The CSI Journal on Computer Science & Engineering

The CSI Journal on
Computer Science and Engineering
Vol. 8, No. 2 & 4 (b), 2010
Pages 42-50
Regular Paper

Finding All the Upper Boundary Points of a Stochastic-Flow Network with Budget Constraints

Majid Forghani-elahabad Nezam Mahdavi-Amiri

Department of Mathematical Sciences, Sharif University of Technology, Tehran, Iran

Abstract

Recently, several algorithms have been proposed for computing system reliability and unreliability as two frequent performance indices. Here, we make use of a performance index to measure the performance for a stochastic-flow network. The index is the probability of being the maximum flow from source to sink being equal to a given demand level *d* subject to a budget constraint. The index can be determined in terms of all upper boundary points. Presenting some useful results to decrease the number of candidates, an improved algorithm is proposed to find all the constrained upper boundary points. The time complexity of the algorithm is established, showing its efficiency as compared to other algorithms. Moreover, a medium-size network is employed to provide an intuitive understanding of the efficiency of the algorithm. Using the obtained results, the performance index of the example is computed by the sum of disjoint product approach.

Keywords: Performance Index, Stochastic-Flow Network (SFN), (d, b)-MinCut ((d, b)-MC), Minimal Cut (MC), Budget Constraint.

1. Introduction

The theory of network reliability originates from a series of lectures given by Von Neumann in 1952 [1]. Thence, much of the research effort in network reliability has been expended on the evaluation (exact or approximate) of three measures: (1) R_N with N being the set of all nodes in the network, called all-terminal reliability, as the probability that all nodes in the network have communication to each other [2, 3], (2) R_K with K being an arbitrary subset of N, called K-terminal reliability, as the probability that all the nodes in K can communicate [4, 5, 6], and (3) $R_{s,t}$ with s and t being two specified nodes in the network, called two-terminal or source-to-terminal reliability, as the probability of the existence of at least one operation al path from node s to node t [7, 8].

Algorithms for computing these metrics belong to the NP-complete class [9], a family of NP-hard problems. However, there are a number of realistic systems with their components having more than two states, and thus the binary state theory cannot properly compute the system reliability [10, 11]. Hence, the multi-state two-terminal reliability problem has attracted attention significantly [12-20]. This way, the aim is to calculate the probability that the maximum flow of the network between two nodes s and t is more than d, or the probability that more than d units of flow can be sent from node s to node t through the network. Mostly, the proposed approaches compute system reliability in terms of d-MinCuts (d-MCs) [12-20]. Once all the d-MCs are determined, reliability can be calculated by some exact methods such as the inclusion-exclusion [7] and sum of disjoint products [21], or approximating methods such as the Monte-Carlo simulation [5, 19]. Thus, determination of all the d-MCs is an important step in computing system reliability.

Xue [12] proposed an algorithm using discrete function theory, modular decomposition, and system enlarging to generate all d-MCs for a multistate system having multistate components. Jan et al. [13] pointed out that many steps in Xue's algorithm were superfluous when applied to a stochastic-flow network and then by introducing the notion

of d-MC candidate proposed a more efficient algorithm that first finds all d-MC candidates obtained from each MC and then checks every candidate for being a d-MC. Yan and Qian [15] proved new results to decrease the number of the obtained d-MC candidates, found some d-MCs without the need for testing and eliminated some duplicate d-MCs, and then proposed an improved algorithm to find all the d-MCs. Yeh [16] presented some new results, and for the first time proposed an algorithm that avoided the production of the duplicate d-MCs. Salehi-Fathabadi and Forghani-elahabad [17] proposed an algorithm working merely on the definition of d-MC. By introducing a new data structure to eliminate the duplicates and a simple technique to find the Lower Capacity Limits, Forghani-elahabad and Mahdavi-Amiri [18] improved the proposed algorithm in [17] and demonstrated the efficiency of their algorithm in comparison with other proposed algorithms in [15] and [16].What is certain is that budget constraint is a significant element for designing and evaluating reliability of flow networks. Some algorithms have been proposed for computing system reliability or unreliability under budget constraint in terms of Minimal Paths (MPs) [22, 23] or MCs [24-28]. Since the number of MCs is usually less than the number of MPs [29], working on MCs is preferred. Considering the maintenance cost constraint, Yeh [24] proposed a simple algorithm for finding all the upper boundary points meeting the cost constraint (called (d, b)-MCs) in a stochastic-flow network with perfect nodes. Lin [25] considered multi commodity networks and proposed a simple algorithm to find all the maximal vectors meeting the budget constraint and the demand level d. In [26], an algorithm was proposed to evaluate system unreliability of a stochastic-flow network whose nodes and arcs have several possible capacities and may fail. Afterwards, Lin [27] used the proposed algorithm in [26] for calculating all the (d, b)-MCs in a multi commodity stochastic-flow network. Applying a genetic algorithm and recursive sum of disjoint products, Lin and Yen [28] proposed an optimization algorithm for the network reliability based transmission line assignment problem with the budget constraint.

Authors in [25-28] compared multi-state vectors to find maximal vectors, resulting in algorithms having high time complexities. Furthermore, the proposed algorithms in [25-28] may produce duplicate (d, b)-MCs from different MCs, contributing to further complexity. Yeh [16] presented a technique to avoid the generation of the duplicate d-MCs (without budget constraint), but the technique turned to have a high time complexity. Later, in absence of the budget constraint, Forghani-elahabad and Mahdavi-Amiri [18] introduced a more efficient technique.

Here, using the introduced data structure in [18], some useful existing results and presenting new results, an improved algorithm for finding all the (d, b)-MCs is proposed and compared with existing algorithms in terms of complexity results and performance on a medium-size example.

