), (0, 2, 2, 1), (1, 1, 2, 1), (2, 0, 2, 1), (0, 1, 2, 2), (1, 0, 2, 2), (0, 0, 2, 3), (0, 2, 3, 0), (1, 1, 3, 0), (2, 0, 3, 0), (0, 1, 3, 1), (1, 0, 3, 1), (0, 0, 3, 2), (0, 1, 4, 0), (1, 0, 4, 0), (0, 0, 4, 1), (0, 0, 5, 0)}.
Step 2. Directly generate the set Λ_{min} from $\mathbf{Y}$.
(a) $Y = (0, 5, 0, 0)$
(b) $X = (0, 5, 0, 5, 0, 0)$
(c) $i = 2$ and $x_2 > 0$
(d) $R = (0, 0, 0, 0, 0, 0, 0)$
(e) $x_2 = 5 - 1 = 4$
(f) $\delta_1 = 0$, $\delta_2 = 4$, $\delta_3 = 0$, $\delta_4 = 0$

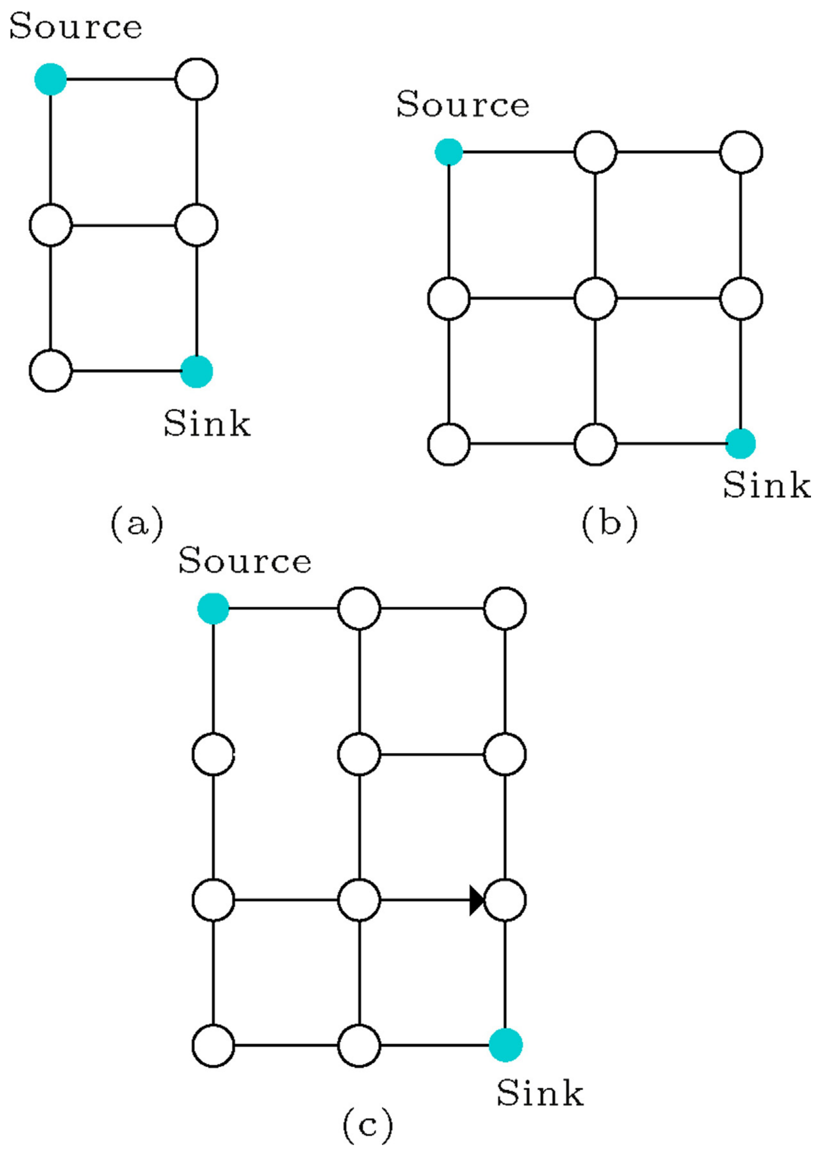

Fig. 2. Three rectangular networks.

