Adv Syst Sci Appl 2020; **04**; 1-10
Published online at https://ijassa.ipu.ru.

An Algorithm to Search for All Minimal Cuts in a Flow Network

Majid Forghani-elahabad[1*], Nezam Mahdavi-Amiri[2]

[1)] *Centro de Matemática, Computação e Cognição – CMCC, Universidade Federal do ABC – UFABC, Santo André, SP, Brazil*

E-mail: m.forghani@ufabc.edu.br

[2)] *Faculty of Mathematical Sciences, Sharif University of Technology, Tehran, Iran*

E-mail: nezamm@sharif.edu

Abstract: Reliability evaluation approaches exploit a variety of tools for system modeling and calculation of reliability indices. Most proposed approaches in the literature are in terms of minimal cuts (MCs) or minimal paths. Hence, finding all the MCs in a network is an attractive problem in network flow and reliability theory. Here, investigating some works related to the MC problem, we first point out a number of defects in certain proposed algorithms, and present certain new techniques to rectify the flaws in two algorithms. Moreover, we present some techniques to improve the algorithms. Using our results, we propose an improved algorithm for finding all the MCs in a flow network. A benchmark network example is used to illustrate the performance of the algorithm. The proposed algorithm is shown to be correct. Finally, after establishing the complexity results, we demonstrate the proposed algorithm to be more efficient than some existing algorithms.

Keywords: minimal cut, network flow, algorithm, reliability

1. INTRODUCTION

Most proposed algorithms for evaluating the reliability of a flow network make use of minimal cuts (MCs) [3, 4, 9, 10, 16-18, 19, 20] or minimal paths [5, 6, 7, 8, 11, 12, 14, 15, 22]. In the algorithms based on MCs, we usually need to know all the MCs in advance. This has promoted the MC problem that is, finding all the MCs in a network, as one of the most attractive problems in network flow and reliability theory. A cut is a set of edges whose removal from the network results in disconnection between the source (s) and sink (t) nodes. An MC is a cut with none of its proper subsets being a cut. The MC problem is an NP-hard problem [20]. Several algorithms have been proposed to obtain all the MCs in both undirected flow networks [1, 2, 13, 25, 26] and directed flow networks [1, 21]. To determine all the MCs, Fard and Lee [2] considered both link and node failures and proposed an algorithm for finding all the MCs of such networks by the use of networks having perfect nodes. The algorithm adopts the concept of common-cause failure and does not re-enumerate MCs for the additional supposition of node failures. Yeh [25] defined an MC using a node set (called MCV), and then proposed a simple algorithm to find all the MCVs between the two special nodes s and t. Unfortunately, the algorithm of [25] turned to be faulty and thus might not find all the MCVs of a network [13]. Gomes and Fernandes [13] tried to identify some but not all the existing defects of [25]. In fact, the modified algorithm of [13] has still some flaws which we will point out later on. Yeh et al. [26] proposed an algorithm for the problem of finding all the MCs in a modified network. They showed the algorithm to be more efficient than the existing algorithms. A number of available related algorithms in the

* Corresponding author: m.forghani@ufabc.edu.br, phone: +55(11)4996-8330.

literature were stated in [1]. Assuming nodes with *k*-out-of-*n* property, Tan [21] extended the traditional directed *s*-*t* networks and presented an approach for finding all the MCs for all the nodes by using the definition of MC for the nodes and starting with the source node *s* and ending with the sink node *t*.

Here, the techniques of [13] for modifying the proposed algorithm of Yeh [25] are improved. We note that the algorithms of [13] and [25] first determine all the MCVs. Then, the algorithms transform each MCV to an MC by determining all the existing arcs between the two associating node sets of the MCV by using the adjacency matrix of the network. This may be a very time consuming step for large networks. Here, some new techniques are presented to improve upon the existing algorithms. Before that, we first state all the flaws of the algorithms of [13] and [25].

In the remainder of our work, in Section 2 we provide the required definitions and state the flaws in the algorithms of [13, 25]. In Section 3, some more efficient techniques are presented to modify the algorithm of [25]. Then, an approach with less time complexity is presented to transform each MCV to an MC. Afterwards, an improved algorithm is proposed to determine all the MCs in a network flow by using the presented results. The correctness and complexity results are provided in Section 3. Section 4 gives our concluding remarks.

2. FINDING ALL THE MINIMAL CUTS

The required notations are first described, and then the algorithm of [25] is exposed to explain its flaws in finding all the MCs. We also state some flaws of the modified algorithm of [13]. For brevity, here we only state the proposed algorithm of [25] and do not show the details of the algorithm of [13].

