"Mircea cel Batran" Naval Academy Scientific Bulletin, Volume XVIII – 2015 – Issue 2
Published by "Mircea cel Batran" Naval Academy Press, Constanta, Romania // The journal is indexed in:
PROQUEST SciTech Journals, PROQUEST Engineering Journals, PROQUEST Illustrata: Technology, PROQUEST Technology Journals, PROQUEST Military Collection PROQUEST Advanced Technologies & Aerospace

ALGORITHM FOR GENERATING ALL k VERTICES SUBGRAPHS AND FINDING THE VALUES FOR EACH OF THE SUBGRAPHS

Paul VASILIU[1]
Tiberiu PAZARA[2]
[1]Lecturer PhD Eng. Department of Electrical Engineering and Electronics, „Mircea cel Batran" Naval Academy
[2]Assistant PhD, Department of Electrical Engineering and Electronics, „Mircea cel Batran" Naval Academy

***Abstract**: In this paper we will prove how all subgraphs with k vertices and weighted edges of a graph can be generated and how can be computed the value of each subgraph. The paper will include a written C++ program that implements the presented algorithm. Moreover, a use case scenario for this algorithm will be described.*
***Keywords**: graph, subgraph, compute, algorithm, program*

INTRODUCTION

Let it be $X=\{x_1,x_2,\cdots,x_n\}$ the set of vertices, we have the application $\Gamma: X\to P(X)$and the graph $G=(X,\Gamma)$. Let it be $A=(a_{ij})_{\substack{i=1,2,\cdots,n\\ j=1,2,\cdots,n}}$ with the elements $a_{ij}=\begin{cases}1 & daca \quad x_j\in\Gamma(x_i)\\ 0 & daca \quad x_j\notin\Gamma(x_i)\end{cases}$ the adjacency matrix of graph $G=(X,\Gamma)$.

Let it be $U=\{(x_i,x_j)|x_j\in\Gamma(x_i)\}$ the set of edges of the graph. Let us assume that the graph has weighted edges $l(x_i,x_j)$ with values greater than 0 for every $(x_i,x_j)\in U$.

We define as value for the graph $G=(X,\Gamma)$, the number $l(G)=\sum_{(x_i,x_j)\in U} l(x_i,x_j)$.

Let it be $X'=\{x_{i_1},x_{i_2},\cdots,x_{i_k}\}\subset\{x_1,x_2,\cdots,x_n\}$ the subset, not null, of vertices of set X.

Let it be $\Gamma': X'\to P(X')$ the application defined $\Gamma'(x_{i_p})=\Gamma(x_{i_p})\cap X'$, by for every $x_{i_p}\in X'$. Pair $G'=(X',\Gamma')$ is called the subgraph of the graph $G=(X,\Gamma)$ defined by the set of vertices X'.

Let it be $U'=\{(x_i,x_j)|x_j\in\Gamma'(x_i)\}$ the set of edges of the subgraph $G'=(X',\Gamma')$.

Let it be defined the number $(G')=\sum_{(x_i,x_j)\in U'} l(x_i,x_j)$for the subgraph $G=(X,\Gamma)$.

In this paper we will prove how can be generated all the subgraphs of k vertices of a given graph and how their values can be computed.

We will present a program written in C++, that receives as input a graph and the number k, representing the number of vertices. The program generates all the subgraphs defined by k vertices and their values. The paper includes an usage example for the program. In the end, we will present the future work for this research.

ALGORITHM

In this section we will show how can be generated all the subgraphs defined by k vertices of a given graph and also, how the values for the subgraphs are computed.

For this purpose, we generated through backtracking all the subsets of k distinct elements of the set $\{1,2,\cdots,n\}$in which the order doesn't matter. In other words, we generated all the combinations of k elements of set $\{1,2,\cdots,n\}$. All these subsets are saved in a file named sg.txt. The elements of each subset $\{i_1,i_2,\cdots,i_k\}$ represent the indexes of the subgraph vertices that will be generated and which value will be computed. The value for the subgraph $G'=(X',\Gamma')$ defined by the vertices $x_{i_1},x_{i_2},\cdots,x_{i_k}$equals to $l(G')=\sum_{(x_i,x_j)\in U'} l(x_i,x_j)$.