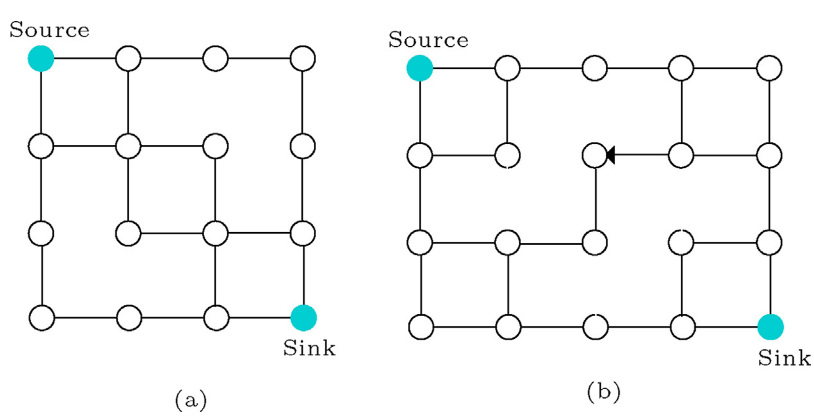

Fig. 3. More complicated networks.

(g) **CallMergeSort** on $\{\delta_j | 1 \le j \le z\}$ and get the sorted set
$\delta_{(1)} = 4$, $\delta_{(2)} = 0$, $\delta_{(3)} = 0$, $\delta_{(4)} = 0$
(h) $j = 1$
(i) $\delta_{(1)} = 4$
(j) $r_1 = 0$, $r_2 = 4$, $r_3 = 0$, $r_4 = 4$, $r_5 = 0$, $r_6 = 0$
(h) $j = 2$
(i) $\delta_{(2)} = 0$
(j) $r_1 = 0$, $r_2 = 4$, $r_3 = 0$, $r_4 = 4$, $r_5 = 0$, $r_6 = 0$
(h) $j = 3$
(i) $\delta_{(3)} = 0$

