Iranian Journal of Science & Technology, Transaction B, Engineering, Vol. 30, No. B6, PP 749-761
Printed in The Islamic Republic of Iran, 2006

RECURSIVE CONTRACTION ALGORITHM: A NOVEL AND EFFICIENT GRAPH TRAVERSAL METHOD FOR SCANNING ALL MINIMAL CUT SETS*

A. R. SHARAFAT** AND O. R. MA'ROUZI

Dept. of Electrical and Computer Engineering, Tarbiat Modarres University, Tehran, I. R. of Iran
Email: Sharafat@isc.iranet.net

Abstract– We propose a novel algorithm called RCA_MC, in which we use the breadth first search method (BFS) in conjunction with edge contraction and connectivity properties of a given undirected graph to enumerate and scan all its minimal edge cutsets. It is known that the problem of enumerating all minimal edge cutsets of a given graph is #P-complete. In addition, we introduce the concepts of pivot vertex and absorbable clusters, and use them to develop our enhanced recursive contraction for scanning all mimimal edge cutsets, called ERCA_MC, of a given graph. Simulation results provide empirical evidence that the complexity of the ERCA_MC algorithm is linear per cutset.

Keywords– Breadth first search (BFS), cutset scanning, edge contraction, minimal edge cutset, #P-complete

1. INTRODUCTION

Scanning all minimal edge cutsets is an important issue in many applications [1], such as evaluating the reliability of networks [2-4], calculating the flow through a network [5], designing optimized networks [6], and analyzing stochastic networks [7]. A closely related problem is finding all (s,t)-cuts of a graph in which (s,t) is a vertex pair. This problem is equivalent to finding all minimal cutsets that separate vertices in a vertex pair (s,t) from one another in a given graph [8]. It is also extensively used for calculating the 2-terminal reliability of networks [9, 10].

Provan and Ball [11] proved that the problem of enumerating all minimal edge cutsets of a given graph is #P-complete. The basic conventional approach for scanning all edge cutsets of a given graph uses state-space enumeration [12, 13]. Although it is conceptually simple, the state-space approach is impractical because the size of the state space grows exponentially with the number of nodes. Certain improvements to this approach focus directly on topological properties of a given graph to substantially reduce its state-space size. Using this concept, Tsukiyama et al. [14] proposed an algorithm for finding the (s,t)-cuts of a given graph in linear time per cutset. Their recursive algorithm enumerates all s-t separating subsets of vertices which induce connected subgraphs by considering only certain extensions (1-point extensions) for all possible s-t separating subsets. An efficient method for enumerating all (s,t)-cuts in directed graphs is the paradigm of Provan and Shier [8].

A fast random approach for enumerating all suboptimal edge cutsets within a multiplicative factor k of the minimal cutset was proposed by Karger [15]. This algorithm is based on iterative contraction of edges in a graph. It can be extended to obtain, with a high probability of success, a list of all edge cutsets of a graph, while the relative error can be reduced arbitrarily by increasing the number of trials in the algorithm. The complexity of the Karger algorithm is also $O(n^{2k})$, which is exponential with respect to the maximum rank of all edge cutsets in a given graph.

In this paper, we modify the method proposed by Tsukiyama et al. in [14] by using the concept of iterative contraction [15] and BFS ordering of vertices to develop a novel *recursive contraction algorithm*

*Received by the editors June 12, 2004; final revised form September 19, 2006.
**Corresponding author

(RCA_MC) for scanning (enumerating and visiting) all minimal edge cutsets of a given graph. In addition, we introduce the concepts of a *pivot vertex* and *absorbable and inabsorbable clusters*, and use them to develop the *enhanced recursive contraction algorithm* (ERCA_MC).

The RCA_MC algorithm is a modification of Tsukiyama's algorithm in [14]. Both algorithms are recursive methods that examine the connectivity of subgraphs obtained by deleting edge cutsets. Besides, they both exhaustively scan all connected subgraphs. However, they are different in their scanning methods; where in Tsukiyama's algorithm the notion of 1-point extensions are used for this purpose, as compared to our proposed scheme which uses BFS ordering of vertices adjacent to a contracted node. This difference enables us to develop the ERCA_MC algorithm, which skips over those subgraphs that do not contribute to the list of minimal cutsets. As finding cutsets is a recursive process, significant reductions in the processing time can be achieved. In cases where no subgraph can be skipped, both the RCA_MC and the ERCA_MC algorithms have the same number of recursions. We use simulations to provide empirical evidence that the complexity of the ERCA_MC algorithm is linear per edge cutset.

This paper is organized as follows. In Section 2 we present our notations and introduce some basic concepts. In Section 3 we develop the recursive contraction algorithm and its enhanced version for finding all minimal cutsets of a given undirected graph. In Section 4 we present the results of applying the proposed algorithm to different sample graphs and show that the complexity of our algorithm is linear per cutset. Finally, in Section 5 we present the conclusion and a summary of results.

2. PRELIMINARIES

An undirected graph $G=(V,E)$ consists of a set V of vertices and a set E of edges whose elements are unordered pairs of vertices. The edge $e=(u,v)\in E$ is said to be incident with vertices u and v, where u and v are the end points of e. These two vertices are called adjacent. The set of vertices adjacent to v is denoted by $\Gamma(v)$, and the degree of v is the number of vertices adjacent to v and is denoted by $|\Gamma(v)|$. Throughout, we will reserve n for $|V|$ and m for $|E|$. In a graph $G=(V,E)$, a partition (X,X') is defined as two proper disjoint subsets of V. The complement of $X \subseteq V$ is denoted by $X' = V-X$. The open neighborhood of X is defined as $\Gamma(X)=\{v\in X' \mid (u,v)\in E \text{ for some } u\in X\}$. The induced subgraph $G[X]$ is a graph $H = (X,F)$, where $F=\{(u,v)\in E \mid u,v\in X\}$.