2.1. Problem description

We use the same notations, nomenclature, and assumptions of [13, 25]. Let $G(V, E)$ be a connected network with the set of nodes $V=\{s, 1, 2, \ldots, n\text{-}2, t\}$ and the set of edges E, where s and t are the source and sink nodes, respectively, and $e_{uv} \in E$ denotes an edge between nodes u and v. For an arbitrary set of nodes $U \subseteq V$, let $\overline{U} = V - U$, $E(U) = \{e_{uv} \in E \mid u, v \in U\}$ be the associated edges with the set of nodes U, $G(U, E(U))$ be the sub network of $G(V, E)$ including only the set of nodes U and its associated edges, and $MC(U) = \{e_{uv} \in E \mid u \in U$ and $v \in \overline{U}\}$ be the corresponding cut. An MCV candidate is a subset of nodes whose removal will cause a disconnection of nodes s and t. An MCV candidate in $G(V, E)$, say U, is an MCV when $MC(U)$ is a minimal cut. Let σ be the number of MCVs (or MCs), and for each $i=1, 2, \ldots, \sigma$, $C_i=MC(U_i)$ be the associated MC with MCV, U_i. Moreover, for each MCV, U, and each node $v \notin U$, let $E(v,U) = \{e_{vu} \in E \mid u \in U\}$.

2.2. Flaws in the existing algorithms

Here, all the flaws in the algorithm of [25] are explained. For convenience, we rewrite the proposed algorithm of [25] as 'Algorithm 1' below.

Algorithm 1
The algorithm of [25] for finding all the MCVs in a network $G(V, E)$.
Step 0. Let $i = k = 0$, $S = U_0 = \{s\}$, $T = V - \{s\}$, $N_0 = \{t\}$, and $P = \phi$.
Step 1. If there is a node $u \in T - N_i$ adjacent to S, then $S \cup \{u\}$ is an MCV candidate and go to Step 2, else go to Step 4.
Step 2. If $G(T-\{u\}, E(T-\{u\}))$ is a connected network then $S \cup \{u\}$ is an MCV and go to Step 3, else any node set containing $S \cup \{u\}$ is not an MCV.
Step 3. Let $i = i+1$, $k = k+1$, $U_k = S = S \cup \{u\}$, $P = P \cup \{U_k\}$, $T = T-\{u\}$, $N_i = N_{i-1}$, and go to Step 1.

Step 4. If $i = 1$ then stop, else remove the last node, say v, in S, let $i = i-1$, $N_i = N_i \cup \{v\}$, $T = T \cup \{v\}$, and go to Step 1.

We first use a similar example as the one given in [25] to show how "Algorithm 1" fails in some cases, and also its obtained solution depends on the order of node selection in Step 1.

Example 1. Consider Fig. 1 as a network, and find all its MCVs by using 'Algorithm 1'.

Solution:

Step 0. Let $i = k = 0$, $S = U_0 = \{s\}$, $T = V-\{s\}$, $N_0 = \{t\}$, and $P=\phi$.
Step 1. Since $T - N_0 = \{1, 2, 3, 4\}$, node 2 is selected and transfer is made to Step 2.
Step 2. Since $G(\{1, 3, 4, t\}, E(\{1, 3, 4, t\}))$ is connected, $\{s,2\}$ is found as an MCV, and transfer is made to Step 3.
Step 3. Let $i = 1$, $k = 1$, $U_1 = S = \{s,2\}$, $P = \{U_1\}$, $T = \{1, 3, 4, t\}$, $N_1 = \{t\}$, and transfer is made to Step 1.
Step 1. Since $T - N_1 = \{1, 3, 4\}$, node 4 is selected and transfer is made to Step 2.
Step 2. Since $G(\{1, 3, t\}, E(\{1, 3, t\}))$ is not connected, the algorithm deduces that any set containing the set $\{s, 2, 4\}$ is not an MCV.

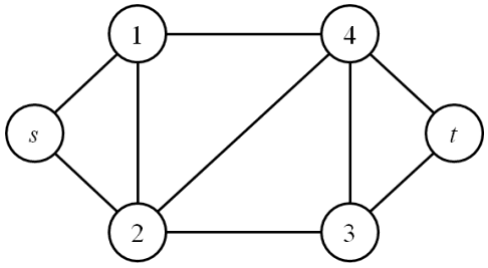

Fig. 1. A benchmark network

Next, if transfer is made to Step 1, the algorithm may select node 4 recursively, and the algorithm does not terminate. If transfer is made to Step 3, 'Algorithm 1'saves the set $\{s, 2, 4\}$ as an MCV which is an incorrect result. If transfer is made to Step 4, since $i = 1$, the algorithm stops without finding the other MCVs. Hence, in any case, the algorithm leads to an incorrect result, and consequently it cannot find all the MCVs. It should be noted that if in the first place one selects node 1 instead of node 2, the algorithms end up with a very different final result. In fact, the obtained solutions by "Algorithm 1" depend on the order of node selection in Step 1.

