Instituto de Engenharia de Sistemas e Computadores de Coimbra
Institute of Systems Engineering and Computers INESC - Coimbra

Teresa Gomes
Luís Fernandes

A note on
"A simple algorithm to search all MCs in networks"

No. 11 2010

ISSN: 1645-2631

Instituto de Engenharia de Sistemas e Computadores de Coimbra
INESC - Coimbra
Rua Antero de Quental, 199; 3000-033 Coimbra; Portugal
www.inescc.pt

Work partially financially supported by programme COMPETE of the EC Community Support Framework III and cosponsored by the EC fund FEDER and national funds (FCT - PTDC/EEA-TEL/101884/2008).

A note on "A simple algorithm to search all MCs in networks"

Teresa Gomes[1,2] and Luís Fernandes[1,2]

[2]Departamento de Engenharia Electrotécnica e de Computadores da FCTUC,
Pólo 2 da Univ. Coimbra, 3030-290 Coimbra, Portugal

[3]INESC-Coimbra, Rua Antero de Quental 199,
3000-033 Coimbra, Portugal.

e-mail: teresa@deec.uc.pt, a2003107513@alunos.deec.uc.pt

August 6, 2010

Abstract

Minimal Cuts are relevant tools for evaluating a network's reliability. In [3] an algorithm is proposed for searching all the MCs in a network. This note has the purpose of clarifying the algorithm in [3], so that it can be correctly implemented.

1 Introduction

Minimal cuts (MCs) are used as tools in network reliability [3, 2], which is a difficult problem [1].

In [3] an algorithm is proposed for obtaining all MCs which separate a specific s-t node pair. After careful analysis we detected that the algorithm was incomplete (or ambiguous) and that it contained two incorrections (or possibly misprints). It is also noted that the author failed to point out that the algorithm requires the previous removal of all nodes irrelevant for the calculation of the searched MCs.

The purpose of this note is the clarification of an efficient algorithm so that it can be correctly implemented.

2 Notation

The notation proposed in [3] will be used in this document. Let $G = (V, E)$ be a connected undirected network with node set $V = \{s, t, 1, 2, \ldots, n-2\}$ and edge set E (where n

designates the number of nodes in G). The nodes s and t, are the designated source and sink node respectively.

Each element in E is an unordered pair of adjacent nodes (for example e_{uv}, e_{vu}, is an arc between nodes u and v). Using the network in figure 1, e_{12} or e_{21} is the edge between nodes 1 and 2.

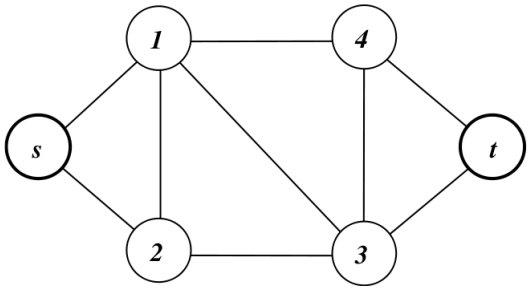

Figure 1: Example network from [3].

Let $E(U)$ designate the set of edges of the sub-graph of G induced by node set U, $U \subset V$.

Definition 2.1. $E(U) = \{e_{uv} : \forall\ u, v \in U \subseteq V \wedge e_{uv} \in E\}$

The residual sub-network of $G(V, E)$ after removing $V \backslash U$ (also represented by $U - V$) nodes, where $U \subset V$, is designated by $G(U, E(U)$. For example, if $U = \{s, 1, 2\}$, the network $G(U, E(U))$ (after removing $V \backslash U = \{3, 4, t\}$ from $G(V, E)$) is given by: $G(U = \{s, 1, 2\}, E(U) = \{e_{s1}, e_{s2}, e_{12}\})$.

A set of edges E' is a cut if this set disconnects the network. A cut E' is a Minimal Cut (MC) if no subset of E' disconnects the network. A mininmal cut can also be designated by min-cut. An s-t MC is a min-cut that separates nodes s and t.

