Global Journal of Computer Science and Technology
Software & Data Engineering
Volume 12 Issue 14 Version 1.0 Year 2012
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

# Efficient Algorithm to Determine Whether a given Graph is Hamiltonian or not with all Possible Paths

By Narendra Pratap Singh, Ramu Agrawal & Indra Paliwal
*BSA College of Engineering and Technology, Mathura, U.P, India*

*Abstract -* Given a Graph G (V, E), We Consider the problem of deciding whether G is Hamiltonian, that is- whether or Not there is a simple cycle in E spanning all vertices in V. [1] However to Verify that the given cycle is Hamiltonian by checking whether it is permutation of the vertices of V and whether each of the consecutives edges along the cycle actually exists in the Graph. This Verification Algorithm can certainly be implemented to run in O (n2) time, where n is the length of the encoding of G [2]. But to predict in Advance that the Graph has Hamiltonian Cycle or not was still Exponential before this Algorithm. This Problem is known to be NP-Complete hence cannot be solved in Polynomial time in |V| unless P=NP. However till today there was no known Criterion we can apply to determine the existence Hamiltonian Circuit in General [3]. For its Exponential time We can Refer to theorems: - Vertex Cover problem is polynomially transformable to the Hamiltonian circuit Problem for Directed graphs, hence the Hamiltonian Circuit problem for Directed Graph is NP-Complete and the Hamiltonian Circuit Problem for Directed Graph is Polynomialy transformable to Hamiltonian Cycle Problem for Undirected Graph, hence the Hamiltonian Cycle Problem for undirected Graph is NP-complete [4]. Note that these derivations are based on the CNF- Satisfiability.

Through this Paper we have introduced a Newer Algorithm with different approach to determine whether a given Graph is Hamiltonian or Not with all possible Paths, by applying Few Mathematical and logical Operations. This provides necessary and sufficient condition for a graph to be Hamiltonian.

*Keywords :* *Adjacency matrix, Adjacency List, Nodes, Vertices, Edges, Hamiltonian circuit.*

*GJCST-C Classification :* *E.1*


*Strictly as per the compliance and regulations of:*

# Efficient Algorithm to Determine Whether a given Graph is Hamiltonian or not with all Possible Paths

Narendra Pratap Singh [α], Ramu Agrawal [σ] & Indra Paliwal [ρ]

*Abstract -* Given a Graph *G (V, E),* We Consider the problem of deciding whether *G* is Hamiltonian, that is- whether or Not there is a simple cycle in *E* spanning all vertices in *V.* [1] However to Verify that the given cycle is Hamiltonian by checking whether it is permutation of the vertices of *V* and whether each of the consecutives edges along the cycle actually exists in the Graph. This *Verification Algorithm* can certainly be implemented to run in $O(n^2)$ time, where *n* is the length of the encoding of *G* [2]. But to predict in Advance that the Graph has Hamiltonian Cycle or not was still Exponential before this Algorithm. This Problem is known to be NP-Complete hence cannot be solved in Polynomial time in |V| unless P=NP. However till today there was no known Criterion we can apply to determine the existence Hamiltonian Circuit in General [3]. For its Exponential time We can Refer to theorems: - Vertex Cover problem is polynomially transformable to the Hamiltonian circuit Problem for Directed graphs, hence the Hamiltonian Circuit problem for Directed Graph is NP-Complete and the Hamiltonian Circuit Problem for Directed Graph is Polynomialy transformable to Hamiltonian Cycle Problem for Undirected Graph, hence the Hamiltonian Cycle Problem for undirected Graph is NP-complete [4]. Note that these derivations are based on the CNF- Satisfiability.

Through this Paper we have introduced a Newer Algorithm with different approach to determine whether a given Graph is Hamiltonian or Not with all possible Paths, by applying Few Mathematical and logical Operations. This provides necessary and sufficient condition for a graph to be Hamiltonian.

