Łukasz PIĄTEK

Częstochowa University of Technology

A new algorithm for generating cuts set in undirected graphs representing electric systems

Abstract. *A new algorithm for finding cut set of a undirected graph is presented. The algorithm finds all cut in graph which can be used in the cut set method of reliability evaluation. The algorithm is based on division of graph vertices set into subset in which vertices have the same distance from the sink. An example of application of the algorithm for analysis of sample electric system is presented.*

Streszczenie. *Prezentowany jest nowy algorytm znajdowania przekrojów w grafach o krawędziach niezorientowanych. Algorytm znajduje zbiór przekrojów grafu, który może być użyty w metodzie analizy niezawodności. Algorytm bazuje na podziale zbioru wierzchołków grafu na podzbiory w których każdy wierzchołek leży w tej samej odległości od źródła. Pokazano przykład znajdowania przekrojów w grafie systemu elektrycznego.* (***Nowy algorytm znajdowania przekrojów w grafach o krawędziach nieskierowanych reprezentujących systemy elektryczne***)

Keywords: graph cut, graph, reliability
Słowa kluczowe: przekrój w grafie, graf, niezawodność.

Introduction

For evaluation of reliability of electric systems the method of cut sets is used [1,2]. In order to find all cuts in graph corresponding to considered electric system a new algorithm is introduced. The algorithm is based on division of graph vertices set into subset called layer. Each vertice v in layer have the same length of minimal path from source s to v. A combination of vertices in every layer is generated. The result of algorithm work is a set of all cuts in graph.

The simplest known algorithm for finding all cuts in graph consists in enumerating all graph edges combination. For each combination check for graph connectivity is performed.

The other algorithm [3,4] requires finding of all paths in graph in order to determine all cuts set in graph.

Cuts in undirected graph

Let $G=(V,E)$ be an graph with undirected edges. V is the graph vertices set and E is the graph edges set. There are two vertices in graph called the source s and the sink t. The cut of the graph is such a division of graph vertices set into two subsets A and B which satisfy the following conditions :

- subsets A and B are a division of graph vertices set

(1) $$A \bigcup B = V$$

- subsets A and B are disjoint

(2) $$A \bigcap B = \varnothing$$

- the source is in A and the sink is in B

(3) $$s \in A, t \in B$$

- A and B establish a subset C of graph edges $C \subseteq E$ such that every edge $e \in C$ connect two graph vertices i,j which $i \in A$ and $j \in B$

(4) $$e = \left\{ \{i, j\} : i \in A, j \in B \right\}$$

- for every subset C of graph edges which is a graph cut there is no cut C' which is a subset of C

(5) $$C \bigcup C' \neq C'$$

From formulae (1)-(5) a conclusion can be derived that in graph with undirected edges when the cut as a division of graph vertices is known, a cut as a subset of graph edges can be found.

Algorithm for finding all cuts in undirected graph

The algorithm is based on division of entire graph vertices set into subsets called layers. Every layer is a set of vertices which have the same distance to source vertice. The distance between vertice v and s is defined as the number of edges in the shortest path from s to v. Two layer are processed in each iteration, layer L_i and layer L_{i+1}. The layer L_{i+1} is one unit more away from the source than L_i. On the start of the algorithm the first layer contains only the source $L_0=\{s\}$. The following operations are performed during one iteration of the algorithm:

- with the use of vertices of layer L_i vertices of layer L_{i+1} are found where the adjacency list is used to find layer L_{i+1} ,
- all combinations k-elements (k=1, ... , $|L_{i+1}|$) of layer L_{i+1} vertices are generated,
- every combination found in recent step is combined with vertice division which have been already found for layer L_i. The new division of graph vertices set is checked if it satisfy condition (5) and is remembered as a division relaying to layer L_{i+1},
- all vertices in layer L_{i+1} are checked as visited,
- vertices of layer L_{i+1}. become the new layer L_i.

The algorithm finds cuts through edges of the graph. It is possible to use it for finding also cuts through vertices. For achieve this, a preprocessing of the graph structure is done before the start of the algorithm. Each vertice is replaced by two vertices connected by one new edge. When algorithms finds cut through this new edge, the cut is interpreted as a cut through vertice.

Pseudocode of the algorithm

The pseudocode of the algorithm is written in a notation similar to notation of C++ language [5,6].

As the data input of the algorithm a structure *VLIST* is established. The *VLIST* is an array containing information about the structure of the graph. For every i-vertice of the graph, *VLIST*[i] contains list of vertices connected to vertice i. In the start of the algorithm the source and the sink are chosen. The *VLIST* structure is the adjacency list of the graph.

