Raport Badawczy

Research Report

RB/18/2011

# Determining s-t and K-terminal path sets and cut sets in graph-modeled networks by a unified approach

J. Malinowski

Instytut Badań Systemowych
Polska Akademia Nauk

Systems Research Institute
Polish Academy of Sciences

**POLSKA AKADEMIA NAUK**

**Instytut Badań Systemowych**

ul. Newelska 6

01-447 Warszawa

tel.: (+48) (22) 3810100

fax: (+48) (22) 3810105

Kierownik Zakładu zgłaszający pracę:
Prof. zw. dr hab. inż. Olgierd Hryniewicz

Warszawa 2011

## 1. Introduction

The current paper presents a set of algorithms for finding minimal path and cut sets of various types in graph-structured networks (computer, telecommunication, transportation, etc.) Those sets are crucial in reliability analysis of network systems. The presented methodology is based on the construction of the acyclic paths tree, where each path connects two nodes of the considered network. Based on this tree the s-t minimal paths are readily obtained, and after minor transformations the so-called simplified fault tree is constructed from which the s-t minimal cuts are determined. In consequence, we have a common starting point for finding both the s-t path and cut sets, while many existing algorithms use very different approaches for paths and cuts. Apart from that, fairly simple methods are demonstrated allowing to obtain K-terminal (all-terminal) minimal paths or cuts from s-t minimal paths or cuts respectively. It is important that all path or cut sets can be composed of both nodes and links, while many known algorithms do not take node failures into account. Furthermore, the presented methods can be applied both in the case of directed and undirected links. Thus we have a coherent set of procedures for determining essential network reliability characteristics, founded on a common basis and with a wide area of applicability

A network under consideration will be modeled by a graph denoted as $\Gamma = (V,E_1,E_2)$, where $V = \{v_1,\ldots,v_n\}$ is the set of vertices which represent the network's nodes, $E_1$ – the set of directed edges which represent one-way links, $E_2$ – the set of undirected edges which represent two-way links. A directed edge can be regarded as an ordered pair of vertices, while an undirected one – a set of two vertices. If all edges are directed then $E_2=\varnothing$, if they are undirected then $E_1=\varnothing$. An ordered set of components $(v_{i(1)}, e_{j(1)}, v_{i(2)}, e_{j(2)},\ldots, e_{j(k-1)},v_{i(k)})$ such that $v_{i(1)}=v$, $v_{i(k)}=w$, and $e_{j(h)}$ is the directed edge $(v_{i(h)},v_{i(h+1)})$ or the undirected edge $\{v_{i(h)},v_{i(h+1)}\}$, is called a path from v to w. A loop-free path will be called acyclic. Let $\sim$ and $\rightarrow$ be relations in the set V defined as follows: $v\sim w$ or $v\rightarrow w$ if there exists a path from v to w such that all respective nodes and links are operable; the former relation is applicable if $E_1=\varnothing$ (all edges are undirected), the latter – if $E_1\neq\varnothing$ (some edges are directed). Clearly, both relations are transitive, and "$\sim$" is symmetric, unlike "$\rightarrow$".

A set of nodes and links, such that it contains (is) an acyclic path from a source node $v_s$ to a terminal node $v_t$ will be called a (minimal) s-t path. An s-t path set is minimal if none of its subsets is an s-t path set. Clearly, there is a connection from $v_s$ to $v_t$ if and only if all components in at least one minimal s-t path are operable.

A set of nodes and links such that it contains at least one component of each minimal s-t path set will be called an s-t cut set. An s-t cut set is minimal if none of its subsets is an s-t cut set. Clearly, there is no connection from $v_s$ to $v_t$ if and only if all components in at least one minimal s-t cut set are failed. Apart from s-t path and cut sets, the so called K-terminal path and cut sets are also considered in this paper. Their definitions will be given in chapters 4 and 6.

## 2. Generating all minimal s-t path sets

In this chapter a well-known method of finding all minimal s-t path sets, based on the breadth first search algorithm, is described. It consists in constructing the tree of acyclic paths, denoted as $T_{s-t}$, connecting two distinguished vertices of $\Gamma$ – the source vertex $v_s$ with the terminal vertex $v_t$. As in general $\Gamma$ has directed edges, it is important which of the two vertices is the source, and which the terminal one. The rules for constructing $T_{s-t}$ are very simple. To the nodes and links of $T_{s-t}$ the vertices and edges of $\Gamma$ are assigned. To the root node of $T_{s-t}$ the source vertex $v_s$ is assigned. To each other node $v \in T_{s-t}$ are assigned those vertices of G directly reachable from v, which in $T_{s-t}$ are not located between the root node and v. If $v=v_t$ or v has no child nodes then v is a leaf node. A path which connects $v_s$ to a leaf node other than $v_t$ will be called a dead-end path. Clearly there is one-to-one correspondence between all paths in $T_{s-t}$ beginning at $v_s$ and ending at $v_t$, and all minimal s-t path sets ensuring a connection from $v_s$ to $v_t$. For illustration, an exemplary network structure is presented in Fig.1, and the tree of acyclic paths connecting A with F – in Fig.2. Note that C, although it is not a terminal vertex, is a leaf node of $T_{A-F}$, and {A, 1, B, 4, D, 6, C} is a dead-end path.