*Keywords :* *Adjacency matrix, Adjacency List, Nodes, Vertices, Edges, Hamiltonian circuit.*

## I. Introduction

Hamiltonian Problem is Decision Problem in which *G (V, E)* should be traversed from any one vertex to same vertex without repeating any vertex again (means, Vertex should be traverse exactly once). We look for *n* long sequence of vertices $v_0, v_1, v_2, \ldots\ldots, v_{n-1}$ visit all vertices in *v* such that $0 \le i \le n$, *(vi, v(i+1)mod n)*$\in E$, along with the element of Adjacency Matrix *Ai, j= 1,if* $\forall E \in (i, j)$, *0,* otherwise. From the general prediction as prescribed in literature that Hamiltonian Cycle exists if and only if there is an *n*- long tour that cover all the vertices and returns to the standing point.

Scientist around the globe deduced the Method based on the number of edges and degrees of graph, some for planarity and some for connected but they all failed for a general graph and was not sufficient. It follows the CNF- satisfiability also [4].

But we put through the above statement from the mathematical and logical point of view.

By this algorithm, now the scientist will have reasonable condition to determine the Hamiltonian Circuit in Advance without traversing it vertex to vertex manually on the paper.

Till today this problem which spurred the computer scientist around the globe to be able to draw an Algorithm which Culminate the possibilities, the usage of global information was shown to speed up the process: however it has cost in communication and complexity of individual agent. Now in our Algorithm there is no foundation for an undirected, directed, planarity, colorability, and connectedness of a graph, it can be applied to the all types of graphs.

Rest of the paper is organized as follows. Section2 present the related work. The proposed method algorithm has been described in section3. In section4, experimental results and sample run have been presented and paper is concluded in section5.

## II. Related Work

Since, its (Hamiltonian Cycle Problem) origin, by famous Irish Mathematician Sir William Rowan Hamilton, 1859, was still unsolved. There was no known criterion we could apply to determine the existence of Hamiltonian circuit in general. A circuit is a connected graph *G* is said to be Hamiltonian if it includes every vertex of *G.* Hence a Hamiltonian Circuit in a Graph of *n* vertices cost of exactly *n* edges. Obviously, not every connected Graph has Hamiltonian Circuit. For example, neither of the Graph shown in figures (2.1 and 2.2) and has a Hamiltonian circuit. This raise the Question: What is the necessary and sufficient condition for a connected Graph G to have Hamiltonian Circuit? [5]

Also, No known Characterization to determine Hamiltonian graph in any given Graph G has been found [6].

*Author α σ ρ : Department of Computer Science, BSA College of Engineering and Technology, Mathura, U.P, India.*
*E-mail α : narendrapratapbsa@gmail.com*
*E-mail σ : ramuagrawalbsa@gmail.com*
*E-mail ρ : indrapaliwal@gmail.com*

However several Scientist has proposed several methods on the basis of degree and edges with the reference to any specific graph (like connected, planarity, etc.). But they had not found full success with necessary condition to predict Hamiltonian cycle in advance for every Graph. Most of works has been presented before the 1975. Hence there was no programming based algorithmic approach had been considered? Some famous works are as follows:

- Every Graph G with $n \geq 3$ vertices and minimum degree at least $n|2$ has a Hamiltonian Cycle [7] (Dirac 1956).

[Note that this theorem bound prediction within limit of $n \geq 3$.]

- Every Graph G with $/G/ \geq 3$ and $K(G) \geq \alpha(G)$ has a Hamiltonian Cycle.

[Note that the theorem is bounded within condition $/G/ \geq 3$[7](Dirac 1956).

- Every 4- Connected planar Graph has a Hamiltonian Cycle. (Tutte 1956)[8, 9, 10]
- Historically, Dirac's Theorem formed the point of departure for the discovery of a series of weaker degree conditions, all sufficient for Hamiltonian circuit. The development of our algorithm culminates all the theorem s and encompasses all the earlier results.

If G is graph with n vertices and degrees $d_1 < d_2 < \ldots\ldots < d_n$, then the n-tuples $(d_1, d_2, \ldots, d_n)$ is called the degree of sequence of G. Note that this Sequence is unique, even though G has a several vertex enumeration giving emphasis to its degree sequence $(a_1, a_2, \ldots\ldots, a_n)$ Hamiltonian if Every Graph with $n$ vertices and a degree sequence point wise greater than $(a_1, \ldots . a_n)$ *if* $(d_i \geq a_i$ *for all i)*

The following theorem characterizes all Hamiltonian Sequences.

