A new approach for enumerating minimal cut-sets in a network

Ahmed R. Abdelaziz
Electrical Eng. Dep., Faculty of Eng., Alexandria University, Alexandria, Egypt

This paper deals with the enumeration of all minimal cut sets separating an undirected graph into two sub-graphs. The algorithm does not enumerate trees or spanning trees as a first step. This approach has yielded an algorithm, which generates minimal cut sets at O (½n(n-1)), where n is the number of vertices in the graph, computational effort per cut set. Formal proofs of the algorithm and its complexity are presented. Results of some computational experience show that, a) this algorithm is appreciably faster than previous algorithms, and b) can handle much larger graphs due to less memory requirements.

في هذا البحث نقدم طريقة لتعيين مجموعات القطع والتي تفصل الشبكة إلى قسمين منفصلين تماما. هذه الطريقة لا تحتـــاج إلـــى تعريف مسبق لشجرة الشبكة. وهي تقتصد في زمن تشغيل الحاسب إلى ما يقرب من $O(\frac{1}{2}n(n-1))$ حيث n هي عدد نقـــاط الشبكة، وفي هذا البحث سنقدم إثبات ذلك. وسنجد أيضا في سرد نتائج البحث أن هذه الطريقة أسرع بكثير وكذلك لها القدرة علـــى تناول الشبكات كبيرة الحجم وذلك لقلة ما تحتاجه من ذاكرة الحاسب أثناء تعينها للقواطع. ويتضمن البحث العديد من الأمثلة لبيـــان مقدرة هذه الطريقة.

Keywords: Minimal cut set, Undirected graph, Enumeration, Algorithm, Cut sets-based Proble

1. Introduction

This paper considers the problem of enumerating all the minimal cut sets separating a graph into two sub-graphs: this problem is one of the most fundamental problems in graph theory as well as being a basic step in evaluating the reliability of networks encountered in computer, communication, electrical-power systems... etc.

Several algorithms have been developed for directed and undirected graphs [1-8]. The approaches in these papers are based on the use of two broad categories of methods:

1- Gaussian elimination and Boolean algebra.
2- Implicit or explicit enumeration scheme.

Enumeration appears to be the most computationally efficient. Nevertheless, the enumeration approach of all minimal cut sets still appears as a fundamental step in many algorithms for evaluating the reliability of networks. However, the enumeration of minimal cut sets in general graphs is known to be an NP-hard problem [1-8].

The most widely cited reliability problems are:

i. Terminal-pair reliability: is a commonly used measure of connectivity. It is the probability of obtaining service between a pair of operative centers, called source and sink, in terms of reliability for each communication link/node in the network [9].

ii. K-Terminal reliability: is the probability that all nodes in some specified set K of target nodes are joined by paths of non-failed edges [10].

iii. All-Terminal reliability (Global reliability): is the probability of existence of a minimal set of up-state edges such that all the nodes of the network are joined by paths of non-failed edges [11]. Such a minimal set of edges is known as spanning tree of the network [12].

Several algorithms exist for finding the minimal cut sets of a network. Some of these conventional methods [6-8], the minimal cut sets were deduced by first finding all paths and then evaluating the combinations of failure that break these minimal paths. Some of published papers [1-8], did not consider multi-sources/multi-sinks, which is the case of large power systems. A vertex cut sets enumeration was presented in [3], this is not valid for power systems due to vertices in such systems are more reliable than edges. In [5], the difficulty of enumerating the minimal cut

sets was overcoming via the most probable cut sets to occur was considered only. However, this is introducing another problem, which is how to rank the partitions of a large/ complex network.

The objective of this paper is to construct a polynomial-time algorithm for the enumeration of all minimal cut sets. Section 2 defines necessary terms commonly used in graph theory. Section 3 describes a polynomial-time cut set enumeration algorithm, which is followed by an illustrative example and the computational experiences are presented in section 4.

2. Preliminaries

2.1. Definitions

This section defines some technical terms used in graph theory. The terms are mainly concerned with undirected graph [12].

A graph. A graph G = (V, E) consists of a set of objects V = {v_1, v_2,...} called vertices and another set E = {e_1, e_2, ...} whose elements are called edges. In case of electrical power network graph the buses are represented by vertices, while generators, transmission lines, transformers or lumped loads are represented by edges.

