Malinowski Jacek 0000-0002-4413-1868
Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland, jacek.malinowski (at) ibspan.waw.pl

A fast method for enumerating all minimal d-cut-sets in a flow network

Keywords

flow network, component flow capacity, required flow, maximum flow, min-d-cut-set, min-cut-set

Abstract

A d-cut-set in a flow network is a set of components whose removal or blockage causes the maximum flow (MF) to fall below value *d*, provided that in fully operational network MF is greater or equal to *d*. A min-d-cut-set is a d-cut set such that it does not contain any other d-cut-set. In turn, a cut-set is a set of components whose removal disconnects the source and the sink, which results in MF being equal to 0. A min-cut-set contains no other cut-set. The method works as follows: the min-cut-sets are ordered according to increasing flow capacities, then from every min-cut-set a number of d-cut-sets are generated and each of them is checked for being minimal and unique. The method features two different algorithms respectively applied if $\Phi(C) < 2d$ or $\Phi(C) \geq 2d$. C is the min-cut-set from which d-cut-sets are generated, and $\Phi(C)$ is the flow capacity of C. This distinction results in quick generation of min-d-cut-sets without finding many non-d-cut-sets or non-minimal d-cut-sets. Compared to the similar methods, the new one is highly efficient, due to several original solutions, e.g. an efficient method of checking the redundancy of candidates for min-d-cut-sets. Min-d-cut-sets have two main applications. First, they can be used to compute various reliability characteristics of flow networks. Second, the knowledge of these sets facilitates or even enables management and maintenance of various flow networks such as data transmission, water, power supply, or traffic networks, field drainage systems, etc.

1. Introduction

In this chapter two efficient algorithms are proposed for the purpose of enumerating all minimal d-cut-sets in a flow network with one source and one sink node. Although this topic has been studied by several authors (see [2], [5], [9] and [12], where the last paper contains a comprehensive literature survey), the technique presented herein differs significantly from those applied in the above papers and features several original solutions aimed at high computational efficiency.

Let *D* be a set of components (links and/or nodes) in a flow network with one source and one sink node. *D* is called a d-cut-set if the failure of all components in *D* causes the maximum flow in the network to fall below *D* value *d* (*d* may be regarded as the required network capacity). Clearly, this definition makes sense if the maximum flow in the fully operational network (all its components are in operation) is greater than or equal to *d*. Further, *D* will be called a minimal-*d*-cut-set (abbreviated to min-d-cut-set or m-d-c-s) if no subset of *D* is a d-cut-set. It should be noted that min-d-cut-sets can be named differently throughout the literature, e.g. the term "subset cuts" is used in [2] and [5], but the former term seems more self-explanatory.

For the theoretical considerations conducted in the next section we will need the well-known Ford-Fulkerson theorem stating that the maximum flow from the source to the sink node is equal to the smallest capacity of a minimal cut-set (see [6]). A cut-set is a set of components whose failure results in topological disconnection between the source and the sink node; note that cut-sets should be distinguished from d-cut-sets. A minimal cut-set (shortened to min-cut-set) contains no other cut-set. The capacity of a cut-set or d-cut-set is the sum of capacities of all its components.

As stated in the abstract, the presented method consists of two algorithms, both of which use min-cut-sets as the initial data, as the algorithms of other cited authors do. From each min-cut-set a number of d-cut-sets are generated; each one is checked for being minimal and unique. This leads to

Safety and Reliability of Systems and Processes, Summer Safety and Reliability Seminar 2020

DOI: 10.26408/srsp-2020-13

finding all min-d-cut-sets. The method uses one of two algorithms, depending on whether $\Phi(C) < 2d$ or $\Phi(C) \geq 2d$, where C denotes a min-cut-set from which min-d-cut sets are generated, and $\Phi(C)$ denotes the capacity of C. The different handling of the cases $\Phi(C) < 2d$ and $\Phi(C) \geq 2d$, not encountered in other relevant papers, significantly accelerates the enumeration procedure. Most importantly, it greatly reduces the number of non-d-cut-sets that are generated, and ensures that the produced d-cut-sets include only a small number of non-minimal ones. Other advantages of this approach are given in Sections 4 and 5.

As indicated above, the method requires all min-cut-sets to be found in advance. The problem of finding them is a long studied one and there exist multiple algorithms for that purpose, presented e.g. in [1], [7], and [11].

The list of all (or some) min-d-cut-sets can be used for a number of practical purposes. First of all, it can be used for computing the network reliability which in this context is defined as the probability that the maximum flow is greater or equal to d. However, other uses are also possible, e.g. for drainage or traffic system control.

The paper is organized as follows. In Section 2 the used notation and preliminary assumptions concerning the studied network model are presented. Section 3 contains the necessary theoretical background for the algorithms developed in Sections 5 and 6. In Section 4 a general description of the newly developed method is given, along with a comparison to recent results published in [2] and [5]. In Sections 5 and 6 the algorithms for the cases $\Phi(C) < 2d$ and $\Phi(C) \geq 2d$ respectively are presented. They are illustrated by their application to an example network, and each step is analyzed in detail. Each of the Sections 3, 5 and 6 begins with several lemmas which collectively constitute the method's mathematical background. Some of the lemmas from Section 3 can most likely be found in earlier works, but their proofs are provided for the sake of self-containment. In turn, the lemmas from Sections 5 and 6 are the author's original results. In Section 7 the numerical complexity of the new method is estimated. Section 8 contains its detailed comparison with the method presented in [2]. In Section 9 a number of conclusive remarks are given with some hints regarding possible extensions of the method to networks modeled by (partly) directed graphs, networks with multiple sources and/or sinks, or networks with multi-state components.

2. Notation and preliminary assumptions

- $G = (V, E)$: a network defined as a graph G with V and E as the sets of nodes and links;
- $card(S)$: the number of elements in a set S;
- n: the number of components in G, $n = card(E) + card(V)$;
- x_i: the state of the i-th component; $x_i = 1$ or $x_i = 0$ if the i-th component is operable or failed;
- s, t: the source and the sink nodes;
- $s-t$ path: a path from s to t consisting of components of G;
- $\Gamma_{s,t}^{G}(x)$: the $s-t$ connectivity function of G, i.e. $\Gamma_{s,t}^{G}(x) = 1$ if there is an $s-t$ path composed of operable components, otherwise $\Gamma_{s,t}^{G}(x) = 0$;
- cut-set: a set of components whose removal or failure disconnects s and t;
- min-cut-set: a cut-set such that no its subset is a cut-set;
- m: the number of all min-cut-sets of G;
- $C_1, \ldots, C_m$: all min-cut-sets of G;
- $\varphi(c)$: the flow capacity of a component c, $c \in E \cup V$;
- large capacity component: a component c, such that $\varphi(c) \geq d$;
- $\Phi(C)$: the total flow capacity of all components in a set C, $C \subseteq \mathrm{E} \cup \mathrm{V}$, i.e. $\Phi(C) = \sum_{c \in C} \varphi(c)$;
- $\Psi_{max}(G)$: the maximal flow from s to t through G;
- C, C', D, D', D_1, D_2: sets of network components;
- $\subset, \subseteq$: the "included in and not equal to" and "included in or equal to" relations between sets;
- $G \backslash D$: the subgraph of G obtained by removing all the components of D;
- d-cut-set: a set D such that $\Psi_{max}(G \backslash D) < d$, i.e. D is a set of components whose removal or failure causes the maximal flow through G to fall below d;
- min-d-cut-set: a short form of "minimal d-cut-set"; D is a min-d-cut-set if no subset of D is a d-cut-set, i.e. $\Psi_{max}(G \backslash D') < d$ for each $D' \subset D$;
- m-d-c-s candidate in C_k: a short form of "minimal d-cut-set candidate in C_k"; D is a m-d-c-s candidate in C_k if $D \subseteq C_k$, $\Phi(C_k \backslash D) < d$ and $\Phi(C_k \backslash D') \geq d$ for each $D' \subset D$;
- redundant set: a set is redundant in a family of sets if it contains or is equal to another set in this family;
- IR: internal redundancy; a d-cut-set D is inte rnally redundant in C_k if $D \subseteq C_k$ and there

exists a d-cut-set D' such that $D' \subset D$;

- ER: external redundancy; a d-cut-set D is externally redundant in C_k, $k \geq 2$, if $D \subseteq C_k$ and there exists a d-cut-set D' such that $D' \subset D$ and $D' \subseteq C_j$ for a certain $j < k$;
- $A(C, D)$: the set of all elements in $C \backslash D$ preceding the last element in D, e.g. if $C = \{1, \dots, 10\}$ and $D = \{2, 4, 6\}$, then $A(C, D) = \{1, 3, 5\}$;
- $B(C, D)$: the set of all elements in $C \backslash D$ succeeding the last element in D, e.g. if D and C are defined as above then $B(C, D) = \{7, \dots, 10\}$;
- μ_D or $\mu(D)$: the component of D with the smallest flow capacity, i.e. $\varphi(\mu_D) = min[\varphi(d): d \in D]$.

For convenience, we will equate the component with their indices throughout the whole paper, i.e. the components c_1, c_2,, etc. of G will be referred to as 1, 2, etc.

Preliminary assumption
We assume that C_k, $k \in \{1, \dots, m\}$ are ordered according to the increasing flow capacity, as required by the proposed method. Hence, from the Ford-Fulkerson theorem we have:

$$\Psi_{max}(G) = \Phi(C_1) \leq \dots \leq \Phi(C_m). \quad (1)$$

We also assume that $\Psi_{max}(G) \geq 2d$.

3. Theoretical basis of the proposed method

The proposed method is based on several graph-theoretical properties of flow networks, which, for the sake of clarity, will be formulated as separate lemmas and two key theorems.

Lemma 3.1
If D is a set of network components, then all min-cut-sets of $G \backslash D$ are obtained by removing all redundant sets in the family $(C_1 \backslash D), \dots, (C_m \backslash D)$.