For output data there is one structure called *ESETS*. This is an array of sets named *eset*. Each *eset* contains list of edges of one cut of the graph.

For auxiliary data five data structure and one type of data are defined. The type of data *vset* is a list of graph vertices and represents one division of graph vertices set.

Data structure *VSETS* contains list of *vset* found for each layer in graph. Data structure *CHECKED* contains information about vertices which have been already visited during the algorithm work. Data structure *COMBINATIONS* contains list of combination of vertices in layer. Data structures *LAYER_I* and *LAYER_I_PLUS* contains vertices of layer L_i and respectively L_{i+1} .

The main function is *FIND_CUTS*. Function *ADD_ESET* is responsible for creating cuts as set of vertices with the use of *vset*. Because there is no guarantee that all edges set added by function *ADD_SET* satisfy condition (5), function *CHECK_ESETS* checks it. In function *FIND_CUTS* there is one use of function *CREATE_COMBINATION* which implements algorithm for generating combinations [7]. The pseudocode of this function was not listed here.

```
function FIND_CUTS( V_LIST, s, t )
{
 VSETS[0]:= VSETS[0] ⋃ {s};
 CHECKED[s] =1;
 LAYER_I = LAYER_I ⋃ {s};
 integer layer_i_index =0
 while ( LAYER_I≠Ø )
 {
  LAYER_I_PLUS = Ø
  for each v in LAYER_I
  {
    for each w in VLISTS[v]
    {
      if (w==t) continue;
      if ( CHECKED[w]==1 ) continue;
     LAYER_I_PLUS := LAYER_I_PLUS ⋃ {w};
    }
  }
  if ( LAYER_I_PLUS=Ø ) break;

  for ( i = 1 ; i <|LAYER_I_PLUS| ; i++)
  {
    COMBINATIONS=
      = CREATE_COMBINATIONS( |LAYER_I_PLUS| ,i );
   for each comb_vset in COMBINATIONS
       for (j=0 ;j<| comb_vset | ; j++ )
           comb_vset [j] = LAYER_I_PLUS[ comb_vset [j] ];

   for each comb_vset in COMBINATIONS
   {
     for ( j =0 ; j < |VSETS[layer_i_index]| ; j++ )
     {
       vset_tmp = Ø;
       for each w in VSETS[v][j]
           vset_tmp = vset_tmp ⋃ {w};
       for each w in comb_vset
           vset_tmp = vset_tmp ⋃ {w};
       if ( ADD_ESET( vset_tmp ) )
          VSETS[layer_i_index+1]=
              =VSETS[layer_i_index+1] ⋃ {vset_tmp};
     }// end of for j
   }// end of for each comb_vset

   layer_i_index++;
  LAYER_I= Ø;
   for each v in LAYER_I_PLUS
   {
     LAYER_I= LAYER_I ⋃ {v};
     CHECKED[v] =1;
    }
  } // end of for i
 } //end of while (LAYER_I≠Ø)
 CHECK_ESETS ( );
  return 1;
} //end of function FIND_CUTS

function ADD_ESET ( vset )
{
 new_eset= Ø;
 bool = true;
 for each v in vset
     for each w in VLISTS[v]
         if ( w ∉ vset )      new_eset= new_eset ⋃ { {v,w} }
 for ( i=0 ; i < |ESETS| ; i++ )
   for each eset in ESETS[i]
    if   ( (new_eset ⋂ eset) == eset )
    {
      bool = false;
      if   ( |new_eset| == |eset| ) return 0;
    }
 if (bool==true)
   ESETS[ |new_eset|-1 ] = ESETS[ |new_eset|-1 ] ⋃ new_eset;
 return l;
}//end of function ADD_ESET

function CHECK_ESETS ( )
{
 for (m=|ESETS|-1 ; m≥0 ; m--)
    for each eset in ESETS[i]
    {
      bool=true;
      for ( i=0 ; |ESETS| ; i++ )
           for each eset2 in ESETS[i]
              if   ( (eset ⋂ eset2) == eset )
                   bool = false;
       if (bool==false)
          ESETS[ |eset|-1 ] = ESETS[ |eset|-1 ] \ new_eset;
    }
}//end of function CHECK_ESETS
```

The computer program

A computer program for finding all cuts in graph was created. The computer program was written in *C++* language and uses STL (Standard Template Library) [6]. The program has a command-line interface. The structure of the graph is defined in a textfile as the list of edges. The found cuts are written in output textfile.

Example

A scheme of sample electric system is shown in Figure 1. This is the scheme of electric substation.