An alternating sequence of distinct adjacent vertices and their incident edges, i.e., v_0, (v_0, v_1), v_1, ... , v_{k-1}, (v_{k-1}, v_k), v_k is called a u-v path when $v_0= u$ and $v_k = v$. If a u-v path exists between all vertices u and v in a given graph G, then G is connected. Otherwise, G decomposes into a number of connected subgraphs referred to as components of G. A graph $G-\{u\}$, obtained by deleting a vertex u, is defined as a subgraph of G induced by $V-\{u\}$, i.e. $G-\{u\} = G[V-\{u\}]$.

DEFINITION 1. *Cutset:* For a given graph $G= (V,E)$, a subset of edges $C \subset E$ is a minimal cutset if and only if deleting all edges in C would divide G into two connected components and no subset of C is a cutset. An isolated vertex is considered as a component. A minimal cutset divides the vertices of G into two disjoint subsets X and X', each of which induce a connected subgraph. We denote a cutset by $\langle X,X'\rangle$. For convenience we say cutset instead of minimal edge cutset.

DEFINITION 2. *Cut Vertex:* For a connected graph G, a vertex v is a cut vertex if the graph $G-\{v\} = G[V-\{v\}]$ is not connected.

DEFINITION 3. *Biconnected and Separable Graphs:* A biconnected graph does not have any cut vertex, and a separable graph has at least one cut vertex. The maximal induced subgraphs of a separable graph G which are not separable are called the blocks of G.

OBSERVATION 1. Each cutset of a block is a cutset of a separable graph. So the set of all cutsets of a separable graph is the union of all cutsets of its blocks. Therefore, without any loss of generality we assume that all graphs to which we apply our proposed algorithm are biconnected.

DEFINITION 4. *Edge Contraction:* A graph denoted by G/uv is made by the contraction of an edge uv in Graph G in the following manner. Delete vertices u and v in G and replace them with a new contracted vertex g. Then remove all edges incident to both u and v (i.e., uv). For each edge incident to any one of the vertices u or v (uw or vw), there is an incident edge between g and the other vertex of the incident edge (i.e., gw) in G/uv. We extend this definition for a proper vertex set $F \subset V$ by sequentially applying contraction to all edges of $G[F]$ (the order of contractions is irrelevant). The resulting graph is denoted by G/F.

OBSERVATION 2. For any subset $F \subset V$, if the induced subgraph $G[F]$ is a connected subgraph, then all vertices of F in the contracted graph G/F are contracted to a single contracted vertex g. This is a direct result of Definition 4.

3. CUTSET SCANNING ALGORITHMS

Given an undirected graph $G(V,E)$, we develop an algorithm for scanning the set of all cutsets of G as $\Gamma_G = \{ \langle S,T \rangle \mid S \subset V, T=V-S, G[S] \in \Omega, G[T] \in \Omega \}$, where Ω is the set of all connected graphs.

a) Simple partitioning of vertices

The Simple Partitioning Algorithm (SPA) divides $V(G)$ into two proper disjoint subsets, C_a and C_b. According to Definition 1, if subgraphs $G[C_a]$ and $G[C_b]$ are connected, then partition $\langle S = C_a, T = C_b \rangle$ is a cutset. In [12], a simple method, called SPA, is proposed for finding all cutsets. The SPA lists all possible combinations of vertices that produce two proper disjoint subsets C_a and C_b, and then select those whose induced subgraphs $G[C_a]$ and $G[C_b]$ are connected [12],[13]. The simple partitioning algorithm is shown in Fig. 1.

```
Algorithm SPA:
simple_partitioning_algorithm
{
 input: 1) graph G ;
 output: 1) list of all cutsets S of G ;
 Initialization: S=∅, partition list F=∅ ;
  List all possible (p1, p2) partitions of  V(G) in F;
  For every partition p1 and p2 in F do
 {
    find induced subgraphs H1=G[ p1]  and H2= G[ p2]  over G;
    if H1 and H2 both are connected subgraphs
  {
     add partitions  p1 and p2 to cutset list S;
  }
 }
}
```

Fig. 1. The simple partitioning algorithm (SPA) for scanning all minimal cutsets of a given graph

b) Recursive contraction algorithm

LEMMA 1. A partition $C=\langle S=\{v\}, T = V-\{v\}\rangle$ is a minimal cutset if and only if v is not a cut vertex of G.

PROOF: Suppose that v is not a cut vertex of G. According to Definition 2, $H = G[V-\{v\}]$ is a connected subgraph. The connected subgraphs $F = (\{v\},\varnothing)$ and H are two components of G connected to each other via incident edges in C. Therefore, according to Definition 1 the set of edges $C = \langle S=\{v\}, T = V-\{v\}\rangle$ is a minimal cutset for G. Conversely, let $C = \langle S=\{v\}, T = V-\{v\}\rangle$ be a minimal cutset of G. The subgraph induced by $T_G = V(G)-\{v\}$ must be connected. Therefore, according to Definition 2, v is not a cut vertex of G. □

OBSERVATION 3. A subgraph obtained by deleting two adjacent vertices $u,v \in V$ of a graph G is equivalent to a subgraph obtained by deleting the contracted vertex g from G/uv, i.e., $(G/uv)-\{g\}=G-\{u,v\}$. This is a direct result of Definition 4.

LEMMA 2. For two adjacent vertices u and v in a graph G, the partition $\langle S=\{u,v\},T=V-\{u,v\}\rangle$ is a minimal cutset if and only if the contracted vertex g in G/uv is not a cut vertex in G/uv.

PROOF: Let $\langle S=\{u,v\},T=V-\{u,v\}\rangle$ be a minimal cutset for G. According to Definition 1, $G-\{u,v\}=G[V-\{u,v\}]$ must be a connected subgraph. From Observation 3 we know that $(G/uv)-\{g\}=G-\{u,v\}$. This means that the subgraph obtained by deleting the contracted node g is connected, or equally, g is not a cut vertex of G/uv. Conversely, assume that g is not a cut vertex of G/uv. This means that $(G/uv)-\{g\}$ is a connected subgraph, i.e., $G-\{u,v\}$ is a connected subgraph. We know that u and v are adjacent vertices, so the subgraph induced by them is connected and contains u, v, and the edge(s) uv. Thus from Definition 1, the partition $\langle S=\{u,v\},T=V(G)-\{u,v\}\rangle$ is a minimal cutset of G. □

OBSERVATION 4. For any subset of vertices $F \subset V$, if the induced subgraph $G[F]$ is a connected subgraph, then the induced subgraph $G[V-F]$ of a given graph G is equivalent to a subgraph obtained by deleting the contracted vertex g from G/F, i.e., $(G/F)-\{g\}=G[V-F]$. This is a direct result of Definition 4 and Observation 2.