Proof: Let us first assume that D fulfills the following condition: $C_1 \backslash D \neq \varnothing$ for $k \in \{1, \dots, m\}$, i.e. none of the sets $C_1, \dots, C_m$ is included in D. It is a well-known fact from the reliability theory that $C_1, \dots, C_m$ are all min-cut-sets of G if and only if

$$\Gamma_{s,t}^{G}(x) = \Lambda_{k=1}^{m} \vee_{i \in C_k} x_i. \quad (2)$$

The $s - t$ connectivity function of $G \backslash D$ is obtained by setting x_i to 0 for $i \in D$ in $\Gamma_{s,t}^{G}$ given by (2). Let K_D be obtained from $\{1, \dots, m\}$ by removing each k such that $(C_k \backslash D)$ is redundant in $(C_1 \backslash D), \dots, (C_m \backslash D)$. We have:

$$\Gamma_{s,t}^{G \backslash D}(x) = \Lambda_{k=1}^{m} \vee_{i \in C_k \backslash D} x_i = \Lambda_{k \in K_D} \vee_{i \in C_k \backslash D} x_i \quad (3)$$

which ends the first part of the proof.
Let now $C_k \subseteq D$ for a certain $k \in \{1, \dots, m\}$. It holds that $C_k \backslash D = \varnothing \subseteq C_j \backslash D$ for each $j \neq k$, i.e. each $C_j \backslash D$ is redundant in regard to $C_k \backslash D$ which is the only (empty) min-cut-set of $G \backslash D$. Indeed, since C_k is a cut-set and $C_k \subseteq D$, there is no $s - t$ path in $G \backslash D$, hence no components have to be removed from $G \backslash D$ in order to separate s, and t, which means that the empty set is the only min-cut-set of $G \backslash D$. The whole proof is thus completed.

Lemma 3.2
If D is a d-cut-set, then there exists a min-cut-set C_k, $k \in \{1, \dots, m\}$, such that $C_k \cap D$ is also a d-cut-set. In other words, each d-cut-set is redundant w.r.t. a d-cut-set included in one of the min-cut-sets.

Proof: Let D be any set of network components, not necessarily a d-cut-set. We will show that

$$\Psi_{max}(G \backslash D) = min[\Phi(C_k \backslash D): k \in K_D] = min[\Phi(C_k \backslash D): k \in \{1, \dots, m\}]. \quad (4)$$

The first equality in (4) follows from *Lemma 3.1* and the Ford-Fulkerson theorem. For the proof of the second one let us note that for each $k \notin K_D$ there exists $j \in K_D$ such that $(C_1 \backslash D) \subseteq (C_k \backslash D)$, hence $\Phi(C_1 \backslash D) \leq \Phi(C_k \backslash D)$ which means that each set redundant in $(C_1 \backslash D), \dots, (C_m \backslash D)$ is irrelevant for determining the second minimum in (4). Let

$$k^*(D) = arg\, min_{k \in \{1, \dots, m\}} \Phi(C_k \backslash D) \quad (5)$$

i.e. $k^*(D)$ is one of these k for which $\Phi(C_k \backslash D)$ attains its minimum over $k \in \{1, \dots, m\}$. Replacing D with $C_{k^*(D)} \cap D$ in (4) yields:

$$\Psi_{max}\left[G \backslash \left(C_{k^*(D)} \cap D\right)\right] = min_{k \in \{1, \dots, m\}} \Phi\left[C_k \backslash \left(C_{k^*(D)} \cap D\right)\right]. \quad (6)$$

Let us note that the right hand side of (6) fulfills the following inequality:

$$min_{k \in \{1, \dots, m\}} \Phi\left[C_k \backslash \left(C_{k^*(D)} \cap D\right)\right] \leq \Phi\left[C_{k^*(D)} \backslash (C_{k^*(D)} \cap D)\right]. \quad (7)$$

In turn, the identity $P \backslash Q = P \backslash (P \cap D)$, where P and Q are arbitrary sets, yields:

$$\Phi\left[C_{k^*(D)}\backslash(C_{k^*(D)} \cap D)\right] = \Phi\left(C_{k^*(D)}\backslash D\right). \quad (8)$$

From (6), (7) and (8) we obtain:

$$\Psi_{max}\left[G\backslash\left(C_{k^*(D)} \cap D\right)\right] \leq \Phi\left(C_{k^*(D)}\backslash D\right). \quad (9)$$

Let now D be a d-cut-set. This assumption along with (4), (5) and (9) yield:

$$\begin{aligned} & d > \Psi_{max}(G\backslash D) = \Phi(C_{k^*(D)}\backslash D) \\ & \geq \Psi_{max}\left[G\backslash\left(C_{k^*(D)} \cap D\right)\right] \end{aligned} \quad (10)$$

which means that $C_{k^*(D)} \cap D$ is a d-cut-set. Thus, $C_{k^*(D)}$ is the sought min-cut-set, Q.E.D.

Theorem 1
Each min-d-cut-set is a subset of a certain min-cut-set, i.e. If D is a min-d-cut-set, then there exists C_k, such that $D \subseteq C_k, k \in \{1, \dots, m\}$. More precisely, $D \subseteq C_{k^*(D)}$, where $k^*(D)$ is defined by (6). It also holds that $\Phi\left(C_{k^*(D)}\backslash D\right) < d$.

Proof: If D is a min-d-cut-set, then, by *Lemma 3.2*, $C_{k^*(D)} \cap D$ is a d-cut-set. Since D is a min-d-cut-set and $C_{k^*(D)} \cap D \subseteq D$, it holds that $C_{k^*(D)} \cap D = D$, because there cannot be a d-cut-set smaller than D. In consequence $D \subseteq C_{k^*(D)}$. The postulated inequality follows from (10).

Theorem 2
Let D be a subset of a min-cut-set, i.e. $D \subseteq C_k$, for a certain $k \in \{1, \dots, m\}$.. Then D is a min-d-cut-set, externally non-redundant in C_k, if and only if the following conditions hold:

1. $\Phi(C_k\backslash D) < d$,
2. $\Phi(C_k\backslash D') \geq d$ for each $D' \subseteq D$,
3. $\Phi\left(C_j\backslash D\right) \geq d$ for each $j < k$, $k \geq 2$.

Such a set D is referred to as a min-d-cut-set obtained from C_k.

Proof: We first prove the rightward implication. Let D be a min-d-cut-set non-redundant w.r.t. any min-d-cut-set which is a subset of C_j, $j < k$. Assuming that *Condition 1* does not hold, i.e. $\Phi(C_k\backslash D) \geq d$, for each $j > k$ we have:

$$\begin{aligned} & \Phi\left(C_j\backslash D\right) = \Phi\left(C_j\backslash C_j \cap D\right) \\ & = \Phi\left(C_j\right) - \Phi\left(C_j \cap D\right) \geq \Phi(C_k) - \Phi(D) \\ & = \Phi(C_k\backslash D) \geq d. \end{aligned} \quad (11)$$

The first two equalities in (11) are due to the fact that $C_j\backslash D = C_j\backslash C_j \cap D$ and $C_j \cap D \subseteq C_j$. The first inequality in (11) is a consequence of the second preliminary assumption in Section 2 and the fact that $\Phi\left(C_j \cap D\right) \leq \Phi(D)$. The third equality is due to the inclusion $D \subset C_k$. However, since D is a d-cut-set, by virtue of (4) we have:

$$\begin{aligned} & \Psi_{max}(G\backslash D) \\ & = min[\Phi(C_1\backslash D), \dots, \Phi(C_m\backslash D)] < d, \end{aligned} \quad (12)$$

hence it must hold that $\Phi\left(C_{j^*}\backslash D\right) < d$ for a certain $j^* < k$. In consequence,

$$\Phi\left(C_{j^*}\backslash D\right) = \Phi\left(C_{j^*}\backslash C_{j^*} \cap D\right) < d \quad (13)$$

which means that

$$\begin{aligned} & \Psi_{max}(G\backslash D') \\ & = min[\Phi(C_1\backslash D'), \dots, \Phi(C_m\backslash D')] < d, \end{aligned} \quad (14)$$

where $D' = C_{j^*} \cap D$. Formula (14) and the inclusion $D' \subseteq D$ yield that D is redundant w.r.t. D' which is a subset of C_{j^*}. This contradicts the non-redundancy assumption. Thus, *Condition 1* holds, i.e. $\Phi(C_k\backslash D) < d$. Let us note that *Condition 3* is also fulfilled, because, as just shown, it cannot hold that $\Phi\left(C_j\backslash D\right) < d$ for a certain $j < k$. Finally, *Condition 2* is fulfilled too, because otherwise (14) would hold for a certain $D' \subset D$ and D would be redundant w.r.t. D'.
Let us now prove the opposite implication. *Condition 1* and formula (4) yield that

$$\Psi_{max}(G\backslash D) \leq \Phi(C_k\backslash D) \leq d \quad (15)$$

i.e. D is a d-cut-set in G. In order to prove that D is minimal it has to be shown that $\Psi_{max}(G\backslash D') \geq d$ for each $D' \subset D$. Let us take any D' such that $D' \subset D$. Formula (4) yields:

$$\Psi_{max}(G\backslash D') = min_{j\in\{1,\dots,m\}}\Phi\left(C_j\backslash D'\right). \quad (16)$$

Let $j < k$. Since $D' \subset D$, by virtue of *Condition 3* we have:

$$\Phi\left(C_j\backslash D'\right) \geq \Phi\left(C_j\backslash D\right) \geq d \quad (17)$$

while *Condition 2* implies that

$$\Phi\left(C_j\backslash D'\right) \geq d. \quad (18)$$

Finally, with regard to (18), for $j > k$ we have:

$$\begin{aligned} & \Phi\left(C_j\backslash D'\right) = \Phi\left(C_j\backslash C_j \cap D'\right) \\ & = \Phi\left(C_j\right) - \Phi\left(C_j \cap D'\right) \geq \Phi(C_k) - \Phi(D') \\ & = \Phi(C_k\backslash D') \geq d. \end{aligned} \quad (19)$$

Formula (19) is justified analogously to (11). Formulas (16)-(19) yield that $\Psi_{max}(G\backslash D') \geq d$, hence, in view of (15), D is a min-d-cut-set in G.
It now remains to show the external non-redundancy of D. Let D' be an externally non-redundant min-d-cut-set included in a certain C_j, $j < k$. If we assume that D is redundant w.r.t. D', then $D = D'$ ($D' \subset D$ cannot hold, because D would not be minimal). Also, $\Phi(C_j\backslash D') < d$, because D' is included in C_j, hence it fulfills *Condition 1* as shown in the first part of the proof. Thus, $\Phi(C_j\backslash D) = \Phi(C_j\backslash D') < d$ which contradicts *Condition 3*. If D' is an externally redundant min-d-cut-set included in C_j, then it is also included in a certain $C_{j'}$, $j' < j$, where it is non-redundant. Redundancy of D w.r.t. D' yields that $\Phi(C_{j'}\backslash D) = \Phi(C_{j'}\backslash D') < d$ which also contradicts *Condition 3*. This completes the whole *proof*.