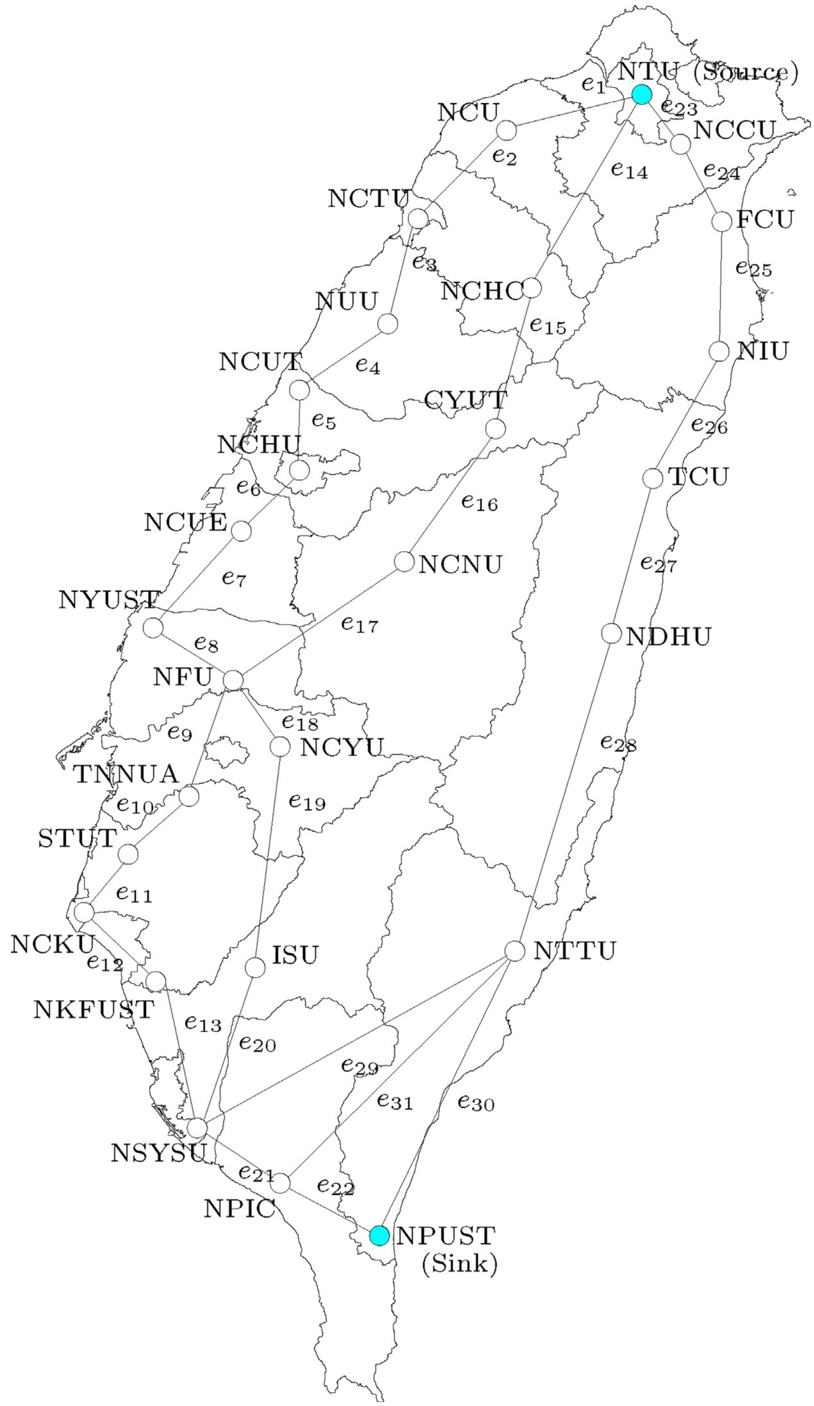

Fig. 4. A part of TANET network [10].

(j) $r_1 = 0, r_2 = 4, r_3 = 0, r_4 = 4, r_5 = 0, r_6 = 0$
(h) $j = 4$
(i) $\delta_{(4)} = 0$
(j) $r_1 = 0, r_2 = 4, r_3 = 0, r_4 = 4, r_5 = 0, r_6 = 0$
(k) $\sum_{k=1}^{z} \delta_k = 4 \geq 5$ is false.
(c) $i = 4$ and $x_4 > 0$
(d) $R = (0, 0, 0, 0, 0, 0, 0)$
(e) $x_4 = 5 - 1 = 4$
(f) $\delta_1 = 0, \delta_2 = 4, \delta_3 = 0, \delta_4 = 0$
(g) **CallMergeSort** on $\{\delta_j | 1 \leq j \leq z\}$ and get the sorted set $\delta_{(1)} = 4, \delta_{(2)} = 0, \delta_{(3)} = 0, \delta_{(4)} = 0$
(h) $j = 1$
(i) $\delta_{(1)} = 4$
(j) $r_1 = 0, r_2 = 4, r_3 = 0, r_4 = 4, r_5 = 0, r_6 = 0$
(h) $j = 2$
(i) $\delta_{(2)} = 0$
(j) $r_1 = 0, r_2 = 4, r_3 = 0, r_4 = 4, r_5 = 0, r_6 = 0$
(h) $j = 3$
(i) $\delta_{(3)} = 0$
(j) $r_1 = 0, r_2 = 4, r_3 = 0, r_4 = 4, r_5 = 0, r_6 = 0$
(h) $j = 4$
(i) $\delta_{(4)} = 0$
(j) $r_1 = 0, r_2 = 4, r_3 = 0, r_4 = 4, r_5 = 0, r_6 = 0$
(k) $\sum_{k=1}^{z} \delta_k = 4 \geq 5$ is false.