Example 1 illustrated some flaws in 'Algorithm 1'. In the sequel, we mention all the defects in this algorithm in detail.

(1) During steps 1-4, 'Algorithm 1' never determines the MCV=$\{s\}$, and so we should consider it as an initial input in Step 0. To address this flaw, we can replace "$P= \phi$" with "$P = \{U_0\}$" in Step 0.
(2) As seen in Example 1, there is an ambiguity in the second part of Step 2 in 'Algorithm 1', where $G(T-\{u\}, E(T-\{u\}))$ is not connected. In fact, in such a case, it is not clear how the algorithm should proceed.
(3) In Step 2 of 'Algorithm 1', when $G(T-\{u\}, E(T-\{u\}))$ is not connected, the algorithm leads to a wrong result, namely, "any node set containing $S \cup \{u\}$ is not an MCV". To see this more clearly, we note that in Example 1, since $G(\{1, 3, t\}, E(\{1, 3, t\}))$ is not connected, the algorithm concludes that any set containing the set $\{s, 2, 4\}$ is not

an MCV, whereas it is apparent that the sets $\{s, 1, 2, 4\}$ and $\{s, 1, 2, 3, 4\}$ contain the set $\{s, 1, 3\}$ and are indeed MCVs.

(4) The stopping criterion for the algorithm, i.e., $i = 1$, is not appropriate and may lead to the loss of some MCVs. In fact, with this stopping condition, the algorithm may stop before determining all the MCVs.

Although Goems and Fernandes [13] stated the flaws (1), (2), and (4), but the algorithm of [13] still contained the flaws (3) and (4); see lines 15 (Step 2) and 27 (Step 4) in the proposed algorithm of [13]. To rectify the stated defect (2), the authors of [13] used a new set D in their proposed algorithm and added lines 4, 16, and 17 to the algorithm. Even though this way they removed the flaw, but there is room for improvement. In fact, the proposed algorithm of [13], in such a case (flaw (2)), removes the node u form the set D (see line 16 of the algorithm of [13]) and does not change the sets T and N_i. In this case, the algorithm may again add the node u to the set D going through lines 5-7, 11-13, 19-24, and 4. Hence, the algorithm of [13] may check a node several times in a time consuming process, deteriorating the efficiency of the algorithm.

Furthermore, a common inefficiency of both algorithms of [13] and [25] is that both algorithms first determine all the MCVs and then transform each MCV into the associated MC. However, as we state in the next section, if one determines an MC, whenever the associated MCV is obtained, then the computing time would decrease.

3. NEW RESULTS

Here, we first make some proposals to modify the flaws in the proposed algorithm of [25], and then present some results to improve the efficiency of the algorithm. An improved algorithm is presented and shown to be correct. The complexity results are also provided.

3.1. *Modifications and improvements*

To address flaw (1), as we mentioned in Section 2.2, we can simply replace "$P = \phi$" with "$P = \{U_0\}$" in Step 0. To address flaw (2), we use a new set B and make some more changes in steps 0-2 as given in the proposed algorithm below, Algorithm 2. To address the flaw (3), we just need to remove the wrong assertion, "any node set containing $S \cup \{u\}$ is not an MCV", from Step 2. To address flaw (4), the stopping criterion for the algorithm is replaced by $i = 0$.

In addition to these modifications, we also make some improvements. We note that the final aim of the algorithm is to determine all the MCs in a flow network. For this, the algorithms of [13] and [25] first find all the MCVs and then transform each MCV into an MC by determining all the arcs between the two associated node sets. Let U be an MCV. It is vivid that the associated MC with U, i.e., $MC(U)$, given by

$$MC(U) = \bigcup_{u \in U} E(u, \overline{U}). \tag{3.1}$$

For each $u \in \overline{U}$, the time complexity of obtaining $E(u, \overline{U})$ is $O(n)$. Since n is an upper bound for the number of nodes in U, the time complexity of determining each MC is $O(n^2)$, and consequently the time complexity for determining all the MCs is $O(n^2\sigma)$. However, if one determines each MC, whenever a new MCV is obtained, then the time complexity can be lessened to $O(n\sigma)$. We note that a new MCV is obtained by adding to or removing some nodes from the current MCV in the algorithm, and hence considering this, we can determine the associated MC by adding and removing some edges to the current MC. Let U be an MCV with the associated MC, $MC(U)$, and $W = U \bigcup \{v\}$ be another MCV. In this case, we have $MC(W) = MC(U) \bigcup E(v, \overline{U} - \{v\}) - E(v, U)$. Similarly, when we remove a node, say u, from

MCV, U, we should add the edges of the set $E(u,\overline{U})$ to and remove the edges of the set $E(u,U-\{u\})$ from the associated MC to update the MC. For this, corresponding to each node, the time complexity for updating the MC is $O(n)$. As a result, the time complexity for determining all the MCs is $O(n\sigma)$.