- (Chvatal 1972), An integer Sequence $(a_1, \ldots, a_n)$ such that $0 \leq a_1 \leq \ldots\ldots \leq a_n < n$ and $n \geq 3$ is Hamiltonian if and only if the following holds for every $i < n/2$:

$$a_i \leq i \Rightarrow a_{n-1} \geq n\text{-}i.$$ [11]

- An integer sequence (a1, …an) such that $n \geq 2$ and $0 \leq a1 \leq \ldots\ldots \leq an < n$ is path Hamiltonian if and only if every $i \leq n/2$ is such that $a_i < i \Rightarrow a_{n+i-2} \geq n\text{-}1$ Hamiltonian Cycle in the square of a graph
- (Fleischner 1974), if $G$ is a 2- Connected graph then $G^2$ has a Hamiltonian Graph. [11]
- (Seymour 1974), let $G$ be a Graph of order $n \geq 3$, and k be positive integer. If $G$ has a minimum degree $\delta(G) \geq (k/k+1)*(n)$, there $G$ has Hamiltonian Cycle $H$ such that $H^k \subseteq G$. [12].

For $k=1$, this is preciselyDirac's theorem the case $k=2$ had already been conjecture by Posa in 1963 and was proved for large n by kamlos, Sarojy & Szemerdi, 1996.[13]

Beyond the above given thesis, the age comes to programming and computer scientist developed several algorithms in the same context but did not get succeed to make sure exact prediction for a graph to be Hamiltonian or not. These are given as follows:

- Frieze [14] introduced a heuristic polynomial – time algorithm, Ham, for finding Hamiltonian cycle in random graphs with high Probability.

[Note that High probability terms indicate not to be fully assured for being a Graph, Hamiltonian.

- Improved version of Ham, Semi Ham [15] by keydar.
- Vandegrind analyzes the knight's tour problem in some details. He gave several existence and Non-existence theorems for different parameters values. He also reports on existence compute Experiments.
- An inspired Heuristic for regonizing Hamiltoion graphs, by Israel A. Wagner.

From IBM Hafia research lab, Matan, Hafia, Dept. of C.S. Technion City, Haifa 32000, Israel and Alfred M. Bruckstein from AT &T Bells at Murray Hill NJ0794, USA. Remain some challenging Question.

1. Vertex ant walk (VAW) has the Hamiltonian cycles as its limit cycle; however we do know if those are the only limit cycles of the process which are longer than the n variables.
2. A probabilistic Version of VAW rule does not determine the next neighbor specifically, but assigns each neighbor a probability according to its current ($\mu$, t) mark (e.g) the probability of jumping from u to v be

$$Prob(u \rightarrow v) = (1/(1+\mu(v)))/(\sum_{w \in N(u)}(1/(1+\mu(w))$$

Where, clearly, $\sum_{w \in N(u)} prob\ (u \rightarrow w) = 1N(u)$ stands for the sets of vertices $v \in V$ such that $(u, v) \in \sum$ ). In such a semi-Random process faster, (In recognizing a Hamiltonian Graph), on the average, or lower than the deterministic one?

[Note that, the prediction in above method is probabilistic, means of having some amount of Uncertainty.]

- Solving the Hamiltonian cycle Problem using symbolic determinant by V.Ejov[1], J.A.Filar[2], S.K Lucas[3]& J.L. Nelson[4](1, 2,3 school of mathematics and srtatistics, University of south Australia, Dept. of mathematics, Harvey mudd college ,1250 N. Dartmouth Ave Clare mount CA91711 USA. has applied Algebra but Complexity remain for this algorithm is Exponential.
- *Bitonic Euclidean traveling-salesman problem*[2]

The *Euclidean traveling-salesman problem* is the problem of determining the shortest closed

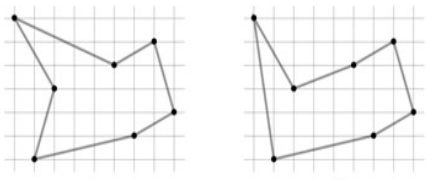
Tour that connects a given set of *n* points in the plane Figure 2.1(a) shows the solution to a 7- point problem. The general problem is NP-complete, and its solution is therefore believed to require more than polynomial time

Figure 15.9: Seven points in the plane, shown on a unit grid. *(a)* The shortest closed tour, with Length approximately 24.89. This tour is not bitonic. *(b)* The shortest bitonic tour for the same set of points. Its length is approximately 25.58.