The remainder of our work is organized as follows. Section 2 describes the required notations, nomenclature, and assumptions. In Section 3, some useful existing results as well as some new results are given. Using the presented results, we propose an improved algorithm and demonstrate its efficiency by establishing the complexity results in Section 4. Moreover, a medium-size network is worked out to have an intuitive understanding of the efficiency of the proposed algorithm in comparison with other existing ones. In Section 5, we make use of a performance index. Then, by using the obtained result in Section 4 and applying the sum of disjoint product approach, the performance index of a network is computed. Finally, we conclude in Section 6.

2. Notations, Nomenclature, and Assumptions

2.1. Notations

G (N, A, M ,C)	a stochastic-flow network with the set of nodes N ={1, 2, ..., n}, the set of arcs A={a_i| 1 ≤ i ≤ m}, M=(M_1, M_2, ..., M_m) with M_i= M(a_i) denoting the max-capacity of a_i, for 1 ≤ i ≤ m, and C=(c_1,c_2, ..., c_m) with c_i denoting the cost of sending one unit flow through a_i, for i=1, 2, ..., m. Moreover, node 1 is the source node and node n is the sink node.
b	The total budget of the system.
n, m	The number of nodes in N and arcs in A, respectively.
X (a_i)	The capacity level of arc a_i under the system-state vector X =(x_1, x_2,…, x_m).
e_i	e_i = 0(a_i) is a system-state vector in which the capacity level is1 for a_i and 0 for other arcs.
G (N, A, X, C)	The corresponding network to G (N, A, M, C) with current system state vector X =(x_1, x_2, ..., x_m).
R (N, A, X^d, C)	the corresponding residual network to G(N,A,X,C) after sending d units of flow from node s to node t.
V (X)	The max-flow from node s to node t in G (N, A, X, C).
υ	V (M), the max-flow from node s to node t in G (N, A, M, C).
U (X)	U(X) = {a ∈A|X (a) <M (a)} is the set of unsaturated arcs in G (N, A, M, C).
C (X)	C(X) = c_1x_1 + c_2x_2 + ... + c_m x_m is the cost associated with X.
$CAP_{K_i}(X)$	The capacity of MC, K_i, under vector X=(x_1, x_2, …, x_m); i. e. , $CAP_{K_i}(X) = \sum_{a_l \in K_i} x_l$.
|•|	the number of elements; e.g., |N| is the number of nodes in N.

2.2. Nomenclature

Cut: A cut is a subset of A, in which there is no path from the source node s to the sink node t after elimination of all its arcs from G (N, A, M).

Minimal Cut (MC): A cut so that none of its proper subsets is a cut.

Demand level D: 0 ≤d<υ is a non-negative integer-valued flow or stress requirement for a given network flow.

System Reliability: If d is a deterministic constant, then the system reliability, R_d, equals pr{X | V (X) >d}. If d is a random variable with distribution f_d, then the system reliability, R_d, is equal to pr {X|V (X) >d}. f_d.

(d, b)-MinCut ((d, b)-MC): A system-state vector X is a (d, b)-MC if and only if it satisfies the budget constraint, V(X) = d, and V (X + e_i) >d, for every a_i ∈ U(X).

Y < X: A vector Y=(y_1,y_2, ..., y_m) is less than a vector X=(x_1,x_2, ..., x_m) if and only if for every i = 1, 2, ..., m, y_i≤ x_i, and for at least one 1 ≤j ≤m, y_j<x_j.

Maximal Vector: A vector X∈ψ is a maximal vector in ψ if and only if there is no vector Y (≠ X) in ψ so that X < Y.

2.3. Assumptions

1. The capacity of each arc $a_i \in A$ is a non-negative integer-valued random number less than or equal to M_i.
2. The capacities of different arcs are statistically independent.
3. The flow in G (N, A, M, C) satisfies the flow conservation law [30].

Moreover, since an SFN with unreliable nodes can be modified to a conventional network with perfect nodes [31], we consider an SFN with perfect reliable nodes here.

3. Upper Boundary Point

Here, we first state some useful new and existing results on the d-MC problem and then some useful results subject to the budget constraint are given.

3.1. D-MC Problem

An upper boundary point, called d-MinCut (d-MC), is a system-state vector, say X, such that V (X) = d and V(X+0(a))>d, for every a ∈ U(X). The notion of d-MC candidate was first defined by Jane et al. [13] as follows.

Definition 1. A system-state vector $X=(x_1, x_2,\ldots, x_m)$ is a d-MC candidate if and only if there exists at least one MC, K_i, so that the followings hold:

$$\begin{cases} (1)\mathrm{CAP}_{K_i}(X) = d \\ (2) 0 \le x_r \le M_r \quad \forall a_r \in K_i \\ (3)\, x_l = M_r \quad \forall a_r \notin K_i \end{cases} \tag{1}$$

Generally, the proposed algorithms in [12-18] for finding all the d-MCs consist of two stages, gathering all the d-MC candidates obtained from all the MCs and finding all the d-MCs among candidates by testing them. Thus, decreasing the number of d-MC candidates or shortening the time of testing process of candidates can be effective for solving the d-MC problem. The following lemma stated in [18] is effective for decreasing the number of d-MC candidates.

Lemma 1. Let X be a d-MC candidate. If |U(X)| = 1, considering U(X) = {a*}, then every d-MC Y (≠X) satisfies Y (a*) >X (a*).

Applying Lemma 1, one can determine a Limit Capacity Level (LCL), L_r, for x_r in inequality (2) of system (1), and thus somewhat avoid generation of repetitive d-MC candidates. In fact, by using Lemma 1, one can replace inequality (2) in (1) with the inequality

$$L_r \le x_r \le M_r \quad \forall a_r \in K_i$$

And thus provide the following system:

$$\begin{cases} (1)\mathrm{CAP}_{K_i}(X) = d \\ (2) L_r \le x_r \le M_r \quad \forall a_r \in K_i \\ (3)\, x_l = M_r \quad \forall a_r \notin K_i \end{cases} \tag{2}$$

To have an intuitive understanding of the effect of Lemma 1, consider Figure 1 and note that by using (1) we obtain five hundred and forty six 12-MC candidates, whereas only twenty 12-MC candidates are generated by applying (2).

