“Mircea cel Batran” Naval Academy Scientific Bulletin, Volume XX – 2017 – Issue 2
The journal is indexed in: PROQUEST / DOAJ / Crossref / EBSCOhost/ INDEX COPERNICUS/ OAJI / DRJI / JOURNAL INDEX / I2OR / SCIENCE LIBRARY INDEX / Google Scholar / Academic Keys / ROAD Open Access / Academic Resources / Scientific Indexing Services / SCIPIO/ JIFACTOR

AUTOMATICALLY FINDING THE HAMILTONIAN PATH IN A DIRECTED GRAPH WITHOUT CYCLES BY USING THE YU CHEN ALGORITHM

Paul VASILIU[1]
[1]Lecturer Ph.D. Eng. Department of Electrical Engineering and Electronics, „MirceacelBătrân” Naval Academy, No.1 Fulgerului Street, Constanţa, Romania
p_vasiliu@yahoo.com

Abstract:*The problem of operations succession (the execution of multiple operations on a machine in the order in which the sum of the consumed times/cycles for getting ready the machine to move from one operation to another to be minimum), the traveling salesman problem (finding the shortest path that visits all the houses from his neighborhood) are problems that can be reduced to finding the Hamiltonian path in a directed graph. “Yu Chen’s algorithm” [1], [3], [5] solves the problem of finding the Hamiltonian path in a directed graph without cycles. The algorithm finds the adjacency matrix in a directed, finite graph, with$n$vertices. If the graph has no cycles, the algorithm computes the powers of reaching the vertices. If the sum of the powers for reaching the vertices is equal to $n \cdot (n-1)/2$and if the powers for reaching the vertices are distinct two by two, then the graph has a Hamiltonian path defined by the powers of reaching the vertices. In this paper, the author presents an implementation in C language of the Yu Chen algorithm. The program presents the computations of the adjacency matrix and test whether there are cycles or not in the graph. If there are no cycles, the program computes the powers of reaching the vertices, checks the requirements from Yu Chen’s theory and if they are satisfied, determines the Hamiltonian path in the graph.*
Keywords:*algorithm, directed graph,hamiltonianpath, programming*

INTRODUCTION

The problem of finding the Hamiltonian paths in a directed graph and without cycles has been solved by Yu Chen’s algorithm. Applying it manually is difficult if the number n of vertices of the graph is high. In this paper, the author proposes an implementation in C language of the Yu Chen algorithm for finding the Hamiltonian path (if it exists), in a directed graph and without cycles.

YU CHEN’s ALGORITHM

Let us consider $X = \{x_1, x_2, \cdots, x_n\}$ the set of vertices of graph $G = (X, U)$ which is directed and without cycles, and let it be U the set of edges. Let it be $A = (a_{ij}),\ i,j = 1,2,\cdots,n$, $a_{ij} = \begin{cases} 1 & for \quad (x_i, x_j) \in U \\ 0 & for \quad (x_i, x_j) \notin U \end{cases}$ the adjacency matrix.
Let it be l_i row i from the adjacency matrix. Let it be $\oplus$ the symbol of the boolean sum. For all $x, y \in \{0,1\}$the boolean sum is defined by: $x \oplus y = \begin{cases} 0 & if \quad x = 0\ and\ y = 0 \\ 1 & otherwise \end{cases}$. For the following rows $l_i = (a_{i1}, a_{i2}, \cdots, a_{in})$ and $l_j = (a_{j1}, a_{j2}, \cdots, a_{jn})$ from the A matrix we defined the operation $l_i \oplus l_j = (a_{i1} \oplus a_{j1}, a_{i2} \oplus a_{j2}, \cdots, a_{in} \oplus a_{jn})$ as boolean sum between rows $l_i$and l_j.
The matrix of paths $D = (d_{ij}),\ i,j = 1,2,\cdots,n$ of graph $G = (X, U)$, can be found with another algorithm written by Yu Chen and presented in this paper (C1 algorithm):

```
start
write adjacency matrix $A$ of the graph;

fori=1 to $n$ // for each row $l_i$ from $A$
do
repeat
find$J = \{j \mid a_{ij} = 1\}$
for$j \in J$
assign$l_i \leftarrow l_i \oplus l_j$
assign$D \leftarrow A$
until$l_i$is unchanged
endfor
end
```