J. L. Bentley has suggested that we simplify the problem by restricting our attention to *bitonic tours*, that is, tours that start at the leftmost point, go strictly left to right to the rightmost point, and then go strictly right to left back to the starting point. Figure (b) shows the shortest bitonic tour of the same 7 points. In this case, a polynomial-time algorithm is possible.

Describe an $O(n^2)$-time algorithm for determining an optimal bitonic tour. You may assume

That no. two points have the same *x*-coordinate. (*Hint:* Scan left to right, maintaining optimal Possibilities for the two parts of the tour.)



(a) (b)

But, we have introduced a Newer Algorithm which over comes all the Hurdles suggested in above given recent research, by applying logical and mathematical operations. Hence it proves itself as a sufficient condition for a Graph to be Hamiltonian.

## III. The Proposed Method

In this section, we will present our original approach to determine all Hamiltonian paths for a Given Graph G. G(V, E) is the ordered pair consists two sets, V for vertices(v1, v2, v3, …., vn) and E for edges (e1, e2, e3, …., en)[or ek∈E, k∈ $N$ . the vertices are the basic nodes types which stores the information of Graph, itself a mathematical structures and finds its application in many areas of interest in which problem need to solved using Computers. In case of Hamiltonian, e.g., Indian Railways may need to expand its tracking belongs to each station in such a manner that the train running on this must visit each station exactly once by starting from any station and reached at the same. It represented in the Data Structures. These representations commonly used Adjacency Matrix, Adjacency list and Multi- list representation. Firstly, we will use these for Adjacency Matrix $A_{ij} = 1$, if $\forall e \in (vi \Leftrightarrow v_j)$, $A_{ij} = 0$, otherwise, takes $O(n^2)$ space to represent graph with n vertices, even for Sparse Graph.

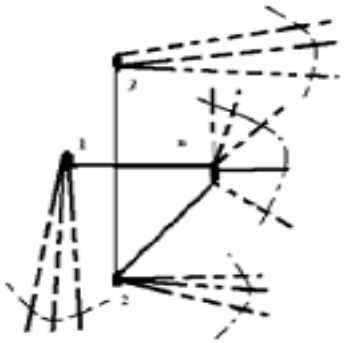
Algorithm *(for First Basic method*)*:

1. Draw the Adjacency matrix ( Inputting graph):

Let take adjacency matrix for any undirected graph without parallel edges or loops.

| | [1] | [2] | [3]…… | [n] |
|---|---|---|---|---|
| [1] | 0 | 0 | 1 …… | 0 |
| [2] | 0 | 0 | 1 …… | 1 |
| [3] | 1 | 1 | 0 …… | 0 |
| . | . | . | . …… | |
| . | . | . | . …… | |
| . | . | . | . …… | |
| [n] | 1 | 1 | 0 …… | 0 |

*Fig. 3.1 :* n*n- matrix for undirected graph

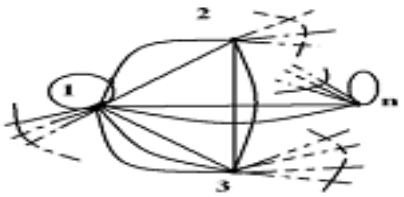


Certainly, it would represent a Graph, connected, planar, directed, and undirected or of having some self loops and parallel edges to put p∈W, (p, no. of edges)

Adjacency matrix for directed/undirected having loops and parallel edges.

| | [1] | [2] | [3] | ……..[n] |
|---|---|---|---|---|
| [1] | 1 | 2 | 3 | …….. 2 |
| [2] | 2 | 0 | 2 | …….. 0 |
| [3] | 3 | 2 | 0 | …….. 0 |
| . | . | . | . | …… |
| . | . | . | . | …… |
| [n] | . | . | . | ……. 1 |

*Fig. 3.2 :* n*n matrix for graph having loop and parallel edges



2. Make function for traversing from vertex to vertex:

For a completely connected graph there will be *n!* Path (possible arrangements of all the vertices), which provides highest probability to find out the Hamiltonian paths, hence for the cases having parallel edges the path increases by the factor of *2*. Let take any

graph, fig (3.1), if we make all possible arrangements of vertices as paths.