The algorithm proposed in [3], which searches for all MCs which separate a specific s-t node pair, requires the definition of the additional terms:

- an node u is an adjacent node to a node set U ($U \subset V$), if there is an edge between node u and at least one node in U
- a s-t cut is an edge set such that no path from the source node s to the sink node t exists after removing it

- A MC is a cut set such that if any arc is removed from this cut set, then the remaining set is no longer a cut set.
- A MCV *candidate* is a node subset such that its removal will disconnect the source node (s) and the sink node (t) in the corresponding network.
- A MCV is an MCV *candidate* in $G(V, E)$, say U, such that the edge set from the nodes in U to the nodes in $V \backslash U$ is a MC.

So a MCV *candidate* is associated with a s-t cut and and a MCV is associated with a s-t MC.

A $MC(U)$ is a set of edges that separates nodes U from the nodes in $V \backslash U$ in $G(V, E)$.

Definition 2.2. *Given a graph $G = (V, E)$, the edge set from the nodes in U to the nodes in $V \backslash U$ is $MC(U)$, $MC(U) = \{e_{uv} : \forall\ u \in U \subseteq V \wedge v \notin U \wedge e_{uv} \in E\}$.*

A set U is a MCV in $G(V, E)$ if and only if $MC(U)$ is a MC which separates nodes s and t – by Theorem 1 in [3] – assuming $s \in U$ and $t \in V \backslash U$.

The network is assumed to satisfy the following conditions [3]:

- Nodes do not fail.
- The networks graph is connected and free of self-loops.
- Each edge has two states: working or failed.
- All flows in the network obey the conservation law.

We do not see the relevance of the last condition because no other reference to flows is made in [3].

3 The algorithm

The original algorithm is in appendix A. The corrected version of the algorithm differs from the original versions as follows.

Algorithm Min-Cut: Modified version of the Algorithm in appendix, assuming all irrelevant nodes for the calculation of the s-t MCs have been removed.

Data: A connected graph $G = (V, E)$, with node set V, edge set E, a source node s and a target node t

Result: P, the set of MCVs in $G = (V, E)$

1 $i \leftarrow 0$, $k \leftarrow 0$, $S \leftarrow \{s\}$, $U_0 \leftarrow \{s\}$, $T \leftarrow V - \{s\}$, $N_0 \leftarrow \{t\}$
2 $P \leftarrow \{s\}$
3 **Step 1:**
4 $D = T - N_i$
5 **if** *Exists a node* $u \in D$ *adjacent to* S **then**
6 $S \cup \{u\}$ is a MCV *candidate*
7 Go to Step 2
8 **else**
9 Go to Step 4
10 **Step 2:**
11 **if** $G = (T - \{u\}, E(T - \{u\}))$ *is a connected network* **then**
12 $S \cup \{u\}$ is a MCV
13 Go to Step 3
14 **else**
15 Any node set containing $S \cup \{u\}$ is not a MCV
16 `erase`(u, D)
17 Go back to line 5 (in Step 1)
18 **Step 3:**
19 $i \leftarrow i + 1$, $k \leftarrow k + 1$
20 $S \leftarrow S \cup \{u\}$
21 $U_k \leftarrow S$
22 $P \leftarrow P \cup \{U_k\}$
23 $T \leftarrow T - \{u\}$
24 $N_i \leftarrow N_{i-1}$
25 Go back to Step 1
26 **Step 4:**
27 **if** $i = 1$ **then**
28 STOP
29 **else**
30 Remove the last node, say v, in S
31 $i \leftarrow i - 1$
32 $N_i \leftarrow N_i \cup \{v\}$
33 $T \leftarrow T \cup \{v\}$
34 Go back to Step 1

- In line 2 the initial set in P is made equal to $\{s\}$, instead of $\emptyset$, otherwise that MCV would never be added to P.