In the next section, using the stated modifications, an improved algorithm is proposed to find all the MCs in a flow network.

3.2. *Proposed algorithm*

Here, we propose an improved algorithm to solve the MC problem, and then provide its correctness and complexity results. Note that since in our algorithm, Algorithm 2 below, all the MCs are obtained straightforwardly, there is no need to save the MCVs.

Algorithm 2

An improved algorithm for finding all the MCs in a network $G(V, E)$.

Step 0. Let $i = k = 0$, $S = \{s\}$, $T = V-\{s\}$, A = $C_0 = D_0 = E(s, T)$, $N_0 = \{t\}$, $B = \phi$, and $P = \{C_0\}$.

Step 1. If there is a node $v \in T - \{B \cup N_i\}$ adjacent to S then go to Step 2, else go to Step 4.

Step 2. If $G(T-\{v\}, E(T-\{v\}))$ is a connected network then let $B = \phi$ and go to Step 3, else let $B = B \cup \{v\}$ and go to Step 1.

Step 3. Let $i=i+1$, $k=k+1$, $T = T - \{v\}$, $A = D_i = C_k = A \bigcup E(v,T) - E(v,S)$, $S = S \cup \{v\}$, $P = P \cup \{C_k\}$, $N_i = N_{i-1}$, and go to Step 1.

Step 4. If $i = 0$ then stop, else remove the last node, v, in S, let $i=i-1$, $A = D_i$, $N_i = N_i \cup \{v\}$, $T = T \cup \{v\}$, and go to Step 1.

For a more convenient description of Algorithm 2, a flowchart of the algorithm is given in Fig. 2. Next, we use a benchmark network named modified ARPANET as given in Fig. 1 to illustrate Algorithm 2, and also show how our introduced changes turn the algorithm to work correctly with reasonable efficiency.

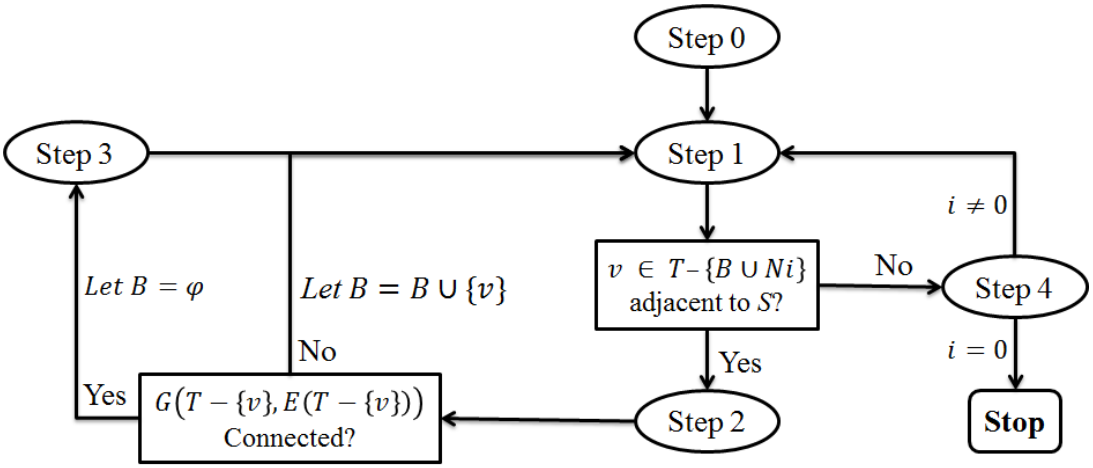

Fig. 2. A flowchart of Algorithm 2

Solution:

Step 0. Let $i = k = 0$, $S = \{s\}$, $T = \{1, 2, 3, 4, t\}$, $A = C_0 = E(s, T) = \{e_{s1}, e_{s2}\}$, $N_0 = \{t\}$, $B = \phi$, and $P = \{C_0\}$.

Step 1. Since $T - \{B \cup N_0\} = \{1, 2, 3, 4\}$, node $v = 1$ is selected and transfer is made to Step 2.

Step 2. Since $G(\{2, 3, 4, t\}, E(\{2, 3, 4, t\}))$ is connected, $B = \phi$ and transfer is made to Step 3.