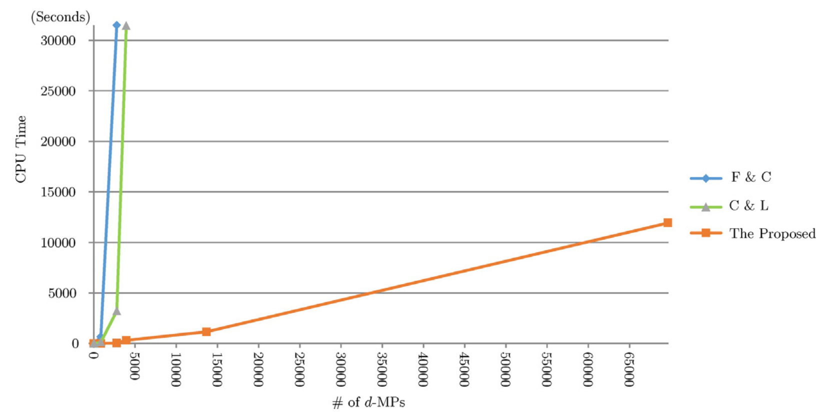

Fig. 5. The plots of CPU times vs. # of 5-MPs.

Table 2
Benchmarking between the well-known algorithms and the proposed algorithm.

Figures	# of MPs	# of 5-MPs	CPU time (s)		
(Nodes × edges)			F &M	C &L	The proposed
(6 × 7)	4	36	0.032	0.031	0.0031
(9 × 12)	12	859	655.527	123.759	4.243
(12 × 16)	16	2782	NA[a]	3214.648	45.006

[a] Not completed within one day.

(l) $\Lambda_{min} = \Lambda_{min} \cup \{X = (0, 5, 0, 5, 0, 0)\}$.

1. ...

At the end of the step: $\Lambda_{min} = \{X_1 = (0, 5, 0, 5, 0, 0)$, $X_2 = (1, 4, 1, 4, 0, 0)$, $X_3 = (2, 3, 2, 3, 0, 0)$, $X_4 = (3, 2, 3, 2, 0, 0)$, $X_5 = (4, 1, 4, 1, 0, 0)$, $X_6 = (5, 0, 5, 0, 0, 0)$, $X_7 = (0, 5, 1, 4, 0, 1)$, $X_8 = (1, 4, 2, 3, 0, 1)$, $X_9 = (2, 3, 3, 2, 0, 1)$, $X_{10} = (3, 2, 4, 1, 0, 1)$, $X_{11} = (4, 1, 5, 0, 0, 1)$, $X_{12} = (0, 5, 2, 3, 0, 2)$, $X_{13} = (1, 4, 3, 2, 0, 2)$, $X_{14} = (2, 3, 4, 1, 0, 2)$, $X_{15} = (3, 2, 5, 0, 0, 2)$, $X_{16} = (0, 5, 3, 2, 0, 3)$, $X_{17} = (1, 4, 4, 1, 0, 3)$, $X_{18} = (2, 3, 5, 0, 0, 3)$, $X_{19} = (0, 5, 4, 1, 0, 4)$, $X_{20} = (1, 4, 5, 0, 0, 4)$, $X_{21} = (0, 5, 5, 0, 0, 5)$, $X_{22} = (1, 4, 0, 5, 1, 0)$, $X_{23} = (2, 3, 1, 4, 1, 0)$, $X_{24} = (3, 2, 2, 3, 1, 0)$, $X_{25} = (4, 1, 3, 2, 1, 0)$, $X_{26} = (5, 0, 4, 1, 1, 0)$, $X_{27} = (2, 3, 0, 5, 2, 0)$, $X_{28} = (3, 2, 1, 4, 2, 0)$, $X_{29} = (4, 1, 2, 3, 2, 0)$, $X_{30} = (5, 0, 3, 2, 2, 0)$, $X_{31} = (3, 2, 0, 5, 3, 0)$, $X_{32} = (4, 1, 1, 4, 3, 0)$, $X_{33} = (5, 0, 2, 3, 3, 0)$, $X_{34} = (4, 1, 0, 5, 4, 0)$, $X_{35} = (5, 0, 1, 4, 4, 0)$, $X_{36} = (5, 0, 0, 5, 5, 0)\}$.