Remark: Let us note that if D fulfills *Conditions 1* and *2*, then D is a m-d-c-s candidate in C_k. Also note that if *Condition 3* is not fulfilled, then D is externally redundant w.r.t. $C_j \cap D$ for a certain $j < k$. If, in turn, *Condition 3* holds, then D is an externally non-redundant min-d-cut-set.

Lemma 3.3
Let D be a subset of C_k such that $\Phi[(C_k\backslash D) \cup \mu_D] \geq d$, where μ_D is the component of D with the smallest capacity. It then holds that $\Phi[(C_k\backslash D') \geq \Phi[(C_k\backslash D) \cup \mu_D]$ for each $D' \subset D$.

Proof: Let $D' \subset D$. The lemma's assumptions yield that $\mu_D \notin C_k\backslash D$ and $D' \subset C_k$. Thus

$$\Phi[(C_k\backslash D) \cup \mu_D] = \Phi(C_k) - \Phi(D) + \varphi(\mu_D) \quad (20)$$

and

$$\Phi(C_k\backslash D') = \Phi(C_k) - \Phi(D'). \quad (21)$$

Also, $card(D') \leq card(D\backslash\mu_D)$ and D' is obtained from D by removing components with capacities no smaller than $\varphi(\mu_D)$, hence

$$\Phi(D') \leq \Phi(D) - \varphi(\mu_D). \quad (22)$$

Thus, finally, we have:

$$\Phi(C_k\backslash D') \geq \Phi(C_k) - \Phi(D) + \varphi(\mu_D) = \Phi[(C_k\backslash D) \cup \mu_D]. \quad (23)$$

Corollary: $\Phi(C_k\backslash D') \geq d$ for each $D' \subseteq D$ if and only if $\Phi[(C_k\backslash D) \cup \mu_D] \geq d$. Thus, D is a m-d-c-s candidate if $\Phi(C_k\backslash D) < d$ and $\Phi[(C_k\backslash D) \cup \mu_D] \geq d$. This means that *Condition 2* in *Theorem 2* can be replaced by the latter inequality.

Lemma 3.4
If $\Phi(C_j\backslash C_k) \geq d$, for a certain $j < k$, then $\Phi(C_j\backslash D) \geq d$ for each $D \subseteq C_k$.

Proof: If $D \subseteq C_k$, then $C_j\backslash C_k \subseteq C_j\backslash D$, hence $\Phi(C_j\backslash D) \geq \Phi(C_j\backslash C_k)$, and the lemma follows from the assumption that $\Phi(C_j\backslash C_k)] \geq d$.

Corollary: If the lemma's assumption holds and D is a m-d-c-s candidate in C_k, then the check if $\Phi(C_j\backslash D)] \geq d$ can be skipped, because it would give a positive result. Thus, *Condition 3* in *Theorem 2* only needs to be verified for j such that $\Phi(C_j\backslash C_k)] < d$.

Lemma 3.5
Let C_{k^*} be the set of all "large capacity" components in C_k, $k \in \{1, \ldots, m\}$. Then the following two statements hold:

1. Each m-d-c-s candidate obtained from C_k must include C_{k^*};
2. If C_{j^*} is a min-d-cut-set and $C_{j^*} \subseteq C_{k^*}$ for a certain $j < k$, then each m-d-c-s candidate obtained from C_k is redundant w.r.t. C_{j^*}.

Proof: Let us assume that D is a m-d-c-s candidate obtained from C_k and D does not include C_{k^*}. Then there exists at least one $c \in C_{k^*}\backslash D$ and, since $C_{k^*} \subseteq C_k$, it holds that $C_{k^*}\backslash D \subseteq C_k\backslash D$, hence

$$\Phi(C_k\backslash D) \geq \Phi(C_{k^*}\backslash D) \geq \varphi(c) \geq d. \quad (24)$$

In view of (24), D is not a m-d-c-s candidate, which contradicts the initial assumption. Thus *Statement 1* holds. Accordingly, if D is a m-d-c-s candidate obtained from C_k, then $C_{k^*} \subseteq D$. The assumption that $C_{j^*} \subseteq C_{k^*}$ yields $C_{j^*} \subseteq D$, hence D is redundant w.r.t. C_{j^*}, Q.E.D.

Corollary: *Lemma 3.5* allows to easily ascertain if each m-d-c-s candidate in C_k is redundant in regard to a m-d-c-s composed of large capacity components of C_j, $j < k$. Clearly, C_k is to be omitted in the process of generating min-d-cut-sets from min-cut-sets.

4. The considered method in outline

Now we will present in outline the two algorithms whose detailed pseudo-codes are given in the next two sections. We will also briefly explain why two algorithms have been developed instead of one.

From *Theorems 1* and *2* we conclude that, in order to find all the min-d-cut-sets of G, we have to find all the subsets of each C_k, $k = 1, \dots, m$, satisfying the three conditions in *Theorem 2*. In more detail, min-d-cut-sets are generated from each successive C_k, $k = 1, \dots, m$, with the use of *Alg. 1* if $\Phi(C_k) < 2d$, or *Alg. 2* if $\Phi(C_k) \geq 2d$. *Algorithm 1* arranges the components of C_k according to decreasing flow capacities, then it generates successive subsets of C_k and checks them for being min-d-cut-sets. In turn, *Algorithm 2* arranges the components of C_k according to increasing flow capacities, then it generates successive subsets of C_k and checks their complements for being min-d-cut-sets. The following subset generation policy is used: in step 1 the successive one-element subsets of C_k are generated (empty set is augmented with the successive elements of $B(C_k, \varnothing)$); in step $j > 1$ each j-element subset C of C_k such that $B(C_k, C) = \varnothing$ is augmented with the successive elements of $B(C_k, C)$, $j = 1, \dots, k-1$. For better understanding, let us apply the above policy to the set $C_1 = \{2, 3, 4, 5\}$ (see *Figure 1*). The following subsets are generated in the successive steps:

- step 1 – $\{2\}$, $\{3\}$, $\{4\}$, $\{5\}$;
- step 2 – $\{2, 3\}$, $\{2, 4\}$, $\{2, 5\}$;
  note that $B(C_1, \{2\}) = \{3, 4, 5\}$;
- step 3 – $\{3, 4\}$, $\{3, 5\}$;
  note that $B(C_1, \{3\}) = \{4, 5\}$;
- step 4 – $\{4, 5\}$;
  note that $B(C_1, \{4\}) = \{5\}$;
- step 5 – $\{2, 4, 5\}$, $\{2, 3, 5\}$, $\{2, 3\}$;
  note that $B(C_1, \{2, 3\}) = \{4, 5\}$;
- step 6 – $\{2, 4, 5\}$;
- step 7 – $\{3, 4, 5\}$;
- step 8 – $\{2, 3, 4, 5\}$.

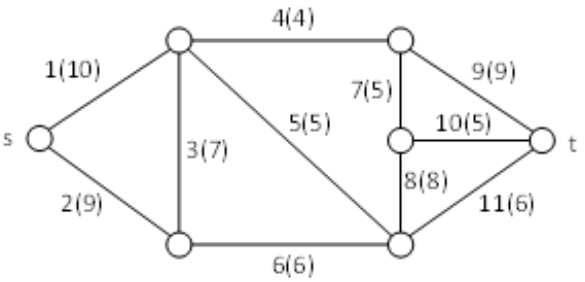

Figure 1. An example network system (the network structure is the same as in [2]). The links' capacities are given in parentheses next to the links' numbers

This standard procedure generates one instance of each subset of a finite set. However, if thus obtained subset of C_k is a min-d-cut-set, it is not further augmented, because non-minimal d-cut-sets would be generated. In consequence, the number of subsets of C_k generated by *Algorithm 1* or *2* is substantially smaller than $2^{card(C_k)} - 1$ which is the number of all non-empty subsets of C_k.

The min-cut-sets of the above network, ordered by the increasing flow capacity, are listed below.

- $C_1 = \{6, 5, 4\}$, $\Phi(C_1) = 15$,
- $C_2 = \{8, 11, 4\}\Phi(C_2) = 18$,
- $C_3 = \{1, 2\}$, $\Phi(C_3) = 19$,
- $C_4 = \{10, 11, 9\}$, $\Phi(C_4) = 20$,
- $C_5 = \{4,7,10,11\}$, $\Phi(C_5) = 20$,
- $C_6 = \{6,3,1\}$, $\Phi(C_6) = 23$,
- $C_7 = \{4,5,3,2\}$, $\Phi(C_7) = 25$,
- $C_8 = \{5,7,6,9\}$, $\Phi(C_8) = 25$,
- $C_9 = \{7,11,8,9\}$, $\Phi(C_9) = 28$,
- $C_{10} = \{5,10,6,8,9\}$, $\Phi(C_{10}) = 33$,
- $C_{11} = \{5,7,3,2,9\}$, $\Phi(C_{11}) = 35$,
- $C_{12} = \{5,11,3,8,1\}$, $\Phi(C_{12}) = 36$,
- $C_{13} = \{5,7,10,11,3,1\}$, $\Phi(C_{13}) = 38$,
- $C_{14} = \{5,10,3,8,2,9\}$, $\Phi(C_{14}) = 43$.

Let C be a currently generated subset of C_k, and let $D = C$ (*Algorithm 1*) or $D = C_k \backslash C$ (*Algorithm 2*). The check whether D is a min-d-cut-set is done as follows. First it is checked if $\Phi(C_k \backslash D) < d$ (*Condition 1* in *Theorem 2*). In case of a positive check, D is a d-cut-set in G (see first part of the proof of *Theorem 2*) and it is checked whether $\Phi[C_k \backslash (D \backslash \{\mu_D\})] \geq d$ (see corollary to *Lemma 3.3*). If this inequality holds, D is marked as a m-d-c-s candidate, i.e. in order to state whether D is a min-d-cut-set it has yet to be verified whether *Condition 3* in *Theorem 2* holds. It is important that once D turns out to be a m-d-c-s candidate, then C is not further augmented, because d-cut-sets larger than $D = C$ are not min-d-cut-sets (*Algorithm 1*), while subsets of $D = C_k \backslash C$ are not even d-cut-sets (*Algorithm 2*).