Sub-graph. A graph g is said to be a sub-graph of a graph G if all the vertices and the edges of g are in G and each edge of g has the same end vertices in g as in G. The symbol from set theory, $g \subset G$, is used in stating g is a sub-graph of G.

Vertex-disjoint sub-graph. Two (or more) sub graphs g_1 and g_2 of a graph G are said to be vertex disjoint if g_1 and g_2 if they don't have any vertices in common.

A path. A path is defined as a finite alternating sequence of vertices and edges beginning and ending with vertices. Such that each edge is incident with the vertices preceding and following it. Neither edges nor vertex appears more than once.

Connected graph. A graph G has said to be connected if there is at least one path between every pair of vertices in G.

Tree. A Tree is a connected graph without any closed loops.

Minimal cut-set (MCS). In a connected graph G, a cut set is a set of edges whose removal from G leaves G disconnected, provided removal of no proper subset of these edges disconnect G. A cut-set always cuts a graph into two subgraphs. So, another way of looking at a cut-set is this: if we partition all the vertices of a connected graph into two vertex-disjoint sub-graphs (Note that one or both of these two sub-graphs may consist of just one vertex).

Vertex cut set. A cut set involving vertices only.

Edge cut set. A cut set involving edges only.

g_1, g_2. g_1 and g_2 are vertex disjoint sub-graphs of G, where $g_1=(V_1, E_1)$, and $g_2=(V_2, E_2)$, where $V_1, V_2 \subset V$, and $E_1, E_2 \subset E$, and also $V_1 \cup V_2 = V$, and $E_1 \cup E_2 = E$.

Cut-sets are of great importance in studying properties of communication, power and transportation networks. In these fields of study, our concern is to look at all cut-sets of the graph, and the one with the smallest number of edges is the most vulnerable.

2. 2. Data representation

Data structure is an important aspect of designing efficient algorithms. Some of the data structures are described as follows.

2.2.1. Bit representation

Soh [13] has discussed the advantages and disadvantages of one kind of data representation, namely bit vector representation In Soh's work, path in a graph with e edges is presented by an identifier having e bits. An up edge of the graph is denoted by a binary 1. A binary 0 sums for a "do not care" state. The logic of the suggested representation is as follows:

Vertex identifier. In a vertex identifier, an incident edge to that vertex is denoted by a binary 1 and a binary 0 otherwise.

Cut set identifier. In a cut set identifier a binary 1 represents a removed edge and a binary 0 represents a do not care edge. The ability of bit representation to handle set theoretic operations like union, intersection, subset... etc are shown in the following definitions:

Union. If A and B are two bit identifiers, the union of these sets is A OR B.

Intersection. The intersection of A and B is A

AND B.
Subset. If A AND B= B, then set B is a subset of A.

2.2.2. Code representation

In modern software, all the previous relations can be implemented on a TURBO PASCAL V-6.0, by using an integer variable and not an identifier representation. So, a form of the data representation, namely code representation was suggested by Abdelaziz [14].

A path/node/cut identifier in a graph with e edges is represented by an integer code (long integer declaration for large graphs). The integer code representation is a decimal form of the bit vector representation.

Example:

Consider the cut 1, 4, 6, 7 in figs 1, and vertex 2, which is the connection of edges 1, 2, and 6. The cut is stored in the memory as identifier {1001011} or as code (105). The vertex is also stored as {1100010} or as code (35).

2.2.3. Set operator

Sets are intended to bridge the gap between the theoretic, artificial world of the computer program and the hard, cold, concrete world of everyday existence. Because sets define collections of objects that represent the real world, it seems only fitting that the common logic of everyday life can be applied to their manipulation.

A set is a finite collection of elements that share the same previously defined type, called the base type. A set variable is declared as in fig 2.

The maximum number of elements in a set is 256. Further, the upper and lower bounds of the base type must have ordinal values that are themselves within the range 0 through 255.

The relational set operators test either for equality or inclusion. The only consideration is the presence or absence of elements; neither the ordering of elements in a set operand nor the relative magnitude of individual elements has any bearing on the result of the test. Table 1 summarizes these operators.

Just as it does with arithmetic relation operators, the NOT operator negates (reverses) the state of a Boolean value. Therefore, if Set 1 = Set 2 is True, NOT (Set 1 = Set 2) is False.