The example is running on a PC with CPU of Intel Core i7 4790 @ 3.60 GHz, 3.60 GHz and 16 GB RAM. Step 1 takes 0.00001 s CPU time to get **Y**. Step 2 takes 0.000001 s CPU time to get 5-MPs. There are totally 36 of 5-MPs found. The final network reliability for this example is $\omega_5 = 0.9999975$.

4.1. Benchmarking

We use three rectangular networks of different sizes to do the benchmarking comparisons with F & M's algorithm [15], C & L's algorithm [11], and the proposed algorithm. The demand is 5 for all these cases. The capacities for all these edges are set to 5. Fig. 2 presents these networks.

All the algorithms were coded by PROLOG running on the same PC denoted above. Table 2 gives the comparisons between them. There are 4 MPs found in Fig. 2(a), 12 MPs found in Fig. 2(b), and 16 MPs found in Fig. 2(c). They can be derived 36 of 5-MPs for Fig. 2(a), 859 of 5-MPs for Fig. 2(b), and 2782 of 5-MPs for Fig. 2(c), respectively. The CPU times for the corresponding algorithms are also listed in the table. It shows that the proposed algorithm has better performance than the other two.

Table 3
More complicated networks explored by the proposed algorithm.

(Nodes × arcs)	# of MPs	# of 5-MPs	CPU time (s)
(16 × 20)	20	3948	307.328
(20 × 25)	32	69,674	11,938.279
(27 × 31)	19	13,679	1153.266
(TANET)			

4.2. More examples

More complicated networks are explored in Fig. 3. Fig. 3(a) is a network of size (16 × 20) (Nodes × arcs), and (b) is a network of size (20 × 25). Table 3 is the results derived by the proposed algorithm. They have 20 and 32 MPs found respectively, and can be derived 3948 and 69,674 of 5-MPs respectively. They spend CPU times 307.328 and 11,938.279 s respectively. Fig. 4 shows a part of TANET (Taiwan Academic Network) network connected with Universities and Colleges around the island [10]. TANET is a network in Taiwan for academic purposes. They have a demand of 5 (in 10^3 GB), and all the capacities of edges are assumed 5 (in 10^3 GB) for illustrative purposes. There are 19 MPs found in TANET and is derived 13,679 of 5-MPs, and spend CPU time 1153.266 seconds to derive them. Fig. 5 shows the trend of CPU times for all these cases compared with the well-known algorithms.

5. Conclusion

This paper proposes a simple algorithm to search for d-MPs with given MPs in terms of fast enumeration (FE) and maximal flow evaluation. Network theories have been widely applied to solve many real-life applications. One of the popular methods for network reliability evaluation is the three-stages method. This paper presents a novel maximal flow searching method along with FE to improve the efficiency of reliability evaluation. Several benchmarking comparisons are conducted to explore the benefits between the well-known algorithms and the proposed one. The results show that the proposed algorithm can significantly improve the efficiency in searching for d-MPs in SFN.

Future researches are encouraged to inspect multi-commodities, multi-terminal or multistate networks to solve more real-life problems.

Acknowledgments

This work was supported in part by the Ministry of Science and Technology, Taiwan, Republic of China, under Grant Nos. MOST 105-2221-E-011-101-MY3 and MOST 105-2221-E-236-004.

