MBNA Publishing House Constanta 2022

Proceedings of the International Scientific Conference SEA-CONF

SEA-CONF PAPER • OPEN ACCESS

Automatic determination of all Hamiltonian cycles in acompletely undirected graph

To cite this article: P. Vasiliu, Proceedings of the International Scientific Conference SEA-CONF 2022, pg. 67-72.

Available online at www.anmb.ro

ISSN: 2457-144X; ISSN-L: 2457-144X

doi: 10.21279/2457-144X-22-008

Automatic determination of all Hamiltonian cycles in a completely undirected graph

Paul Vasiliu[1]

[1] "Mircea cel Bătrân" Naval Academy, Constanţa, România

E-mail: [1] paul.vasiliu@anmb.ro

Abstract. The Hamiltonian cycles in a completely undirected graph have applicability in various fields, among which we mention: computer networks, transportation, logistics, economy. In this paper the author presents an algorithm and a program written in the C ++ language for the automatic determination of all Hamiltonian cycles in a completely undirected graph with $n \geq 3$ vertices.

1. The Hamiltonian cycle in a completely undirected graph

Let the graph $G = (X, U)$ be completely undirected with n vertices.

A cycle is defined as simple chain in which the first vertex coincides with the last vertex. An elementary cycle is defined as a cycle in which the vertices are distinct.

The length of a cycle is equal to the number of edges of the cycle. The minimum length of a cycle is equal to 3.

An elementary cycle that contains all the vertices of a graph is called a Hamiltonian cycle.

An undirected graph that has a Hamiltonian cycle is called a Hamiltonian graph.

The problem of determining Hamiltonian cycles is also known as the traveler's problem. The issue of the travel agent is one of the most important issues in the field combinatorial optimization. This problem consists in calculating a turn in a weighted graph (ie a cycle that visits each node exactly once), so the sum of the weights of the edges in this circuit to be minimal.

The problem of the traveling salesman is known to be NP-strong, which means that none currently known algorithm does not find an optimal circuit in polynomial time.

Given a set of markings on a hardware board and the distance between each possible pair, the traveler's problem is to find the best possible way to visit all the markings (when we will make the holes on the board) and return to the point of departure that minimized travel costs (given the distance between the holes).

2. Generating Hamiltonian cycles in a completely undirected graph. Algorithm

Let's consider the completely undirected graph $G = (X, U)$ with n vertices, $n \geq 3$, where X is the set of vertices, $X = \{1,2, \cdots, n\}$ and U the set of edges.

To determine all Hamiltonian cycles we start from the Hamiltonian cycle formed by the first three vertices: (1 2 3 1).

By constructing the cycles formed by the first four vertices by inserting vertex 4 into the Hamiltonian cycle formed by the first three vertices, in all possible ways, are obtained three Hamiltonian cycles with four vertices: (1 4 2 3 1), (1 2 4 3 1) and (1 2 3 4 1).

The vertex 5 is then inserted into each of the four-pointed Hamiltonian cycles and the five-pointed Hamiltonian cycles are obtained.

Insert vertex 5 into the Hamiltonian cycle $(1\ 4\ 2\ 3\ 1)$ in four ways and obtain the Hamiltonian cycles: $(1\ 5\ 4\ 2\ 3\ 1)$, $(1\ 4\ 5\ 2\ 3\ 1)$, $(1\ 4\ 2\ 5\ 3\ 1)$ and $(1\ 4\ 2\ 3\ 5\ 1)$.

Insert vertex 5 into the Hamiltonian cycle $(1\ 2\ 4\ 3\ 1)$ in four ways and obtain the Hamiltonian cycles: $(1\ 5\ 2\ 4\ 3\ 1)$, $(1\ 2\ 5\ 4\ 3\ 1)$, $(1\ 2\ 4\ 5\ 3\ 1)$ and $(1\ 2\ 4\ 3\ 5\ 1)$.

Insert vertex 5 into the Hamiltonian cycle $(1\ 2\ 3\ 4\ 1)$ in four ways and obtain the Hamiltonian cycles: $(1\ 5\ 2\ 3\ 4\ 1)$, $(1\ 2\ 5\ 3\ 4\ 1)$, $(1\ 2\ 3\ 5\ 4\ 1)$ and $(1\ 2\ 3\ 4\ 5\ 1)$.