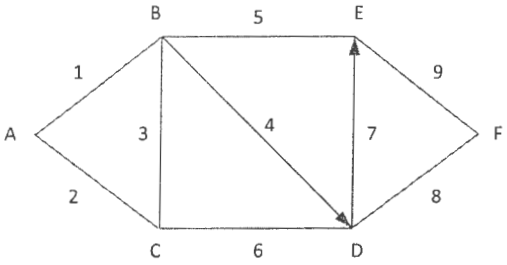


Fig.1 An exemplary network structure

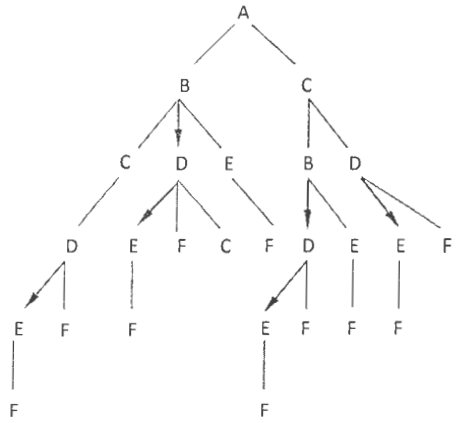


Fig. 2. The tree of acyclic paths from A to F for the network in Fig.1

Analyzing Fig. 2 we obtain the following A-F minimal path sets:

$P_1$ = {A,1,B,3,C,6,D,7,E,9,F}, $P_2$ = {A,1,B,3,C,6,D,8,F}, $P_3$ = {A,1,B,4,D,7,E,9,F},

$P_4$ = {A,1,B,4,D,8,F}, $P_5$ = {A,1,B,5,E,9,F}, $P_6$ = {A,2,C,3,B,4,D,7,E,9,F},

$P_7$ = {A,2,C,3,B,4,D,8,F}, $P_8$ = {A,2,C,3,B,5,E,9,F}, $P_9$ = {A,2,C,6,D,7,E,9,F},

$P_{10}$ = {A,2,C,6,D,8,F}

3. Generating all minimal s-t cut sets

In this chapter a new efficient method of finding all minimal s-t cut sets is presented. It uses the already constructed tree of minimal s-t paths, which after minor modifications is transformed into a fault tree wherefrom s-t cuts are obtained. Proceeding to the details, let $\Phi(x_1,\ldots,x_n)$ be the Boolean function equal to 1 if there is no connection from $v_s$ to $v_t$, or to 0 otherwise. The variables $x_1,\ldots,x_n$ represent reliability states of respective components, i.e. $x_i = 1$ if the i-th component is failed, $x_i = 0$ otherwise. For each node v let

$$(2.1)\quad \Phi_v(x_1,\ldots,x_n) = x_{i(v)} \vee \Lambda_{w\in L(v)}\left[\left(\vee_{v<u<w} x_{i(u)}\right) \vee \Phi_w(x_1,\ldots,x_n)\right]$$

where i(u) is the index of the $\Gamma$'s element (vertex or edge) assigned to $u\in T_{s-t}$, and L(v) is the set of all nodes $w\in T_{s-t}$ fulfilling the following conditions:

1) $v < w$

2) w has more than one child link or w is a leaf node

3) if u is a node (link) such that $v < u < w$ then u has only one child link (node)

Obviously, if v is a leaf node, then L(v) is empty and $\Phi_v(x_1,\ldots,x_n) = x_{i(v)}$. It can be easily shown that

$$(2.2)\quad \Phi(x_1,\ldots,x_n) = \Phi_r(x_1,\ldots,x_n)$$

where r is the root node of T. Based on (2.1) and (2.2), the tree of acyclic paths can be easily transformed into a fault tree by placing an OR gate directly above the root node, an AND gate directly below each node having more than one child node, and a number of OR gates directly below each AND gate – one OR gate per each child node. The rules for constructing OR gates are the following:

1) a leaf node v plus all elements located between v and the lowest AND gate above v become inputs to the respective OR gate;

2) if AND gates $G_x$ and $G_y$ are in the parent-child relation (there are no AND gates between $G_x$ and $G_y$), then $G_y$ plus all elements located between $G_x$ and $G_y$ become inputs to the respective OR gate;

3) the top AND gate and all elements located above it become inputs to the top OR gate. Thus obtained fault tree can be reduced to a simpler form by making OR gates default, i.e. not placing them in the fault tree diagram. In consequence, the simplified fault tree is isomorphic to the acyclic paths tree, whereas its analysis can be performed in the same way as if OR gates were present. In Fig. 3 a simplified fault tree obtained from the acyclic paths tree in Fig. 2 is presented.

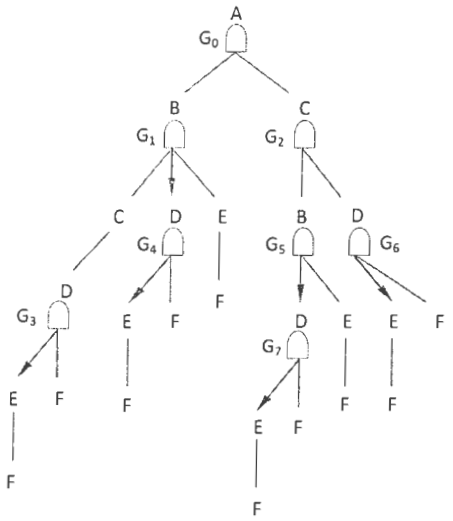


Fig.3. The simplified fault tree obtained from the acyclic paths tree in Fig.2

Once the system's fault tree is constructed it can be used to determine all minimal s-t cut sets of that system. For this purpose the following Boolean expression is defined for each AND gate:

$$\Phi(G) = \Psi_1(G) \wedge \ldots \wedge \Psi_{H(G)}(G) \quad (2.3)$$

where G denotes the considered gate, H(G) is the number of OR gates placed immediately below G, and $\Psi_h(G)$ is the expression for the h-th OR gate, $1 \leq h \leq H(G)$. The expression for an OR gate is the sum of all its inputs. For example, referring to Fig. 3, we have:

$\Phi(G_1) = \Psi_1(G_1) \wedge \Psi_2(G_1) \wedge \Psi_3(G_1)$,

$\Psi_1(G_1) = x_3 \vee x_C \vee x_6 \vee x_D \vee \Phi(G_3)$

$\Psi_2(G_1) = x_4 \vee x_D \vee \Phi(G_4)$

$\Psi_3(G_1) = x_5 \vee x_E \vee x_9 \vee x_F$

The expression for AND gates are found recursively, starting from bottom level gates, passing through intermediate levels, until the top AND gate is reached. E.g., for the fault tree in Fig.3, first the expression for $G_7$ (third-level gate) is found, then the expressions for $G_3$, $G_4$, $G_5$, $G_6$ (second-level gates), then the expressions for $G_1$, $G_2$ (first-level gates), and finally the expression for $G_0$ (zero-level or top gate).

Clearly, if G is an AND gate, then $\Phi(G)$ is a sum of products. Let $J(\pi)$ denote the set of indices of the factors of a product $\pi$, e.g. $J(x_1x_3x_6) = \{1,3,6\}$. If $J(\pi)$ is a cut set then $\pi$ will be called a cut set product. As follows from the principles of fault tree analysis, the expression for the top OR gate, after removing all redundant products (a product is redundant if it includes all variables of any other product in the same expression), corresponds to the family of all minimal s-t cut sets, which will be denoted by L. For example, the expression for the top (default) OR gate in Fig. 3 is equal to $x_A \vee \Phi(G_0)$.

However, in order to generate the list L it is not necessary to include in the expressions for AND gates all cut set products obtained at the lower levels. Those products can be removed as soon as they occur, and the respective cut sets added to L (provided they are not redundant), due to the following proposition:

Proposition 1

If G is an AND gate, and $\Phi(G)$ contains a cut set product $\pi$, then in each AND gate located above G the product $\pi$ is expanded to the product $\rho$ such that $J(\pi) \subseteq J(\rho)$.

Proof: Indeed, the above proposition holds true, as constructing the expressions for gates located above G involves multiplying $\pi$ by at least one variable.

Let $\Phi^*(G)$ be the expression obtained by removing from $\Phi(G)$ all cut set products, and $\Psi^*_h(G)$ – the expression for the respective OR gate, in which $\Phi$ is replaced by $\Phi^*$ for the

respective child AND gate of G (if any). In view of Proposition 1, the list L can be generated by means of the following procedure:

1. L=∅.
2. For each successive AND gate G do:

   calculate the expression $\Psi^*_1(G) \wedge \ldots \wedge \Psi^*_{H(G)}(G)$ and search it for products corresponding to cut sets;

   add new minimal cut sets to L;

   assign to G the expression $\Phi^*(G)$ obtained by removing from $\Psi^*_1(G) \wedge \ldots \wedge \Psi^*_{H(G)}(G)$ all products which correspond to cut sets

Thus constructed procedure is highly efficient, because products which correspond to cut sets are not passed from lower to higher level gates.

Before proceeding to the detailed presentation of the above outlined procedure, two auxiliary lemmas will be formulated. The first lemma states that a different sums which are in inclusion relation for detecting **redundant** products – they are irrelevant for an expression being a sum of products, and hence can be removed from it. Clearly, shorter expressions make the procedure run faster.

Lemma 1

Let $S_1 = \pi_1 \vee \ldots \vee \pi_g$, $S_2 = \rho_1 \vee \ldots \vee \rho_h$

If $I(\pi_i) \subset I(\rho_j)$ for some $1 \le i \le g$, $1 \le j \le h$, then $(\pi_1 \vee \ldots \vee \pi_g)\ \rho_j = \rho_j$

If $I(\pi_i) \supset I(\rho_j)$ for some $1 \le i \le g$, $1 \le j \le h$, then $= \pi_i\ (\rho_1 \vee \ldots \vee \rho_h) = \pi_i$

If $I(\pi_i) = I(\rho_j)$ for some $1 \le i \le g$, $1 \le j \le h$, then $= \pi_i\ (\rho_1 \vee \ldots \vee \rho_h) = (\pi_1 \vee \ldots \vee \pi_g)\ \rho_j = \pi_i$

Proof: To prove the first implication it is sufficient to note that $I(\pi_i) \subset I(\rho_j)$ implies $\pi_i \rho_j = \rho_j$, hence $(\pi_1 \vee \ldots \vee \pi_g)\ \rho_j = (\pi_1 \vee \ldots \vee \pi_{i-1} \vee 1 \vee \pi_{i+1} \vee \ldots \vee \pi_g)\ \rho_j = \rho_j$. The remaining two implications are proved similarly.

The second lemma shows how to check whether a product corresponds to an s-t cut set.

Lemma 2

Let Q(i) be the set of numbers of these minimal s-t path sets which contain $e_i$, i.e.

(2.5) $Q(i) = \{j: e_i \in P_j\}$

A product $\pi = \Lambda_{i \in I(\pi)} x_i$ corresponds a cut set if and only if

(2.6) $\cup_{i \in I(\pi)} Q(i) = \{1, \dots, w\}$

Proof: (2.6) is fulfilled if and only if each minimal s-t path set contains at least one element of I(π), which is equivalent to I(π) being a cut-set (not necessarily minimal).