LEMMA 3. For any subset of vertices $F \subset V$, the partition $\langle F,V-F\rangle$ is a minimal cutset of graph G if and only if (i) the subgraph induced by F is connected, and (ii) the contracted node g is not a cut vertex in the graph G/F.

PROOF: Let $\langle F,V-F\rangle$ be a minimal cutset for G. According to Definition 1, the induced subgraphs of $G[F]$ and $G[V-F]$ must be connected. From Observation 4 we know that $(G/F)-\{g\}=G[V-F]$. Thus $(G/F)-\{g\}$ is a connected subgraph. Therefore, according to Definition 2 the contracted vertex g is not a cut vertex in the graph G/F. Conversely, assume g is not a cut vertex in the graph G/F. So the subgraph obtained by deleting the contracted vertex g from G/F (which is the same as the induced subgraph $G[V-F]$) is a connected subgraph. From (i) we know that $G[F]$ is also a connected subgraph. Therefore, according to Definition 1 we conclude that $\langle F,V-F\rangle$ is a minimal cutset of G. □

OBSERVATION 5. For any subset of vertices $F \subset V$ that has more than one vertex, if the subgraph induced by F is connected, then there is at least one adjacent vertex for any vertex in F. This is a direct result of connectivity and path definitions in graphs.

LEMMA 4. Finding all connected subgraphs of G/uv that include the contracted vertex g is equal to finding all connected subgraphs of G that include u and v.

PROOF: Suppose we find the set of all connected subgraphs $S(g)$ that include the contracted vertex g of the graph G/uv, as well as the set of all connected subgraphs $S(u,v)$ that include the vertices u and v of a given graph G. For each connected subgraph $F \in S(g)$ there is a connected subgraph $H \in S(u,v)$ obtained by replacing g with u and v in F. Therefore finding $S(g)$ for the graph G/uv is equivalent to finding $S(u,v)$ for G. □

1. Scanning connected subgraphs. We use the results of Lemmas 1-4, and propose the Recursive Contraction Algorithm for scanning all Connected SubGraphs (RCA_CSG) of a given graph. In this algorithm, we produce a list of all connected subgraphs $S(v)$ in G that include the vertex v. In doing so, we use topological properties of a given graph G to reduce the number of partitions whose associated subgraphs must be checked for connectivity. We begin by choosing a vertex of G as the *seed vertex*. We denote all proper subsets of V that include the seed vertex v and induce the connected subgraphs as $S(v)$,

i.e., $S(v)=\{ F\subset V \mid v\in F,\ G[F]\in \Omega \}$. Suppose the neighborhood of v is $\Gamma(v)=\{u_1,\ldots,u_k\}$. Using Observation 5, we conclude that every proper subset $F\subset V$ in $S(v)$ except $\{v\}$ must include an adjacent vertex u_i, i.e.,

$$S(v)= \{v\}\cup S(v,u_1)\cup\ldots\cup S(v,u_k) \tag{1}$$

where $S(v,u_i)=\{F\subset V \mid v\in F,\ u_i \in F,\ G[F] \in \Omega\}$. But the sets $S(v,u_i)$, $i=1,\ldots,k$ are not disjoint, and therefore, some partitions will be visited several times. To explain this, suppose that the vertex $z\in \Gamma(v)$ is adjacent to $y\in \Gamma(v)$. Then $\{v, y, z\}$ belongs to both $S(v,z)$ and $S(v,y)$, and all subsets of vertices in $S(v)$ that include these three vertices belong to both $S(v,z)$ and $S(v,y)$ as well. We must exclude parts of $S(v,z)$ that were previously scanned in $S(v,y)$.

2. BFS ordering constraint. To prevent re-examination of recurring subsets, we use the inclusion-exclusion principle and rewrite (1) as

$$S(v) = \{v\} \cup S(v,u_1) \cup S(v,u_2,\overline{u}_1) \cup \cdots \cup S(v,u_k,\overline{u}_1,\cdots,\overline{u}_{k-1}) \tag{2}$$

where $S(v,u_j,\overline{u}_1,\cdots,\overline{u}_{j-1}) = \{F \subset V \mid v \in F, u_j \in F, u_i \notin F, i = 1,\cdots, j-1, G[F] \in \Omega\}$. In (2), $S(v,u_j,\overline{u}_1,\cdots,\overline{u}_{j-1})$ denotes all connected subgraphs which include v and u_j, but do not include u_1, u_2, …, and u_{j-1}. To find $S(v)$, we arrange the vertices in the order of their position in the BFS tree of the graph when the seed vertex v is considered as its root. In (2) we arrange the adjacent vertices $u_j \in \Gamma(v)$ in the same BFS order. According to Lemma 4, in order to find $S(v,u_1)$, we must find all connected subgraphs of G/vu_1 that include the contracted vertex g. Furthermore, to find $S(v,u_j,\overline{u}_1,\cdots,\overline{u}_{j-1})$ we must find all Connected Sub-Graphs (CSGs) of G-$\{u_i, i=1,\ldots, j\text{-}1\}/vu_j$ that include the contracted vertex g. It means that we can recursively find all CSGs of a connected graph CG by finding all CSGs of contracted graphs. In order to preserve the BFS ordering of vertices in all consecutive subproblems, we derive the BFS order of each contracted CSGs G/vu_i from the previous BFS ordering of G. The *Recursive Contraction Algorithm for scanning all Connected SubGraphs* (RCA_CSG) is now developed as shown in Fig. 2. As shown in the first line of the recursion loop in Fig. 2, in order to exclude the vertices that have already contributed to the recursion in (1), in each recursion, we use a new reduced graph TG by deleting all vertices that have a BFS order lower than the current BFS order of the contracted vertex g (the current BFS order of the last vertex contracted to vertex g), from the contracted graph CG.