References

[1] U. Abel, R. Bicker, Determination of all minimal cut-sets between a vertex pair in an undirected graph, IEEE Trans. Reliab. R-31 (1982) 167–171.
[2] K.K. Aggarwal, Y.C. Chopra, J.S. Bajwa, Capacity consideration in reliability analysis of communication systems, IEEE Trans. Reliab. 31 (1982) 177–180.
[3] K.K. Aggarwal, J.S. Gupta, K.B. Misra, A simple method for reliability evaluation of a communication system, IEEE Trans. Commun. 23 (1975) 563–565.
[4] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, Network Flows – Theory, Algorithms, and Applications, Prentice-Hall, NJ, 1993.
[5] S.G. Chen, Efficiency improvement in explicit enumeration for integer programming problems, in: 2013 IEEE International Conference on Industrial Engineering and Engineering Management, Bangkok, Thailand, 2013, pp. 98–100.
[6] S.G. Chen, Reduced recursive inclusion–exclusion principle for the probability of union events, in: 2014 IEEE International Conference on Industrial Engineering and Engineering Management, Selangor, Malaysia, 2014, pp. 1–3.
[7] S.G. Chen, Reduced recursive sum of disjoint product in network reliability, in: 2014 the 20th ISSAT International Conference on Reliability and Quality in Design, Seattle, Washington, USA, 2014, pp. 170–173.
[8] S.G. Chen, Search for all MCs with backtracking, Int. J. Reliab. Qual. Perform. 6 (2015) 101–106.
[9] S.G. Chen, Sort in vector space, in: 2016 the 22th ISSAT International Conference on Reliability and Quality in Design, 2016, pp. 244–248.
[10] S.G. Chen, Y.K. Lin, Search for all minimal paths in a general large flow network, IEEE Trans. Reliab. 61 (2012) 949–956.
[11] S.G. Chen, Y.K. Lin, Searching for *d*-MPs with fast enumeration, J. Comput. Sci. 17 (2016) 139–147.
[12] C.J. Colbourn, The Combinatorics of Network Reliability, Oxford University Press, UK, 1987.
[13] J. Edmonds, R.M. Karp, Theoretical improvements in algorithmic efficiency for network flow problems, J. ACM 19 (1972) 248–264.
[14] L.R. Ford, D.R. Fulkerson, Flows in Networks, Princeton University Press, NJ, 1962.
[15] M. Forghani-elahabad, N. Mahdavi-Amiri, A simple algorithm to find all minimal path vectors to demand level *d* in a stochastic-flow network, in: 2013 the 5th Iranian Conference on Applied Mathematics, Bu-Ali Sina University, Hamedan, Iran, 2013, pp. 1–4.
[16] C.C. Jane, J.S. Lin, J. Yuan, On reliability evaluation of a limited-flow network in terms of minimal cutsets, IEEE Trans. Reliab. 42 (1993) 354–361, 368.
[17] G.B. Jasmon, K.W. Foong, A method for evaluating all the minimal cuts of a graph, IEEE Trans. Reliab. R-36 (1987) 538–545.
[18] S.H. Lee, Reliability evaluation of a flow network, IEEE Trans. Reliab. 29 (1980) 24–26.
[19] J.S. Lin, C.C. Jane, J. Yuan, On reliability evaluation of a capacitated-flow network in terms of minimal pathsets, Networks 25 (1995) 131–138.
[20] Y.K. Lin, On reliability evaluation of a stochastic-flow network in terms of minimal cuts, J. Chin. Inst. Ind. Engrs. 18 (2001) 49–54.
[21] Y.K. Lin, A simple algorithm for reliability evaluation of a stochastic-flow network with node failure, Comput. Oper. Res. 28 (2001) 1277–1285.
[22] J.E. Ramirez-Marquez, B.A. Gebre, A classification tree based approach for the development of minimal cut and path vectors of a capacitated network, IEEE Trans. Reliab. 56 (2007) 474–487.
[23] M.L. Rebaiaia, D. Ait-Kadi, An efficient algorithm for enumerating minimal pathsets in communication networks, in: 4th International Conference on Networked Digital Technologies, volume 293 of Communications in Computer and Information Science, Springer Berlin Heidelberg, Dubai, UAE, 2012, pp. 444–458.
[24] A. Schrijver, On the history of the transportation and maximum flow problems, Math. Program. 91 (2002) 437–445.
[25] Y. Shen, A new simple algorithm for enumerating all minimal paths and cuts of a graph, Microelectron. Reliab. 35 (1995) 973–976.
[26] J. Xue, On multistate system analysis, IEEE Trans. Reliab. 34 (1985) 329–337.
[27] W.C. Yeh, A simple algorithm to search for all *d*-MPs with unreliable nodes, Reliab. Eng. Syst. Saf. 73 (2001) 49–54.
[28] W.C. Yeh, A simple approach to search for all *d*-MCs of a limited-flow network, Reliab. Eng. Syst. Saf. 71 (2001) 15–19.
[29] W.C. Yeh, A simple method to verify all *d*-minimal path candidates of a limited-flow network and its reliability, Int. J. Adv. Manuf. Technol. 20 (2002) 77–81.
[30] W.C. Yeh, A simple algorithm to search for all MCs in networks, Eur. J. Oper. Res. 174 (2006) 1694–1705.
[31] M.J. Zuo, Z. Tian, H.Z. Huang, An efficient method for reliability evaluation of multistate networks given all minimal path vectors, IIE Trans. 39 (2007) 811–817.