It is easily verified that the probability of obtaining d-MC candidates with only one unsaturated arc from an MC, say K, is increased as much as the capacity of K, CAP_K (M), is decreased. Thus, to have a more efficient use of Lemma 1 in the proposed algorithm, all the MCs are sorted in an ascending order of the capacities; if there is a tie, then the one with fewer arcs is arranged first.

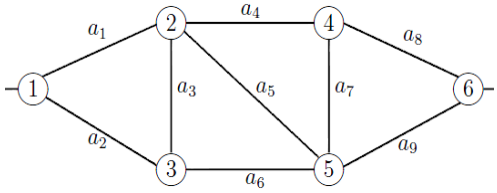

Figure 1. A network flow with M= (8, 6, 2, 5, 5, 4, 1, 6, 8)

For determining a number of d-MCs without the need for the time-consuming examination, the following result provides a practical tool.

Theorem 1. If K_i is an MC in G (N, A, M, C) with $\mathrm{CAP}_{K_i}(X) = \nu$, then all the d-MC candidates obtained from K_i are indeed d-MC.

Proof. Suppose that the system-state vector X is an arbitrary d-MC candidate obtained from K_i. First, it is demonstrated that V (X) = d. Since X (a) = M (a), for all a ∉ U(X), we have

$$\sum_{a\in U(X)} \big(M(a) - X(a)\big) = \mathrm{CAP}_{K_i}(M) - \mathrm{CAP}_{K_i}(X) = \nu - d.$$

On the contrary, assume that V (X) <d, and so there exists an MC, say K_j (≠K_i), in G (N, A, X, C) so that $\mathrm{CAP}_{K_j}(X) < d$. this leads to

$$\mathrm{CAP}_{K_j}(M) = \mathrm{CAP}_{K_j}(X) + \sum_{a\in U(X)\cap K_j} \big(M(a) - X(a)\big) \le \mathrm{CAP}_{K_j}(X) + \sum_{a\in U(X)} \big(M(a) - X(a)\big) < d + \nu - d = \nu$$

With the specification $\mathrm{CAP}_{K_j}(M) < \nu$, contradicting the definition of ν. Therefore, $V(X) = \mathrm{CAP}_{K_i}(X) = d$. Now, assume that a is a given arc in U(X). It is apparent that X+0(a) is a (d+1)-MC obtained from K_i. Analogously, by replacing d withd+1, it is concluded that V(X+0(a)) =d+1>d. This completes the proof. ■

Considering Theorem 1, it is observed that every computable d-MC candidate from each MC with capacity υ is really a d-MC and no extra testing is required. For instance, consider Figure 1 as a network flow. It is seen that υ = 14 and $K_1 = \{a_1, a_3, a_6\}$ is an MC. Since $\mathrm{CAP}_{K_1}(M) = 8 + 2 + 4 = 14$, Theorem 1 concludes that every d-MC candidate obtained from K_1 for each demand level d<14 is a d-MC. For example, all the 10-MC candidates obtained from K_1, X_{21}=(8, 6, 2, 5, 5, 0, 1, 6, 8), X_{22} = (7, 6, 2, 5, 5, 1, 1, 6, 8), X_{23} = (6, 6, 2, 5, 5, 2, 1, 6, 8),X_{24} =(5, 6, 2, 5, 5, 3, 1, 6, 8), X_{25} =(4, 6, 2, 5, 5, 4, 1, 6, 8), X_{26} =(8, 6, 1, 5, 5, 1, 1, 6, 8),X_{27} =(8, 6, 0, 5, 5, 2, 1, 6, 8), X_{28} =(7, 6, 1, 5, 5, 2, 1, 6, 8),

X_{29} =(7, 6, 0, 5, 5, 3, 1, 6, 8), $X_{2,10}$ = (6, 6, 1, 5, 5, 3, 1, 6, 8), $X_{2,11}$ = (6, 6, 0, 5, 5, 4, 1, 6, 8), and $X_{2,12}$ = (5, 6, 1, 5,5, 4, 1, 6, 8), are really 10-MC and no further check is necessary. This illustrates that employing Theorem 1 significantly reduces the number of tested d-MC candidates for being d-MC.

The following theorem is the very famous and basic theorem regarding the cut and flow values whose proof can be found in textbooks on network flow theory (e.g., see [30]).

Theorem 2. (Max-Flow Min-Cut Theorem) The maximum value of the flow from a source node s to a sink node t in a capacitated network equals the minimum capacity among all MCs.

Theorem 2 shows that there exists at least one MC with its capacity being equal to υ, the max-flow of the network (usually there exist more than one such MC). Thus, Theorem 1 can be notably useful.

Another notable fact is the possibility of duplicate d-MCs being produced from different MCs. The proposed algorithms in [12-14, 17] generate duplicate d-MCs, an undesired feature. Yan and Qian [15], Yeh [16], and Forghani-elahabad and Mahdavi-Amiri [18] presented different approaches to remove duplicates or avoid generating them. The proposed data structure in [18] turns to be more efficient than other proposed approaches (see [18] for comparative results). In fact, in [18], a unique number is associated with every d-MC candidate, as described below.

For each d-MC, say $X=(x_1, x_2, ..., x_m)$, we know that $x_i \le M_i$, i=1, 2, ...,m. Let l be the number of digits in $M^*=\max\{M_i | i= 1, 2, \ldots, m\}$. Obviously, the number of digits in every component of X is less than or equal to l. Thus, corresponding to each component x_i of a d-MC, we can associate an l-digit number, n_i, as follows:

1) If x_i has l digits, then let $n_i = x_i$.
2) If x_i has q digits with q < l, then place (l−q) zeros at the left side to make it an l-digit number, that is, $n_i=\underbrace{00\ldots0}_{l-q\ times}\ x_i$.

This way, we obtain an ml-digit number, $N_X = \overline{n_1 n_2 \ldots n_m}$, associated with each d-MC, X. Then, a divide and conquer approach is used to detect duplicates, instead of comparing with all the candidates. Here, we apply the proposed data structure to avoid consideration of the duplicates.