The different handling of the cases $\Phi(C_k) < 2d$ and $\Phi(C_k) \geq 2d$ allows for obtaining m-d-c-s candidates without first generating internally redundant d-cut-sets (*Algorithm 1*), or generating only a small number of them before a m-d-c-s candidate is obtained (*Algorithm 2*). Also, it ensures that only a small number of non-d-cut-sets is generated by both algorithms. Other advantages of this distinction will be given in Sections 4 and 5.

The above outlined method is an improvement of the procedures presented in [2] and [5], also in the context of internal and external redundancy – the concepts introduced by the authors of the aforementioned papers, whose definitions are given in

Notation Section. It is essential to check whether a newly generated subset of C_k is internally or externally redundant, because if it occurs to be a min-d-cut-set, then its supersets need not be generated, since none of them is a min-d-cut set. Let us note that *Condition 3* in *Theorem 2* is a criterion used for checking the absence of external redundancy. Indeed, let us suppose that *Condition 3* does not hold, i.e. D is a d-cut-set obtained from C_k and $\Phi(C_j \backslash D) < d$ for a certain $j < k$. Since $C_j \backslash D = C_j \backslash (C_j \cap D)$, it follows from (4) that $\Psi_{max}(G \backslash (C_j \cap D)) < d$. Moreover, $C_j \cap D \subseteq C_j$, thus $C_j \cap D$ is a d-cut set obtained from C_j. Thus, the inclusion $C_j \cap D \subseteq D$ implies that D is externally redundant w.r.t. $C_j \cap D$. In consequence, D is not a min-d-cut-set, or D is a duplicate one. Let us also note that, in view of *Condition 3* in *Theorem 2*, it is only required to compare a d-cut-set obtained from C_k with C_j, $j < k$, but not with C_j, $j > k$, in order to state whether it is a min-d-cut-set.

5. Generating min-d-cut-sets in decreasing flow capacities

The case $\Phi(C_k) < 2d$ is handled by *Algorithm 1* which requires that the components of C_k be ordered by decreasing flow capacities. *Algorithm 1* is based on the following four lemmas.

Lemma 4.1
Let C be a subset of C_k such that

1. $\Phi(C_k \backslash C) \geq d$,
2. $\Phi(C_k \backslash (C \cup \{b\})) < d$,
   for a certain $b \in B(C_k, C)$.

$C \cup \{b\}$ is then a min-d-cut-set candidate in C_k. The second assumption says that the failure of all components in $C \cup \{b\}$ causes the capacity of C_k to fall below value d. The lemma is also valid for $C = \varnothing$, in which case $\{b\}$ is a one-element min-d-cut-set candidate.

Proof: Let us put $D = C \cup \{b\}$ and note that, due to the ordering of components in C_k, b is the component of D with the smallest capacity. Now *Lemma 4.1* is a consequence of corollary to *Lemma 3.3*.

Remark: *Lemma 4.1* provides a simple criterion for stating whether $C \cup \{b\}$, where C is a subset of C_k, is a m-d-c-s candidate in C_k. Let us note that the lemma's assumptions are equivalent to the first two conditions in *Theorem 2*, where $D = C \cup \{b\}$.

Lemma 4.2
Let C be a subset of C_k such that $\Phi[A(C_k, C)] \geq d$. It then holds that $\Phi[C_k \backslash (C \cup C')] \geq d$ for each $C' \subseteq B(C_k, C)$, i.e. no superset of C is a m-d-c-s candidate in C_k.

Proof: If $C' \subseteq B(C_k, C)$ then

$$A(C_k, C) \subseteq C_k \backslash (C \cup C'),$$

because $A(C_k, C) \subseteq C_k$ and $A(C_k, C)$ has no common elements with either C or C'. Thus

$$\Phi[C_k \backslash (C \cup C')] \geq \Phi[A(C_k, C)] \geq d,$$

which ends the proof.

Remark: Lemma 4.2 provides a simple way to check whether a subset of C_k is non-augmentable, thus allowing to reduce the number of subsets generated from C_k.

Lemma 4.3
Let $b_1, b_2 \ldots$ be the consecutive elements of $B(C_k, C)$. Then $\Phi[C_k \backslash (C \cup \{b_i\})]$ is a non-decreasing sequence with respect to $i \geq 1$.

Proof: The lemma follows from the fact that $\varphi(b_i)$, $i \geq 1$, is a non-increasing sequence.

Corollary: if $\Phi[C_k \backslash (C \cup \{b_i\})] \geq d$ then $\Phi[C_k \backslash (C \cup \{b_j\})] \geq d$ for $j > i$. This property allows not to compute the latter capacities. *Lemma 4.3* is not referenced in *Algorithm 1*, and its role is further explained in the example following *Algorithm 1*.

Lemma 4.4
If $\Phi(C_k) < 2d$ and $\varphi(c^*) \geq d$ for certain $c^* \in C_k$ (i.e. c^* is a large capacity component), then $\{c^*\}$ is the only m-d-c-s candidate in C_k.

Proof: From the assumptions we have:

$$\Phi(C_k \backslash \{c^*\}) \leq \Phi(C_k) - \varphi(c^*) < 2d - d = d. \quad (25)$$

The second preliminary assumption in Notation Section yields that

$$\Phi(C_k \backslash \varnothing) = \Phi(C_k) \geq \Phi(C_1) = \Psi_{\max}(G) \geq d. \quad (26)$$

Since $\varnothing$ is the only subset of $\{c^*\}$, (25) and (26) yield that $\{c^*\}$ is a m-d-c-s candidate in C_k. Let us suppose that D is a m-d-c-s candidate in C_k, and $D \neq \{c^*\}$. D must include c^*, because otherwise c^*

would belong to $C_k \backslash D$, the inequalities $\Phi(C_k \backslash D) \geq \varphi(c^*) \geq d$ would hold, and D would not be a m-d-c-s candidate. In turn, (25) implies that for $D' = \{c^*\} \subset D$ we have:

$$\Phi(C_k \backslash D') < d \tag{27}$$

thus D does not fulfill the second criterion to be a m-d-c-s candidate. This ends the proof.

If $\Phi(C_k) < 2d$ then, based on the *Lemmas 4.1-4.4*, and the rules from Section 4, min-d-cut-sets are obtained from C_k by the following algorithm:

Algorithm 1

For k = 1 to max[k: $\Phi(C_k) < 2d$] do
  Check if $\Phi(\{c_{1,k}\}) \geq d$, where $c_{1,k}$ is the first element of C_k. If so, mark $\{c_{1,k}\}$ as m-d-c-s candidate (lemma 4.4), check it for ER*, and pass to the next k;
  $D \leftarrow \varnothing$;
  For j = 0 to $|C_k| - 1$ do
    If each j-element set generated from C_k is marked, pass to the next k;
    Else
      For each unmarked j-element set C generated from C_k do
        For the successive $b \in B(C_k, C)$ do
          $D \leftarrow C \cup \{b\}$ (augment C with $\{b\}$);
          In the case "$\Phi(C_k \setminus D) < d$" do
            mark D as m-d-c-s candidate (lemma 4.1), check it for ER*, and pass to next b or to next j-element C (if b is the last element of C_k);
          In the case "$\Phi(C_k \setminus D) \geq d$" do
            If b is the last element of C_k, mark D as non-augmentable (D cannot be augmented, because $B(C_k, D) = \varnothing$) and pass to next j-element C;
            If $\Phi[A(C_k, D)] \geq d$, mark D as non-augmentable (lemma 4.2) and pass to next j-element C;

Remark to the command marked with asterisk:

ER denotes external redundancy. If ER occurs, i.e. $\Phi(C_j \backslash D) < d$ for a certain j such that $j < k$ and $\Phi(C_j \backslash C_k) < d$, then D is externally redundant w.r.t. $Cj \cap D$. Otherwise D is a min-d-cut-set, non-externally redundant in C_k (see *Lemma 3.4* and *Theorem 2* with the remark following it).

It should be noted that no internally redundant d-cut-sets are generated by *Algorithm 1*, because its logic and *Lemma 4.1* yield that if the obtained set D is a d-cut-set, then D is a m-d-c-s candidate. Although *Algorithm 1* can generate non-d-cut-sets (the case $\Phi(C_k \backslash D) \geq d$), their number is small due to the condition $\Phi(C_k) < 2d$.

Let us now trace the flow of *Algorithm 1* for the example network in *Figure 1*. Let $d = 10$. There are three min-cut-sets with capacities lower than $2d$, i.e. $C_1 = \{6,5,4\}$, $C_2 = \{8,11,4\}$ and $C_3 = \{1,2\}$ with $\Phi(C_1) = 15$, $\Phi(C_2) = 18$, $\Phi(C_3) = 19$. The results of the successive operations are presented in the tables below – one table for each k. The markings used in the 4-th and 6-th column have the following meanings:

- * – D is an externally non-redundant min-d-cut-set (thus D is non-augmentable);
- # – D is non-augmentable, because $\Phi[A(C_k, D)]$ exceeds d (see *Lemma 4.2*);
- | – D is non-augmentable, because b is the last component in C_k. This marking is only used if * and # do not apply;
- × – cell with a value that need not be computed.

Starting from $k = 2$, the values of $\Phi(C_j \backslash C_k)$, $j < k$, are listed at the top of each table in order to indicate which C_j can be omitted when D is checked for ER redundancy, i.e. when *Condition 3* in *Theorem 2* is verified. C_j is omitted in the following three cases:

1. $\Phi(C_j \backslash C_k) \geq d$ (*Lemma 3.4*),
2. $C_i^* \subseteq C_j^*$ for some $i < j$ and C_i^* is a min-d-cut-set (*Lemma 3.5*),
3. the above two conditions do not hold, but all m-d-c-s candidates obtained from C_j are redundant (e.g. $j = 6$ or $j = 10$ as shown in Section 6).

It is important that the number of j fulfilling the last condition grows rapidly as k increases, thus, in practice, not a large number of ER checks has to be performed.