THEOREM 1. Considering the BFS ordering constraint in the RCA_CSG algorithm, all connected subgraphs of G that include the seed vertex v are scanned once.

PROOF: Consider a connected subgraph induced by a subset of vertices $F\subset V$, where $v\in F$. There is a unique BFS ordering for all vertices in F, denoted by v-u_2-…-u_k where $k<n$. From Observation 5 we know that there is at least one adjacent vertex in F for each vertex in F. The RCA_CSG algorithm scans all adjacent vertices in the vertex list F to construct all possible connected subgraphs in the next recursion level. This ensures that in the recursion level j-1, a vertex u_j that is adjacent to one of the vertices v-u_2-…-u_{j-1} is added to the list of vertices in F, and after k level of recursion, the desired connected subgraph is scanned. Since the BFS order of vertices is unique, a connected subgraph cannot be scanned more than once without contradicting the BFS ordering constraint. □

3. The recursive contraction algorithm for scanning all minimal cutsets (RCA_MC). The RCA_CSG algorithm applies constraint (i) in Lemma 3 and produces a list of all connected subgraphs. To develop the *Recursive Contraction Algorithm for scanning all Minimal Cutsets* (RCA_MC) of a graph G we must also apply the second constraint in Lemma 3 and check the connectivity of the induced subgraph $G[V\text{-}F]$.The RCA_MC algorithm shown in Fig. 3 finds all cutsets in a given graph G by generating all possible connected subgraphs induced by $F\subset V$ in G and checking that the contracted node g is not a cut vertex in G/F.

```
Algorithm RCA_CSG:
all_connected_subgraphs_recursive_algorithm
{
 inputs: 1) graph G,   2) seed vertex v;
 output: 1) list S(v) of all connected subgraphs of G which contain the seed vertex v;
 Initialization: S(v)=∅, vertex list F={v}, vertex index list ORDER={1,2,…,|v|},
                 vertex index list BFS_ORDER=∅, graph CG=G;

BFS_ORDER= BFS ordering tree of the graph G with seed vertex  v as its root and
           ORDER as the vertices selection order for the same level;

  recursive subroutine: find_all_connected_subgraphs (CG , v, F, BFS_ORDER)
  {
   Local Variables: graph TG, vertex v', vertex list H, vertex index list NXT_BFS_ORDER,
                    Vertex list Γ(v);

   add the vertex list F to S(v);
   find the neighborhood set Γ(v) for vertex v;
   if  Γ(v) is empty
   {
    return;
   }
   else
   {
    recursion loop: for all vertices u of Γ(v) do
     {
      TG= CG –{ all vertices of Γ(v)  whose BFS order is smaller than u } ;
      NXT_BFS_ORDER= BFS ordering tree of the graph TG with seed vertex u as its
             root and BFS_ORDER as the vertices selection order for the same level;
      copy the vertex list F to the vertex list H;
      contract the edge (v, u) and find TG/uv;
      add vertex u to the vertex list H;
      set the seed vertex v' to the new contracted vertex g of TG/uv;
      find_all_connected_subgraphs (TG/uv, v', H, NXT_BFS_ORDER);
     }
   }
  }
 }
```

Fig. 2. The RCA_CSG algorithm for scanning all connected subgraphs of a given graph *G*

THEOREM 2. Considering the BFS ordering constraint in the RCA_MC algorithm, all cutsets of a given graph *G* are scanned once.

PROOF: Each cutset divides a given graph *G* into two unique connected subgraphs, one of which includes the seed vertex *v*. In order to specify a cutset, it is sufficient to find its corresponding connected subgraph that includes the seed vertex *v*. We know from Theorem 1 that the RCA_CSG algorithm scans only once all connected subgraphs that include the seed vertex *v*. Since the RCA_MC algorithm uses the same approach and examines all possible connected subgraphs, we conclude that this algorithm scans only once all cutsets of the graph *G*. □

c) Enhanced recursive contraction algorithm

Now, we develop the *Enhanced Recursive Contraction Algorithm to scan all Minimal Cutsets* (ERCA_MC) of a given graph. We will provide empirical evidence that the computational complexity of the ERCA_MC algorithm, similar to the complexity of the RCA_MC algorithm, is linear per cutset, but the former is less than the latter.

DEFINITION 5. *A cluster and its pivot:* According to Definition 2, when we delete a cut vertex from a graph *G*, more than one distinct component remains in the resulting graphs. Each one of such components is called a cluster, and the deleted cut vertex is called the pivot vertex of the respective clusters.

DEFINITION 6. *The absorbable cluster:* A cluster in which all its vertices have a BFS order greater than the BFS order of its pivot (the BFS order of the last vertex contracted in the pivot) is an absorbable cluster. Otherwise it is an inabsorbable cluster (Fig. 4). To find absorbable and inabsorbable clusters we must consider the BFS order of the original graph G.

```
Algorithm RCA_MC:
all_cutset_recursive_algorithm
{
  inputs: 1) graph G,   2) seed vertex v;
  output: 1) list of all cutsets S(v) of G ;
    Initialization: S(v)=∅, vertex list F={v},  vertex index list ORDER={1,2,…,|v|},
                    vertex index list BFS_ORDER=∅, graph CG=G;

  BFS_ORDER= BFS ordering tree of the graph G with seed vertex  v as its root and
                 ORDER as the vertices selection order for the same level;

  recursive subroutine: find_all_cutsets (CG,v, F, BFS_ORDER)
  {
    Local Variables: graph TG, vertex v', vertex list H, vertex index list NXT_BFS_ORDER,
                     vertex list Γ(v);

    if vertex v is not a cut vertex of the graph G/F
    {
      add the vertex list F to S(v);
    }
    find the neighborhood set Γ(v) for vertex v in CG;
    if  Γ(v) is empty
    {
     return;
    }
    else
    {
      recursion loop: for all vertices u of Γ(v) do
      {
       TG= CG –{all vertices of Γ(v) whose BFS order is smaller than u};
       NXT_BFS_ORDER= BFS ordering tree of the graph TG with seed vertex u as its
                      root and BFS_ORDER as the vertices selection order for the same level;
       copy the vertex list F to the vertex list H;
       contract the edge (v, u) and find TG/uv;
       add vertex u to the vertex list H;
       set the seed vertex v' to the new contracted vertex g of TG/uv;
       find_all_cutsets(TG/uv, v', H, NXT_BFS_ORDER);
      }
    }
  }
}
```