In [18], the introduced data structure was used to remove the duplicates after generating all the d-MCs. Here, however, we improve the efficiency by inserting each generated d-MC candidate into the existing sorted list of candidates by a binary search, immediately after the candidate's appearance.

3.2. Budget Constraint

Considering the definition of d-MC, an upper boundary point meeting budget constraint b, called ((d, b)-MC), is a d-MC, say X, such that $C(X) \le b$. The following results are straightforwardly concluded from the definitions of d-MC and (d, b)-MCs.

Corollary 1. Every (d, b)-MC is a d-MC.

Corollary 2. Every d-MC whose total cost is less than or equal to b is a (d, b)-MC.

Hence, we can find all the d-MCs and then determine the ones for which the budget is satisfied. In the previous section, we provided two very useful results to decrease the computations needed for finding all the d-MC candidates. However, our main aim here is to propose an approach to solve the (d, b)-MC problem. In fact, existence of the budget constraint by itself leads to a decrease in the number of candidates obtained in (1). We first solve (1) to find the d-MC candidates and the d-MCs. Then, by testing the budget constraint, we obtain the (d, b)-MCs. Although we do not use the budget constraint for determining the candidates, but we can use them to decrease the number of computations.

Depending on the network, we may have an MC with no (d, b)-MC candidate or an MC with all the d-MC candidates being (d, b)-MC candidates. The following lemmas, whose proofs are easily established, are useful.

Lemma 2. Let K_i be an MC and $c_{min}=\min\{c_r \mid a_r \in K_i\}$. If $d \times c_{min} + \sum_{a_l \notin K_i} c_l M_l > b$, then there is no possible (d, b)-MC obtained from K_i.

Lemma 3. Let K_i be an MC and $c_{Max}=\max\{c_r | a_r \in K_i\}$.If $d \times c_{Max} + \sum_{a_l \notin K_i} c_l M_l < b$, then every d-MC obtained from K_i is a (d, b)-MC.

Moreover, one can find both the most costly and the most cost-effective d-MC candidate producible from each MC, and so verify whether any d-MC candidate can satisfy the budget constraint. Furthermore, we may find some MCs such that every d-MC candidate obtained from them is a (d, b)-MC candidate. Next, we construct the most costly and the most cost-effective d-MC candidate obtained from MC, $K=\{a_{n1}, a_{n2}, a_{nr}\}$, as follows.

Without loss of generality, assume that $c_{n1} \le c_{n2} \le \ldots \le c_{nr}$. Let us construct the system-state vectors $Y_K^d = (y_1, y_2, \ldots, y_m)$ and $Z_K^d = (z_1, z_2, \ldots, z_m)$ as follows:

$$\begin{cases} (1)\ y_{n_1} = \min\{d, M_{n_1}\}, \\ (2)\ y_{n_i} = \max\{0, \min\{M_{n_i}, d - \sum_{j=1}^{i-1} y_{n_j}\}\}, \\ \qquad\qquad i = 2,3, \ldots, r, \\ (3)\ z_{n_i} = \max\{0, \min\{M_{n_i}, d - \sum_{j=i+1}^{r} z_{n_j}\}\}, \\ \qquad\qquad i = 1,2, \ldots, r-1, \\ (4)\ z_{n_r} = \min\{d, M_{n_r}\}, \\ (5) y_l = z_l = M_l, \quad \forall\, a_l \notin K. \end{cases} \tag{3}$$

It is seen that the system vectors $Y_K^d$and Z_K^d are the d-MC candidates generated from K. The following results can be directly deduced.

Corollary 3. The system state vectors $Y_K^d$and Z_K^d are the most costly and the most cost-effective d-MC candidates obtained from K, respectively.

Theorem 3. Considering the constructions (3), we have

1) If $\sum_{i=1}^{m} c_i y_i > b$,there is no possible (d, b)-MC candidate from K,
2) If $\sum_{i=1}^{m} c_i z_i \le b$, then every d-MC obtained from K is a (d, b)-MC.

Consider the MC, K= {a_1, a_3, a_6}, in Figure 1. Employing the constructions (3), we find Z_K^{10} = (4, 6, 2, 5, 5, 4, 1, 6, 8). Since C (Z_K^{10}) = 645 ≤ 650, it is concluded from Theorem 3 that every 10-MC obtained from K is a (10,650)-MC. Therefore, there is no need for checking the budget constraint for the other 10-MCs obtained from K.

4. An Improved Algorithm

Here, using theorems 1, 3, Lemma 1 and the data structure proposed in [18] to avoid of the generation of the duplicates, an improved efficient algorithm, Algorithm 1, is proposed. Note that Algorithm 1 first finds the d-MC candidates obtained from each MC and then checks the budget constraint when it is needed. Moreover, we assume that the number of existing MCs is p, say K_1, K_2, …, K_p, in the stochastic-flow network.

Algorithm 1. An algorithm for finding all the (d, b)-MCs in a stochastic-flow network.

Step 0. Let i = j = 1, L_r = 0, for r = 1, 2, ..., m, Q ={}, and arrange all the MCs in an ascending order of capacities; if there is a tie, then arrange the one with fewer elements first. Then, considering constructions (3), calculate all the vectors $Z_{K_l}^d$ and$Y_{K_l}^d$, for l = 1, 2, …, p.

Step 1. If $C(Y_{K_i}^d) > b$, then there is no possible (d, b)-MC candidate from K_i and go to Step 10.

Step 2. Use an implicit enumeration to find a feasible solution (a d-MC candidate from K_i), say X_{ij}, for system (2). If no such solution exists, then go to Step 10.

Step 3. If |U (X_{ij})|=1, considering U (X_{ij})={a_k}, then let L_k=X_{ij}(a_k) + 1.

Step 4. If $CAP_{K_i}(M) > v$, and then go to Step 6.