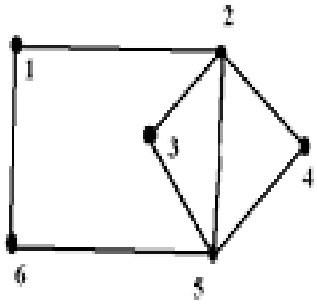



*Fig. 3.3 :* Undirected graph for without Hamiltonian

1-2-3-4-5-6-1
1-2-3-4-6-5-1
1-2-3-5-4-6-1
1-2-3-5-6-4-1
.
.
.
.
Up to *6!* (Possible constructions)

However the path of the fashion of that arrangement may not be occurred, but our main concern is to make them.

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 |
| 2 | 1 | 0 | 1 | 1 |
| 3 | 0 | 1 | 0 | 0 |
| 4 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 1 | 1 |
| 6 | 1 | 0 | 0 | 0 |

Observe carefully, in the case of Hamiltonian the possibility to move on from one vertex[i] to another vertex [j] occurs only when if the $[i][j]_{th}$ element of matrix is 1(one). If the value of [i][j]th element in adjacency matrix is 0(zero) then it halt the path to proceed.

In the Succeeding way if all the values in the given path is 1 in the reference of [i][j]th element therefore it has Hamiltonian path because in the path (say) 1-24-5-3-6-1, all the element of matrix([1][2], [2][4], [4][5], [5][3], [6][1] are 1(one), except [3][6]. If this value becomes one the path will be successfully turn into Hamiltonian circuit.

Let us take a Hamiltonian Graph

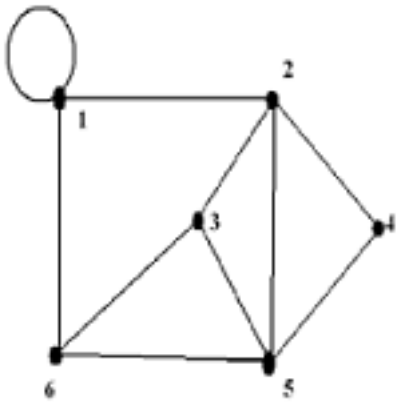



*Fig. 3.4 :* Undirected graph with parallel edges and loops

Adjacency matrix for fig (3.4)

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 0 | 2 |
| 2 | 1 | 0 | 1 | 1 | 1 | 0 |
| 3 | 0 | 1 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 | 1 | 0 |
| 5 | 0 | 1 | 1 | 1 | 0 | 1 |
| 6 | 2 | 0 | 1 | 0 | 1 | 0 |

We observe carefully for path (say) 1-2-4-5-3-6-1, all the element of adjacency matrix [1][2], [2][4], [4][5], [3][6], [5][3], [6][1] have some value *k, k*$\in W$ ,whole numbers.

A. We enumerate a logic from this, let take the array elements in the specific sequences (generated by permutation, all possible paths, drawn by *interchanging array function*).that to put *logical AND (&&)* between these elements. Interchange them for above given sequences.

*If* $P_k$ = *[1][2]&&[2][4]&&[4][5]&&[4][5]&&[5][3]&&[3][6]&&[6][1] = 1*

*There, Hamiltonian circuit exists.*

*Else*

*Not exist.*

Functionally in C programming, we have done this as:

```
Visit (int*, int, int)
Visit (int*NODES, int N, int k)
{
Static level= -1;
Level =level+1;
NODES[k] =level;
If (level == N)
{
 Int j=0, p=1, lock=0;
 While (j<N-1)
P = GRAPHIN [NODES[j]}[NODES[j+1]&&P;
J++
}
P=GRAPHIN [NODES][j] NODES[j+1]}&&P;
```

Do the same operation for all the sequences, we get $P_1$, $P_2$, $P_3$,…,, $P_n$.