Remarks:

1. As follows from *Lemma 4.3*, if $\Phi(C_k \backslash (C \cup \{b\})) \geq d$ for a certain $b \in B(C_k, C)$, then $\Phi[C_k \backslash (C \cup \{b^+\})] \geq d$ for each b^+ that succeeds b in $B(C_k, C)$, thus

there is no need to compute $\Phi[C_k\backslash(C \cup \{b^+\})]$, because we only need to know if it is greater or equal to d.

2. As follows from *Algorithm 1*, $\Phi[\mathrm{A}(C_k\backslash D)]$ is only computed if $\Phi(C_k\backslash D) \geq d$.

Table 1. Output of *Algorithm 1* for $k = 1$

j	C	b	$D = C \cup \{b\}$	$\Phi(C_k\backslash D)$	$\Phi[\mathrm{A}(C_k\backslash D)]$
0	∅	6	{6}*	9	×
0	∅	5	{5}	10	6
0	∅	4	{4}# (L. 4.2)	≥10 (L. 4.3)	11
1	{5}	4	{5,4}*	6	×

Table 2. Output of *Algorithm 1* for $k = 2$:
$\Phi(C_1\backslash C_2) = 11$ (ER check not needed)

j	C	b	$D = C \cup \{b\}$	$\Phi(C_k\backslash D)$	$\Phi[\mathrm{A}(C_k\backslash D)]$
0	∅	8	{8}	10	0
0	∅	11	{11}	≥10 (L. 4.3)	8
0	∅	4	{4}# (L. 4.2)	≥10 (L. 4.3)	14
1	{8}	11	{8,11}*	4	×
1	{8}	4	{8,4}*	6	×
1	{11}	4	{11,4}*	8	×

$k = 3$: $\Phi(C_1\backslash C_3) = 15$, $\Phi(C_2\backslash C_3) = 18$ (ER check not needed).
Output of *Algorithm 1* for $k = 3$:
{1}*.

Remark: According to *Lemma 4.4*, {1}* is the only m-d-c-s candidate obtained from C_3.

6. Generating min-d-cut-sets in increasing flow capacities

The case $\Phi(C_k) \geq 2d$ is handled by *Algorithm 2* which requires that the components of C_k be ordered according to increasing flow capacities. *Algorithm 2* is based on the following four lemmas.

Lemma 5.1
Let C be a subset of C_k such that

1. $\Phi(C) < d$,
2. $\Phi[C \cup \mu(C_k\backslash C)] \geq d$.

Then $(C_k\backslash C)$ is a m-d-c-s candidate in C_k.

Proof: Let us put $D = C_k\backslash C$ and note that, due to the ordering of components in C_k, $\mu(C_k\backslash C)$ is the component of D with the smallest capacity. Now *Lemma 5.1* is a consequence of corollary to *Lemma 3.3*.

Remark: *Lemma 5.1* provides a simple criterion for stating whether $C_k\backslash C$, where C is a subset of C_k, is a m-d-c-s candidate in C_k. Let us note that the lemma's assumptions are equivalent to the first two conditions in *Theorem 2*, where $D = C_k\backslash C$.

Lemma 5.2
If the components of C_k are ordered according to increasing capacities, C is a j-element subset of C_k, C is composed of consecutive elements of C_k, and $\Phi(C) \geq d$, then $\Phi(C') \geq d$ for each j-element C' generated subsequently to C.

Proof: Let $C = \{c_1, \dots, c_j\}$ where $c_1, \dots, c_j$ are consecutive components of C_k, and $C' = \{c'_1, \dots, c'_j\}$ be a j-element set generated subsequently to D. First, it will be proved by induction that $c'_1 > c_1$. This fact is obvious for $j = 1$, in which case $c'_1 > c_1$, and let us assume that it holds for a certain $j \geq 1$. Let $C^+ = \{c_1^+, \dots, c_{j+1}^+\}$ be a $(j+1)$-element subset of C_k composed of its consecutive components. Clearly, C^+ is the first $(j+1)$-element set obtained by augmenting $\{c_1^+, \dots, c_j^+\}$, and, according to the subset generating policy, each subsequent $(j+1)$-element set $C^{+\prime}$, is obtained by augmenting either $\{c_1^+, \dots, c_j^+\}$ or $\{c_1^{+\prime}, \dots, c_j^{+\prime}\}$ which is one of j-element sets generated subsequently to $\{c_1^+, \dots, c_j^+\}$. Thus, the first element of $C^{+\prime}$ is equal either to c_1^+ or to $c_1^{+\prime}$. The induction assumption yields that $c_1^{+\prime} > c_1^+$, hence the first element of $C^{+\prime}$ is greater or equal to c_1^+, which means that the fact to be proved holds for $j + 1$, and, in consequence, for any $j \geq 1$.
Now we can pass to the proper proof. Since the components of C and C' are ordered according to increasing flow capacities, we have:

$$\varphi(c_1) \leq \cdots \leq \varphi(c_j) \tag{28}$$

and, in view of the fact proved above,

$$\varphi(c_1) \leq \varphi(c'_1) \leq \cdots \leq \varphi(c'_j). \tag{29}$$

As $c_1, \dots, c_j$ are consecutive components of C_k, from (28) and (29) it follows that $\varphi(c'_1) \geq \varphi(c_1)$,…,

$\varphi(c'_j) \geq \varphi(c_j)$ hence $\Phi(C') \geq \Phi(C)] \geq d$, q.e.d.

Corollary: If C fulfills the assumptions of *Lemma 5.2*, then the capacities of all j-element subsets of C_k generated subsequently to C are greater or equal to d, hence the complements (w.r.t. C_k) of these subsets are not m-d-c-s candidates.

Lemma 5.3
If the components of C_k are ordered according to increasing capacities, and b_x is the first element of $B(C_k, C) = \{b_1, b_2, \ldots\}$ such that $\Phi[C \cup b_x] \geq d$, then the capacities of all one-element augmentations of C subsequent to $C \cup \{b_x\}$ are greater or equal to d, i.e. $\Phi[C \cup b_y] \geq d$, where $x \leq y \leq card[B(C_k, C)]$.

Proof: The lemma's assumptions yield that $\Phi[C \cup b_y] \geq \Phi[C \cup b_x] \geq d$, q.e.d.

Corollary: If C fulfills the assumptions of *Lemma 5.3*, then the complements (w.r.t. C_k) of all one-element augmentations of C subsequent to $C \cup \{b_i\}$ are not m-d-c-s candidates.

Lemma 5.4
Let $B(C_k, C) = \{b_1, b_2, \ldots\}$, where b_i, $i \geq 1$, are ordered according to increasing capacities. Then $\Phi[C \cup b_i \cup \mu[C_k \backslash (C \cup b_i)]]$ is non-decreasing with respect to $i \geq 1$.

Proof: We will consider two cases: $A(C_k, C) = \varnothing$ and $A(C_k, C) = \varnothing$. In the first case we have:

$$\mu[C_k \backslash (C \cup b_1)] = b_2, \tag{30}$$

$$\mu[C_k \backslash (C \cup b_i)] = b_1,\ i \geq 2. \tag{31}$$

It thus follows that

$$\Phi\big[C \cup b_1 \cup \mu[C_k \backslash (C \cup b_1)]\big] = \Phi(C) + \varphi(b_1) + \varphi(b_2), \tag{32}$$

$$\Phi\big[C \cup b_2 \cup \mu[C_k \backslash (C \cup b_2)]\big] = \Phi(C) + \varphi(b_2) + \varphi(b_1), \tag{33}$$

$$\Phi\big[C \cup b_i \cup \mu[C_k \backslash (C \cup b_i)]\big] = \Phi(C) + \varphi(b_i) + \varphi(b_1),\ i \geq 3. \tag{34}$$

In the second case, i.e. $A(C_k, C) = \varnothing$, we have:

$$\mu[C_k, C \cup \{b_i\}] = a_1,\ i \geq 1, \tag{35}$$

where a_1 is the first element of $A(C_k, C)$. In consequence

$$\Phi\big[C \cup b_i \cup \mu[C_k \backslash (C \cup b_i)]\big] = \Phi(C) + \varphi(b_i) + \varphi(a_1),\ i \geq 1. \tag{36}$$

Since $\varphi(b_i)$ is non-decreasing in i, $i \geq 1$, the lemma's thesis follows from (34) and (36).

Corollary: If b is the first element in $B(C_k, C)$ such that $\Phi\big[C \cup b \cup \mu[C_k \backslash (C \cup b)]\big] \geq d$, b^+ succeeds b in $B(C_k, C)$, and $C^+ = C \cup b^+$, then $\Phi\big[C^+ \cup \mu[C_k \backslash C^+]\big] \geq d$. Thus, the last inequality need not be checked in order to ascertain, as per *Lemma 5.1*, whether $C_k \backslash C^+$ is a m-d-c-s candidate. *Lemma 5.4* is not referenced in *Algorithm 2*, and its role is further explained in the example following

Algorithm 2.
If $\Phi(C_k) \geq 2d$, then, based on the *Lemmas 5.1-5.3*, and the rules from Section 4, min-d-cut-sets are obtained from C_k by the following algorithm:

<u>*Algorithm 2*</u>
For k = min [k: $\Phi(C_k) \geq 2d$] to m do
  If C_k fulfills lemma 3.5 pass to the next k;
  $D \leftarrow \varnothing$;
  For j = 0 to $|C_k| - 1$ do
    If each j-element set generated from C_k is marked, pass to the next k;
    For each unmarked j-element C generated from C_k do *
      For each $b \in B(C_k, C)$, where $\{b\}$ is unmarked and $\varphi(b) < d$, do *
        $C^+ \leftarrow C \cup \{b\}$ (augment C with $\{b\}$);
        In the case "$\Phi(C^+) < d$" do
          If $\Phi[\ C^+ \cup \mu(C_k \setminus C^+)\ ] \geq d$, mark C^+ as non-augmentable, mark $C_k \setminus C^+$ as m-d-c-s candidate, and check it for ER; **
        In the case "$\Phi(C^+) \geq d$" do
          Mark C^+ as non-augmentable;
          If C^+ is composed of consecutive elements of C_k, pass to the next j; ***
          Pass to the next j-element C; ****

Remarks to the commands marked with asterisks:

- adding any b to a marked set C, or adding b such that $\varphi(b) \geq d$ to any C, yields C^+ such that $\Phi(C^+) \geq d$, i.e. $C_k \backslash C^+$ is not a m-d-c-s candidate;
- ** see *Lemma 5.1*;
- *** the capacities of all $(j+1)$-elements subsets of C_k generated subsequently to C^+ would be greater or equal to d (see *Lemma 5.2*);
- **** the capacities of subsequent one-element augmentations of C would be greater or equal to d (see *Lemma 5.3*).