Subsets are extracted successively from file sg.txt. For each subset $\{i_1,i_2,\cdots,i_k\}$ is generated the graph defined by vertices $x_{i_1},x_{i_2},\cdots,x_{i_k}$ and computed its value.

When generating each subgraph, we take into consideration the adjacency matrix $A'$of the subgraph $G'=(X',\Gamma')$, can be obtained from the adjacency matrix $A$of graph $G=(X,\Gamma)$ keeping the rows and columns with indexes $i_1,i_2,\cdots,i_k$ and canceling all the rows and columns from matrix A with indexes $i\notin\{i_1,i_2,\cdots,i_k\}$.

IMPLEMENTATION

Below we present the program written in C++ that implements the described algorithm.

The program receives as input the text file associated to the graph $G=(X,\Gamma)$, the number of vertices and the vertices $x_{i_1},x_{i_2},\cdots,x_{i_k}$ of the subgraph that will be generated. The program outputs the number of vertices, the number of edges, the adjacency matrix and the edges of graph $G=(X,\Gamma)$. For the subgraph $G'=(X',\Gamma')$ we display the adjacency matrix and the edges of the subgraph.

```
#include <stdio.h>
#include <conio.h>
#include <malloc.h>
#define dim 1000

int va[100];
typedef struct arc
{
int x; int y; int val;
}edge;
```

```
edge mu[dim];   // mu array of edges

// Matrix allocation
int ** alocmat (int n)
{
 int i;
 int ** p=(int **) malloc ((n+1)*sizeof (int *));
 if ( p != NULL)
 for (i=0; i<=n ;i++)
 p[i] =(int *) calloc ((n+1),sizeof (int));
 return p;
}

// Function for allocating the array
int *alocv(int n)
{
 int *p;
 p=(int *)malloc(n*sizeof(int));
 return p;
}

// Function for displaying the array
void displayv(int n,int *v) //
{
 int i;
 printf(" { ");
 for(i=0;i<n-1;i++)
 printf(" %d , ",v[i]);
 printf(" %d ",v[i]);
 printf(" } \n");
}

// Function for displaying matrix n x n
void displaym(int **a,int n,char *c)
{
 int i,j;
 printf("\n %s \n\n",c);
 for(i=1;i<=n;i++)
 {
 for(j=1;j<=n;j++)
 printf(" %2d ",a[i][j]);
 printf("\n");
 }
}

// Function for displaying the subgraph
void displaysg(char *c,int k,int *v)
{
 int i,j;
 printf("\n\n %s ",c);
 displayv(k,v);
 printf("\n\n");
}

// Function for displaying matrix n x n
void displaymm(int **a,int n,char *c,int k,int *v)
{
 int i,j;
 printf("\n %s ",c);
 displayv(k,v);
 for(i=1;i<=n;i++)
 {
 for(j=1;j<=n;j++)
 printf(" %2d ",a[i][j]);
 printf("\n");
 }
}

// Function for searching k in v
int search(int k,int n,int *v)
{
 int i,found;
 found=0;
 for(i=0;i<n;i++)
 if(v[i]==k)
 {
 found=1;
 break;
 }
 return found;
}

//Function for reading n, m
int read_n_m(int &n, int &m, char *name)
{
 FILE *f;
 if((f=fopen(name,"r"))!= NULL)
 {
 fscanf(f,"%d %d",&n,&m);
 fclose(f);
 return 1;
 }
 else
 return 0;
}

// Function for displaying the edges of the graph
void display_edges(int m)
{
  int i,val=0;
  printf("\n Graph G has %d edges \n\n",m);
  for(i=1;i<=m;i++)
  {
  printf(" Edge %d:\t ( x%d , x%d ) with value %d
\n",i,mu[i].x,mu[i].y,mu[i].val);
  val+=mu[i].val;
  }
  printf(" The value of the graph is %d \n",val);
  printf("\n");
}

//Function for reading the input file and
//building the adjacency matrix
int read_graph(char *name, int **a)
{
 int i,j,x,y,val,n,m;
 FILE *f;
 if((f=fopen(name,"r")) != NULL)
 {
 fscanf(f,"%d %d",&n,&m);
 for(i=1;i<=m;i++)
 {
```

```
 fscanf(f,"%d %d %d",&mu[i].x,&mu[i].y,&mu[i].val);
 a[mu[i].x][mu[i].y]=1;
 }
 fclose(f);
 return 1;
 }
 else
 return 0;
 }

//Function for generating
//the subgraph adjacency matrix
void gen(int n,int **a,int k,int *v,int **ap)
{
 int i,j;
 for(i=1;i<=n;i++)
 if(search(i,k,v)==1)
 for(j=1;j<=n;j++)
 if(search(j,k,v)==1)
 ap[i][j]=a[i][j];
}

// Function for displaying the edges of the subgraph
// Compute the value of the subgraph
int display_edges_s(int n,int **ap)
{
  int i,j,m,val=0;
  for(m=1,i=1;i<=n;i++)
  for(j=1;j<=n;j++)
  if(ap[i][j]==1)
  {
  mu[m].x=i;
  mu[m].y=j;
  m++;
  }
  m--;
  printf("\n The subgraph has %d edges \n\n",m);
  for(i=1;i<=m;i++)
  {
  printf(" The edge %d:\t ( x%d , x%d ) of value %d \n",i,mu[i].x,mu[i].y,mu[i].val);
  val+=mu[i].val;
  }
  printf("\n");
  return val;
}

//Function for reading the vertices of the next subgraph
void readf(FILE *f,int n,int *v)
{
 int i;
 for(i=0;i<n;i++)
 fscanf(f,"%d ",&v[i]);
}

//Matrix initialization
void initm(int n,int **a)
{
 int i,j;
 for(i=1;i<=n;i++)
 for(j=1;j<=n;j++)
 a[i][j]=0;
}

//Function for initializing array x
void init(int *x,int k)
{
 x[k]=0;
}

//Function for testing the solution
int sol(int k,int p)
{
 return k==p+1;
}

//Function for generating the successor
int successor(int *x,int k,int n)
{
 int as;
 if(x[k]<n)
 {
 as=1;
 x[k]=x[k]+1;
 }
 else
 as=0;
 return as;
}

//Function for validating the solution
int validation(int *x,int k)
{
 int ev,i;
 ev=1;
 if(k>=2 && x[k]<=x[k-1])
 ev=0;
 return ev;
}

// Function for writing the solution in a file
void writef(FILE *f,int k, int *x) //writeF
{
 int i;
 for(i=1;i<=k-1;i++)
 fprintf(f,"%d ",va[x[i]]);
 fprintf(f,"\n");
}

// Function for recursive backtracking
void back(int *x,int k,int n,int p, FILE *f)
{
 if(sol(k,p))
 writef(f,k,x);
 else
 {
 init(x,k);
 while(successor(x,k,n))
 if(validation(x,k))
```