Step 5. If $C(Z_{K_i}^d) \leq b$, X_{ij} is a (d, b)-MC, then go to Step 9, else go to Step 8.

Step 6. If V (X_{ij}) ≠d, then X_{ij} is not a (d, b)-MC, let j = j +1 and go to Step 2.

Step 7. If there is a ∈ U(X_{ij}) such that V (X_{ij} + 0(a)) ≤ d, then X_{ij} is not a(d, b)-MC, let j = j + 1 and go to Step 2.

Step 8. If C(X_{ij}) ≤b, X_{ij} is a (d, b)-MC, then go to Step 9, else go to Step 10.

Step 9. Construct the associating number with X_{ij}, i.e., n_{ij}, and search Q. If n_{ij} is not in Q, then add it to Q. Let j = j + 1 and go to Step 2.

Step 10. If i < p, then let i = i + 1, j = 1 and go to Step 1.

First, by an example, we show how the algorithm searches for and finds all the (d, b)-MCs in a stochastic-flow network.

Example 1. The bridge benchmark [used in 13-17, 24-28] given in Figure 2 shows a simple computer network in which each arc represents a transmission line (it consists of several physical transmission lines, e.g., T_3 cable, E_1 cable, optical fiber) and each node represents a computer center. The probability distributions of arcs are given in Table 1. Thus, M= (4, 3, 3, 4, 3, 3). Also, assume that C= (15, 10, 20, 20, 20, 10) and b= 290. The supervisor would like to know the probability that the maximum flow equals the demand level of 5 with the budget of b = 290.

There are four MCs in Figure 2: D_1 = {a_1, a_3}, D_2 = {a_1, a_4, a_5}, D_3= {a_2, a_3, a_5}, and D_4 = {a_2, a_4}. The maximum flow of the network is υ = 7. Here, we find all the (5,290)-MCs by using Algorithm 1. In the next section, busing the sum of disjoint product approach, we compute the performance index of the given network for demand level 5.

Step 0. Let i=j=1, L_r=0, r = 1, 2, …, 5, Q= {}. We have K_1=D_1, K_2=D_4, K_3=D_3, and K_4=D_2. Then, υ=7 and we also have $Y_{K_1}^5$=(4, 3, 1, 4, 3, 3), $Z_{K_1}^5$=(2, 3, 3, 4, 3, 3), $Y_{K_2}^5$=(4, 3, 3, 2, 3, 3), $Z_{K_2}^5$ =(4, 1, 3, 4, 3, 3),$Y_{K_3}^5$= (4, 3, 0, 4,3, 2), $Z_{K_3}^5$= (4, 0, 3, 4, 3, 2), $Y_{K_4}^5$= (4, 3, 3, 1, 0, 3), $Z_{K_4}^5$= (0, 3, 3, 2, 3, 3).

Step 1. C ($Y_{K_1}^5$) = 280 ≤ 290.

Step 2. X_{11}= (2, 3, 3, 4, 3, 3) is obtained.

Step 3. Since U(X_{11}) = {a_1}, let L_1 = X_{11}(a_1) + 1 = 3.

Step 4. $CAP_{K_1}(M) = 7 \not> 7$.

Step 5. Since C($Z_{K_1}^5$) = 290 ≤ 290, X_{11} is a 5-MC and transfer is made to Step 9.

Step 9. Since n_{11}=233433 is not in Q, then let Q= {233433}, j= 2 and transfer is made to Step 2.

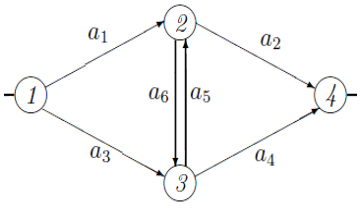

Figure 2. The network for Example 1.

Table 1. Probability distributions of the arc capacities for Figure 2.

Arc	0	1	2	3	4
a_1	0	0	0.05	0.05	0.9
a_2	0	0	0.1	0.9	0
a_3	0	0	0.1	0.9	0
a_4	0	0	0.05	0.05	0.9
a_5	0	0	0.1	0.9	0
a_6	0	0	0.1	0.9	0

Step 2. X_{12} = (4, 3, 1, 4, 3, 3) is obtained.

Step 3. Since U(X_{12}) = {a_3}, let L_3 = X_{12}(a_3) + 1 = 2.

Step 4. $CAP_{K_1}(M) = 7 \not> 7$.

Step 5. Since C ($Z_{K_1}^5$) = 290 ≤ 290, X_{12} is a 5-MC and transfer is made to Step 9.

Step 9. Since n_{12} = 431433 is not in Q, then let Q = {233433, 431433}, j = 3and transfer is made to Step 2.

Step 2.X_{13} = (3, 3, 2, 4, 3, 3) is obtained.

⋮

and continuing the process, we obtain X_{21}=(4, 3, 3, 2, 3, 3), X_{31}=(4, 3, 2, 4, 3, 0), X_{32}=(4, 2, 3, 4, 3, 0), X_{33}=(4, 2, 2,4, 3,1) as the (5,290)-MCs.

Note that there are thirty four producible (5,290)-MC candidates, seven of which are real (5,290)-MCs in Figure 2. Using Lemma 1 causes Algorithm 1 to obtain only eight candidates instead of all the producible candidates.

Because of applying Theorem 1, Algorithm 1 did determine six (5,290)-MCs without the need for testing based on the definition. Moreover, owing to employing Theorem 3, Algorithm 1 did not need to incur the cost of (5,290)-MCs generated from K_1 and K_3.

4.1. Time Complexity of Algorithm 1

For calculating the capacity of all the MCs and arranging them in order of their capacities, we need at most O(mp+ plop) computations. Also, the time complexity of computing the vectors $Y_{K_i}^d$ and $Z_{K_i}^d$, for i=1, 2, …, p, is $O(m^2p)$. Since $O(p) = O(2^{n-2})$ [32] and $O(n) \leq O(m) \leq O(n^2)$, then the time complexity of Step 0 is $O(m^2p)$. The time complexity of Step 1 is vividly O(m). Let σ be an upper bound for the number of all the solutions obtained in Step 2 corresponding to each MC. So, pσ is an upper bound for the number of all the obtained candidates in Step 2of the algorithm. The time complexities of steps 3, 4, and 5 are obviously O(m).