It should be noted that *Algorithm 2* generates as few non-d-cut-sets as possible (the case $\Phi(C^+) \geq d$), and although it can generate internally redundant d-cut-sets (it happens if $\Phi(C^+) < d$ and $\Phi[C^+ \cup \mu[C_k \backslash C^+]] < d$), their number is small due to the condition $\Phi(C_k) \geq 2d$. Furthermore, the IR check is done instantaneously (it is positive if $\Phi[C^+ \cup \mu[C_k \backslash C^+]] < d$) and no comparison with the previously found d-cut-sets is required.

Algorithm 2 will be illustrated by applying it to the network in *Figure 1*. The results of the successive operations are presented in tables, one table for each $k = 4, \ldots, m$. The algorithm starts with $k = 4$, the first k for which $\Phi(C_k) \geq 2d$. The markings used in the 4-th and 7-th column have the following meanings:

- * – $C_k \backslash C^+$ is an externally non-redundant min-d-cut-set;
- ** – $C_k \backslash C^+$ is an ER m-d-c-s candidate;
- # – C^+ fulfills the assumptions of *Lemma 5.2* or *5.3*;
- | – b is the last component of C_k (this marking is only used if * and # do not apply);
- × – cell with a value that need not be computed.

At the top of each table the values of $\Phi(C_j \backslash C_k)$, $j < k$, are listed in order to indicate which C_j can be omitted when D is checked for ER redundancy, i.e. when *Condition 3* in *Theorem 2* is verified.

Remark: As follows from *Lemma 5.4*, if $\Phi[C^+ \cup \mu[C_k \backslash C^+]] \geq d$, where $C^+ = C \cup \{b\}$, then $\Phi[(C \cup b^+) \cup \mu[C_k(C \cup b^+)]] \geq d$, where b^+ succeeds b in $B(C_k, C)$. Thus, there is no need to compute $\Phi[C_k \backslash (C \cup \{b\})]$, because we only need to know if it is greater or equal to d.

$k = 6$: Each m-d-c-s obtained from C_6 is redundant (*Lemma 3.5*).

Output of *Algorithm 2* for $k = 6$: none.

Table 3. Output of *Algorithm 2* for $k = 4$: $\Phi(C_1 \backslash C_4) = 15$, $\Phi(C_2 \backslash C_4) = 12$, $\Phi(C_3 \backslash C_4) = 19$, (ER check not needed)

j	C	b	C^+	$\Phi(C^+)$	$\Phi[C^+ \cup \mu[C_k \backslash C^+]]$	$D = C_k \backslash C^+$
0	∅	10	{10}*	5	11	{11,9}*
0	∅	11	{11}*	6	≥11 (L. 5.4)	{10,9}*
0	∅	9	{9}*	9	≥11 (L. 5.4)	{10,11}*

Table 4. Output of *Algorithm 2* for $k = 5$: $\Phi(C_1 \backslash C_5) = 11$, $\Phi(C_2 \backslash C_5) = 12$, $\Phi(C_3 \backslash C_5) = 19$, $\Phi(C_4 \backslash C_5) = 9$ (check ER for $k = 4$)

j	C	b	C^+	$\Phi(C^+)$	$\Phi[C^+ \cup \mu[C_k \backslash C^+]]$	$D = C_k \backslash C^+$
0	∅	4	{4}	4	9	Not m-d-c-s
0	∅	7	{7}	5	9	Not m-d-c-s
0	∅	10	{10}	5	9	Not m-d-c-s
0	∅	11	{11}*	6	10	{4,7,10}*
1	{4}	7	{4,7}*	9	14	{10,11}**
1	{4}	10	{4,10}*	9	≥14 (L. 5.4)	{7,11}*

Table 5. Output of *Algorithm 2* for $k = 7$: $\Phi(C_1 \backslash C_7) = 6$, $\Phi(C_2 \backslash C_7) = 14$, $\Phi(C_3 \backslash C_7) = 10$, $\Phi(C_4 \backslash C_7) = 20$, $\Phi(C_5 \backslash C_7) = 16$ omit C_6 (each m-d-c-s obtained from C_6 is redundant); check ER for $k = 1$

j	C	b	C^+	$\Phi(C^+)$	$\Phi[C^+ \cup \mu[C_k \backslash C^+]]$	$D = C_k \backslash C^+$
0	∅	4	{4}	4	9	Not m-d-c-s
0	∅	5	{5}	5	9	Not m-d-c-s
0	∅	3	{3}*	7	11	{4,5,2}**
0	∅	2	{2}*	9	≥11 (L. 5.4)	{4,5,3}**
1	{4}	5	{4,5}*	9	16	{3,2}*

Summing up, the following min-d-cut-sets ($d = 10$) in the network from *Figure 1* have been found by *Algorithms 1* and *2*: {6}, {5,4}, {8,11}, {8,4}, {11,4}, {1}, {11,9}, {10,9}, {10,11}, {4,7,10}, {7,11}, {3,2}, {5,7,9}, {7,8,9}. They are listed in the same order in which they have been generated.

Table 6. Output of *Algorithm 2* for $k = 8$:
$\Phi(C_1\backslash C_8) = 4,\ \Phi(C_2\backslash C_8) = 18,\ \Phi(C_3\backslash C_8) = 19,$
$\Phi(C_4\backslash C_8) = 11,\ \Phi(C_5\backslash C_8) = 15,\ \Phi(C_7\backslash C_8) = 20$
omit C_6; check ER for $k = 1$

j	C	b	C^+	$\Phi(C^+)$	$\Phi[C^+ \cup \mu[C_k\backslash C^+]]$	$D = C_k \backslash C^+$
0	∅	5	{5}*	5	10	{7,6,9}**
0	∅	7	{7}*	5	≥10 (L.5.4)	{5,6,9}**
0	∅	6	{6}*	6	≥10 (L.5.4)	{5,7,9}*
0	∅	9	{9}*	9	≥10 (L.5.4)	{4,5,3}**

Table 7. Output of *Algorithm 2* for $k = 9$:
$\Phi(C_1\backslash C_9) = 15,\ \Phi(C_2\backslash C_9) = 4,\ \Phi(C_3\backslash C_9) = 19,$
$\Phi(C_4\backslash C_9) = 5,\ \Phi(C_5\backslash C_9) = 9,\ \Phi(C_7\backslash C_9) = 25,$
$\Phi(C_8\backslash C_9) = 11$ omit C_6; check ER for $k = 2,4,5$

j	C	b	C^+	$\Phi(C^+)$	$\Phi[C^+ \cup \mu[C_k\backslash C^+]]$	$D = C_k \backslash C^+$
0	∅	7	{7}*	5	11	{11,8,9}**
0	∅	11	{11}*	6	≥11 (L.5.4)	{7,8,9}*
0	∅	8	{8}*	8	≥11 (L.5.4)	{7,11,9}**
0	∅	9	{9}*	9	≥11 (L.5.4)	{7,11,8}**

Table 8. Output of *Algorithm 2* for $k = 10$:
$\Phi(C_1\backslash C_{10}) = 4,\ \Phi(C_2\backslash C_{10}) = 10,$
$\Phi(C_3\backslash C_{10}) = 19, \Phi(C_4\backslash C_{10}) = 6,$
$\Phi(C_5\backslash C_{10}) = 9, \Phi(C_7\backslash C_{10}) = 20,$
$\Phi(C_8\backslash C_{10}) = 5, \Phi(C_9\backslash C_{10}) = 11$ omit C_6; check ER for $k = 1,4,5,8$

j	C	b	C^+	$\Phi(C^+)$	$\Phi[C^+ \cup \mu[C_k \backslash C^+]]$	$D = C_k\backslash C^+$
0	∅	5	{5}*	5	10	{10,6,8,9}**
0	∅	10	{10}*	5	≥10 (L.5.4)	{5,6,8,9}**
0	∅	6	{6}*	6	≥10 (L.5.4)	{5,10,8,9}**
0	∅	8	{8}*	8	≥10 (L.5.4)	{5,10,6,9}**
0	∅	9	{9}*	9	≥10 (L.5.4)	{5,10,6,8}**

Table 9. Output of *Algorithm 2* for $k = 11$:
$\Phi(C_1\backslash C_{11}) = 10,\ \Phi(C_2\backslash C_{11}) = 18,$
$\Phi(C_3\backslash C_{11}) = 10,\ \Phi(C_4\backslash C_{11}) = 16,$
$\Phi(C_5\backslash C_{11}) = 15,\ \Phi(C_7\backslash C_{11}) = 4,$
$\Phi(C_8\backslash C_{11}) = 6,\ \Phi(C_9\backslash C_{11}) = 14,$ omit C_6 and C_{10} (each m-d-c-s obtained therefrom is redundant) check ER for $k = 7,8$.

j	C	b	C^+	$\Phi(C^+)$	$\Phi[C^+ \cup \mu[C_k\backslash C^+]]$	$D = C_k \backslash C^+$
0	∅	5	{5}*	5	10	{7,3,2,9}**
0	∅	7	{7}*	5	≥10 (L.5.4)	{5,3,2,9}**
0	∅	3	{3}*	7	≥10 (L.5.4)	{5,7,2,9}**
0	∅	2	{2}*	9	≥10 (L.5.4)	{5,7,3,9}**
0	∅	9	{9}*	9	≥10 (L.5.4)	{5,7,3,2}**

$k = 12, 13$: each m-d-c-s obtained from C_{12} or C_{13} is redundant (*Lemma 3.5*).
Output of *Algorithm 2* for $k = 12, 13$: none.