"Mircea cel Batran" Naval Academy Scientific Bulletin, Volume XVIII – 2015 – Issue 2
Published by "Mircea cel Batran" Naval Academy Press, Constanta, Romania // The journal is indexed in:
PROQUEST SciTech Journals, PROQUEST Engineering Journals, PROQUEST Illustrata: Technology, PROQUEST Technology Journals, PROQUEST Military Collection PROQUEST Advanced Technologies & Aerospace

```
 back(x,k+1,n,p,f);
 }
}

//Main function
int main()
{
 int n,m,k,**a,**ap,*v,nsg=0,i,x[100];
 char name[30];
 FILE *f;
 printf("\n File name : ");
 fflush(stdin);
 gets(name);
 if(read_n_m(n,m,name))
 {
 printf("\n Number of vertices %d \n",n);
 printf(" Number of edges %d \n",m);
 getch();
 a=alocmat(n);
 if(read_graph(name,a))
 {
 displaym(a,n," Graph adjacency matrix \n");
 display_edges(m);
 printf(" The number of vertices of the subgraph = ");
 scanf("%d",&k);
 if(1<=k && k<=n)
 {
 f=fopen("sg.txt","w");
 for(i=1;i<=n;va[i]=i,i++);
 back(x,1,n,k,f);
 fclose(f);
 v=alocv(k);
 ap=alocmat(n);
 f=fopen("sg.txt","r");
 while(!feof(f))
 {
  readf(f,k,v);
  nsg++;
  gen(n,a,k,v,ap);
  displaysg(" Subgraph ",k,v);
  displaymm(ap,n," Subgraph adjacency matrix ",k,v);
  printf(" The value of the subgraph is %d \n",display_edges_s(n,ap));
  initm(n,ap);
  getch();
  }
  printf("\n\n Number of subgraphs with %d vertices = %d \n",k,nsg);
  fclose(f);
  }
  else
  printf(" Undefinied problem \n");
  getch();
}
}
}
```