For example, consider the sub-graph g_1 and g_2 in fig. 1 V_1 is stored in the memory as set [2,3,4], E_1 as [1,2,3,4,6,7], while V_2 is stored in the memory as set [1,5] and E_2 as [1,4,5,6,7]. As a good notice, the MCS [1,4,6,7] is the intersection of E_1, and E_2.

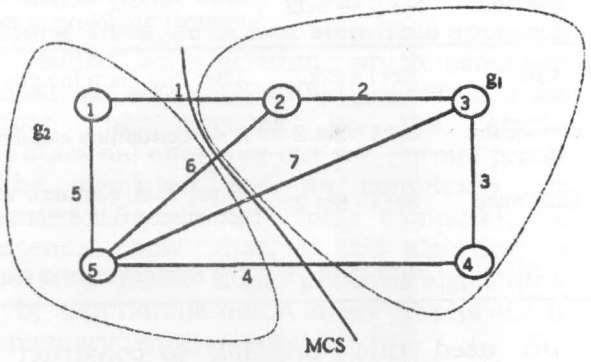


Fig.1. A 5-vertices /7 edges graph.

```
TYPE
Set 1                    = set of 1. .e;
Set 2                    = set of 1. .n;
VAR
Cut set                    : Set 1;
Vertex                   : Set 2;
```

Fig. 2. Implementation of set operator.

3. Algorithm

Our algorithm (ARAZ) is an enumeration scheme derived from the basic approach of Jensen & Bellmore (JB) [1]. This idea was later extended to (YTL) [4] to improve the computational efficiency and space requirements of the algorithm.

- JB proved that, in a 2-terminal graph, the removal of the edges of a minimal cut set partitions the graph into two vertex-disjoint sub-graphs such that:
- the source terminal belongs to one sub-graph and the sink terminal belongs to the other,
- each sub-graph should be a connected sub-graph,
- each edge of a minimal cut has one terminal in one set and the other terminal in the other set.

Table 1
Set operators

Function	Syntax	Returns with
Equality	Set 1 = Set2	True if Set 1 and Set 2 are identical. Every element in Set 1 is contained in Set2, and every element in Set2 is contained in Set1.
Not equal	Set 1 <>Set2	True if one of the sets contains at least one element that is not in the other set.
Sub-set	Set 1 <= Set2	True if every element in Set 1 is also in Set 2. In addition, Set 2 contains at least one other element not found in Set1.
Union	Set 1 + Set2	A set that contains one of every element found in either operand.
Intersection	Set 1 * Set 2	A set containing elements common to both set operands.
Difference	Set 1 - Set 2	A set that contains all elements of the first set that is not found in the second set, sometimes called the complement operator.
IN	Elem IN Set 1	The element elem is found in Set 1.

JB used this principle to construct a systematic binary-tree search that generates all minimal cut sets by sequentially adding vertices & branches until each path culminates into a minimal cut set.

YTL [4], shows that the JB partitioning principle can be implemented without constructing a binary search-tree. Indeed, YTL constructs each minimal cut set in 1 step by adding & removing arcs in a manner that satisfies the 113 partitioning principle assuming that the graph satisfies the triangular property. The minimality procedure is necessary at the termination of both JB and YTL techniques to exclude all non-minimal cut sets, which raises the computational time exponentially.

Our algorithm (ARAZ) is a further development to JB [1], but it can be considered as a generalization of a cut set enumeration techniques because it is useful for multi-source/multi-sink graphs.

Simple example:

This example demonstrates the general idea of ARAZ. In fig 1, all vertices {1..5 } are considered to be either source and/or sink vertices of the graph. The edges are represented by undirected edges {1..7}. Let V_1 and V_2 are the two mutually exclusive sets of vertices defined by the JB principle.

Step 1: consider n first order sub-graphs, each of them contains one vertex of the graph. Therefore we have n vertices-disjoint sub-graphs and the minimal cut set associated with this partitioning listing as follows:

i	g_1	g_2	Cut set
1	{1}	{2,3,4,5}	{1,5}
2	{2}	{1,3,4,5}	{1,2,6}
3	{3}	{1,2,4,5}	{2,3,7}
4	{4}	{1,2,3,5}	{3,4}
5	{5}	{1,2,3,4}	{4,5,6,7}

Step 2: Initialize counter $k_0=0$, k=n, and $k_1=k$.
Step 3: for $i=k_0+1$ to k_1 do the following steps.
Step 4: for j=1 to n; remove one vertex from set g_2 and add it to set g_1, so we will obtain second order g_1, in a manner that does not violate the following six conditions:

i. The vertex removed from g_2 has an edge to a vertex in g_1.
ii. g_1 is not previously studied as a second order sub-graph, whose were stored in memory locations $[(k_1+1). .k]$.
iii. Sub-graph g_1 does not contain vertex j.
iv. Only two sub-graphs resulted after cutting.
v. A resulting cut set does not include any loops.
vi. Cut set does not contain more than two arcs.