To find $V(X_{ij})$ in Step 6 and $V(X_{ij} + 0(a))$ in Step 7, we utilize the max flow algorithm in [30]. Since the time complexity of the max-flow algorithm is $O(n^2\sqrt{m})$ and m is an upper bound for $|U(X_{ij})|$, the number of arcs in $U(X_{ij})$, the time complexity of steps 6 and 7 is $O(m\sqrt{m}n^2)$.

The time complexity of Step 8 is clearly O(m). In Step 9, Algorithm 1 uses a divide and conquers approach to insert the number or find a duplicate. Thus, since the number of all the (d, b)-MCs is bounded by pσ, the time complexity of Step 9 is O(pσlog (pσ)). The time complexity of Step 10 is vividly O(1).

Now, since Step 0 is executed only once and pσ is an upper bound of the number of all the candidates obtained in Step 2, the time complexity of Algorithm 1 is $O(m^2p+p\sigma\log(p\sigma)+mp) + O(p\sigma(m+m+n^2\sqrt{m}+m^2+m)) = O(p\sigma(n^2\sqrt{m}+m^2)) < O(m^2\sqrt{n}p\sigma)$.
Thus, we have the following result.

Theorem 4. The time complexity of Algorithm 1 is less than $O(m^2\sqrt{n}\ p\sigma)$.

The proposed algorithm in [26] employed a comparative method to find all the maximal vectors among all the (d, b)-MC candidates. Since the number of all the candidates is bounded by pσ and each candidate is an m-tuple vector, the time complexity of comparing all the candidates and so the time complexity of the proposed algorithm in [26] is $O(mp^2\sigma^2)$. The proposed algorithm in [24] generates all the producible candidates and tests every candidate for being a (d, b)-MC. However, the algorithm generates duplicate (d, b)-MCs, and the duplicates need to be removed by comparing them. Therefore, the time complexity of the proposed algorithm in [24] is $O(mp^2\sigma^2)$. As a result, according to Table 2, Algorithm 1 is far more efficient than the proposed algorithms in [24] and [26]. In the next section, to have an intuitive understanding of the efficiency of Algorithm 1 in comparison with other existing algorithms in [24] and [26], a medium-size network is employed.

4.2. Performance on a Medium-Size Network

Here, to realize the effect of using Lemma 1 and Theorem 1, we assume that there is enough budget (b=∞).

Consider Figure1 as a network flow. There are nine MCs: $K_1=\{a_1, a_2\}$, $K_2=\{a_1, a_3, a_6\}$, $K_3=\{a_1, a_3, a_5, a_7, a_9\}$, $K_4=\{a_2, a_3, a_4, a_5\}$, $K_5=\{a_2, a_3, a_5, a_7, a_8\}$,$K_6=\{a_4, a_5, a_6\}$, $K_7=\{a_4, a_7, a_9\}$, $K_8=\{a_5, a_6, a_7, a_8\}$, and $K_9=\{a_8, a_9\}$. We use the proposed algorithms in [24, 26] and Algorithm 1 here to find all the (12,∞)-MCs of Figure 1 and the final results are given in Table 3. In Table 3, there are 3 columns. N_{Can} shows the number of all the obtained candidates, N_{Test} shows the number of tested candidates, and N_{Real} is the number of obtained (12,∞)-MCs by the algorithms.

Note that in the given flow network in Figure 1, there are 4286520 possible system state vectors, 546 of which are producible 12-MC candidates. As seen in the table, our proposed algorithm obtained only twenty candidates, nineteen of which were correctly determined as (12,∞)-MC.

Table 2. The time complexity of the proposed algorithms in [24, 26] and Algorithm 1

Algorithms	Time complexities
Algorithm 1 here	less than $O(m^2\sqrt{n}p\sigma)$
The proposed algorithm in [24]	$O(mp^2\sigma^2)$
The proposed algorithm in [26]	$O(mp^2\sigma^2)$

Table 3. The final results obtained by the algorithms on Figure 1

Algorithms	N_{Can}	N_{Test}	N_{Real}
Algorithm 1 here	20	1	19
The proposed algorithm in [24]	546	546	19
The proposed algorithm in [26]	546	546	19

Without testing and only one candidate was tested and determined for not being a (12, ∞)-MC. However, the proposed algorithm in [24, 26] obtained all the five hundred and forty six producible (12,∞)-MCs and needed to test all of them to find the nineteen (12,∞)-MCs.

5. Performance Index

Recall that if the demand level d is a constant, then the system reliability equals $R_d = pr\{X|V(X)>d\}$, for $X=(x_1, x_2, …, x_m)$ being a system-state vector. There are several methods such as the inclusion-exclusion [7] and sum of disjoint products [21], or approximating methods such as the Monte-Carlo simulation [5, 19] to calculate the system reliability.

To analyze a network flow, a performance index, PI_d, is considered here as the probability of being the maximum flow of the network being equal to the demand level d. According to the definition of system reliability, it is easily deduced that the performance index can be computed from the following formula:

$$PI_d = R_{d-1} - R_d \tag{4}$$

Note that it is generally assumed that the capacity for each arc a_i is an integer-valued random variable from the set $\{0, 1, 2, …, M_i\}$, with a given distribution. Moreover, the capacities of the arcs are considered to be statistically independent.