If there is $i \in \{1,2,\cdots,n\}$ with the property that $d_{ii} = 1$ then the graph $G = (X, U)$ has cycles and in this case, Yu Chen’s algorithm for finding the Hamiltonian path cannot be applied. If $d_{ii} = 0$ for every $i \in \{1,2,\cdots,n\}$ then Yu Chen’s algorithm can be applied to find the Hamiltonian path. It is named power for reaching the vertices x_i and is identified as p_i, the number of vertices x_j for which there is at least one path from x_i to x_j. It is straightforward to notice that $p_i = \sum_{j=1}^{n} d_{ij}$ (the number of values equal to 1 from row i from the path matrix D).
The algorithm is based on the following theoretical results from Yu Chen [1], [3], [5], that we will use without a proof.
Sentence 1. Let it be $G = (X, U)$ a directed graph, without cycles. For each edge $(x_i, x_j) \in U$ there is the following inequality $p_i \geq p_j$.
Sentence 2. Let it be $G = (X, U)$ a directed graph, without cycles in which there is a path from $x_i$to x_j, then we have the following statement $p_i \geq p_j$.

DOI: 10.21279/1454-864X-17-I2-017

Theorem 1. (Chen) Let it be $G=(X,U)$ a directed graph, without cycles and with n vertices. Graph $G=(X,U)$ contains a Hamiltonian path if and only if $\sum_{i=1}^{n} p_i = \frac{n\cdot(n-1)}{2}$.
Theorem 2. Let it be $G=(X,U)$ a directed graph, without cycles. If the graf contains a Hamiltonian path then this the path is unique.
Let it be $G=(X,U)$ a directed graph, without cycles. Yu Chen's algorithms for finding the Hamiltonian path in graph $G=(X,U)$ follows the below steps:

Step 1. Write the adjacency matrix A.
Step 2. Build the matrix of paths $D$by using the C1 algorithm.
If $(\exists) i \in \{1,2,\cdots,n\}$ having $d_{ii}=1$
then end
elsegoto step 3.
Step 3. Find the powers of reaching the vertices p_i, $i \in \{1,2,\cdots,n\}$ and compute $\sum_{i=1}^{n} p_i$.
if$\sum_{i=1}^{n} p_i \neq \frac{n\cdot(n-1)}{2}$
then end
else go to step 4.
Step 4. We sort in descending order the powers of reaching the vertices. Let it be $\sigma \in S_n$ the permutation with the following property $p_{\sigma(1)} > p_{\sigma(2)} > \cdots > p_{\sigma(n)}$. The Hamiltonian path from graph $G=(X,U)$ will be the path $d_H=\{x_{\sigma(1)}, x_{\sigma(2)}, \cdots, x_{\sigma(n)}\}$.
We will go through an example with a simple graph for the Yu Chen algorithm. Let it be $G=(X,U)$ the graph defined by the set of vertices $X=\{x_1,x_2,x_3,x_4\}$ and the set of edges $U=\{(x_1,x_2),(x_1,x_3),(x_3,x_2),(x_3,x_4),(x_4,x_2)\}$, with the below representation from figure 1.

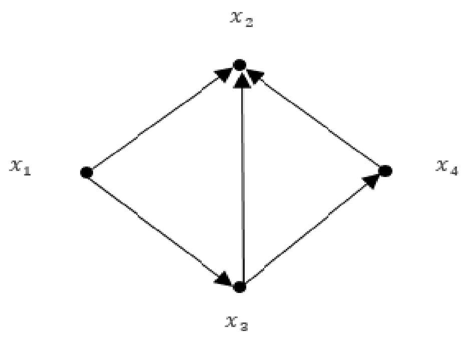

Figure 1. Graph $G=(X,U)$

We can easily notice that the graph from figure 1 is directed and without cycles. We can also identify that path $d_H=\{x_1,x_3,x_4,x_2\}$ is the unique Hamiltonian path from this graph.
We will find these results manually by applying Yu Chen's algorithm (C1 algorithm).