Fig. 3. The RCA_MC Algorithm for scanning all minimal cutsets of a given graph *G*

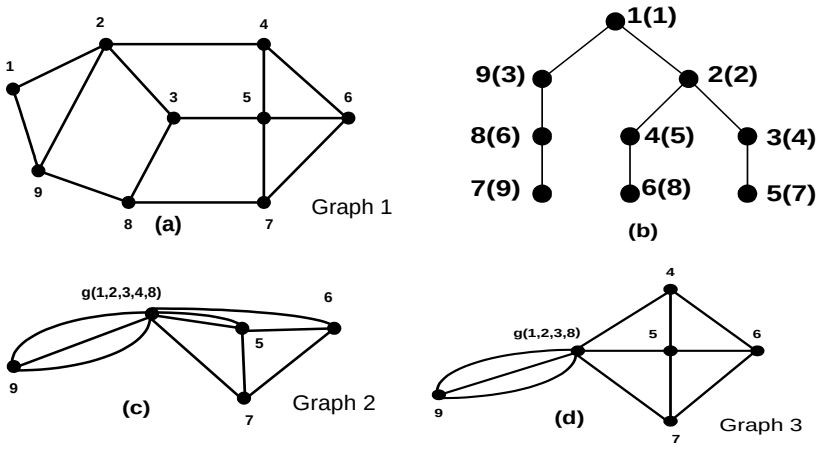

Fig. 4. a) A sample graph, b) The breadth first search ordering *m* of node *n* is shown in the form of *n(m)*, c) Graph 2 is generated by contracting nodes 1,2,3,4,8. The BFS order of the contracted node g(1,2,3,4,8) is 6 (BFS order of the last node, i.e., node 8). The cluster <9> is inabsorbable, but the cluster <5,6,7> is absorbable, d) Graph 3 is generated by contracting nodes 1,2,3,8. The BFS order of the contracted node is 6. Both clusters <9> and <4,5,6,7> are inabsorbable, since the BFS order of node 4 is 5, which is lower than the BFS order of the pivot

THEOREM 3. In the RCA_MC algorithm, if the contraction vertex g in G/F is a pivot with more than one inabsorbable cluster, then there is no cutset $<X, X'>$ for which $F \subset X$, and that would be found by the algorithm by continuing with recursive calls, therefore, the remaining recursions are skipped.

PROOF: The contraction of vertices in the RCA_MC algorithm eventually results in complete absorption of all absorbable clusters in the contracted node. When only one cluster remains, the contracted vertex is transformed into a non-cut vertex of the graph. If there is more than one inabsorbable cluster, the contraction of vertices in the RCA_MC algorithm does not transform the contracted vertex into a non-cut vertex of the graph. This is due to the fact that there is at least one vertex in each inabsorbable cluster whose BFS order is lower than the BFS order of the last vertex contracted in the pivot vertex. According to the BFS ordering constraint, this vertex cannot be selected for contraction, resulting in at least two distinct clusters in the graph. Thus, the pivot vertex always remains a cut vertex of the graph. □

OBSERVATION 6. A pivot vertex with one inabsorbable cluster cannot become a non-cut vertex unless the vertex with the highest current BFS order among all vertices of absorbable clusters is contracted in the pivot vertex.

The ERCA_MC Algorithm. As shown in Fig. 5, we use Theorem 3 to modify the RCA_MC algorithm with a view to preventing further contractions when the contracted vertex is a pivot vertex with more than one inabsorbable cluster. When the contracted vertex is a pivot vertex with one inabsorbable cluster, we use Observation 6; this algorithm goes through a cut-through transient phase until the contracted node is transformed into a non-cut vertex. At this time, the algorithm returns to a normal recursive form. During the cut-through transient phase, the contraction process is carried out as before, but the first part of the recursion loop (lines 6 to 28 in Fig. 5), which is the most time consuming part of the algorithm, and which checks if the contracted node is a non-cut vertex, is not performed. This is due to the fact that according to Observation 6, the contracted node is a cut vertex for the resulting contracted graph until the highest current BFS order vertex of the absorbable clusters is contracted to the contracted node. Therefore, the end of the transient phase is detected by the contraction of the highest current BFS order vertex among all vertices of the absorbable clusters. The complexity of the ERCA_MC algorithm is less than that of the RCA_MC algorithm, because the search space of the ERCA_MC algorithm does not include several groups of connected subgraphs that cannot generate any new cutsets. The number of these groups can increase exponentially with the size of the graph.

4. PERFORMANCE COMPARISONS

In this Section, we present the results of implementing the SPA, the RCA_MC, and the ERCA_MC algorithms. The SPA algorithm is equivalent to the state-space enumeration method [12], [13] and the RCA_MC algorithm is a modified version of the Tsukiyama [14] algorithm for scanning all minimal cutsets instead of all (s,t)-cuts of an undirected graph.

We compare the number of iterations needed to scan all minimal cutsets in each sample graph shown in Fig. 6 for the above algorithms. The number of cutsets for Graphs 1, 2, 3 and 4 are 66, 2232, 17518, and 28448, respectively.