B. Then we apply logical OR(||) operator between all the path values.

*$P = P_1//P_2//P_3//\ldots\ldots, //P_n$*

*If P = 1*

*Then it is Hamiltonian Graph*

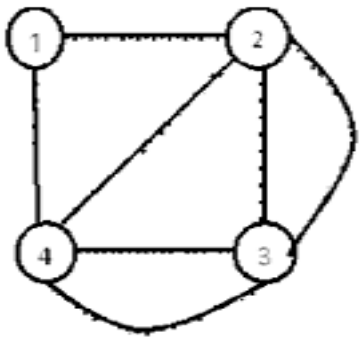
*Else*

*Not* Print the Paths:

In this program it gets prints by the function

Input Adjacency Matrix:

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 |
| 2 | 1 | 0 | 2 | 1 |
| 3 | 0 | 2 | 0 | 2 |
| 4 | 1 | 1 | 2 | 0 |




*fortunately this algorithm provides sufficient conditions for a Graph to be Hamiltonian and also print all the possible paths but Unfortunately takes more time for large Graphs, so we were in search for next Algorithms which take less time than this, therefore we developed second Algorithm which facilitate all the requirements.

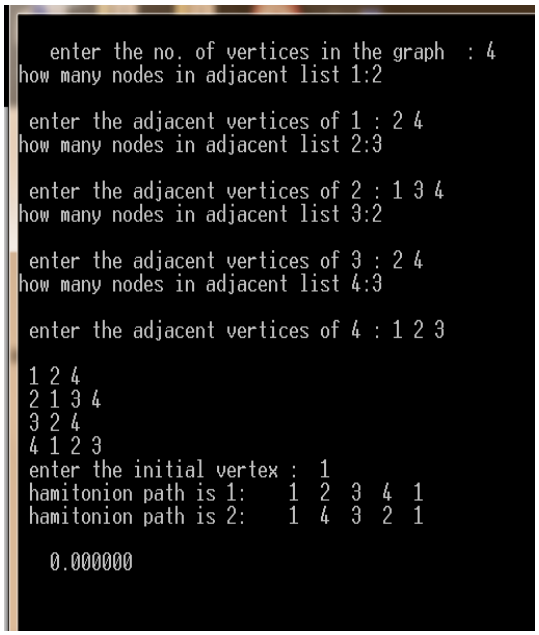
We prepare second Algorithm using the concept of Graph theory, means of having linked list, structures and few pointers.

Second Algorithm:

*Pseudo code for second Algorithm:*

```
Algorithm Hamiltonian path (vertex adj[ ], N, K, a[ ])
{
Create_ham(adj, p → vertex, N)
Print_ham(a, N)
Count: = 0;
If count= N then
{
p: = adj[K]; p:= next;
While p≠ NULL do
   {
      If a[1] = p →vertex than write (a, N); break;
   }
   P: = p →next;
}
Else
{
   P: =adj[K]; p: =p →next;
   While p ≠NULL
    {
         Loc: =0;
         For j ←1 to count do
         {
              If a[j]=p →vertex then
              {
                   Loc++; break;
               }
         }
         If loc=0 then
       {
          Count++;
a[count]:=p →vertex;
create_ham(adj, N, p →vertex)
              count--; a[count]:=0;
        }
        P: = p →next;
  }
 Print_ham(a,N)
 {
      For i ←1 to N do
       Write(a[i]);
Write(a[1]);
}
}
```

Output



## IV. Experiment Result

Let an Example of the Wide Network of Airlines, in which there are *n* Airports, as in the given figure you may assume each node as an Airport and the edges as the route of Airline from one to another.

Hence this will form a type of Adjacency list (linked list), in which each airport has the knowledge of incoming and outgoing flights in various directions. In the given figure ( ), we take *Home Node* A (means, the Plane should have to back this again after traversing all the airports exactly once). You may consider any one airport as a Home Node, in our Algorithm.