The adjacency matrix of graph $G=(X,U)$is $A=\begin{pmatrix}0&1&1&0\\0&0&0&0\\0&1&0&1\\0&1&0&0\end{pmatrix}$. We will build the matrix of paths.
We fix row 1. Because items a_{12} and a_{13} from matrix A are not null, we add the Boolean sum between rows 2 and 3 to the first row from A matrix. We obtainmatrix$D=\begin{pmatrix}0&1&1&1\\0&0&0&0\\0&1&0&1\\0&1&0&0\end{pmatrix}$.
Because the elements d_{12}, d_{13} şi d_{14} from matrix D are not nulls, we add the Boolean sum between rows 2,3, 4 to row 1 from matrix D . We obtain matrix $D=\begin{pmatrix}0&1&1&1\\0&0&0&0\\0&1&0&1\\0&1&0&0\end{pmatrix}$. Because row 1 hasn't changed, we fix the next row (row 2) from matrix D.
We fix the second row. Because all the elements on row 2 are nulls from matrix D, we fix the next rows.
We fix the third row. Because the elements d_{32}, and d_{34} from matrix D are not null, we add the Boolean sum between rows 2 and 4 to row 3 in matrix D. We obtain matrix $D=\begin{pmatrix}0&1&1&1\\0&0&0&0\\0&1&0&1\\0&1&0&0\end{pmatrix}$.
Because row 3 hasn't changed, we fix the next row (row 4) in matrix D.
We fix row 4. Because on row 4 the element $d_{42}$is not null, we add the Boolean sum between row 2 to row 4 from matrix D.We obtain matrix $D=\begin{pmatrix}0&1&1&1\\0&0&0&0\\0&1&0&1\\0&1&0&0\end{pmatrix}$. Because row 4 hasn't changed, Yu Chen's algorithm for finding the paths matrix is completed. We obtain the path matrix $D=\begin{pmatrix}0&1&1&1\\0&0&0&0\\0&1&0&1\\0&1&0&0\end{pmatrix}$. We can notice that all the elements from the main diagonal are null, so we conclude that the graph has no cycles.
We will compute the powers of reaching the vertices as below:
$p_1=\sum_{j=1}^{4} d_{1j}=3$, $p_2=\sum_{j=1}^{4} d_{2j}=0$, $p_3=\sum_{j=1}^{4} d_{3j}=2$ and $p_4=\sum_{j=1}^{4} d_{4j}=1$.
We verify the equality $\sum_{i=1}^{n} p_i = \frac{n\cdot(n-1)}{2}$ from Chen's theorem. In our case, we have $n=4$and $\sum_{i=1}^{4} p_i = 3+0+2+1=6=\frac{4\cdot(4-1)}{2}$. This proves that the graph has a single Hamiltonian path.
We will identify now the Hamiltonian path. We sort in descending order the powers of reaching the vertices and we obtain: $p_1>p_3>p_4>p_2$. Based

DOI: 10.21279/1454-864X-17-I2-017

on this, we conclude that the Hamiltonian path is: $d_H = \{x_1, x_3, x_4, x_2\}$.

Let us consider graph $G_1 = (X, U_1)$ defined by the set of vertices $X = \{x_1, x_2, x_3, x_4\}$ and the set of edges:
$U_1 = \{(x_1, x_2), (x_1, x_3), (x_2, x_3), (x_3, x_4), (x_4, x_2)\}$, having the representation from figure 2.We can notice that the graph is directed and has cycles. A cycle in graph $G_1 = (X, U_1)$is: $\{x_2, x_3, x_4, x_2\}$. In this case, Yu Chen's algorithm for finding the Hamiltonian path cannot be applied.

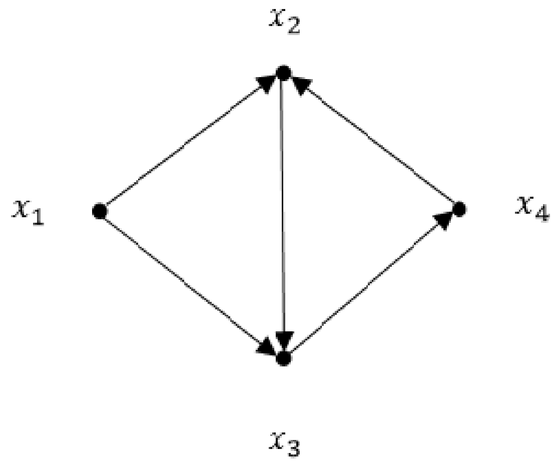

Figure 2. Graph $G_1 = (X, U_1)$

IMPLEMENTATION

The program written in C language, reads the input data from a text file named graphdh.txt with the following configuration: on the first row, we find the number of vertices n and the number of edges m from the graph, separated by an empty space. On the next m lines from the file we find the indices of the edges in the graph. The program follows the below steps:

Step 1. Read input file graphdh.txt.
Step 2. Compute adjacency matrix.
Step 3. Test whether the graph has cycles. If we find cycles, we exit the program. If there are no cycles, we continue with step 4.
Step 4. Compute the powers for reaching the vertices.
Step 5. If the condition from the Chen's theorem is not fullfiled (theorem 1) we exit the program with the conclusion that the graph doesn't have any Hamiltonian path. Otherwise, we continue the execution of the program with step 6.
Step 6. Sort in descending order the powers of reaching the vertices.
Step 7. Find and print the Hamiltonian path.
Step 8. End.

```
// Yu Chen's algorithm
// The matrix of the paths in a directed graph
// Hamiltonian path in a directed graph without cycles
// Input files graphdh.txt, graphdh1.txt

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

typedefstruct arc
{
int x; int y;
}edge;
edge mu[dim];   // mu array of edges

struct date
{
intval,poz;
};

// Matrix allocation
int ** alocmat (int n)
{
inti;
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 date
date *alocvs(int n)
{
date *p;
p=(date *)malloc(n*sizeof(date));
return p;
}
// Function for allocating the array
int *alocv(int n)
{
int *p;
p=(int *)malloc((n+1)*sizeof(int));
return p;
}

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

// Function for displaying matrix n x n
voiddisplaym(int **a,intn,const char *c)
```

DOI: 10.21279/1454-864X-17-I2-017

```
{
inti,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 reading n, m
intread_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
voiddisplay_edges(int m)
{
inti;
printf("\n Graph G has %d edges \n\n",m);
for(i=1;i<=m;i++)
printf("   Edge   %d:\t   (   x%d   ,   x%d   )
\n",i,mu[i].x,mu[i].y);
printf("\n");
}

//Function for reading the input file and
//building the adjacency matrix
intread_graph(char *name, int **a)
{
inti,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",&mu[i].x,&mu[i].y);
a[mu[i].x][mu[i].y]=1;
}
fclose(f);
return 1;
}
else
return 0;
}
" [6]

// Boolean sum
intsumb(inta,int b)
{
if(a==0 && b==0)
return 0;
else
return 1;
}

// Boolean sum of two arrays
voidsumbv(int *v,int *v1,int *v2, int n)
{
inti;
for(i=1;i<=n;i++)
v[i]=sumb(v1[i],v2[i]);
}

// Extracting row l from matrix a
voidextrag(int **a,int *v,intl,int n)
{
int j;
for(j=1;j<=n;j++)
v[j]=a[l][j];
}

// Comparing arrays
intcompar(int *v1,int *v2,int n)
{
intcomp,i;
comp=0;
for(i=1;i<=n;i++)
if(v1[i] != v2[i])
comp++;
return comp;
}

// Assign v1 <- v2
voidassignv(int *v1,int *v2,int n)
{
inti;
for(i=1;i<=n;i++)
v1[i]=v2[i];
}

// Generating the matrix of the paths
void gen(int **a,int **d,int n)
{
inti,j,k,*li,*li1,*li2,*li3;
li=alocv(n);
li1=alocv(n);
li2=alocv(n);
li3=alocv(n);
for(i=1;i<=n;i++)
{
extrag(a,li,i,n);
assignv(li1,li,n);
assignv(li3,li,n);
do
{
for(j=1;j<=n;j++)
```

DOI: 10.21279/1454-864X-17-I2-017

```
{
if(li3[j]==1)
{
extrag(a,li2,j,n);
sumbv(li1,li1,li2,n);
}
assignv(li3,li1,n);
}
}while(compar(li1,li3,n) != 0);
for(j=1;j<=n;j++)
d[i][j]=li1[j];
}
}

// Function for finding cycles
int cycles(int **d,int n)
{
inti,exist;
exist=0;
for(i=1;i<=n;i++)
if(d[i][i]==1)
exist++;
return exist;
}

// Sorting in descending order the array of the
// reaching powers
voidsortd(int *p,int *poz,int n)
{
inti,j;
date *pa,aux;
pa=alocvs(n);
for(i=1;i<=n;i++)
 {
pa[i-1].val=p[i];
pa[i-1].poz=i;
  }
for(i=1;i<=n;i++)
for(j=i+1;j<=n;j++)
if(pa[i-1].val<=pa[j-1].val)
{
aux=pa[i-1];
pa[i-1]=pa[j-1];
pa[j-1]=aux;
}
for(i=1;i<=n;i++)
poz[i]=pa[i-1].poz;
}

// Printing the Hamiltonian path
voidafisdh(int *poz,int n)
{
inti;
printf("\n\n Hamiltonian path is : { ");
for(i=1;i<n;i++)
printf(" x%d, ",poz[i]);
printf(" x%d } \n",poz[i]);
}

// Function for implementing Chen's algorithm for
// finding the Hamiltonian path in directed graphs
// without cycles
voidchen(int **d,int n)
{
int *p,i,j,n1,*poz;
p=alocv(n);
poz=alocv(n);
if(cycles(d,n) == 0)
{
for(n1=0,i=1;i<=n;i++)
for(j=1;j<=n;j++)
if(d[i][j]==1)
n1++;
if(n1 != n*(n-1)/2)
printf("\n\n The graph doesn't have a Hamiltonian path. \n");
else
{
printf("\n\n The graph has a Hamiltonian path. \n\n");
for(i=1;i<=n;i++)
{
p[i]=0;
for(j=1;j<=n;j++)
if(d[i][j]==1)
p[i]++;
}
for(i=1;i<=n;i++)
printf(" The power of reaching vertex x%d is %d \n",i,p[i]);
sortd(p,poz,n);
afisdh(poz,n);
}
}
else
{
printf(" The graph has cycles \n");
printf(" Yu Chen's algorithm is not applicable. \n");
}
}

// The main function
int main()
{
intn,m,**a,**d;
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);
d=alocmat(n);
if(read_graph(name,a))
```