All three algorithms give the correct values of cutsets irrespective of the seed vertex. We calculate the number of recursions as a measure of complexity for different seed vertices. The complexities of the SPA, the RCA_MC, and the ERCA_MC algorithms are compared in Fig. 7. It is evident that the complexity of the ERCA_MC algorithm is substantially lower compared to those of the SPA and the RCA_MC

algorithms for all seed vertices. The number of iterations in the RCA_MC and the ERCA_MC algorithms for larger graphs are several orders of magnitude less than that of the SPA Algorithm. In the RCA_MC algorithm, the minimum number of recursions is obtained by choosing the lowest degree node as the seed vertex. This is not the case for the ERCA_MC algorithm, for which the variance of the number of recursions as a function of the seed vertex is low compared to the RCA_MC algorithm.

```
Algorithm ERCA_MC:
Enhanced_all_cutset_recursive_algorithm
{
 inputs: 1) graph G,   2) seed vertex v;
 output: 1) list of all cutsets S(v) of G ;
 Initialization: S(v)=∅, vertex list F={v}, vertex index list ORDER={1,2,…,|v|},
         vertex index list BFS_ORDER=∅, graph CG=G, dummy_flag=FALSE;

 BFS_ORDER= BFS ordering tree of the graph G with seed vertex  v as its root and
                 ORDER as the vertices selection order for the same level;

 recursive subroutine: find_all_cutsets (CG,v, F, BFS_ORDER)
  {
   Local Variables: graph TG, vertex v', vertex list H, vertex list Γ(v),
                    vertex index list NXT_BFS_ORDER;
   If (dummy_flag is FALSE)
   {
     if vertex v is not a cut vertex of the graph G/F
     {
       add the vertex list F to S(v);
     }
     else
     {
       find the set of clusters clst for pivot vertex v in G/F;
        if there is more than one inabsorbable cluster in clst
        {
          return;
        }
        else
        {
           if there is one inabsorbable cluster in clst
           {
             highest_BFS_order_vertex = find the highest current BFS order vertex
                                        among vertices of absorbable clusters;
             dummy_flag=TRUE;
           }
        }
     }
   }
   find the neighborhood set Γ(v) for vertex v in CG;
    if  Γ(v) is empty
    {
      return;
    }
    else
   {
     recursion loop: for all vertices u of Γ(v) do
     {
      TG= CG –{ all vertices of Γ(v) whose BFS order is smaller  than u};
      NXT_BFS_ORDER= BFS ordering tree of the graph TG with seed vertex u as its
                     root and BFS_ORDER as the vertices selection order for the same level;
      copy the vertex list F to the vertex list H;
      contract the edge (v, u) and find TG/uv;
      add vertex u to the vertex list H;
      set the seed vertex v' to the new contracted vertex g of TG/uv;
      if (dummy_flag is TRUE)
       {
         If (u= highest_BFS_order_vertex) dummy_flag=FALSE;
       }
      find_all_cutsets (TG/uv, v', H, NXT_BFS_ORDER);
     }
   }
  }
}
```

Fig. 5. The ERCA_MC Algorithm for scanning all minimal cutsets of a given graph *G*

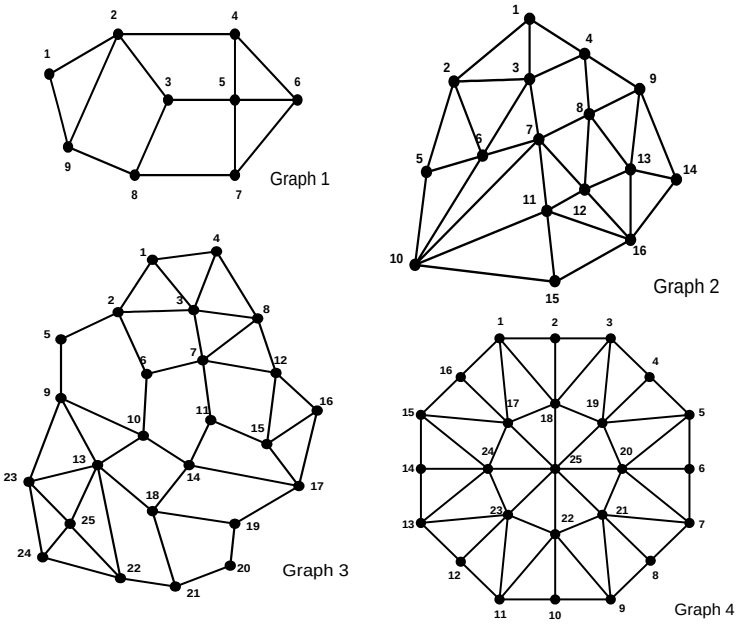

Fig. 6. Sample graphs for comparing the performances of the SPA, the RCA_MC and the ERCA_MC algorithms

In order to show that the number of recursions in the RCA_MC and ERCA_MC algorithms is proportional to the number of cutsets in a given graph, we generate 10 random graphs for each network size and average-node-degree, and apply the RCA_MC and ERCA_MC algorithms to them. Fig. 8 shows the results for a graph with 20 nodes and different average-node-degrees from 2 to 10. In Fig. 9 we show the results for random graphs with an average node degree equal to 3 and different network sizes from 10 to 36 nodes.

For each sample point in Figs. 8 and 9, we apply the RCA_MC and ERCA_MC algorithms ten times for different random biconnected graphs and calculate the mean values for the number of recursions as well as for the number of cutsets. It is evident from Figs. 8 and 9 that for the RCA_MC and the ERCA_MC algorithms, the number of recursions follows the number of cutsets. We conclude that the complexities of both the RCA_MC and the ERCA_MC algorithms are linear per cutset irrespective of the size of the graph, but the complexity of the latter is less than that of the former. This is because the latter avoids scanning some classes of disconnected subgraphs due to utilizing the notion of inabsorbable clusters. If the node-degree is increased, the difference in the performances of the RCA_MC and the ERCA_MC algorithms become smaller and for larger node degrees, both asymptotically approach the performance of the SPA algorithm.

Figures 10 and 11 show the ratio of the average number of cutsets to the average number of iterations, which we call the *efficiency factor*. As this value gets closer to 1, the efficiency of the algorithm is improved. For a ring network in which the average-node-degree is 2, all iterations correspond to a cutset, so the efficiency factor of the RCA_MC and the ERCA_MC algorithms is equal to 1. However, in mesh networks, in which the average node degree is greater than 2 and less than the number of nodes in the graph, the efficiency factor increases gradually towards 1 as the average-node-degree of the graph increases. In Figs. 10 and 11, it is shown that the ERCA_MC algorithm is more efficient than the RCA_MC algorithm for all node degrees and network sizes. As shown in Fig. 11, the efficiency factor of the ERCA_MC algorithm is relatively independent of graph size, and is about 0.56 when the average node degree is 3, but the efficiency of the RCA_MC algorithm decreases for larger network sizes. The difference in the performances of the RCA_MC and the ERCA_MC algorithms is more significant for

large bounded degree mesh networks, which represent the majority of actual telecommunications networks.