Step 3. Let $i = 1$, $k = 1$, $T = \{2, 3, 4, t\}$, $A = D_1 = C_1 = \{e_{s1}, e_{s2}\} \bigcup \{e_{12}, e_{14}\} - \{e_{s1}\} = \{e_{s2}, e_{12}, e_{14}\}$, $S = \{s, 1\}$, $P = \{C_0, C_1\}$, $N_1 = N_0 = \{t\}$, and transfer is made to Step 1.
Step 1. Since $T - \{B \cup N_1\} = \{2, 3, 4\}$, node $v = 2$ is selected and transfer is made to Step 2.
Step 2. Since $G(\{3, 4, t\}, E(\{3, 4, t\}))$ is connected, $B = \phi$ and transfer is made to Step 3.
Step 3. Let $i = 2$, $k = 2$, $T = \{3, 4, t\}$, $A = D_2 = C_2 = \{e_{14}, e_{24}, e_{23}\}$, $S = \{s, 1, 2\}$, $P = \{C_0, C_1, C_2\}$, $N_2 = \{t\}$ and transfer is made to Step 1.
Step 1. Since $T - \{B \cup N_2\} = \{3, 4\}$, node $v = 3$ is selected and transfer is made to Step 2.
Step 2. Since $G(\{4, t\}, E(\{4, t\}))$ is connected $B = \phi$ and transfer is made to Step 3.
Step 3. Let $i = 3$, $k = 3$, $T = \{4, t\}$, $A = D_3 = C_3 = \{e_{14}, e_{24}, e_{34}, e_{3t}\}$, $S = \{s, 1, 2, 3\}$, $P = \{C_0, C_1, C_2, C_3\}$, $N_3 = \{t\}$ and transfer is made to Step 1.
Step 1. Since $T - \{B \cup N_3\} = \{4\}$, node $v = 4$ is selected and transfer is made to Step 2.
Step 2. Since $G(\{t\}, E(\{t\}))$ is connected $B = \phi$ and transfer is made to Step 3.
Step 3. Let $i = 4$, $k = 4$, $T = \{t\}$, $A = D_4 = C_4 = \{e_{4t}, e_{3t}\}$, $S = \{s, 1, 2, 3, 4\}$, $P = \{C_0, C_1, C_2, C_3, C_4\}$, $N_4 = \{t\}$ and transfer is made to Step 1.
Step 1. Since $T - \{B \cup N_4\} = \phi$, transfer is made to Step 4.
Step 4. Since $i = 4 \neq 0$, let $v = 4$, $S = S - \{4\} = \{s, 1, 2, 3\}$, $i = i - 1 = 3$, $A = D_3 = \{e_{14}, e_{24}, e_{34}, e_{3t}\}$, $N_3 = \{4, t\}$, T = $\{4, t\}$, and transfer is made to Step 1.
Step 1. Since $T - \{B \cup N_3\} = \phi$, transfer is made to Step 4.
Step 4. Since $i = 3 \neq 0$, let $v = 3$, $S = S - \{3\} = \{s, 1, 2\}$, $i = i - 1 = 2$, $A = D_2 = \{e_{14}, e_{24}, e_{23}\}$, $N_2 = N_2 \cup \{3\} = \{3, t\}$, $T = \{3, 4, t\}$, and transfer is made to Step 1.
Step 1. Since $T - \{B \cup N_2\} = \{4\}$, node $v = 4$ is selected and transfer is made to Step 2.
Step 2. Since $G(\{3, t\}, E(\{3, t\}))$ is connected $B = \phi$ and transfer is made to Step 3.
Step 3. Let $i = 3$, $k = 5$, $T = \{3, t\}$, $A = D_3 = C_5 = \{e_{23}, e_{34}, e_{4t}\}$, $S = \{s, 1, 2, 4\}$, $P = \{C_0, C_1, C_2, C_3, C_4, C_5\}$, $N_3 = \{3, t\}$, and transfer is made to Step 1.
Step 1. Since $T - \{B \cup N_3\} = \phi$, transfer is made to Step 4.
Step 4. Since $i = 3 \neq 0$, let $v = 4$, $S = S - \{4\} = \{s, 1, 2\}$, $i = i - 1 = 2$, $A = D_2 = \{e_{14}, e_{24}, e_{23}\}$, $N_2 = N_2 \cup \{4\} = \{3, 4, t\}$, $T = \{3, 4, t\}$, and transfer is made to Step 1.
Step 1. Since $T - \{B \cup N_2\} = \phi$, transfer is made to Step 4.
Step 4. Since $i = 2 \neq 0$, let $v = 2$, $S = S - \{2\} = \{s, 1\}$, $i = i - 1 = 1$, $A = D_1 = \{e_{s2}, e_{12}, e_{14}\}$, $N_1 = N_1 \cup \{2\} = \{2, t\}$, $T = \{2, 3, 4, t\}$, and transfer is made to Step 1.
Step 1. Since $T - \{B \cup N_1\} = \{3, 4\}$, node $v = 3$ is selected and transfer is made to Step 2.
Step 2. Since $G(\{2, 4, t\}, E(\{2, 4, t\}))$ is disconnected, $B = \{3\}$, and transfer is made to Step 1.
Step 1. Since $T - \{B \cup N_1\} = \{4\}$, node $v = 4$ is selected and transfer is made to Step 2.
Step 2. Since $G(\{2, 3, t\}, E(\{2, 3, t\}))$ is connected $B = \phi$, and transfer is made to Step 3.
Step 3. Let $i = 2$, $k = 6$, $T = \{2, 3, t\}$, $A = D_2 = C_6 = \{e_{s2}, e_{12}, e_{24}, e_{34}, e_{4t}\}$, $S = \{s, 1, 4\}$, $P = \{C_0, C_1, C_2, C_3, C_4, C_5, C_6\}$, $N_2 = \{2, t\}$, and transfer is made to Step 1.
Step 1. Since $T - \{B \cup N_2\} = \{3\}$, node $v = 3$ is selected and transfer is made to Step 2.
Step 2. Since $G(\{2, t\}, E(\{2, t\}))$ is disconnected, $B = \{3\}$, and transfer is made to Step 1.
Step 1. Since $T - \{B \cup N_2\} = \phi$, transfer is made to Step 4.
Step 4. Since $i = 2 \neq 0$, let $v = 4$, $S = S - \{4\} = \{s, 1\}$, $i = i - 1 = 1$, $A = D_1 = \{e_{s2}, e_{12}, e_{14}\}$, $N_1 = N_1 \cup \{4\} = \{2, 4, t\}$, $T = \{2, 3, 4, t\}$, and transfer is made to Step 1.
Step 1. Since $T - \{B \cup N_1\} = \phi$, transfer is made to Step 4.
Step 4. Since $i = 1 \neq 0$, let $v = 1$, $S = S - \{1\} = \{s\}$, $i = i - 1 = 0$, $A = D_0 = \{e_{s1}, e_{s2}\}$, $N_0 = N_0 \cup \{1\} = \{1, t\}$, $T = \{1, 2, 3, 4, t\}$, and transfer is made to Step 1.
Step 1. Since $T - \{B \cup N_0\} = \{2, 4\}$, node $v = 2$ is selected and transfer is made to Step 2.
Step 2. Since $G(\{1, 3, 4, t\}, E(\{1, 3, 4, t\}))$ is connected $B = \phi$ and transfer is made to Step 3.
Step 3. Let $i = 1$, $k = 7$, $T = \{1, 3, 4, t\}$, $A = D_1 = C_7 = \{e_{s2}, e_{12}, e_{24}, e_{23}\}$, $S = \{s, 2\}$, $P = \{C_0, C_1, C_2, C_3, C_4, C_5, C_6, C_7\}$, $N_1 = \{1, t\}$ and transfer is made to Step 1.
Step 1. Since $T - \{B \cup N_1\} = \{3, 4\}$, node $v = 3$ is selected and transfer is made to Step 2.
Step 2. Since $G(\{1, 4, t\}, E(\{1, 4, t\}))$ is connected $B = \phi$ and transfer is made to Step 3.