Based on the above premises, a procedure creating the list of all minimal s-t cut sets can now be presented in detail. Its employs an algorithm, run repeatedly for each AND gate of the simplified fault tree, starting from bottom-level gates, passing from lower to higher-level gates, and ending at the top-level gate. This algorithm is composed of two fragments. In the first one it is checked whether products from different inputs to a gate G (OR gates located immediately below G) are in inclusion relation (cf. Lemma 1), and the expression from which Φ*(G) will be obtained is constructed (this expression may still contain cut set products). Below the pseudo-code for this fragment is given.

Algorithm 1 (first part)

$\Phi^*(G) \leftarrow S_1(G)$

$\Phi_{aux} \leftarrow \varnothing$

for h ← 2 to H(G) {

for $\pi \in \Phi^*(G)$ {

for $\rho \in S_h(G)$ {

if $I(\rho) \subset I(\pi)$ then move $\pi$ from $\Phi^*(G)$ to $\Phi_{aux}$

if $I(\pi) \subset I(\rho)$ then move $\rho$ from $S_h(G)$ to $\Phi_{aux}$

if $I(\pi) = I(\rho)$ then move $\pi$ from $\Phi^*(G)$ and $S_h(G)$ to $\Phi_{aux}$

}

}

for $\pi \in \Phi^*(G)$ {

for $\rho \in S_h(G)$ {

add $\pi\rho$ to $\Phi_{aux}$

}

}

$\Phi^*(G) \leftarrow \Phi_{aux}$

$\Phi_{aux} \leftarrow \varnothing$

}

In the second fragment it is checked for each $\pi \in \Phi^*(G)$ whether $\pi$ corresponds to a cut-set (cf. Lemma 2). If not, then $\pi$ remains in $\Phi^*(G)$, otherwise $\pi$ is removed from $\Phi^*(G)$ and compared with each $\rho \in L$. If $J(\rho) \subset J(\pi)$ for some $\rho \in L$ then $\pi$ is not a minimal cut-set and is not added to L. if $J(\pi) \subset J(\rho)$ for some $\rho \in L$ then $\rho$ is not a minimal cut-set and is removed from L. If $J(\pi)$ does not include any $J(\rho)$, $\rho \in L$, then $\pi$ is added to L. Thus it is guaranteed that L will consist of all and only minimal cut-sets after the procedure is run for the top AND gate (excluding one-element cut sets corresponding to components located between the top OR

and top AND gates). Moreover, no two cut sets in L will be identical. Below the pseudo-code for this fragment is given.

Algorithm 1 (second part)

```
for π∈Φ*(G) {
 if J(π) is a cut set then {
  remove π from Φ*(G)
  x ← 1
  for ρ∈L {
   if J(ρ) ⊆ J(π) then {x ← 0; break } ## J(π) is not a minimal cut set or π is already in L
   if J(π) ⊂ J(ρ) then remove ρ from L ## J(ρ) is not a minimal cut set
  }
  if x = 1 then add J(π) to L
 }
}
```

In order to create the full list L, the above algorithm is run for all AND gates, starting from those located at the lowest level, passing from lower to higher levels, and ending at the top AND gate. Next, one-element cut sets – the components being the inputs to the top OR gate are added to L. Prior to running Algorithm 1 for the first AND gate the list L must be empty.

For illustration and better understanding let us find all minimal s-t cut sets for the network in Fig. 1, assuming that s=1, t=9, and only the links are failure-prone (the nodes do not fail therefore they are not cut sets' elements). Applying the presented method to the fault tree in Fig. 2 we obtain:

$\Phi^{**}(G_7) = (x_9 \vee x_7)\, x_8 = x_9 x_8^* \vee x_7 x_8$; $\Phi^*(G_7) = x_7 x_8$; L = {9,8}

$\Phi^{**}(G_6) = (x_9 \vee x_7)\, x_8 = \underline{x_9 x_8} \vee x_7 x_8$; $\Phi^*(G_6) = x_7 x_8$

$\Phi^{**}(G_5) = [\Phi^*(G_7) \vee x_4][x_9 \vee x_5] = \underline{x_7 x_8 x_9} \vee x_7 x_8 x_5^* \vee x_4 x_9 \vee x_4 x_5$; $\Phi^*(G_5) = x_4 x_9 \vee x_4 x_5$;

L = {9,8}, {7,8,5}

$\Phi^{**}(G_4) = (x_9 \vee x_7)\, x_8 = \underline{x_9 x_8} \vee x_7 x_8$; $\Phi^*(G_4) = x_7 x_8$

$\Phi^{**}(G_3) = (x_9 \vee x_7)\, x_8 = \underline{x_9 x_8} \vee x_7 x_8$ ; $\Phi^{*}(G_3) = x_7 x_8$

$\Phi^{**}(G_2) = [\Phi^{*}(G_5) \vee x_3][\Phi^{*}(G_6) \vee x_6] = \underline{x_4 x_9 x_7 x_8} \vee \underline{x_4 x_5 x_7 x_8} \vee x_4 x_9 x_6^{*} \vee x_4 x_5 x_6^{*} \vee x_3 x_7 x_8 \vee x_3 x_6$ ; $\Phi^{*}(G_2) = x_3 x_7 x_8 \vee x_3 x_6$ ; L = {9,8}, {7,8,5}, {4,9,6}, {4,5,6}