Finally, we compare the time complexity of the ERCA_MC and the Tsukiyama algorithms in Table 1. Enumeration of checked partitions is a possible measure for time complexity. The execution times of the ERCA_MC and the Tsukiyama algorithms for 6 different size mesh networks and 5 complete networks [16], K6-K10 on a personal computer with a full cache and a 2 GHz Pentium IV CPU and 256 MB RAM is shown in the last two columns of Table 1. It is evident that the time complexity of the ERCA_MC algorithm is less than that of the Tsukiyama algorithm, except for the small network G1. For large networks, the time complexity of the ERCA_MC algorithm is less than that of the Tsukiyama algorithm by as much as 20 percent and for complete networks by up to 50 percent.

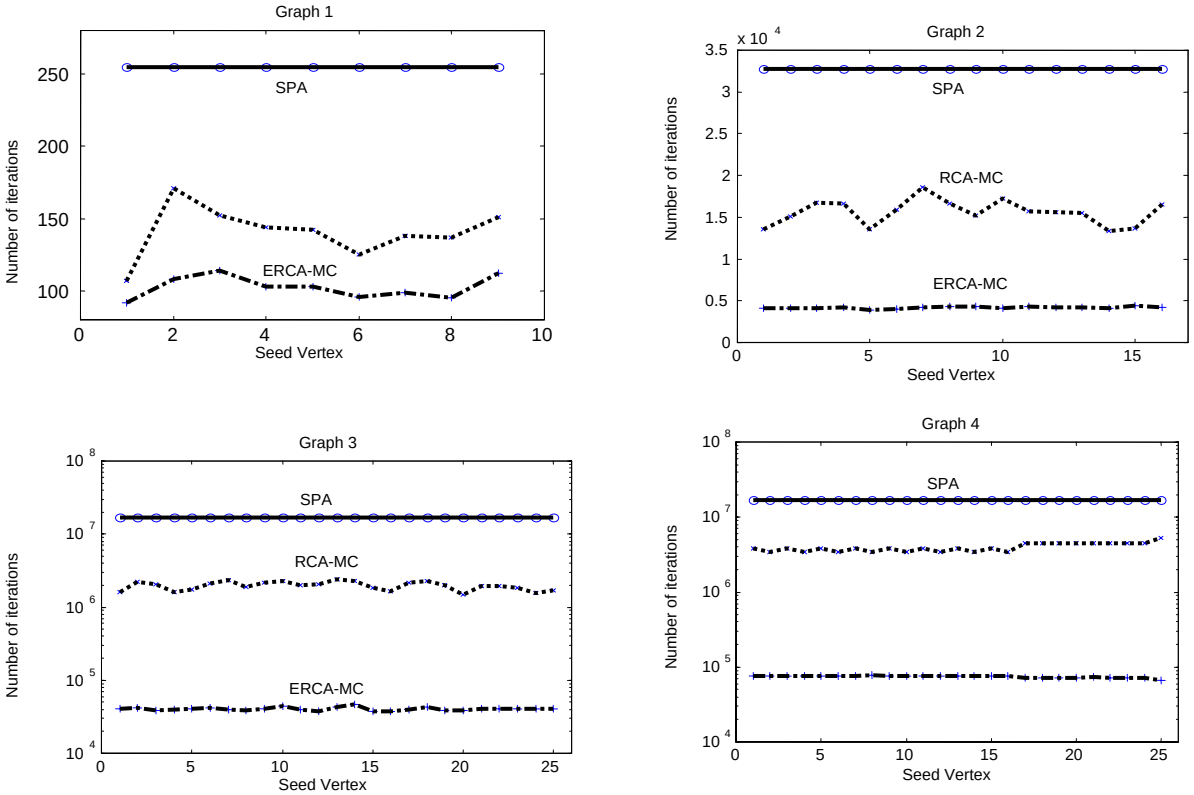

Fig. 7. Comparison of complexities of the SPA, the RCA_MC and the ERCA_MC for sample graphs in Fig. 6

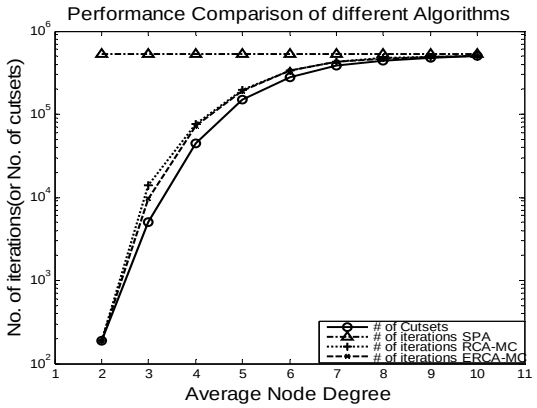

Fig. 8. The number of iterations and the number of cutsets *vs*. average-node-degree for a network of size 20

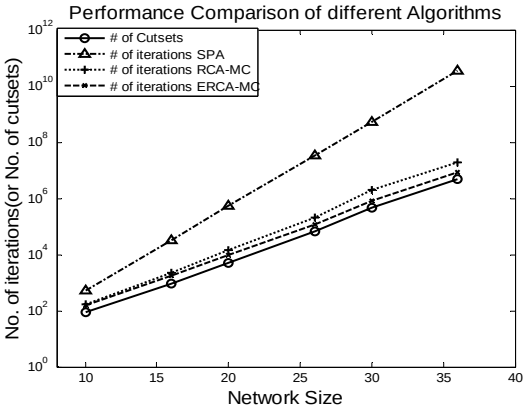

Fig. 9. The number of iterations and the number of cutsets *vs*. network size for different graphs each with average node degree 3

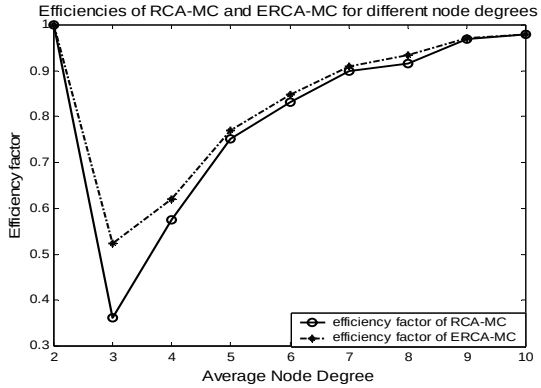

Fig. 10. The efficiency factor for the network of Fig. 8 *vs*. the average node degree