Step 3. Let $i = 2$, $k = 8$, $T = \{1, 4, t\}$, $A = D_2 = C_8 = \{e_{s1}, e_{12}, e_{24}, e_{34}, e_{3t}\}$, $S = \{s, 2, 3\}$, $P = \{C_0, C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8\}$, $N_2 = \{1, t\}$, and transfer is made to Step 1.
Step 1. Since $T - \{B \cup N_2\} = \{4\}$, node $v = 4$ is selected and transfer is made to Step 2.
Step 2. Since $G(\{1, t\}, E(\{1, t\}))$ is disconnected, $B = \{4\}$ and transfer is made to Step 1.
Step 1. Since $T - \{B \cup N_2\} = \phi$, transfer is made to Step 4.
Step 4. Since $i = 2 \neq 0$, let $v = 3$, $S = S - \{3\} = \{s, 2\}$, $i = i - 1 = 1$, $A = D_1 = \{e_{s1}, e_{12}, e_{24}, e_{23}\}$, $N_1 = N_1 \cup \{3\} = \{1, 3, t\}$, $T = \{1, 3, 4, t\}$, and transfer is made to Step 1.
Step 1. Since $T - \{B \cup N_1\} = \phi$, transfer is made to Step 4.
Step 4. Since $i = 1 \neq 0$, let $v = 2$, $S = S - \{2\} = \{s\}$, $i = i - 1 = 0$, $A = D_0 = \{e_{s1}, e_{s2}\}$, $N_0 = N_0 \cup \{2\} = \{1, 2, t\}$, $T = \{1, 2, 3, 4, t\}$, and transfer is made to Step 1.
Step 1. Since $T - \{B \cup N_0\} = \{3\}$, there is no node adjacent to S and transfer is made to Step 4.
Step 4. Since $i = 0$, the algorithm terminates.