As an illustration, if we remove vertex 2 from the 1st g_2, the resulting sub-graph are g_1= {1,2} and g_2= {3,4,5}, resulting a cut set {2,5,6} which is acceptable, while if we remove vertex 5 from the 2nd g_2, the resulting sub-graphs are g_1={2,5} and g_2= {1,3,4}, resulting a cut set {1,2,4,5,7} which does not satisfy conditions (iii) and (vi).

Step 5: let k=k+1, store the kth g.

Step 6: let $k_0=k_1$, and $k_1=k$, then go to step 3 to generate the next order of g.

Step 7: repeat steps 3-6, until maximum order of g_1 (=n/2) was reach.

4. Computational experience

At this time, we can summarize the features of ARAZ as follows:

i. No need to delta/star transformation.
ii. No need to check up the minimality of produced cut sets.
iii. No need to storage memory.
iv. It can handle multi-source/multi-sink networks.

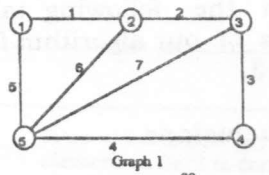
In the following table, we show some results of our algorithm for nine graphs given in fig. 3.

5. Conclusions

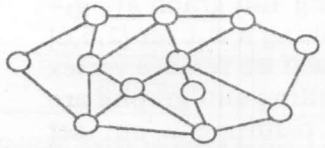
In this paper, the enumeration of all minimal cut sets separating an undirected graph into two sub-graphs is introduced. The algorithm does not enumerate trees or spanning trees as a first step. This approach has yielded an algorithm, which generates minimal cut sets at O(½n(n-1)), where n is the number of vertices in the graph, computational effort per cut set. Formal proofs of the algorithm and its complexity are presented. Results of some computational experience show that, a) this algorithm is appreciably faster than previous algorithms, and b) can handle much larger graphs due to less memory requirements.

Table 2
Results for ARAZ algorithm

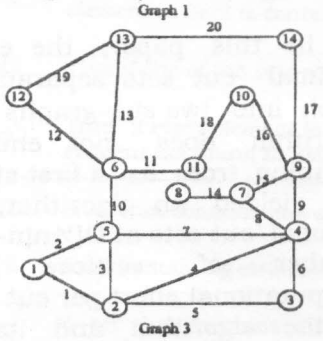
Graph	n	e	Cut sets	CPU time (sec)		O[½n(n-1)]	Time/cut (msec)
				The present algorithm ARAZ	Published algorithm [2]		
1	5	7	10	0.0	0.45	8	0.0
2	12	20	228	0.16	703	66	0.7
3	14	20	111	0.1	666	91	0.9
4	17	36	405	0.76	1750	88	1.9
5	20	38	5322	29.93	-	190	5.6
6	24	38	335	0.76	1833	276	2.3
7	30	41	1677	5.32	-	435	3.2
8	37	44	2156	4.55	-	648	2.1
9	57	80	15349	163.56	-	1568	10.6