Now, assume that $X^1, X^2, ..., X^q$ are d-MCs and $X=(x_1, x_2,..., x_m)$ is a system state vector. Now, the system reliability for level d, the probability that the maximum flow of the network is more than d, using the sum of disjoint product method, is given by

$$R_d = pr(A) = \sum_{i=1}^{q} pr(E_i) \tag{5}$$

Where,

$$A = \bigcup_{i=1}^{q} A_i = \bigcup_{i=1}^{q} E_i\,, A_i = \{X \mid X > X^i\},$$

$$E_1 = A_1, E_i = A_i - \bigcup_{j=1}^{i-1} A_j\,, i = 2,3, \dots, q,$$

$$pr(E_i) = \sum_{X \in E_i} pr(X), \text{ and } pr(X) = \prod_{j=1}^{m} pr(x_j).$$

According to the definition of d-MC, it is clearly observed that V(X)> d if and only if $X \in A = \bigcup_{i=1}^{q} E_i$. Thus, R_d=pr{X|V (X)>d} = pr {X|X ∈ A} = pr (A), and consequently (5) is correct.

Likewise, considering

$$B_i = \{X | X \le X^i\}, i = 1,2, \dots, q,$$
$$F_1 = B_1, F_i = B_i - \bigcup_{j=1}^{i-1} B_j\,, i = 2,3, \dots, q,$$

And $F = \bigcup_{i=1}^{q} F_i$, one can also compute R_d as follows:

$$R_d = 1 - pr(F) = 1 - \sum_{i=1}^{q} pr(F_i). \quad (6)$$

It is seen that both (5) and (6) can be applied to compute the system reliability of the network. However, to lessen the computations, we should use (5) when d ≥υ/2, and otherwise use (6).

Note that for the sum of disjoint product method is an improved version of inclusion-exclusion approach. In fact, the number of computations for the sum of disjoint product method is less than the one for inclusion-exclusion method [29].

Example 2. The probability distributions of the arc capacities of Figure 2 are listed in Table 1. Find the introduced performance index for the demand level 5 meeting the budget constraint b=290 in Figure 2.

Solution:

According to (4), we first must find all the (4,290)-MCs in Figure 2 employing Algorithm 1. The final results are given in Table 4. Now, from (5) we obtain R_4=0.853659 and R_5=0.788049, and consequently it is deduced from (4) that $PI_5 = 0.853659 - 0.788049 = 0.06561$.

Table 4. The final results obtained by Algorithm 1 for Figure 2

Minimal cuts	(4, 290)−MCs	(5, 290)−MCs
$K_1 = \{a_1, a_3\}$	(4,3,0,4,3,3), (3,3,1,4,3,3), (2,3,2,4,3,3), (1,3,3,4,3,3)	(2,3,3,4,3,3), (4,3,1,4,3,3), (3,3,2,4,3,3)
$K_2 = \{a_2, a_4\}$	(4,0,3,4,3,3), (4,3,3,1,3,3), (4,2,3,2,3,3), (4,1,3,3,3,3)	(4,3,3,2,3,3)
$K_3 = \{a_2, a_3, a_5\}$	(4,1,1,4,3,2), (4,1,2,4,3,1), (4,1,3,4,3,0), (4,2,1,4,3,1), (4,2,2,4,3,0), (4,3,1,4,3,0)	(4,3,2,4,3,0), (4,2,3,4,3,0), (4,2,2,4,3,1)
$K_4 = \{a_1, a_4, a_5\}$	- -	- -

6. Conclusions

There are a number of approaches to solve the d-MC problem. However, budget constraint is an important issue in the design of network flows. Several authors proposed algorithms to solve the (d, b)-MC problem, the d-MC problem with a budget constraint. Here, we used certain useful existing results as well as our own new results for the (d, b)-MC problem to propose an improved algorithm.

The time complexity of the proposed algorithm was established and compared with other existing ones to show the efficiency of our algorithm. Moreover, a medium size network was employed to illustrate the efficiency of the proposed algorithm for large scale examples. We also made use of a performance index for evaluating a stochastic-flow network. Using the obtained results by our algorithm and the sum of disjoint product approach, we computed the performance index for a medium size example.

Acknowledgements

The authors thank the Research Council of Sharif University of Technology for its support.

References

[1] J. Von Neumann, *Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components,* New Jersey, Princeton University Press, 1956.

[2] A. R. Sharafat, and O. R. Ma'rouzi, "All-terminal network reliability using recursive truncation algorithm," *IEEE Trans. Reliability*, vol. 58, no. 2, pp. 338-347, 2009.

[3] A. Rodionov, D. Migov, and O. Rodionova, "Improvements in the efficiency of cumulative updating of all-terminal network reliability," *IEEE Trans. Reliability*, vol. 61, no. 2, pp. 460-465, 2012.

[4] R. K. Wood, "Factoring algorithms for computing K-terminal network reliability," *IEEE Trans. Reliability*, vol. 35, no. 3, pp. 269-278, 1986.

[5] H. Cancela, and M. El Khadiri, "Series-parallel reductions in Monte Carlo network reliability evaluation," *IEEE Trans. Reliability*, vol. 47, no. 2, pp. 159-164, 1998.

[6] G. Hardy, C. Lucet, and N. Limnios, "K-terminal network reliability measures with binary decision diagrams," *IEEE Trans. Reliability*, vol. 56, no. 3, pp. 506-515, 2007.

[7] J. A. Buzacott, and J. S. K. Chang, "Cut-set intersections and node partitions," *IEEE Trans. Reliability*, vol. 33, no. 5, pp. 385-389, 1984.

[8] S. Y. Kuo, S. K. Lu, and F. M. Yeh, "Determining terminal-pair reliability based on edge expansion diagrams using OBDD," *IEEE Trans. Reliability*, vol. 48, no. 3, pp. 234-246, 1999.

[9] M. O. Ball, "Computational complexity of network reliability analysis: an overview," *IEEE Trans. Reliability*, vol. 35, no. 3, pp. 230-239, 1986.

[10] R. A. Boedigheimer, and K. C. Kapur, "Customer-driven reliability models for multistate coherent systems," *IEEE Trans. Reliability*, vol. 43, no. 1, pp. 46-50, 1994.

[11] R. Billinton, and W. Zhang, "State extension for adequacy evaluation of composite power systems-applications," *IEEE Trans. Power Systems*, vol. 15, no. 1, pp. 427-432, 2000.