EXAMPLE

Let us consider the directed graph $G = (X, \Gamma)$, with weighted edges, definied by the set of vertices $X = \{x_1, x_2, x_3, x_4, x_5\}$and the application $\Gamma: X \to P(X)$ definied by the equalities:$\Gamma(x_1) = \{x_1, x_3, x_5\}$, $\Gamma(x_2) = \{x_1, x_4, x_5\}$, $\Gamma(x_3) = \{x_2, x_4, x_5\}$, $\Gamma(x_4) = \{x_2, x_3\}$, $\Gamma(x_5) = \{x_1, x_3, x_4\}$.
Edges values are given in Table 1:

Edge	Value	Edge	Value
(x_1, x_1)	3	(x_3, x_4)	7
(x_1, x_3)	4	(x_3, x_5)	1
(x_1, x_5)	2	(x_4, x_2)	4
(x_2, x_1)	5	(x_4, x_3)	6
(x_2, x_4)	2	(x_5, x_1)	2
(x_2, x_5)	6	(x_5, x_3)	1
(x_3, x_2)	4	(x_5, x_4)	6

Table 1 Edges values

The adjacency matrix of the graph $G = (X, \Gamma)$is:

$$A = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix}.$$

The program read as input a text file with name graph.txt The file has on the first line the number n, representing the number of verices, and the number m, representing the number of edges separated by one blank space. On the next m lines we find the edges of the subgraphand their values. For each graph from our example, the input file graph.txt is the following:

```
5 14
1 1 3
1 3 4
1 5 2
2 1 5
2 4 2
2 5 6
3 2 4
3 4 7
3 5 1
4 2 4
4 3 6
5 1 2
5 3 1
5 4 6
```

For k = 4, file sg.txt is:

```
1 2 3 4
1 2 3 5
1 2 4 5
```

1 3 4 5
2 3 4 5

A run execution example of the program is:

File name : graph.txt

Number of vertices 5
Number of edges 14

Graph adjacency matrix

```
 1  0  1  0  1
 1  0  0  1  1
 0  1  0  1  1
 0  1  1  0  0
 1  0  1  1  0
```

Graph G has 14 edges

Edge 1: (x1 , x1) with value 3
Edge 2: (x1 , x3) with value 4
Edge 3: (x1 , x5) with value 2
Edge 4: (x2 , x1) with value 5
Edge 5: (x2 , x4) with value 2
Edge 6: (x2 , x5) with value 6
Edge 7: (x3 , x2) with value 4
Edge 8: (x3 , x4) with value 7
Edge 9: (x3 , x5) with value 1
Edge 10: (x4 , x2) with value 4
Edge 11: (x4 , x3) with value 6
Edge 12: (x5 , x1) with value 2
Edge 13: (x5 , x3) with value 1
Edge 14: (x5 , x4) with value 6
The value of the graph is 53

The number of vertices of the subgraph = 4

Subgraph { 1 , 2 , 3 , 4 }

Subgraph adjacency matrix { 1 , 2 , 3 , 4 }

```
 1  0  1  0  0
 1  0  0  1  0
 0  1  0  1  0
 0  1  1  0  0
 0  0  0  0  0
```

The subgraph has 8 edges

The edge 1: (x1 , x1) of value 3
The edge 2: (x1 , x3) of value 4
The edge 3: (x2 , x1) of value 2
The edge 4: (x2 , x4) of value 5
The edge 5: (x3 , x2) of value 2
The edge 6: (x3 , x4) of value 6
The edge 7: (x4 , x2) of value 4
The edge 8: (x4 , x3) of value 7

The value of the subgraph is 33

Subgraph { 1 , 2 , 3 , 5 }

Subgraph adjacency matrix { 1 , 2 , 3 , 5 }

```
 1  0  1  0  1
 1  0  0  0  1
 0  1  0  0  1
 0  0  0  0  0
 1  0  1  0  0
```