Also in Example 1 [3, page 1701, line -4] in Step 3 $P = P \cup \{U_1\}$ where P was initially $\emptyset$ and $U_1 = \{s, 1\}$ (and U_0 would never be added to P).

- In line 4 the auxiliary set D was included, and initially is equal to $T - N_i$. The purpose of this set is to keep track of the elements of $T - N_i$ that have not been excluded as elements that may be part of an MCV candidate. When for a given $u \in D$ the condition evaluated in line 11 returns the value false, then according to line 15 "any node set containing $S \cup \{u\}$ is not a MCV" and the algorithm must search for another node in $T - N_i$, adjacent to S in Step 1, but before that can occur node u is erased from D (see line 16), so that it is never selected again in Step 1.

- In the original algorithm the step sequence is clearly indicated in every step except at the end of Step 2 after "Otherwise, any node set contains $S \cup \{u\}$ is not a MCV".

  Clearly Step 3 can not follow Step 2 when $S \cup \{u\}$ is not an MCV. Step 4 can not follow Step 2, because the algorithm needs to try other nodes in $T \backslash N_i$, without changing i. Moreover, if the algorithm proceeded to step 4 it would not be a Depth First Search as stated in [3]. Hence the only logical next step, in the original algorithm at the end of Step 2 after "Otherwise, any node set contains $S \cup \{u\}$ is not a MCV", is Step 1.

  Because of the introduction of the auxiliary set D, we have in line 17 of the modified version "Go back to line 5" instead of simply "go to step 1" (which is missing in the original version).

- In Step 4, line 27 the stopping condition of the algorithm was modified to $i = 0$, in order to allow the generation of all the problem solutions.

With $i = 0$ the algorithm would stop after generating $U_6 = \{s, 1, 4\}$, as can be seen if we resume Example 1 in [3] where it was left (with $T = \{2, 3, 4, t\}$, $i = 1$, $N_1 = \{2, t\}$) and after correcting Step 1 in page 1073, where is $T \backslash N_2$ should be $T \backslash N_1$. In our continuation of that example it is assumed that in Step 2 whenever "any node set containing $S \cup \{u\}$ is not a MCV" the algorithm goes to Step 1 in search for a unexamined node in $T - N_i$.

Resuming Example 1 [3]:

Step 1. Since node $3 \in (T \backslash N_1 = \{3, 4\})$ is adjacent to $S = \{s, 1\}$, go to Step 2.

Step 2. $u = 3$, $G = (T - \{u\}, E(T - \{u\}))$ is not a a connected network, then any node set containing $S \cup \{u\}$ is not a *MCV and go to Step 1.*

Step 1. Since node $4 \in (T \backslash N_1 = \{3, 4\})$ is adjacent to $S = \{s, 1\}$, go to Step 2.

Step 2. $u = 4$, $G = (T - \{u\}, E(T - \{u\}))$ is a connected network, then $S \cup \{u\}$ is a *MCV* and go to Step 3.

Step 3. $i \leftarrow i + 1$, that is $i \leftarrow 2$;

$k \leftarrow k + 1$, that is $k \leftarrow 6$;

$U_k \leftarrow S \cup \{u = 4\}$, that is $U_6 = \{s, 1, 4\}$

$S \leftarrow S \cup \{u\}$, that is $S \leftarrow \{s, 1, 4\}$

$P \leftarrow P \cup \{U_k\}$, that is $P = \{U_1, U_2, U_3, U_4, U_5, U_6\}$

$T \leftarrow T - \{u\}$, that is $T = \{2, 3, t\}$

$N_i \leftarrow N_{i-1}$, that is $N_2 = \{2, t\}$

Go back to Step 1

Step 1. Since node $3 \in (T \backslash N_2 = \{3\})$ is adjacent to $S = \{s, 1, 4\}$, go to Step 2.

Step 2. $u = 3$, $G = (T - \{u\}, E(T - \{u\}))$ is not a a connected network, then any node set containing $S \cup \{u\}$ is not a *MCV and go to Step 1.*

Step 1. Since there are no more nodes in $(T \backslash N_2 = \{3\})$ to evaluate, go to step 4.