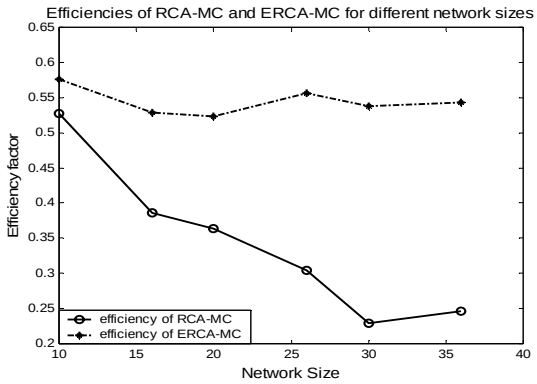

Fig. 11. The efficiency factor for the network of Fig. 9 *vs*. network sizes

Table 1. Time complexities of the ERCA_MC and the Tsukiyama algorithms

Network	Number of vertices	Number of edges	Number of Cutsets	ERCA Iterations	TSU Iterations	ERCA Exec. Time	TSU Exec. Time
G1	9	14	66	95	87	0.43 ms	0.36 ms
G2	16	35	2232	4155	7371	28 ms	35 ms
G3	25	46	17518	45120	78120	535 ms	550 ms
G4	25	56	28448	80747	146590	1170 ms	1450 ms
NSFNET[17]	14	21	799	1469	1658	8 ms	8 ms
COST239[17]	18	39	6049	10783	22723	80 ms	110 ms
K6	6	15	31	31	31	0.09 ms	0.11 ms
K7	7	21	63	63	63	0.2 ms	.29 ms
K8	8	28	127	127	127	0.42 ms	0.68 ms
K9	9	36	255	255	255	0.83 ms	1.5 ms
K10	10	45	511	511	511	1.78 ms	3.52 ms

5. CONCLUSION

In this paper we presented a novel and efficient recursive algorithm for scanning all minimal cutsets of a given undirected graph. This algorithm is based on the consecutive contraction of edges in the graph and uses the concept of cut vertex for checking the connectivity of the induced subgraphs that are generated from partitioning the graph. We used the BFS ordering of vertices to prevent visiting a possible state more than once. We also used the concept of an absorbable cluster for a cut vertex (pivot) in a given graph to substantially reduce the complexity of the algorithm.

We applied our proposed algorithms to different graphs and provided empirical evidence that scanning all cutsets is done in linear time per cutset. We also applied the RCA_MC and the ERCA_MC algorithms to biconnected randomly generated graphs with different sizes and average-node-degrees, and showed that the efficiency of the ERCA_MC algorithm improves significantly as compared to the RCA_MC algorithm for large bounded degree mesh networks. We also showed that the efficiency of the ERCA_MC algorithm does not change considerably with network size when the average node degree is kept constant.

Finally, simulation results indicate that the ERCA_MC algorithm performs better than the Tsukiyama algorithm.

Acknowledgement- The Authors wish to thank the anonymous reviewers for their comments and suggestions.

REFERENCES

1. Picard, J. C. & Queyranne, M. (1980). On the structure of all minimum cuts in a network and applications. *Math. Programming Stud. 13,* 8-16.
2. Colbourn, C. J. (1987). *The combinatorics of network reliability*. International Series of Monographs on Computer Science, 4, Oxford University Press.
3. Fard, N. S. & Lee, T. H. (1999). Cutset enumeration of network systems with link and node failure. *Reliability Engineering and System Safety, 65,* 141-146.
4. Rai, S. (1982). A cutset approach to reliability evaluation in communication networks. *IEEE Trans. on Reliability, 31,* 428-431.
5. Ahuja, R. R., Magnanti, T. L. & Orlin, J. B. (1993). *Network Flows- Theory, Algorithms, and Applications.* Prentice-Hall International.
6. Picard, J. C. & Queyranne, M. (1982). Selected applications of minimum cuts in networks. *INFOR. 20,* 394-422.
7. Sigal, C. E., Pritsker, A. A. B. & Solberg, J. J. (1979). The use of cutsets in Monte Carlo analysis of stochastic networks. *Math. Comput. Simulation, 21,* 376-384.
8. Provan, J. S. & Shier, D. R. (1996). A paradigm for listing (s,t)-cuts in graphs. *Algorithmica, 15,* 351-372.
9. Ball, M. O. (ed.) (1995). *Network Models* (Chapter on Network Reliability). Elsevier, Amsterdam.
10. Shier, D. R. (1988). Algebraic aspects of computing network reliability. In *Applications of Discrete Mathematics,* R. D. Ringeisen and F. S. Roberts (eds.), SIAM, Philadelphia, 135-147.
11. Provan, J. S. & Ball, M. O. (1983). The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal of Computing, 12,* 777-788.
12. Ahmad, S. H. (1988). Simple enumeration of minimal cutsets of acyclic directed graph. *IEEE Trans. on Reliability, 37*(5), 484-487.
13. Hongfen, Z. (1995). A simple enumeration of all minimal cutsets of an all-terminal graph. *The Journal of China Universities of Posts and Telecommunications, 2*(2).
14. Tsukiyama, S., Shirakava, I., Ozaki, H. & Ariyoshi, H. (1980). An algorithm to enumerate all cutsets of a graph in linear time per cutset. *Journal of the ACM, 27,* 619-632.
15. Karger, D. R. (2000). Minimum cuts in near-linear time. *Journal of the ACM, 47,* 46-76.
16. Weisstein, E. W. (Complete Graph). From *MathWorld*-A Wolfram Web Resource. http://mathworld.wolfram.com/CompleteGraph.html
17. Lee, K., Kang, K. C., Lee, T. & Park, S. (2002), An optimization approach to routing and wavelength assignment in wdm all-optical networks without wavelength conversion. *ETRI Journal, 24(*2), 131-141.