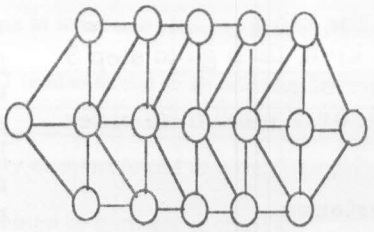
Graph 1



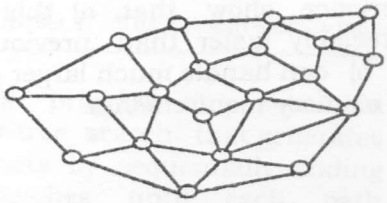
Graph 2



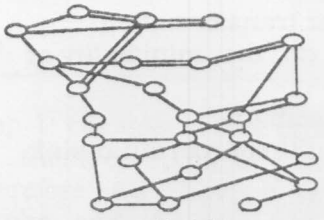
Graph 3



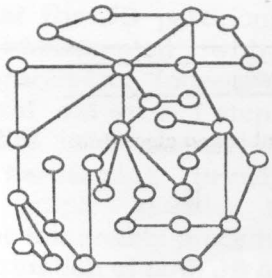
Graph 4



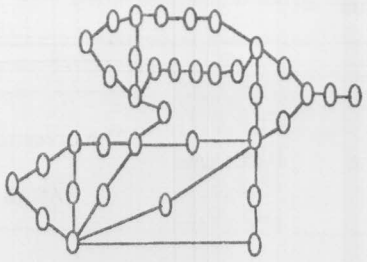
Graph 5



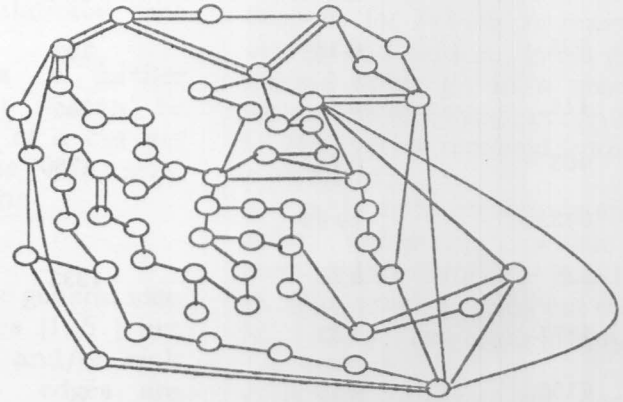
Graph 6



Graph 7



Graph 8



Graph 9

Fig. 3. A nine graph used to test the performance of ARAZ algorithm.

References

[1] P. Jensen, and M. Bellmore, "An algorithm to determine the reliability of a complex system", IEEE Transactions on Reliability, Vol. 18, pp.169-174 (1969).

[2] Y. Chen, and M. Yuang. "A cut-based method for terminal-pair reliability", IEEE Transactions on Reliability, Vol. 45, pp. 413-416 (1996).

[3] C. Patvardhan, V. Prasad, and V. Pyara, "Vertex cut sets of undirected graphs", IEEE Transactions on Reliability, Vol. 44, pp. 347-353,1995.

[4] L. Yan, H. Taha, and I. Landers, "A recursive approach for enumerating minimal cut sets in a network", IEEE Transactions on Reliability, Vol.43, pp. 383-388(1994).

[5] S. Banerjee, and V. Li, "Order-p: an algorithm to order network partitioning", IEEE Transactions on Reliability, Vol. 43, pp. 310-320 (1994).

[6] G. Jasmon, and O. Kai, "A new technique in minimal path and cut sets evaluation", IEEE Transactions on Reliability, Vol. 34, pp. 136-141(1985).

[7] C. Sung, and B. Yoo, "Simple enumeration of minimal cut sets separating 2 vertices in a class of undirected planar graphs", IEEE Transactions on Reliability, Vol. 41, pp. 63-71(1992).

[8] G. Jasmon, and K. Foong, "Cut sets analysis of networks using basic minimal paths and network decomposition", IEEE Transactions on Reliability, Vol. 36, pp. 539-545(1987).

[9] A. R. Abdelaziz, " A fuzzy-based power system reliability", Electric power systems reserch, Vol. 50, pp. 1-5 (1999).

[10] R. K. Wood, "Factoring algorithm for computing K-terminal network reliability", IEEE Transactions on Reliability, Vol. 35, pp. 269-278 (1986).

[11] S. P. Jain, and K. Gopal, "An efficient algorithm for computing global reliability of a network", IEEE Transactions on Reliability, Vol. 37, pp. 488-492 (1988).

[12] N. Deo, "Graph theory with applications to engineering and computer science", Prentice-Hall, Inc., Chapter 4 (1974).

[13] S. Soh, and S. Rai, "Experimental results on processing of path/cut terms in sum of disjoint products technique", IEEE Transactions on Reliability, Vol. 42, pp. 24-33 (1993).

[14] A. R. Abdelaziz, "Calculating the frequency of power system failure including common-cause failures", Electric Machines and power systems Journal, Vol. 24, pp. 533-540 (1996).

[15] M. Yester, "Using Turbo Pascal. 6", 2 edition, QUE Corporation, p. 87 (1991).

Received September 1, 2000
Accepted August 13, 2001