Step 4. Remove the last node, say $v = 4$, in S, that is $S = \{s, 1\}$

$i \leftarrow i - 1$, that is $i \leftarrow 1$

$N_i \leftarrow N_i \cup \{v\}$, that is $N_1 = \{2, 4, t\}$

$T \leftarrow T \cup \{v\}$ that is $T = \{2, 3, 4, t\}$

Go back to Step 1

Step 1. Since node $3 \in (T \backslash N_1 = \{3\})$ is adjacent to $S = \{s, 1\}$, go to Step 2.

Step 2. $u = 3$, $G = (T - \{u\}, E(T - \{u\}))$ is not a a connected network, then any node set containing $S \cup \{u\}$ is not a MCV *and go to Step 1.*

Step 1. Since there are no more nodes in $(T \backslash N_1 = \{3\})$ to evaluate, go to step 4.

Step 4. $i = 1$: the original algorithm would stop without generating U_7 and U_8.

The importance of removing irrelevant nodes before using the (modified) algorithm for obtaining all the s-t MCs, can be easily illustrated adding a spur to the network graph. If a spur is added at node 1 (adding edge e_{15}), when node 1 is examined (with $i = k = 0, S = U_0 = \{s\}, T = \{1, 2, 3, 4, 5, t\}, N_0 = \{t\}$), in Step 1 , $G = (T - \{u\}, E(T - \{u\}))$ is a disconnected network because node 5 becomes an isolated node, and U_1 would not be obtained. Hence U_i, $i = 2, 3, 4, 5, 6$, would not be generated because according to the algorithm any node set containing $S \cup \{1\} = \{s, 1\}$ could not be a MCV! This results from Property 2 in [3] which is only valid if, for each considered s-t node pair, the irrelevant nodes for paths between s and t have been previously removed.

A Original Algorithm

Algorithm Min-Cut: Find MCVs in a network $G(V, E)$

Data: A connected graph $G = (V, E)$, with node set V, edge set E, a source node s and a target node t

Result: P, the set of MCVs in $G = (V, E)$

$i \leftarrow 0$, $k \leftarrow 0$, $S \leftarrow \{s\}$, $U_0 \leftarrow \{s\}$, $T \leftarrow V - \{s\}$, $N_0 \leftarrow \{t\}$

$P \leftarrow \emptyset$

Setp 1:

- **if** *Exists a node* $u \in T - N_i$ *adjacent to* S **then**
  - $S \cup \{u\}$ is a MCV *candidate*
  - Go to Step 2
- **else**
  - Go to Step 4

Step 2:

- **if** $G = (T - \{u\}, E(T - \{u\}))$ *is a connected network* **then**
  - $S \cup \{u\}$ is a MCV
  - Go to Step 3
- **else**
  - Any node set containing $S \cup \{u\}$ is not a MCV

Step 3:

- $i \leftarrow i + 1$, $k \leftarrow k + 1$
- $S \leftarrow S \cup \{u\}$
- $U_k \leftarrow S$
- $P \leftarrow P \cup \{U_k\}$
- $T \leftarrow T - \{u\}$
- $N_i \leftarrow N_{i-1}$
- Go back to Step 1

Step 4:

- **if** $i = 1$ **then**
  - STOP
- **else**
  - Remove the last node, say v, in S
  - $i \leftarrow i - 1$
  - $N_i \leftarrow N_i \cup \{v\}$
  - $T \leftarrow T \cup \{v\}$
  - Go back to Step 1

References

[1] C. J. Colbourn. *The combinatorics of network reliability.* The International Series of Monographs on Computer Science. Oxford University Press, 1987.

[2] D. R. Shier. *Network Reliability and Algebraic Structures.* Claredon Press - Oxford, 1991.

[3] Wei-Chang Yeh. A simple algorithm to search for all MCs in networks. *European Journal of Operational Research*, 175(3):1694–1705, November 2006.