$\Phi^{**}(G_1) = [\Phi(G_3) \vee x_6 \vee x_3][\Phi(G_4) \vee x_4][x_9 \vee x_5] = (x_7 x_8 \vee x_6 x_4 \vee x_3 x_4)(x_9 \vee x_5) = \underline{x_7 x_8 x_9} \vee \underline{x_7 x_8 x_5} \vee \underline{x_6 x_4 x_9} \vee \underline{x_6 x_4 x_5} \vee x_3 x_4 x_9 \vee x_3 x_4 x_5$ ; $\Phi^{*}(G_1) = x_3 x_4 x_9 \vee x_3 x_4 x_5$ ;

$\Phi^{**}(G_0) = [\Phi(G_1) \vee x_1][\Phi(G_2) \vee x_2) = \underline{x_3 x_4 x_9 x_7 x_8} \vee \underline{x_3 x_4 x_9 x_6} \vee \underline{x_3 x_4 x_5 x_7 x_8} \vee \underline{x_3 x_4 x_5 x_6} \vee x_3 x_4 x_9 x_2^{*} \vee x_3 x_4 x_5 x_2^{*} \vee x_1 x_3 x_7 x_8^{*} \vee x_1 x_3 x_6^{*} \vee x_1 x_2^{*}$ ; $\Phi^{*}(G_0) = \varnothing$;

L = {9,8}, {7,8,5}, {4,9,6}, {4,5,6}, {3,4,9,2}, {3,4,5,2}, {1,3,7,8}, {1,3,6}, {1,2}

The underscored products correspond to redundant cut-sets (a cut set is redundant if it includes or is equal to another cut set on L), therefore they are removed from G, the ones marked with an asterisk correspond to non-redundant cut sets, thus they are moved from G to L. If nodes' failures were taken into account, all calculations would be carried out in the same way as above, however, they would be more complex and the obtained cut sets would also include nodes.

Using the above example Algorithm 1 was compared to other known methods of s-t cut sets generation, e.g. Abel and Bicker [1] or Arunkumar and Lee [2], or those surveyed or presented in recently published literature, e.g. [6], [8], [9], and it proved to be faster. Besides, Algorithm 1 can be applied in the same form to directed, partly directed, or undirected graphs. Moreover, it can generate cut sets including both vertices and edges, while most existing methods are limited to cut sets composed of edges alone – node failures are not taken into account, which is unacceptable from the practical point of view. It should also be noted that the algorithm is particularly useful when both paths and cut sets are to be found, as it is based on the acyclic paths (minimal paths sets) tree.

## 4. Generating all minimal K-terminal cut sets

A K-terminal cut set is such a set C of the considered network's components, that the failure of all elements in C causes a disconnection between the nodes of a set $U \subseteq V$, i.e. there exist nodes $v \in U$ and $w \in U$ such that there is no connection from v to w. A K-terminal cut set is minimal if none of its subsets is a K-terminal cut set. In this chapter a simple method of

finding all minimal K-terminal cut sets will be presented. This method assumes that the minimal s-t cut sets for some node pairs $(v_1,v_2)$ are known, and is based on the following lemmas:

Lemma 1

If $\Gamma$ is undirected then all nodes in $U \subseteq V$ are connected if and only if $v \sim w$ for any $v \in U$ and each $w \in U$ such that $w \neq v$.

Proof:

The „=>" implication is obvious, it thus remains to prove the „<=" implication. Let us take two arbitrary nodes $w_1 \in U$, $w_2 \in U$ such that $w_1 \neq v$, $w_2 \neq v$. By assumption, $v \sim w_1$ and $v \sim w_2$, thus symmetricity of the „~" relation implies that $w_1 \sim v \sim w_2$. The „~" relation is transitive, therefore $w_1 \sim w_2$, which completes the proof.

Lemma 2

If $\Gamma$ has directed edges then all nodes in $U=\{v_1,\ldots,v_k\} \subseteq V$ are connected if and only if $v_1 \rightarrow v_2$, $v_2 \rightarrow v_3, \ldots, v_{k-1} \rightarrow v_k$, $v_k \rightarrow v_1$.

Proof:

Lemma 2 is a direct consequence of the transitivity of the "→" relation.

Let $\Gamma$ be an undirected graph, v and w – arbitrary nodes of $\Gamma$, q(v,w) – the number of minimal s-t cut sets separating v from w, and $C_j(v,w)$ – the j-th such cut set, $1 \leq j \leq q(v,w)$. By definition, let

$$(4.1) \quad \Phi_{v \sim w}(x_1, \ldots, x_n) = \bigvee_{j=1}^{q(v,w)} \bigwedge_{i \in C_j(v,w)} x_i$$

where $x_i$, $1 \leq i \leq n$ represent the components' reliability states $x_i = 1$ if the i-th element is failed, $x_i = 0$ if it is operable. Clearly, $\Phi_{v \sim w}$ is a Boolean function equal to 1 if there is no connection between v and w, otherwise $\Phi_{v \sim w}$ is equal to 0. Based on lemma 1 we can formulate the following proposition

Proposition 1

Let $U=\{v_1,\ldots,v_k\}$ be a subset of V. U is disconnected if and only if

$$\Phi_{v_1\sim v_2} \vee \ldots \vee \Phi_{v_1\sim v_k} = 1 \quad (4.2)$$

Proof: Lemma 1 yields that U is disconnected if and only if a node in U is disconnected from one of the remaining nodes in U, i.e. the equality (4.2) holds.