We see that Algorithm 2 found all the 8 MCs in Fig. 1.

3.3. *The correctness and complexity results*

The following theorem demonstrates the correctness of Algorithm 2.

Theorem 1: Algorithm 2 determines all the MCs in a connected flow network.

Before presenting the proof, we note that in the proof below, C_i is an MC with two corresponding node sets U_i and $\overline{U}_i$ such that $\overline{U}_i = V - U_i$ and C_i = MC(U_i) = $\{e_{uv} \mid u \in U_i, v \in \overline{U}_i\}$. Moreover, $V_i = V(C_i) = \{u \in U_i \mid e_{uv} \in C_i\}$. It is clear that $V_i \subseteq U_i$, and one can name this as the extreme nodes in U_i.

Proof. It is vividly seen that Algorithm 2 finds $C_0 = \{e_{sv} \in E\}$ as the first MC in Step 0. Now, to the contrary, assume that there are some MCs missed by the algorithm. Without loss of generality, let C_i be a missed MC with its corresponding node set U_i having a minimal number of nodes. It is clear that V_i is not empty, and so consider a $j \in V_i \subseteq U_i$. According to definition of the set V_i, we know that there is at least one $k \in \overline{U}_i$ so that $e_{jk} \in C_i$, and consequently, $G(U_i - \{j\}, E(U_i - \{j\}))$ and $G(\overline{U}_i \cup \{j\}, E(\overline{U}_i \cup \{j\}))$ are connected. Hence, $U_i' = U_i - \{j\}$ is an MCV. Because U_i is the corresponding node set with a missed MC by the algorithm having a minimal number of nodes, the MC associated with the node set U_i', i.e., C_i', is found by the algorithm. It is observed that Algorithm 2 finds MCs in Step 3, and then goes to Step 1 to find the next possible one. Therefore, if Algorithm 2 finds C_i' in Step 3, then it will determine C_i in the subsequent iterations by selecting j as a node adjacent to $S = U_i'$. This contradicts the earlier assumption that C_i is a missed MC, and thus completes the proof. ■

Now, we compute the time complexity of Algorithm 2. The time complexity of Step 0 is $O(n)$. The time complexity for the construction of each MCV through the steps 1-4 is $O(n+m)$, and as explained in Section 3.1, the time complexity for updating an MC is $O(n)$. Thus, considering σ as the number of all the MCVs (or MCs) in the network, the time complexity of Algorithm 2 to determine all the MCs is $O(n+(m+n)\sigma)$, and since in a

connected flow network we have $O(n) \leq O(m)$ [1], the time complexity of Algorithm 2 is $O(m\sigma)$. Thus, we have the following result.

Theorem 2: The time complexity of Algorithm 2 for determining all the MCs is $O(m\sigma)$, where m and σ are respectively the number of edges and the number of MCs in the network.

We note that Theorem 2 shows that Algorithm 2 is as efficient as the recently algorithm of [26] and is more efficient than the algorithms of [13, 25].

5. CONCLUDING REMARKS

There are a number of algorithms for finding all the minimal cuts in different types of networks such as directed or undirected networks, network with node failures, link failures, or both node and link failures, networks with *k*-out-of-*n* nodes, cyclic or acyclic networks, etc. Here, two available algorithms in the literature were investigated and their flaws were stated. Certain efficient techniques were proposed to modify the flaws and some new techniques were also developed. Using the presented results, an improved algorithm was proposed to solve the MC problem and its correctness was established. The established complexity results showed the algorithm to be more efficient than some available algorithms.

ACKNOWLEDGEMENTS

The second author thanks Sharif University of Technology for supporting this work.