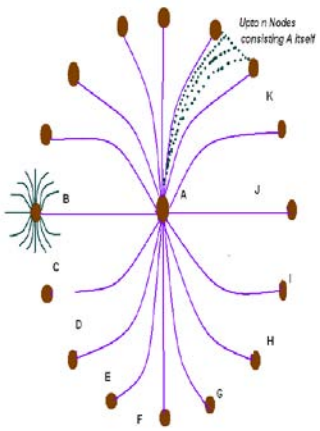



*Figure :* Graph showing Airlines connections

In the general analysis, take this example of Airline system as a Strongly Connected Complete Graph, means each and every Airport is connected to all other Airports.

T (*n*) for Best Case:

Let start from the node A, the Adjacency list will seem like this

$$A \rightarrow B \rightarrow C \rightarrow D.....$$
$$B \rightarrow A \rightarrow C \rightarrow D.....$$
$$C \rightarrow A \rightarrow B \rightarrow D.....$$
$$.....$$
$$.....$$

If all nodes are reachable from A, than flights may seek to any Airport, Note that this step will make just one comparison and proceed to next one (say, B).

On the next point, it will decide to move on the next Airport Except the later traversed Airport (A). If this node is reachable to another then it will move on by making just single Comparison again. This sequence of comparison at one time and moving on the next node will remain next Node will remain continue till it did not reach at the initial vertex (Airport).

$$T(n) = 1+1+1+.....+1 \quad (n \text{ times})$$

$$T(n) = n$$

T (*n*) For Worst Case:

In the worst case, the Airplane will choose the move to specific Airport by making (n-1) comparison for itself; therefore the airplane at the next step will decide the next move to further proceeding after making n-1 comparison. The routine will remain till the Airplane did not reach at the initial Airport (or Standing point).

$$= n\,(n-1)!$$

Hence, this Algorithm runs near to polynomial for small vertices sets.

## V. Conclusion

Our method exploits the ways to find Hamiltonian Circuit using Sequential and predicate logic and opens up opportunities for future researcher interested in this problem succinctly in much advance way. However our method has successfully find out all possible paths but still working near to polynomial not exactly polynomial, but it opens opportunities to think about this problem by applying advance methods of predicate calculus to find out all possible paths in polynomial time.

## References Références Referencias

1. Israel A. Wagner, Alfred M. Bruckestein, An Ant Inspired Heuristic for Recognizing Hamiltonian Graphs, 0-78-3-5536-9/99/1999 IEEE.
2. T. H. Cormen, C. E. Leiserson and R. L. Rivest, Introduction to Algorithms, MIT Press Cambridge, MA, 2009.
3. Narsingh Deo, Graph Theory with Application to Engineering and Computer Science, Prentice Hall India, pp. 20-32, 20007.
4. Alfred V. Aho, John E. Hopcroft, Jefferey D. Ulman, The Design and Analysis of Computer Algorithms, pp.387-393, 2008.
5. Ellis Horowitz, Sataj Sahni, Sanguthevar Rajshekharan, University press 2nd Edition 2009.
6. Reinhard Diestel, Graph Theory Electronic edition, 2000, Springer-verlog, New york, 2000, pp.222-237.
7. B.Bollobas, External Graph Theory, Academic Press 1978.
8. R.Halin, Graph theorie, Wissen Schatliche Buchgessellschaft, 1980.
9. C.Thomassen, J.Graph Theory7 (1983), pp.169-176.
10. J.A.Bondy & U. S. R. Murthy, Graph Theory With Applications, Macvmillan 1976
11. Fleischner, J. Combin Theory (1991), 117-123.
12. P. D. Seymour, Problem3, in (T. P. Mc Donough and V.C. Marron, OD) Combinatorics, Camridge University Press 1974.
13. J. Komlos, G. N. Sarkozy & E. Szemredi, Random Structures and Algorithms, 1976, pp. 193-211.

14. B. Bollobas, T. I. Fenmer and A.M. Frieze, An Algorithm for finding Hamiltonian Paths and Cycles in Random Graphs, Combinatorica,7(4), pp. 327-341, 1987.
15. Eram keydar, Finding Hamiltonian cycle in Semi random Graphs. Mastros Thesis, Weizerman Institute of Science, 2002.

This page is intentionally left blank