Yi-Kuei Lin is currently a Chair Professor of Industrial Management Department, National Taiwan University of Science and Technology, Taiwan, Republic of China. He received a Bachelor degree in Applied Mathematics Department from National Chiao Tung University, Taiwan. He obtained his Master degree and Ph.D. degree in the Department of Industrial Engineering and Engineering Management at National Tsing Hua University, Taiwan, Republic of China. He has the honor to get the Outstanding Research Awards from National Science Council of Taiwan in 2008, 2010 and 2013, respectively. His research interest includes performance evaluation, stochastic network reliability, operations research, and telecommunication management. He has published over 170 papers in refereed journals including *European Journal of Operational Research; Computers and Operations Research; Reliability Engineering & System Safety; IEEE Transactions on Reliability; IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans; International Journal of Industrial Engineering; Computers and Mathematics with Applications; International Journal of Advanced Manufacturing Technology; International Journal of Reliability, Quality and Safety Engineering; Journal of the Operations Research Society of Japan; Mathematical and Computer Modelling; Computers and Industrial Engineering; International Journal of Production Economics; and Applied Mathematics and Computation.*

After 2010, he has more than 100 papers published or accepted by SCI/SSCI journals. He is the Editor-in-Chief of the *International Journal of Reliability and Quality Performance*, and serves as Associate Editor of several international Journals such as *Mathematical Problems in Engineering* (SCI), *International Journal of Electronic Business Management* (EI), *Journal of Industrial and Production Engineers* (EI), *International Journal of Reliability and Applications*, etc. He also serves as the Chair of Steering Committee of the Asia-Pacific International Conference on Advanced Reliability and Maintenance Modeling (2013–2022), the Program Chair of the ISSAT International Conference on Reliability and Quality in Design (2010–2017), International Conference on Quality, Reliability, Risk, Maintenance and Safety Engineering (2014–2016), 18th IEEE International Conference on Computer Supported Cooperative Work in Design (2014), IIE Asian 2013, and 9th International Conference on Optimization: Techniques and Applications (2013), as well as a member of Advisory Board of International Conference on Advanced Manufacturing and Automation (2009), The 13th Asia Pacific Industrial Engineering and Management Systems Conference (2012), International Conference on Engineering Applications (2013), etc.

Shin-Guang Chen received his B.Sc. in Computer Engineering, National Chiao Tung University, Taiwan, and his Ph.D. in Industrial Engineering, National Chiao Tung University, Taiwan. Since 2001, he has been with the Department of Industrial Management, Tungnan University, Taiwan, where he holds the rank of 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 Information Sciences, JOCS, AOR, IJPR, IJIE, ESWA, JORSJ, CSTM, IJOR, JCIIE, IJRQP, APEN, CIE, OMEGA, IEEE TOR, APM, etc. He is a member of IEEE, IEEE TOR, POMS, CIIE, CERPS, and CSQ.