Do the same with the vertices $6, 7, \cdots, n$.

We will determine the number of Hamiltonian cycles with $n \geq 3$ vertices.

Let C_n be the number of Hamiltonian cycles of the completely undirected graph $G = (X, U)$ with n vertices, $n \geq 3$.

Because n vertices Hamiltonian cycles are obtained from $n-1$ vertices Hamiltonian cycles by inserting the n vertices in $(n-1)$ ways and the number of $n-1$ vertices Hamiltonian cycles is equal to C_{n-1} recurrence:

$$C_n = \begin{cases} (n-1) \cdot C_{n-1} & pentru \quad n \geq 4 \\ 1 & pentru \quad n = 3 \end{cases}.$$

Using this recurrence relation, the equalities are obtained:

$C_4 = 3 \cdot C_3$

$C_5 = 4 \cdot C_4$

……

$C_n = (n-1) \cdot C_{n-1}$

By computing the product member by member of these relationships the following equality results/ is obtained:

$C_4 \cdot C_5 \cdot \cdots \cdot C_{n-1} \cdot C_n = 3 \cdot 4 \cdot \cdots \cdot (n-1) \cdot C_3 \cdot C_4 \cdot C_5 \cdot \cdots \cdot C_{n-1}$

Simplify non-zero factors $C_4,\ C_5,\ \cdots, C_{n-1}$ and equality is obtained:

$C_n = 3 \cdot 4 \cdot \cdots \cdot (n-1) \cdot C_3$

Since $C_3 = 1$ and $3 \cdot 4 \cdot \cdots \cdot (n-1) = \frac{(n-1)!}{2}$ we obtain that $C_n = \frac{(n-1)!}{2}$.

We have thus shown that the number of all Hamiltonian cycles of the completely undirected graph $G = (X, U)$ with n vertices, $n \geq 3$ is equal to $\frac{(n-1)!}{2}$.

We prove this result by using the mathematical induction method.

For $n = 3$ we obtain $C_3 = \frac{(3-1)!}{2} = 1$, which is true.

For $n = 4$ we obtain $C_4 = \frac{(4-1)!}{2} = 3$, which is true.

For $n = 5$ we obtain $C_5 = \frac{(5-1)!}{2} = 12$, which is true.

The verification of the induction hypothesis is thus complete.

Let's suppose that $C_n = \frac{(n-1)!}{2}$ and will prove that $C_{n+1} = \frac{n!}{2}$.

Hamiltonian cycles with $n+1$ vertices are obtained from Hamiltonian cycles with n vertices by inserting the vertex $n+1$ in n modes in each of the $C_n = \frac{(n-1)!}{2}$ Hamiltonian cycles with n vertices.

It results that the number of Hamiltonian cycles with $n+1$ vertices is equal to

$n \cdot C_n = n \cdot \frac{(n-1)!}{2} = \frac{n!}{2}$ and therefore $C_{n+1} = \frac{n!}{2}$.

According to the principle of mathematical induction it results that: $C_n = \frac{(n-1)!}{2}$ for any $n \geq 3$.

It can be observed that the set of Hamiltonian cycles of the completely undirected graph $G = (X, U)$ with n vertices, $n \geq 3$, excluding the last vertex that coincides with the first vertex of each cycle, is a subset of the set of permutations of n objects.

By using this observation, the algorithm for generating the set of permutations of n objects can be used to generate Hamiltonian cycles.

For this purpose, we used the backtracking method to generate permutations of n objects and we kept those permutations that coincide with the Hamiltonian cycles.