From (4.2) it follows that the list $L_U$ of all minimal K-terminal cut sets separating the nodes of $U=\{v_1,\ldots,v_k\}\subseteq V$ is the union of the lists $L_{1\sim 2},\ldots,L_{1\sim k}$ with redundant cut sets removed, where $L_{1\sim i}$ is the list of all minimal s-t cut sets separating $v_1$ and $v_i$, $1 \le i \le k$. Thus we obtain a simple algorithm to generate $L_U$.

Algorithm 2

$L_U = L_{1\sim 2}$;
for i=3 to k
 for j=1 to $q(v_1,v_i)$
  for $C\in L_U$ {
   if $C\subseteq C_j(v_1,v_i)$ then continue;
   if $C_j(v_1,v_i)\subseteq C$ then {replace C with $C_j(v_1,v_i)$; continue}
   $L_U = C\cup C_j(v_1,v_i)$
 }

The "if" commands in the innermost "for" loop safeguard against adding redundant cut sets to $L_U$.

Example 2

Let us consider the network depicted in Fig. 4 with U={A, B, C}. Applying Algorithm 1 we obtain the following lists of minimal s-t cut sets:

$L_{A,B}$ = {1,2}, {1,3,4}, {1,3,5}
$L_{A,C}$ = {1,2}, {2,3,4}, {2,3,5}

The above lists serve as input data for Algorithm 2 which yields:

$L_U$ = {1,2}, {1,3,4}, {1,3,5}, {2,3,4}, {2,3,5}

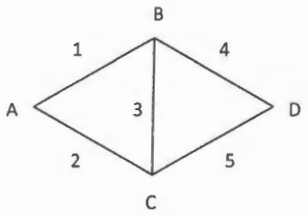


Fig. 4. A network structure with undirected links

Let now $\Gamma$ be a graph with directed edges. By definition, let

(4.3) $\Phi_{v\to w}(x_1,\dots,x_n) = \bigvee_{j=1}^{q(v,w)} \bigwedge_{i\in C_j(v,w)} x_i$

where the symbols used in (4.3) have the same meanings as those used in (4.1). However, it should be noted that for a graph with directed edges $q(v,w) \neq q(w,v)$, $C_j(v,w) \neq C_j(w,v)$, or $\Phi_{v\to w} \neq \Phi_{w\to v}$ may hold. Clearly, $\Phi_{v\to w}$ is a Boolean function equal to 1 if there is no connection from v to w, otherwise $\Phi_{v\to w}$ is equal to 0. Based on lemma 2 we can formulate the following proposition

Proposition 2

Let $U=\{v_1,\dots,v_k\}$ be a subset of V. U is disconnected if and only if

(4.4) $\Phi_{v_1\to v_2} \vee \Phi_{v_2\to v_3} \vee \dots \vee \Phi_{v_{k-1}\to v_k} \vee \Phi_{v_k\to v_1} \equiv 1$

Proof

Lemma 2 yields that U is disconnected if and only if at least one link in the connections cycle $v_1\to v_2\to \dots \to v_k\to v_1$ is failed, i.e. the equality (4.4) holds.

In view of (4.4) the list $L_U$ of all minimal K-terminal cut sets separating the nodes of $U=\{v_1,...,v_k\}\subseteq V$ is the union of the lists $L_{1\to2}$, $L_{2\to3}$,..., $L_{k-1\to k}$, $L_{k\to1}$, with redundant cut sets removed, where $L_{i\to j}$ is the list of all minimal s-t cut sets separating $v_i$ from $v_j$. Thus we obtain the following algorithm to generate $L_U$.

Algorithm3

$L_U = L_{k\to1}$;

for i=1 to k−1

for j=1 to $q(v_i,v_{i+1})$

for $C\in L_U$ {

if $C\subseteq C_j(v_i,v_{i+1})$ then continue;

if $C_j(v_i,v_{i+1})\subseteq C$ then {replace C with $C_j(v_i,v_{i+1})$; continue}

$L_U = C\cup C_j(v_i,v_{i+1})$

}

Example 3

Let us consider the network depicted in Fig. 5 with U={A, B, C}. Applying Algorithm 1 we obtain the following lists of minimal s-t cut sets:

$L_{A\to B}$ = {1,2}, {1,3}

$L_{B\to C}$ = {1,4}, {1,5}, {2,4}, {2,5}

$L_{C\to A}$ = {2,3}, {1,2}

The above lists serve as input data for Algorithm 3 which yields:

$L_U$ = {2,3}, {1,2}, {1,3}, {1,4}, {1,5}, {2,4}, {2,5}

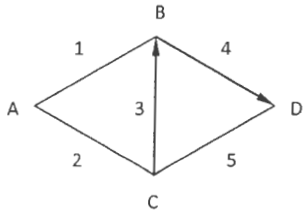


Fig. 5. A network structure with directed links

5. Generating minimal cut sets of other types

In practice cut sets of types other than s-t or K-terminal can be encountered. For example a network administrator may want to divide a complex network into a number of fragments such that unrestricted communication is possible within each fragment, but certain restrictions are imposed on inter-fragment communication. In order to illustrate such a problem let us consider the following example.

Example 4

Four LANs (local area networks) denoted by A, B, C, D are connected by three routers denoted by P, Q, R by means of nine router-to-network interfaces denoted by 1,...,9, as shown in Fig. 6.