DOI: 10.21279/1454-864X-17-I2-017

```
{
displaym(a,n," Graph adjacency matrix \n");
display_edges(m);
gen(a,d,n);
displaym(d,n," Paths matrix \n");
}
chen(d,n);
}
getch();
}
```

We will use as inputs for the program graphs $G = (X, U)$ and$G_1 = (X, U_1)$ from figures 1 and 2. The input file graphdh.txt associated with graph $G = (X, U)$from figure 1 is:

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

After running the program, we obtain the following results:

```
File name : graphdh.txt

Number of vertices 4
Number of edges 5

Graph adjacency matrix

0  1  1  0
0  0  0  0
0  1  0  1
0  1  0  0

Graph G has 5 edges

Edge 1:        ( x1 , x2 )
Edge 2:        ( x1 , x3 )
Edge 3:        ( x3 , x2 )
Edge 4:        ( x3 , x4 )
Edge 5:        ( x4 , x2 )

Paths matrix

0  1  1  1
0  0  0  0
0  1  0  1
0  1  0  0

The graph has a Hamiltonian path

The power of reaching vertex x1 is 3
The power of reaching vertex x2 is 0
The power of reaching vertex x3 is 2
The power of reaching vertex x4 is 1

Hamiltonian path is:  {  x1,  x3,  x4,  x2 }
```

The input file graphdh1.txt associated with graph $G_1 = (X, U_1)$from figure 2 is:

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

After running the program, we obtain the following results:

```
File name : graphdh1.txt

Number of vertices 4
Number of edges 5

Graph adjacency matrix

0  1  1  0
0  0  1  0
0  0  0  1
0  1  0  0

Graph G has 5 edges

Edge 1:        ( x1 , x2 )
Edge 2:        ( x1 , x3 )
Edge 3:        ( x2 , x3 )
Edge 4:        ( x3 , x4 )
Edge 5:        ( x4 , x2 )

Paths matrix

0  1  1  1
0  1  1  1
0  1  0  1
0  1  1  1

The graph has cycles.
Yu Chen’s algorithm is not applicable.
```

CONCLUSIONS

In this paper, the author presented a implementation in C language of Yu Chen’s algorithms for finding the Hamiltonian path (if one exists) in a directed graph, without cycles, that has a maximum number of 1000 vertices. We briefly presented Yu Chen’s algorithm together with the theory behind it. We have also included

DOI: 10.21279/1454-864X-17-I2-017

the proposed implementation in detail. The paper discusses an example which is solved not only manually, but also programmatically by using the program developed by the author. Yu Chen’s algorithm is applicable only for directed graphs that don’t have cycles.
From the large spectrum of future work we will enumerate:

- The development of an algorithm for directed graphs with cycles

Writing an extended program for graphs with cycles.

BIBLIOGRAPHY

[1] Acu, A.M., Acu, D., Acu, M., Dicu, P., Matematiciaplicateîneconomie – Volumul I, Editura ULB, Sibiu, 2001.
[2] Băutu, A., Vasiliu, P., Bazeleprogramăriicalculatoarelor, Ed. A.N.M.B., Constanţa 2009.
[3] Berge, C., Teoriagrafurilorşiaplicaţii, EdituraTehnică, Bucureşti, 1971.
[4] Kernigham, B., Ritchie, D., The C programming language, Prentice Hall,1975.
[5] Tomescu, I., Combinatoricăşiteoriagrafurilor, EdituraUniversităţii din Bucuresti, 1990.
[6] Vasiliu, P., Algorithm for determining the minimum value of all subgraphs with k vertices, Scientific Bulletin. Volume XIX-2016-Issue 2, ISSN 1454-864X, „MirceacelBătrân” Naval Academy Press, pag. 564-573.
[7] Vasiliu, P., Băutu, A., Programareacalculatoarelorînlimbajul C, Ed. Europolis, Constanţa, 2006.

DOI: 10.21279/1454-864X-17-I2-017