3. Implementation in the C++ language

The program, written in the C ++ language, generates all Hamiltonian cycles from a completely unoriented graph with n vertices. The program writes all these cycles are written to a text file.

```
// Generate Hamiltonian cycles in a completely undirected graph with n nodes

#include <iostream>
#include <fstream>

using namespace std;

// Function for writing solution
void writesol(int n, int *x, int &nrsol)
    {
        int i;
        nrsol++;
        cout<<endl;
        cout<<" Cycle number "<<nrsol<<"     \t";
        x[n]=1;
        for(i=0; i<n; i++)
        cout<<" "<<x[i];
        cout<<endl<<endl;
    }

// Function for writing file
void writef(char *namef,int n,int *v, int nrsol)
    {

        int i;
        ofstream f(namef,ios::app);
        f<<" Cycle "<<nrsol<<"\t     ";
        for(i=0;i<n;i++)
        f<<v[i]<<" ";
        f<<endl;
        f.close();
    }

// Function for uniqueness test
int equal(int *x, int n)
    {
        int i,j,f=0;
        for(i=0;i<n;i++)
        for(j=i+1;j<n;j++)
        if(x[i]==x[j])
        f++;
        if(f==0)
        return 1;
        else
```

```
        return 0;
    }

// Continuous generation test function
int cont(int *x, int k)
    {
        int i;
        for(i=0;i<k;i++)
        if(x[i]==x[k] || x[0] != 1 || (x[i]>=x[k] && x[i]==3))
        return 0;
        return 1;
    }

// Function for insert vertice val
void insert(int n, int *v, int poz, int val)
    {
        int i;
        for(i=n;i>=poz;i--)
        v[i+1]=v[i];
        v[poz]=val;
        v[n]=1;

    }

// Backtracking function
void back(int n, int k, int *x, int &nrsol, char *namef)
    {
        int i,j,p;
        if(k==n)
        {
        for(j=1;j<n;j++)
        {
        insert(n,x,j,n);
        if (equal(x,n)==1)
        {
        writesol(n+1,x,nrsol);
        writef(namef,n+1,x,nrsol);
        }
        for(p=j;p<=n;p++)
        x[p]=x[p+1];
        }
        }
        else
        for(i=1;i<=n;i++)
        {
                x[k]=i;
                if(cont(x,k)==1)
                back(n,k+1,x,nrsol,namef);
        }
    }
```

```
// Main function
int main()
    {
        int n,i, x[500], nrsol=0;
        char namef[20];
        cout<<" Number of vertices = ";
        cin>>n;
        cout<<" File name : ";
        cin>>namef;
        back(n,0,x,nrsol,namef);
        cout<<" Number of Hamiltonian cycles "<<nrsol<<endl;
    }
```

The results generated by the program for $n = 4$ are:

```
Number of vertices = 4
File name : f.dat
Cycle number 1        1 4 2 3 1
Cycle number 2        1 2 4 3 1
Cycle number 3        1 2 3 4 1

 Number of Hamiltonian cycles 3
```

The results generated by the program for $n = 5$ are:

```
Number of vertices = 5
File name : f.dat
Cycle number 1        1 5 2 3 4 1
Cycle number 2        1 2 5 3 4 1
Cycle number 3        1 2 3 5 4 1
Cycle number 4        1 2 3 4 5 1
Cycle number 5        1 5 2 4 3 1
Cycle number 6        1 2 5 4 3 1
Cycle number 7        1 2 4 5 3 1
Cycle number 8        1 2 4 3 5 1
Cycle number 9        1 5 4 2 3 1
Cycle number 10       1 4 5 2 3 1
Cycle number 11       1 4 2 5 3 1
Cycle number 12       1 4 2 3 5 1

Number of Hamiltonian cycles 12
```

4. Conclusions and further developments

In this paper we have presented an algorithm and a program written in the C ++ programming language for generating all Hamiltonian cycles of a completely undirected graph. The program can be used for teaching purposes and in practical applications in various fields.

An important application of the travel commissioner's problem is that of making printed circuit boards.

The holes for mounting microprocessors, microchips and other electronic components are made automatically with drilling machines.

The holes have different diameters and require drills with such diameters.

The optimization of the drill change process is based on the problem of determining the Hamiltonian cycles in a completely undirected graph, a problem of the traveler.

The author intends to develop algorithms for optimizing the perforating process of printed plates based on the results of this paper.

References

[1] Horowitz E., Sahni S. - Fundamentals of Computer Algoritms, Computer Science Press, 1985.

[2] Knuth E., D. – Tratat de programarea calculatoarelor, vol. 1, Algoritmi fundamentali, Editura Tehnică, Bucureşti, 1974.

[3] Knuth E., D. - Tratat de programarea calculatoarelor, vol. 2, Sortare şi căutare, Editura Tehnică, Bucureşti, 1976.

[4] Stroustrup B.- The C++ Programming Laguage Ed.Addison – Wesley 1996.