Table 10. Output of *Algorithm 2* for $k = 14$:
$\Phi(C_1\backslash C_{14}) = 10,\ \Phi(C_2\backslash C_{14}) = 10,$
$\Phi(C_3\backslash C_{14}) = 10,\ \Phi(C_4\backslash C_{14}) = 6,$
$\Phi(C_5\backslash C_{14}) = 15,\ \Phi(C_7\backslash C_{14}) = 4,$
$\Phi(C_8\backslash C_{14}) = 11,\ \Phi(C_9\backslash C_{14}) = 11,$ omit C_6, C_{10}, C_{12}, C_{13} (each m-d-c-s obtained therefrom is redundant) check ER for $k = 4, 7$.

j	C	b	C^+	$\Phi(C^+)$	$\Phi[C^+ \cup \mu[C_k \backslash C^+]]$	$D = C_k\backslash C^+$
0	∅	5	{5}*	5	10	{10,3,8,2,9}**
0	∅	10	{10}*	5	≥10 (L.5.4)	{5,3,8,2,9}**
0	∅	3	{3}*	7	≥10 (L.5.4)	{5,10,8,2,9}**
0	∅	8	{8}*	8	≥10 (L.5.4)	{5,10,3,2,9}**
0	∅	2	{2}*	9	≥10 (L.5.4)	{5,10,3,8,9}**
0	∅	9	{9}*	9	≥10 (L.5.4)	{5,10,3,8,2}**

7. Numerical complexity issue

In this section a formula for the presented method's numerical complexity will be derived.

Let us introduce the following additional notation:

- $J^*(C)$ - the smallest j such that:

  $$\varphi(c_1)+\ldots+\varphi(c_j) > \Psi_{max}(G) - d,$$

  where $c_1, c_2, \ldots$ are the components of C orde red according to non-decreasing capacities, i. e.

  $$\varphi(c_1) \leq \varphi(c_2) \leq \ldots;$$

- $j^*(C)$ - the smallest j such that

  $$\varphi(c_1)+\ldots+\varphi(c_j) > \Psi_{max}(G) - d,$$

  where $c_1, c_2, \ldots$ are the components of C ordered according to non-decreasing capacities, i.e.

  $$\varphi(c_1) \geq \varphi(c_2) \geq \cdots .$$

Lemma 6.1
The computational complexity of the presented method, expressed in the "big O" notation, is approximately equal to

$$O\left[\sum_{k=1}^{m} \max_{j^*(C_k)\leq j\leq J^*(C_k)} \binom{card(C_k)}{j}\right]. \quad (37)$$

Proof: Let us note that a min-d-cut-set generated from C_k has $J^*(C_k)$ or $j^*(C_k)$ components if it consists of the components with the smallest or largest capacities respectively. It holds that $J^*(C_k) \geq j^*(C_k)$. In consequence, any min-d-cut set generated from C_k has between $j^*(C_k)$ and $J^*(C_k)$ components. Thus the number of min-d-cut-sets generated from C_k does not exceed the maximum number of j-element subsets of a set whose number of elements equals $card(C_k)$, the maximum being taken over $j^*(C_k) \leq j \leq J^*(C_k)$. In view of the fact that only a very small number of non-minimal d-cut-sets is generated from each C_k, $k = 1,\ldots,m$ (due to different treatment of the cases $\Phi(C_k) < 2d$ and $\Phi(C_k) \geq 2d$), we can approximate the method's complexity by the total number of generated min-d-cut-sets, which, as follows from the first part of the proof, does not exceed (37).

8. Result comparison with other methods

The main difference between the two methods consists in dissimilar ways in which they address the external redundancy problem. The authors of [2] claim that they avoid generating externally redundant d-cut-sets by using a special decomposition technique that modifies L_t (the list of sets from which min-d-cut-sets are generated) when the generation algorithm passes to the next set on L_t. A short description of their method is given below.
In the beginning L_t is composed of all min-cut-sets, and to each $C \in L_t$ a value $w(C)$ is assigned; $w(C)$ is called the required capacity of C and is initially equal to d. The following operations are repeated in a loop: Each subset D of C_1 (the first set on L_t), such that $\Phi(C_1\backslash D) < w(C_1)$, is added to L_d (the list of min-d-cut-sets), then C_1 is deleted from L_t. If L_t is now empty, the algorithm stops, otherwise each recently generated D is checked for inclusion in each C on L_t. If $D = \{d_1,\ldots,d_{|D|}\} \subseteq C$ then C is decomposed into $|D|$ sets $C\backslash\{d_1\},\ldots, C\backslash\{d_{|D|}\}$ whose required capacities are computed as follows:

$$w(C\backslash\{d_i\}) = w(C) - \varphi(d_i),\ i = 1,\ldots,card(D).$$

Then C is substituted on L_t by those sets $C\backslash\{d_1\}$ for which $w(C\backslash\{d_i\}) > 0$.
For greater clarity, let us write down the above procedure in a pseudocode as the following algorithm:

Algorithm 3
1. Put $L_d = \varnothing$ and populate L_t with min-τ-cut-sets ordered by increasing number of components
2. Put on $L_{d,aux}$ each minimal subset D of C_1, such that $\Phi(C_1\backslash D) < w(C_1)$
3. Augment L_d with $L_{d,aux}$ and delete C_1 from L_t
4. If L_t is empty, stop
5. For each set D on $L_{d,aux}$ do
   For each set C on L_t do
   If $D \subseteq C$ then substitute C with those $C\backslash\{d_i\}$
   for which $w(C) - \varphi(d_i) > 0$, $i = 1,2,\ldots, card(D)$
6. Skip to 2

Remarks: $L_{d,aux}$ is the list of min-d-cut-sets generated from one set on L_t, and L_d is the list of min-d-cut-sets generated up to the current step. When the algorithm stops, L_d contains all min-d-cut-sets. A subset D fulfilling the condition in step 2 is minimal if $\Phi(C_1\backslash D') < w(C_1)$ for each $D' \subset D$. $\Phi(C\backslash D)$ must be less than $w(C)$, so that D can be a d-cut set generated from C.
To facilitate the comparison of *Algorithm 3* with *Algorithm 1* and *2*, the detailed trace of *Algorithm 3* for the network in *Figure 1* is now presented. For each C on L_t, such that $w(C) < d = 10$, the value $w(C)$ is given next to C as a superscript. If $w(C) = 10$ then C is not superscripted

$L_d = \varnothing$,

$L_t = \{1,2\}, \{4,5,6\}, \{4,8,11\}, \{9,10,11\}, \{1,3,6\},$

{4,7,10,11}, {2,3,4,5}, {5,6,7,9}, {7,8,9,11}, {5,6,8,9,10}, {2,3,5,7,9}, {1,3,5,8,11}, {1,3,5,7,10,11}, {2,3,5,8,9,10}.

Generate min-d-cut-sets from {1,2}:

$L_{d,aux} = \{1\}$, $L_d = L_d \cup L_{d,aux}$, $L_t = L_t \backslash \{1,2\}$;

Execute step 5 (13 checks and 3 decompositions);

$L_t = \{4,5,6\}, \{4,8,11\}, \{9,10,11\}, \{3,6\}^{(0)}, \{4,7,10,11\}, \{2,3,4,5\}, \{5,6,7,9\}, \{7,8,9,11\}, \{5,6,8,9,10\}, \{2,3,5,7,9\}, \{3,5,8,11\}^{(0)}, \{3,5,7,10,11\}^{(0)},$

$\{2,3,5,8,9,10\} = \{4,5,6\}, \{4,8,11\}, \{9,10,11\}, \{4,7,10,11\}, \{2,3,4,5\}, \{5,6,7,9\}, \{7,8,9,11\}, \{5,6,8,9,10\}, \{2,3,5,7,9\}, \{2,3,5,8,9,10\}.$

Generate min-d-cut-sets from {4,5,6}:

$L_{d,aux} = \{6\}, \{4,5\}$, $L_d = L_d \cup L_{d,aux}$,

$L_t = L_t \backslash \{4,5,6\}$;

Execute step 5 (18 checks and 4 decompositions);

$L_t = \{4,8,11\}, \{9,10,11\}, \{4,7,10,11\}, \{2,3,5\}^{(6)}, \{2,3,4\}^{(5)}, \{5,7,9\}^{(4)}, \{7,8,9,11\}, \{5,8,9,10\}^{(4)}, \{2,3,5,7,9\}, \{2,3,5,8,9,10\}.$

Generate min-d-cut-sets from {4,8,11}:

$L_{d,aux} = \{4,8\}, \{4,11\}, \{8,11\}$, $L_d = L_d \cup L_{d,aux}$,

$L_t = L_t \backslash \{4,8,11\}$;

Execute step 5 (27 checks and 4 decompositions);

$L_t = \{9,10,11\}, \{7,10,11\}^{(6)}, \{4,7,10\}^{(4)}, \{2,3,5\}^{(6)}, \{2,3,4\}^{(5)}, \{5,7,9\}^{(4)}, \{7,9,11\}^{(2)}, \{7,8,9\}^{(4)}, \{5,8,9,10\}^{(4)}, \{2,3,5,7,9\}, \{2,3,5,8,9,10\}.$

Generate min-d-cut-sets from {9,10,11}:

$L_{d,aux} = \{9,10\}, \{9,11\}, \{10,11\}$,

$L_d = L_d \cup L_{d,aux}$, $L_t = L_t \backslash \{9,10,11\}$;

Execute step 5 (30 checks and 8 decompositions);

$L_t = \{7,11\}^{(1)}, \{7,10\}^{(0)}, \{4,7,10\}^{(4)}, \{2,3,5\}^{(6)}, \{2,3,4\}^{(5)}, \{5,7,9\}^{(4)}, \{7,11\}^{(-7)}, \{7,9\}^{(-4)}, \{7,8,9\}^{(4)}, \{5,8,10\}^{(-5)}, \{5,8,9\}^{(-1)}, \{2,3,5,7,9\}, \{2,3,5,8,10\}^{(1)},$

$\{2,3,5,8,9\}^{(5)} = \{7,11\}^{(1)}, \{4,7,10\}^{(4)}, \{2,3,5\}^{(6)}, \{2,3,4\}^{(5)}, \{5,7,9\}^{(4)}, \{7,8,9\}^{(4)}, \{2,3,5,7,9\}, \{2,3,5,8,10\}^{(1)}, \{2,3,5,8,9\}^{(5)}.$

Generate min-d-cut-sets from $\{7,11\}^{(1)}$:

$L_{d,aux} = \{7,11\}$, $L_d = L_d \cup L_{d,aux}$,

$L_t = L_t \backslash \{7,11\}^{(1)}$;

Execute step 5 (8 checks and 0 decompositions);

$L_t = \{4,7,10\}^{(4)}, \{2,3,5\}^{(6)}, \{2,3,4\}^{(5)}, \{5,7,9\}^{(4)}, \{7,8,9\}^{(4)}, \{2,3,5,7,9\}, \{2,3,5,8,10\}^{(1)}, \{2,3,5,8,9\}^{(5)}.$