[12] J. Xue, "On multistate system analysis," *IEEE Trans. Reliability*, vol. 34, no. 4, pp. 329-337, 1985.

[13] C. C. Jane, J. S. Lin, and J. Yuan, "Reliability evaluation of a limited-flow net work in terms of minimal cut sets," *IEEE Trans. Reliability*, vol. 42, no. 3, pp. 354-361,1993.

[14] Y. K. Lin, "Using minimal cuts to evaluate of reliability the stochastic-flow net work with failures at nodes and arcs," *Journal of Reliability Engineering and System Safety*, vol. 75, no. 1, pp. 41-46, 2002.

[15] Z. Yan, and M. Qian, "Improving efficiency of solving d-MC problem in stochastic-flow net work," *Journal of Reliability Engineering and System Safety*, vol. 92, no. 1, pp. 30-39, 2007.

[16] W. C. Yeh, "A fast algorithm for searching all multi-state minimal cuts," *IEEE Trans. Reliability*, vol. 57, no. 4, pp. 581-588, 2008.

[17] H. Salehi-Fathabadi, and M. Forghani-elahabad, "A note on 'A simple approach to search for all d-MCs of a limited-flow network," *Journal of Reliability Engineering and System Safety*, vol. 94, no. 2, pp. 1878-1880, 2009.

[18] M. Forghani-elahabad, and N. Mahdavi-Amiri, "A simple efficient approach for the D-Min Cut problem," *Proc, IEEE Int'l Conf. Operations Research and Optimization*, pp. 72-75, 2013.

[19] J. E. Ramirez-Marquez, and D. W. Coit, "A Monte-Carlo simulation approach for approximating multi-state two-terminal reliability," *Journal of Reliability Engineering and System Safety*, vol. 87, no. 2, pp. 253-264, 2005.

[20] S. Satitsatian, and K. C. Kapur, "An algorithm for lower reliability bounds of multistate two-terminal networks," *IEEE Trans. Reliability*, vol. 55, no. 2, pp. 199-206, 2006.

[21] A. O. Balan, and L. Traldi, "Preprocessing min paths for sum of disjoint products," *IEEE Trans. Reliability*, vol. 52, no. 3, pp. 289-295, 2003.

[22] J. S. Lin, "Evaluate the reliability of limited-flow networks under the cost constraint," *IEEE Trans. Reliability*, vol. 30, no. 3, pp. 1175-1180, 1998.

[23] Y. F. Niu, and X. Z. Xu, "Reliability evaluation of multi-state systems under cost consideration," *Journal of Applied Mathematical Modeling*, vol. 36, no. 1, pp. 4261-4270, 2012.

[24] W. C. Yeh, "Multistate network reliability evaluation under the maintenance cost constraint," *Journal of Production Economics*, vol. 88, no. 2, pp. 73-83, 2004.

[25] Y. K. Lin, "A simple algorithm to generate all (d, b)-MCs of a multi commodity stochastic-flow network," *Journal of Reliability Engineering and System Safety*, vol. 91, no. 2, pp. 923-929, 2006.

[26] Y. K. Lin, "Unreliability evaluation for a limited-flow network with failed nodes subject to budget constraint," *Journal of Computers and Mathematics with Applications*, vol. 51, no. 1, pp. 73-82, 2006.

[27] Y. K. Lin, "On a multi commodity stochastic-flow network with unreliable nodes subject to budget constraint," *European Journal of Operational Research*, vol. 176, no. 2, pp. 347-360, 2007.

[28] Y. K. Lin, and C. T. Yeh, "Using minimal cuts to optimize network reliability for a stochastic computer network subject to assignment budget," *Journal of Computers and Operations Research*, vol. 38, no. 3, pp. 1175-1187, 2011.

[29] G. Rubino, "Network reliability evaluation, in: K. Bagchi, J. Walrand (Eds.), State-of-the-Art in Performance Modeling and Simulation," *Gordon and Breach Books*, pp. 275-302, 1998.

[30] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows-Theory, Algorithms, and Applications*, New Jersey, Prentice-Hall, 1993.

[31] K. K. Aggarwal, J. S. Gupta, and K. B. Misra, "A simple method for reliability evaluation of a communication system," *IEEE Trans. Communications*, vol. 23, no. 3, pp. 563-565, 1975.

[32] D. Shier, *Network Reliability and Algebraic Structures*, New York, Clarendon Press, 1991.

Majid Forghani-elahabad is a Ph.D. student in Operation Research under the supervision of Professor Mahdavi-Amiri at Faculty of Mathematical Sciences, Sharif University of Technology, Tehran. He received his M. Sc. degree in Operation Research under the supervision of Professor Salehi-Fathabadi from University of Tehran, and his B.S. degree in Applied Mathematics from Yazd University. He joined Iranian Operations Research Society in July 2013, and is working on evaluating reliability of stochastic-flow networks.
E-mail: forghanimajid@mehr.sharif.ir

Nezamedin Mahdavi-Amiri is a full professor of Mathematical Sciences at Sharif University of Technology. He received his Ph.D. degree from the Johns Hopkins University in Mathematical Sciences in 1981. He is on the editorial board of several mathematical and computational journals in Iran including Bulletin of the Iranian Mathematical Society (Style-Language Editor), the Iranian Journal of Operations Research (Editor-in-Chief) and the CSI Journal of Computer Science and Engineering. He

was also the Editor-in-Chief of Mathematical Thought and Culture (in Persian), a journal of Mathematical Society of Iran, and on the Executive Council of the Iranian Mathematical Society. He is currently the Vice President of the Operations Research Society of Iran as well as the representative of Operations Research Society of Iran to IFORS.
E-mail: nezamm@sina.sharif.edu

Paper Handling Data:

Submitted: 22.04.2013
Received in revised form: 15.08.2013
Accepted: 20.08.2013
Corresponding author: Dr. Nezam Mahdavi-Amiri, Department of Mathematical Sciences, Sharif University of Technology, Tehran, Iran.