The subgraph has 9 edges

The edge 1: (x1 , x1) of value 3
The edge 2: (x1 , x3) of value 4
The edge 3: (x1 , x5) of value 2
The edge 4: (x2 , x1) of value 5
The edge 5: (x2 , x5) of value 2
The edge 6: (x3 , x2) of value 6
The edge 7: (x3 , x5) of value 4
The edge 8: (x5 , x1) of value 7
The edge 9: (x5 , x3) of value 1

The value of the subgraph is 34

Subgraph { 1 , 2 , 4 , 5 }

Subgraph adjacency matrix { 1 , 2 , 4 , 5 }

```
 1  0  0  0  1
 1  0  0  1  1
 0  0  0  0  0
 0  1  0  0  0
 1  0  0  1  0
```

The subgraph has 8 edges

The edge 1: (x1 , x1) of value 3
The edge 2: (x1 , x5) of value 4
The edge 3: (x2 , x1) of value 2
The edge 4: (x2 , x4) of value 5
The edge 5: (x2 , x5) of value 2
The edge 6: (x4 , x2) of value 6
The edge 7: (x5 , x1) of value 4
The edge 8: (x5 , x4) of value 7

The value of the subgraph is 33

Subgraph { 1 , 3 , 4 , 5 }

Subgraph adjacency matrix { 1 , 3 , 4 , 5 }

```
 1  0  1  0  1
 0  0  0  0  0
 0  0  0  1  1
 0  0  1  0  0
 1  0  1  1  0
```

The subgraph has 9 edges

The edge 1: (x1 , x1) of value 3
The edge 2: (x1 , x3) of value 4
The edge 3: (x1 , x5) of value 2
The edge 4: (x3 , x4) of value 5
The edge 5: (x3 , x5) of value 2
The edge 6: (x4 , x3) of value 6
The edge 7: (x5 , x1) of value 4
The edge 8: (x5 , x3) of value 7
The edge 9: (x5 , x4) of value 1

The value of the subgraph is 34

Subgraph { 2 , 3 , 4 , 5 }

Subgraph adjacency matrix { 2 , 3 , 4 , 5 }

```
0 0 0 0 0
0 0 0 1 1
0 1 0 1 1
0 1 1 0 0
0 0 1 1 0
```

The subgraph has 9 edges

The edge 1: (x2 , x4) of value 3
The edge 2: (x2 , x5) of value 4
The edge 3: (x3 , x2) of value 2
The edge 4: (x3 , x4) of value 5
The edge 5: (x3 , x5) of value 2
The edge 6: (x4 , x2) of value 6
The edge 7: (x4 , x3) of value 4
The edge 8: (x5 , x3) of value 7
The edge 9: (x5 , x4) of value 1

The value of the subgraph is 34

Number of subgraphs with 4 vertices = 5

This execution example of the program highlights the correctness of the obtained theoretical results but also the correctness of the results obtained by the program.

CONCLUSIONS

In this paper we have presented an algorithm and a program written in C++ for generating all the subgraphs defined by K given vertices of a given graph with weighted edges. Furthermore, we computed the values of these subgraphs.

From the large spectrum of future work we will enumerate:

- finding the subgraphs of minimum value;
- finding the subgraph of maximum value.

BIBLIOGRAPHY:

[1] Blăjină A., Produse software aplicate în programarea matematică şi teoria jocurilor, Ed. Albastra, Cluj-Napoca 2006
[2] Gorunescu F., Prodan A., Modelare stochastică şi simulare, Ed. Albastra, Cluj-Napoca 2001
[3] Marin Ghe., Popoviciu I., Vasiliu P., Lupei T., Chiru C., Cercetări operaționale – Programare liniară, Ed. ANMB, Constanța 2002
[4] Marin Ghe., Popoviciu I., Vasiliu P., Cață M., Cercetări operaționale. Grafuri. Probleme de coordonare. Teoria aşteptării. Programare dinamică. Teoria deciziei. Ed. ANMB, Constanța 2003
[5] Vasiliu P., Modelare şi simulare, Ed. ANMB, Constanța 2013
[6] Văduva I., Modele de simulare cu calculatorul, Editura Tehnică, Bucureşti, 1977