Generate min-d-cut-sets from $\{4,7,10\}^{(4)}$:

$L_{d,aux} = \{4,7,10\}$, $L_d = L_d \cup L_{d,aux}$,

$L_t = L_t \backslash \{4,7,10\}^{(4)}$;

Execute step 5 (7 checks and 0 decompositions);

$L_t = \{2,3,5\}^{(6)}, \{2,3,4\}^{(5)}, \{5,7,9\}^{(4)}, \{7,8,9\}^{(4)}, \{2,3,5,7,9\}, \{2,3,5,8,10\}^{(1)}, \{2,3,5,8,9\}^{(5)}.$

Generate min-d-cut-sets from $\{2,3,5\}^{(6)}$:

$L_{d,aux} = \{2,3\}$, $L_d = L_d \cup L_{d,aux}$,

$L_t = L_t \backslash \{2,3,5\}^{(6)}$;

Execute step 5 (6 checks and 8 decompositions);

$L_t = \{3,4\}^{(-4)}, \{2,4\}^{(-2)}, \{5,7,9\}^{(4)}, \{7,8,9\}^{(4)}, \{3,5,7,9\}^{(1)}, \{2,5,7,9\}^{(3)}, \{3,5,8,10\}^{(-8)}, \{2,5,8,10\}^{(-6)}, \{3,5,8,9\}^{(-4)},$

$\{2,5,8,9\}^{(-1)} = \{5,7,9\}^{(4)}, \{7,8,9\}^{(4)}, \{3,5,7,9\}^{(1)}, \{2,5,7,9\}^{(3)}.$

Generate min-d-cut-sets from $\{5,7,9\}^{(4)}$:

$L_{d,aux} = \{5,7,9\}$, $L_d = L_d \cup L_{d,aux}$,

$L_t = L_t \backslash \{5,7,9\}^{(4)}$;

Execute step 5 (3 checks and 6 decompositions)

$L_t = \{7,8,9\}^{(4)}, \{3,7,9\}^{(-4)}, \{3,5,9\}^{(-4)}, \{3,5,7\}^{(-8)}, \{2,7,9\}^{(-2)}, \{2,5,9\}^{(-2)}, \{2,5,7\}^{(-6)} = \{7,8,9\}^{(4)}.$

Generate min-d-cut-sets from $\{7,8,9\}^{(4)}$:

$L_{d,aux} = \{7,8,9\}$, $L_d = L_d \cup L_{d,aux}$,

$L_t = L_t \backslash \{7,8,9\}^{(4)} = \varnothing$ (L_t is empty);

L_d ={1}, {6}, {4,5}, {4,8}, {4,11}, {8,11}, {9,10}, {9,11}, {10,11}, {7,11}, {4,7,10}, {2,3}, {5,7,9}, {7,8,9}.

The comparison of *Algorithm 3* with *Algorithm 1* or *2* leads to the following conclusions.

1) The number of cycles of the main loop in the combined *Algorithm 1* and *2* does not exceed m (the number of min-cut-sets), but is unpredictable in *Algorithm 3* due to variable length of L_t, and may be greater than m.

An ER check in *Algorithm 1* or *2* is numerically equivalent to an inclusion check or decomposition operation in *Algorithm 3*, but the number of ER checks in *Algorithm 1* and *2* can be significantly smaller than the number of inclusion checks and decompositions in *Algorithm 3*. Each m-d-c-s candidate found by *Algorithm 1* or *2* is only compared to selected min-τ-cut-sets preceding the one from which this candidate is generated (see corollary to *Lemma 3.4*). If applied to the network in *Figure 1*, *Algorithm 1* makes no ER checks, and *Algorithm 2* makes 64 ER checks (3 in step 5, 3 in step 7, 4 in step 8, 12 in step 9, 20 in step 10, 10 in step 11, and 12 in step 14). In turn, in *Algorithm 3* each D on $L_{d,aux}$ is checked for inclusion in each set on the current L_t, which amounts to 112 checks and 33 decompositions if *Algorithm 3* is applied to the same network. This difference between *Algorithm 3* and *Algorithm 1* and *2* is likely to grow with the network size.

2) The computational effort afforded in *Algorithm 1* or *2* to obtain m-d-c-s candidates from one min-cut-set is comparable to that required to extract $L_{d,aux}$ from one set on L_t. ($L_{d,aux}$ is obtained using the decomposition technique to prevent generating internally redundant d-cut-sets). In turn, the technique of generating m-d-c-s candidates, based on *Lemmas 4.1–4.4* and *5.1–5.4*, and on capacity-based ordering of min-cut-sets and their elements, minimizes the number of non-d-cut-sets (*Algorithm 1*) or internally redundant d-cut-sets (*Algorithm 2*) generated prior to finding a candidate. It also avoids generating internally redundant d-cut-sets in *Algorithm 1* or allows for instantaneous internal redundancy check in *Algorithm 2*. In [2] the min-cut-sets are initially ordered by increasing number of components, and their elements are ordered by increasing indexes, thus the advantages of capacity-based ordering are overlooked.

3) Last but not least, *Algorithms 1* and *2*, based on *Theorem 2*, guarantee that no ER redundant d-cut-sets are generated. It should be noted that the authors of [2] in the 2-nd paragraph of Section II and the last paragraph of Section V admit that their method can generate a certain number of ER d-cut-sets (although no such sets were generated by *Algorithm 3* for the given example).

Algorithms 1 and *2* are only compared with *Algorithm 3* for two reasons. One – very few methods for min-d-cut-sets enumeration can be found in the literature. Second – the authors of [2] claimed to improve the results of [5] and proved their method to be very efficient. Also, it is difficult to compare the results presented here with those of [9], because networks with multi-state links (not two-state ones) are considered there.

9. Conclusion

As shown in Section 7, the new method of enumerating all min-d-cut-sets in a flow network presented in this chapter is competitive in comparison with the analogous methods described in the relevant literature, particularly in [2] that improves the results of [5].

Let us note that, using *Algorithm 1* or *2*, it is possible to generate m-d-c-s candidates and check them for non-redundancy, in parallel for several successive k. This is because, due to *Condition 3* in *Theorem 2*, non-redundancy check involves the given min-cut-sets C_j, $j < k$, rather than min-d-cut-sets obtained from C_j, $j < k$. Thus, the computing time can be significantly reduced compared to generating min-d-cut-sets sequentially for $k = 1, \ldots, m$.

Although the provided example suggests that the proposed method can only be used for networks with undirected links and failure-free nodes, this is not the case. Since the Ford-Fulkerson theorem also holds for networks with directed links and failing nodes with limited capacities, so do all the lemmas and theorems from Sections 3–5. See [7] and [10] for algorithms that find min-cut-sets in such networks. Moreover, the method can be generalized to the multi-source and/or multi-sink case, but this would require appropriate redefining of min-τ-cut-sets, which will be a topic of further research. It should be noted that a similar issue has been addressed in [3].

Another topic for future research is the adaptation of *Algorithms 1* and *2* to networks with multi-state components. This is not a simple task, because in the multi-state case a d-cut-set is not defined as a subset of components, but as a vector of reduced capacities of all components in a network. For example, (10,9,7,4,5,6,5,8,9,5,6) and (10,9,7,3,3,3,5,8,9,5,6) are vectors of maximum and reduced capacities for the network in *Figure 1* (capacities of components 4, 5 and 6 are reduced to 3). The Ford-Fulkerson theorem allows to easily

check that the second vector reduces $\Psi_{max}(G)$ from 15 to 9, i.e. it is a d-cut-set if $d \geq 10$. Although it seems possible to formulate and prove lemmas and theorems analogous to those from Section 3, the lemmas from Sections 4 and 5 may not have direct counterparts. In any case, the numerical complexity of the method adapted to multi-state components will be significantly higher than that given by (37).

References

[1] Abel, U. & Bicker, R. 1982. Determination of all minimal cut-sets between a vertex pair in an undirected graph, *IEEE Transactions on Reliability* (31), 167–171.

[2] Chakraborty, S. & Goyal, N. K. 2015. Irredundant subset cut enumeration for reliability evaluation of flow networks. *IEEE Transactions on Reliability* 64(4), 1194–1202.

[3] Chakraborty, S. & Goyal, N. K. 2017. An efficient reliability evaluation approach for networks with simultaneous Multiple-Node-Pair flow requirements. *Quality and Reliability Engineering International* 33(5), 1067–1082.

[4] Chatelet, E. et al. 1999. An optimal procedure to generate sums of disjoint products. *Reliability Engineering and System Safety* 65(3), 289–294.

[5] Chaturvedi, S. K. 2007. Irredundant subset cut generation to compute capacity related reliability. *International Journal of Performability Engineering* 3(2), 243–256.

[6] Cormen, T. H. et al. 2009. *Introduction to Algorithms*. MIT Press, London.

[7] Malinowski, J. 2015. A new efficient algorithm generating all minimal s–t cut sets in a graph-modeled network. *Proceedings of the International Conference of Numerical Analysis and Applied Mathematics, AIP Conference Proceedings*, 480030-1–480030-4.

[8] Malinowski, J. 2016. A fast tree-scanning algorithm finding a compact expression for the structure function of a system with known minimal path(cut) sets. *Proceedings of the 26th European Safety and Reliability Conference, ESREL 2016*, In: Risk, Reliability and Safety: Innovating Theory and Practice 1375–1379.

[9] Niu, Y. F. et al. 2017. A new efficient algorithm for finding all d–minimal cuts in multi-state networks. *Reliability Engineering and System Safety* 166, 151–163.

[10] Singh, B. 1994. Enumeration of node cut sets for an s–t network. *Microelectronics Reliability* 34, 559–561.

[11] Soh, S. & Rai, S. 2005. An efficient cutset approach for evaluating communication-network reliability with heterogeneous link-capacities. *IEEE Transactions on Reliability* 54(1), 133–144.

[12] Yeh, W. C. 2006. A simple algorithm to search for all MCs in networks. *European Journal of Operational Research* 174, 1694–1705.

[13] Yeh, W. C. 2007. An improved sum-of-disjoint-products technique for the symbolic network reliability analysis with known minimal paths. *Reliability Engineering and System Safety* 92(2), 260–268.