REFERENCES

1. Ahuja, R.K., Magnanti, T.L. *and* Orlin J.B. (1993). *Network flows theory, algorithms, and applications*. New Jersey: Englewood Cliffs, Prentice-Hall, Int.
2. Fard, N.S., and Lee, T.H. (1999). Cutset Enumeration of Network Systems with Link and Node Failures. *Reliability Engineering and System Safety*, 65: 141–146.
3. Forghani-elahabad, M., and Mahdavi-Amiri, N. (2010). Finding all the upper boundary points of a stochastic-flow network with budget constraints, *The CSI Journal on Computer Science and Engineering*, 8(2): 42–50.
4. Forghani-elahabad, M., and Mahdavi-Amiri, N. (2016a). An Improved Algorithm for Finding All Upper Boundary Points in A Stochastic-Flow Network. *Applied Mathematical Modelling*, 40: 3221–3229.
5. Forghani-elahabad, M., and Mahdavi-Amiri, N. (2016b). An efficient method for generating all minimal vectors for the *q* SMPs reliability problem with time and budget constraints. *IEEE Transactions on Reliability*, 65(2): 828–842.
6. Forghani-elahabad, M. *and* Bonani, L. (2017). Finding all the lower boundary points in a multistate two-terminal network. *IEEE Transactions on Reliability* 66(3): 677–688.
7. Forghani-elahabad, M. *and* Kagan, N. (2019a). Reliability evaluation of a stochastic-flow network in terms of minimal paths with budget constraint, *IISE Transactions*, 51(5): 547–558.
8. Forghani-elahabad, M. *and* Kagan, N. (2019b). A simple improved algorithm to find all the lower boundary points in a multiple-node-pair multistate flow network, *Advances in Systems Science and Applications*, 19(1): 1–11.

9. Forghani-elahabad, M. *and* Kagan, N. (2019c). An approximate approach for reliability evaluation of a multistate flow network in terms of minimal cuts, *Journal of Computational Science*, 33: 61–67.
10. Forghani-elahabad, M. *and* Kagan, N. (2019d). Assessing reliability of multistate flow networks under cost constraint in terms of minimal cuts, *International Journal of Reliability, Quality and Safety Engineering*, 26(05), 1950025.
11. Forghani-elahabad, M., Kagan, N. and Mahdavi-Amiri, N. (2019e) An MP-based approximation algorithm on reliability evaluation of multistate flow networks, *Reliability Engineering and System Safety*, 191, 106566.
12. Forghani-elahabad, M., Mahdavi-Amiri, N. and Kagan, N. (2020) On multi-state two separate minimal paths reliability problem with time and budget constraints. *International Journal of Operational Research*, 37(4), 479–490.
13. Gomes, T., and Fernandes, L. (2011). A Note on "A simple algorithm to search all MCs in networks". No. 11, Available at: http://www.inescc.pt/documentos/11-2010.PDF.
14. Lin, Y.K. (2003). Flow Assignment of a Stochastic Flow Network with Multiple Node Pairs. *International Journal of Industrial Engineering*, 10: 167–174.
15. Lin, Y.K. (2009). A MP-Based Algorithm for A Multicommodity Stochastic-Flow Network with Capacity Weights. *International Journal of Industrial Engineering*, 16(4), 282–292.
16. Niu, Y.F., Gao, Z.Y. and Lam, W.H.K. (2017) Evaluating the reliability of a stochastic distribution network in terms of minimal cuts. *Transportation Research Part E*, 100, 75–97.
17. Niu, Y.F., Gao, Z.Y. and Lam, W.H.K. (2017) A new efficient algorithm for finding d-minimal cuts in multi-state networks. *Reliability Engineering and System Safety*, 66, 151–163.
18. Niu, Y.F., *and* Xu, X.Z. (2019) A new solution algorithm for the multi-state minimal cut problem. *IEEE Transactions on Reliability*, DOI:10.1109/TR.2019.2935630.
19. Padmavathy, N. *and* Chaturvedi, S.K. (2013). Evaluation of mobile ad hoc network reliability using propagation-based link reliability model. Reliability Engineering *and* System Safety, 15: 1–9.
20. Provan, J.S., and Ball, M.O. (1983). The Complexity of Counting Cuts and of Computing the Probability That A Graph Is Connected. *SIAM Journal of Computing*, 12: 777–788.
21. Tan, Z. (2003). Minimal Cut Sets of s-t Networks with k-out-of-n Nodes. Reliability Engineering and System Safety, 82: 49–54.
22. Wu, W.W., Ning, A. *and* Ning, X.X. (2008). Evaluation of the reliability of transport networks based on the stochastic flow of moving objects. Reliability Engineering *and* System Safety, 93: 838–44.
23. Xu, X.Z., Niu, Y.F. and Li, Q. (2019). Efficient enumeration of d-minimal paths in reliability evaluation of multistate networks. *Complexity*, DOI: https://doi.org/10.1155/2019/4561845.
24. Xu, X.Z., Niu, Y.F. and Li, Q. (2018). Performance Assessment of a Freight Network with Stochastic Capacities. Complexity, Article ID 9142542, DOI: https://doi.org/10.1155/2018/9142542.

25. Yeh, W.C. (2006). A Simple Algorithm to Search for All MCs in Networks. European Journal of Operational *Research*, 174: 1694–1705.
26. Yeh, W.C., Ho, H.C., Chen, Y.C., and Yeh, Y.M. (2012). A New Algorithm for Finding All Minimal Cuts in Modified Network. *International Journal of Innovative Computing, Information and Control*, 8(1): 419–430.