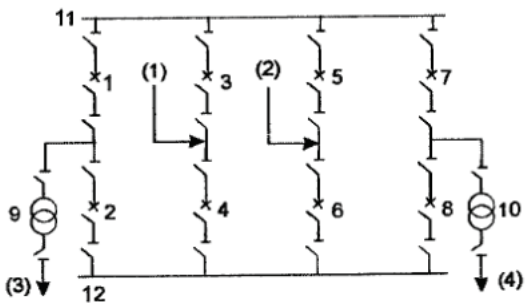

Fig.1. Sample electric system scheme [4]

The corresponding graph of the structure from Figure 1 is shown on Figure 2.

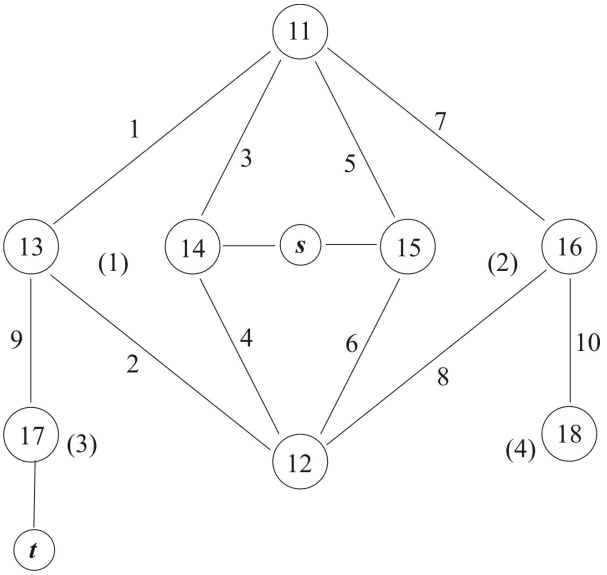

Fig.2. Graph of the electric system shown in Fig. 1.

Table 1. Cut in graph from figure 2

number of edges in cut	number of cuts	list of cuts
1	2	{9},{13}
2	5	{1,12},{1,2},{2,11},{11,12},{14,15}
3	8	{3,4,15},{3,12,15},{3,5,12},{4,11,15}, {4,6,11},{5,6,14},{5,12,14},{6,11,14}
4	19	{1,4,7,15},{1,6,7,14},{1,4,6,7}, {1,4,15,16},{1,6,14,16},{1,4,6,16}, {1,4,8,15},{1,6,8,14},{1,4,6,8}, {2,3,8,15},{2,5,8,14},{2,3,5,8}, {2,3,15,16},{2,5,14,16},{2,3,5,16}, {2,3,7,15},{2,5,7,14},{2,3,5,7},{3,4,5,6}

For the chosen pair of vertices source s and sink t, the program finds all cuts of graph shown in Figure 2. The cuts can be later used in the cut set method of evaluating reliability of the electric system [1].
All cuts found by the computer program are given in Table 1. Only cuts crossing edges and vertices indexed on Figure 2 are listed. Other cuts e.g. cut crossing edges {s,14} and {s,15} are considered to be less important and are skipped.

Conclusions

The presented algorithm allow to enumerate all cuts in graph including cuts through vertices. The algorithm do not require searching all paths in graph and do not require finding for graph connectivity. The algorithm can be used in reliability computation, especially in those issues where all cuts or high-order cuts need to be considered.

REFERENCES

[1] Billinton R., Allan. R.: Reliability evaluation of power systems, 2nd ed., Plenum Press New York 1994.
[2] Grabski F., Jaźwiński. J.: Funkcje o losowych argumentach w zagadnieniach niezawodności, bezpieńczeństwa i logistyki, WKŁ Warszawa 2009.
[3] Billinton R., Lian G., A new technique for active minimal cut set selection used in substation reliability evaluation, *Electric Power Systems Research*, 35 (1995), No. 5, 797-805
[4] Filipiak R.: Metody oceny niezawodności stacji elektroenergetycznych WN/SN, *Metody i systemy Komputerowe w Automatyce i Elektrotechnice*, Częstochowa 2005.
[5] Baron B., Piątek. Ł.: Metody numeryczne w C++, Gliwice 2004.
[6] Stroustrup. B.: Język C++, WNT Warszawa 2002.
[7] Ruskey. F.: Combinatorial generation, Victoria 2003.

Author: *mgr inż. Łukasz Piątek, Politechnika Częstochowska, Instytut Informatyki, al. Armii Krajowej 17, 42-200 Częstochowa, E-mail: l_piatek@interia.pl*