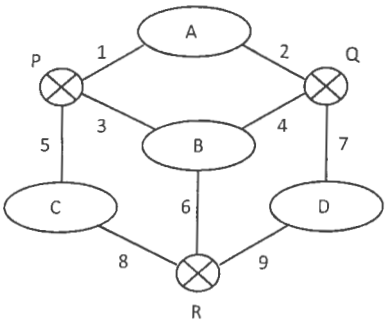


Fig. 6. An exemplary network environment

The considered network can be modeled by a bipartite undirected graph with {A,B,C,D} and {P,Q,R} as the sets of nodes and {1,...,9} as the set of edges. Let us assume that the network administrator has to choose interfaces on which traffic filtering has to be performed in order to check data sent between two groups of LANs - first consisting of A and B, and second – of C and D. This problem can be formulated in terms of cut sets in the following way: assuming that only the interfaces (edges) are subject to failure find all cut sets separating A and B from C and D, but not separating A from B or C from D. Clearly, such cut sets are neither of s-t nor of K-terminal type. The Boolean function corresponding to the family of thus defined cut sets has the following form:

(5.1) $\Phi = \Phi_{A\sim C} \wedge \Phi_{A\sim D} \wedge \Phi_{B\sim C} \wedge \Phi_{B\sim D} \wedge (1 - \Phi_{A\sim B}) \wedge (1 - \Phi_{C\sim D})$

where $\Phi_{v\sim w}$ are defined by (4.1). The arguments of $\Phi$ are the variables $x_1,...,x_9$, where $x_i = 1$ if the i-th interface is failed, $x_i = 0$ otherwise, $1 \leq i \leq 9$. Let us note that

(5.2) $\neg A\sim C \wedge \neg A\sim D \wedge A\sim B \Rightarrow \neg B\sim C \wedge \neg B\sim D$

thus $\Phi$ can be written in the simpler form:

(5.3) $\Phi = \Phi'_{A\sim C} \wedge \Phi'_{A\sim D} \wedge (1 - \Phi_{A\sim B}) \wedge (1 - \Phi_{C\sim D})$

where $\Phi'_{v\sim w}$ is obtained from $\Phi_{v\sim w}$ by removing products which correspond to cut sets separating A from B or C from D – due to the last two factors such products cannot be $\Phi$'s components. We have

(5.4) $L'_{A\sim C} = L'_{A\sim D} = \{1,3,6,7\}, \{2,4,5,6\}, \{5,6,7\}$

where $L'_{A\sim C}$ and $L'_{A\sim D}$ correspond to $\Phi'_{A\sim C}$ and $\Phi'_{A\sim D}$ respectively, thus the list corresponding to $\Phi'_{A\sim C} \wedge \Phi'_{A\sim D}$ is equal to $L'_{A\sim C}$, hence it does not contain cut sets separating A from B or C from D. In consequence L corresponding to $\Phi$ is given by

(5.5) $L = L'_{A\sim C} = \{1,3,6,7\}, \{2,4,5,6\}, \{5,6,7\}$

6. Generating all minimal K-terminal path sets

A K-terminal path set is such a set P, composed of elements of a considered network, that if all elements in P are operable then each two nodes v and w belonging to a set $U \subseteq V$ are connected, i.e. there exists a connection from v to w. A K-terminal path set is minimal if none of its subsets is a K-terminal path set. In this chapter efficient methods of finding all minimal K-terminal path sets will be presented

Let $\Gamma$ be an undirected graph, v and w – arbitrary nodes of $\Gamma$, r(v,w) – the number of minimal s-t path sets separating v from w, and $P_j(v,w)$ – the j-th such cut set, $1 \le j \le r(v,w)$. By definition, let

(6.1) $$\Theta_{v\sim w}(y_1, \dots, y_n) = \bigvee_{j=1}^{r(v,w)} \bigwedge_{i\in P_j(v,w)} y_i$$

where $y_i$, $1 \le i \le n$ represent the components' reliability states; $y_i = 1$ if the i-th component is operable, $y_i = 0$ if it is failed. Clearly, $\Theta_{v\sim w}$ is a Boolean function equal to 1 if there is a connection between v and w, otherwise $\Theta_{v\sim w}$ is equal to 0. Based on lemma 1 we can formulate the following proposition:

Proposition 3

Let $U=\{v_1,\dots,v_k\}$ be a subset of V. U is connected if and only if

(6.2) $$\Theta_{v_1\sim v_2} \wedge \dots \wedge \Theta_{v_1\sim v_k} = 1$$

Proof: Lemma 1 yields that U is connected if and only a node in U is connected to each of the remaining nodes in U, i.e. the equality (6.2) holds.

From (6.2) it follows that there is one-to-one correspondence between the expression on the left-hand side of (6.2), brought to the form of a sum of non-redundant products, and the

list of all K-terminal path sets connecting the nodes of U. The algorithm generating this list is presented in extenso in []. It should be noted that it employs certain techniques, developed by the author of [], which make it highly efficient.

Let now $\Gamma$ be a graph with directed edges. By definition, let

$$\Theta_{v\to w}(y_1,\dots,y_n) = \bigvee_{j=1}^{r(v,w)} \bigwedge_{i\in P_j(v,w)} y_i \qquad (6.3)$$

where the symbols used in (6.3) have the same meanings as those used in (6.1). However, it should be noted that for a graph with directed edges $r(v,w) \neq r(w,v)$, $P_j(v,w) \neq P_j(w,v)$, or $\Theta_{v\to w} \neq \Theta_{w\to v}$ may hold. Clearly, $\Theta_{v\to w}$ is a Boolean function equal to 1 if there is a connection from v to w, otherwise $\Phi_{v\to w}$ is equal to 0. Based on lemma 2 we can formulate the following proposition

Proposition 4

Let $U=\{v_1,\dots,v_k\}$ be a subset of V. U is connected if and only if

$$\Theta_{v_1\to v_2} \wedge \Theta_{v_2\to v_3} \wedge \dots \wedge \Theta_{v_{k-1}\to v_k} \wedge \Theta_{v_k\to v_1} = 1 \qquad (6.4)$$

Proof: Lemma 2 yields that U is connected if and only if all the links in the connections cycle $v_1\to v_2\to \dots \to v_k\to v_1$ are operable, i.e. the equality (6.4) holds.

From (6.4) it follows that there is one-to-one correspondence between the expression on the left-hand side of (6.4), brought to the form of a sum of non-redundant products, and $L_U$ – the list of all K-terminal path sets connecting the nodes of U. By the argument similar to that used in constructing Algorithm 1 we obtain the following algorithm generating $L_U$ from $L_{1\to 2}$, $L_{2\to 3}$,..., $L_{k-1\to k}$, $L_{k\to 1}$, where $L_{i\to j}$ is the list of all minimal s-t path sets providing a connection from $v_i$ to $v_j$.

Algorithm 4

$L_U \leftarrow L_{k\rightarrow 1}$

$L_{aux} \leftarrow \varnothing$

for i ← 1 to k–1 {

for $\pi \in L_U$ {

for $\rho \in L_{i\rightarrow i+1}$ {

if $I(\pi) \subset I(\rho)$ then move $\rho$ from $L_{i\rightarrow i+1}$ to $L_{aux}$

if $I(\rho) \subset I(\pi)$ then move $\pi$ from $L_U$ to $L_{aux}$

if $I(\pi) = I(\rho)$ then move $\pi$ from $L_U$ and $L_{i\rightarrow i+1}$ to $L_{aux}$

}

}

for $\pi \in L_U$ {

for $\rho \in L_{i\rightarrow i+1}$ {

x ← 1;

for $\sigma \in L_{aux}$ {

if $I(\pi\rho) \subset I(\sigma)$ then remove $\sigma$ from $L_{aux}$

if $I(\sigma) \subseteq I(\pi\rho)$ then {x ← 0; break}

}

if x=1 then add $I(\pi\rho)$ to $L_{aux}$

}

}

$L_U \leftarrow L_{aux}$

if i < k–1 then $L_{aux} \leftarrow \varnothing$

}

The first „for $\pi \in L_U$" loop in the body of the „for i = 1 to k–1" loop searches Lu and $L_{i \to i+1}$ for products which are in inclusion relation with products from the other list. Such products have to be "isolated", because they would generate redundancies, as follows from Lemma 1. The second „for $\pi \in L_U$" loop updates $L_U$ seeing to it that it does not contain redundant products (the "for $\sigma \in L_{aux}$" loop). It has to be remarked that for graphs with directed edges there is no simple criterion for stating whether a given set of components is a K-terminal path set. Thus such path sets cannot be "isolated" starting from the first cycle of the "for i ← 1 to k–1" loop, which would accelerate their generation, as was the case with s-t cut sets in Algorithm 1. They are only known once Algorithm 4 has terminated.

Example 5

Let us consider the network depicted in Fig. 5 with U={A, B, C}. Constructing the respective trees of acyclic paths we obtain the following lists of minimal s-t path sets:

$L_{A \to B}$ = {1}, {2,3}

$L_{B \to C}$ = {1,2}, {4,5}

$L_{C \to A}$ = {2}, {1,3}

Applying Algorithm 4 we obtain:

$L_U$ = {2}, {1,3}; $L_{aux} = \varnothing$;

$L_{aux}$ = {2,3}, {1,3}, {1,2};

$L_U$ = {2,3}, {1,3}, {1,2}; $L_{aux} = \varnothing$;

$L_{aux}$ = {1,2}, {2,3,4,5}, {1,3,4,5};

$L_U$ = {1,2}, {2,3,4,5}, {1,3,4,5}.

7. Bibliography


1. Abel U., Bicker R., "Determination of All Minimal Cut-Sets between a Vertex Pair in an Undirected Graph", *IEEE Transactions on Reliability*, Vol. R-31, Issue 2, pp. 167-171 (1982).
2. Arunkumar S., Lee S.H., "Enumeration of all minimal cut-sets for a node pair in a graph", *IEEE Transactions on Reliability*, Vol. R-28, Issue 1, pp. 51-55 (1979).
3. Beichelt, Frank: "Zuverlaessigkeit strukturierter Systeme", VEB Verlag Technik Berlin (1988).
4. Malinowski, Jacek: "A new efficient algorithm for generating all minimal tie-sets connecting selected nodes in a mesh-structured network", *IEEE Transactions on Reliability*, Vol. 59, No 1, pp. 203-211 (2010).
5. Prasad V.C., Sankar V., Prakasa K.S., "Generation of vertex and edge cutsets", *Microelectronics Reliability*, Vol. 32, Issue: 9 , pp. 1291-1301 (1992).
6. Ross S., "Introduction to probability models – 10$^{th}$ edition", Elsevier (2010).
7. Singh B., "Enumeration of node cutsets for an s-t network", *Microelectroics Reliability*, Vol. 34, Issue 3, pp. 559-561 (1994).
8. Yeh, W.-C., " A new algorithm for generating minimal cut sets in *k*-out-of-*n* networks", Reliability Engineering and System Safety, Vol. 99, Issue 1, pp. 36-43 (2006).
9. Yeh, W.-C., "A simple algorithm to search for all MCs in networks", European Journal of Operational Research, Vol. 174, Issue 3, pp. 1694-1705 (2006).