EasyChair Preprint

№ 1416

Probabilistic Algorithm for Finding a Hamiltonian Path/ Cycle in a Graph

Rama Murthy Garimella, Vihan Shah and Dhruv Srivastava

August 24, 2019

PROBABILISTIC ALGORITHM FOR FINDING HAMILTONIAN PATH/CYCLE IN A GRAPH

A PREPRINT

Prof. Garimella Rama Murthy
Department of Computer Science
Mahindra École Centrale
Hyderabad, IND 500043

Vihan Shah
Department of Computer Science
Mahindra École Centrale
Hyderabad, IND 500043

Dhruv Srivastava
Department of Computer Science
Mahindra École Centrale
Hyderabad, IND 500043

July 20, 2019

ABSTRACT

In this research paper, a novel probabilistic algorithm for finding a Hamiltonian Path/ Cycle in a graph is discussed. Thus, a probabilistic polynomial time algorithm (PP class) for finding Hamiltonian path/cycle is proposed.

1 Introduction

Graph theory, as a research field was developed due to the efforts of pure and applied mathematicians. In many applications in Science as well as engineering, various problems related to graphs naturally arise. Researchers in computer science formulated various interesting graph problems and designed efficient algorithms to solve such problems. Several open research problems associated with graphs are also formulated by computer science researchers. One of the interesting open problems is to find efficient algorithm for determining existence of Hamiltonian path/cycle in a graph and finding one such path. This research paper proposes a probabilistic algorithm to find a Hamiltonian path / cycle in a graph if one such path/cycle exists in such a graph. This research paper is organized as follows. In Section 2, the related research literature is reviewed. In Section 3, a novel algorithm for finding Hamiltonian Path/Cycle in a graph is proposed. The research paper concludes in Section 4.

2 Review of Research Literature:

We begin with the definition of a Hamiltonian path/cycle.

• Definition: Hamiltonian path is a graph path in an undirected or directed graph between two vertices of the graph that visits each vertex exactly once [1], [2] .

• Definition: If a Hamiltonian path exists whose end points are adjacent, then the resulting graph cycle is called a Hamiltonian cycle.

• Definition: A graph that possesses a Hamiltonian path is called a Traceable Graph.

Note: In general, the problem of finding a Hamiltonian path in a graph is NP-complete [1].

3 Claims

We consider undirected graphs (weighted or not) and propose a novel probabilistic algorithm for finding a Hamitonian path/cycle if one such exists. We expect such algorithm to be modified for directed graphs (weighted or not). We are proposing 3 claims regarding existing of a Hamiltonian path in a graph G.
The claims are given as follows:

CLAIM 1

"Consider any Graph G, that has a degree 1 vertex u. If there exists a hamiltonian path in G then it either starts or ends at u."

PROOF : BY CONTRADICTION

Consider a Graph G, with a degree 1 vertex u and a Hamiltonian path H such that u is not an endpoint in the path H. The path H looks like:

$$V_1, V_2, V_3 ... V_k, u, V_{k+1}, ... V_n$$

u has to be in the path H since it is hamiltonian. And *u* is connected to V_k and V_{k+1}. $Thus, deg(u) >= 2$.

But we know that deg(u) is 1 and thus we arrive to a contradiction. Thus any degree 1 vertex should be the start or end point of a Hamiltonian path.

CLAIM 2

"Consider any Graph G, with strictly greater than 2, 1 degree vertices u1 , u2, u3 ... uk where k>2. G cannot have a Hamiltonian path."

PROOF : BY CONTRADICTION

Consider Hamiltonian path H in G. Using Claim 1 we can say that

$$u_1, u_2 ... u_k$$

are all either start or end points of H. Using Pigeon Hole Principle, we can say that there are more than 1 start or end points, which is not possible and hence we arrive on a contradiction that G cannot have a Hamiltonian Path. Complexity to check Claim 1 and Claim 2.

Case 1: Adjacency List
We need to go through each vertex and check if it is degree 1. O(1) for each vertex and O(V) for V vertices.

Case 2: Adjacency Matrix
We need to go through each position in the matrix in the worst case O(V^2) for V vertices.

CLAIM 3

"Consider graph G (V,E). The set V can be partitioned into V_1 ,V_2, ... V_k, U where k>=3 and V_k is non-empty and U=u. There does not exist any edge from any node in V_i to any node in V_j for all i,j where i not equal to j. There exists at least 1 edge from a node in Vi to a node in U for all i. G can not have a hamiltonian path."

PROOF : BY CONTRADICTION

Case 1:
H starts from u. The next vertex has to be a part of some Vi . Now no vertex in Vj is reachable where i not equal to j since the only path from a vertex in Vi to a vertex in Vj includes u and u is already covered in H. Thus H can not exist.

Case 2:
Without loss of generality H starts from V1. 'u' has to be the first vertex in H which is not in V1. After visiting u we go to some vertex in V2. Now no vertex in V3 can be reached since the only path from a vertex in V2 to a vertex in V3 includes u and u is already covered in H. Therefore all vertices can not be covered and thus H-path can not exist.

Complexity to check Claim 3

For each vertex do the following:

Remove the vertex and check number of connected components by running a BFS. If at any stage the number of connected components are more than 2, we can say that H-path can not exist.

Case 1: Adjacency List

We need to remove each vertex once O(1) for removing a vertex, (maybe set a flag) O(V + E) for BFS $O(V^2 + V * E)$ in total.

Case 2: Adjacency Matrix

We need to remove each vertex once O(1) for removing a vertex, (maybe set a flag) $O(V^2)$ for BFS $O(V^3)$ in total.

Theorem: ***" While carrying out the procedure given above in graph G, if at any point by removing a vertex V we find 2 connected components C1 and C2 then we can split the path finding problems into 2 sub-problems. The sub-problems are finding a Hamiltonian Path separately in each of the connected components, ending at V."***

PROOF: If there is a Hamiltonian Path H in G then V has to be a part of H. 1) V cannot be an endpoint in H. WLOG H starts in C_1. If H ends at V no vertices in C_2 can be covered thus H is not a Hamiltonian path. Thus V is not an end point in H. WLOG H starts in C1 and ends in C2 with V somewhere in the middle. H looks like this U_1, U_2 ... U_k, V, W_1, W_2 ... W_m. Where U_i belongs to C_1 and W_j belongs to C_2.

Thus we can break H into 2 parts U_1, U_2 ... U_k and W_1, W_2 ... W_k. Where U_i belongs to C_1 and W_j belongs to C_2. Thus we can split the path finding problems into 2 sub-problems, finding a path in C_1 ending at V and finding a path in C_2 ending at V.

COMPRESSION

Given graph G to find a graph G* such that G has a Hamiltonian Path if and only if G* has hamiltonian path and G* has lesser number of vertices (a sense of merging the vertices without changing the Hamiltonian property).

Theorem: ***"In a graph G, there is a vertex V of degree 2 having neighbours V1 and V2 and V1 is a degree 2 vertex. Consider graph G* which has the same vertices and edges as G except vertex V is removed and there is an edge between V1 and V2. G has Hamiltonian Path iff G* has Hamiltonian Path. In other words a 2 degree vertex can be compressed."***

PROOF:

Path in G => Path in G*

CASE 1: V is in the middle Consider a path H in G. H looks like U_1, U_2 ... V_1, V, V_2 ... U_k. A path H* in G* will look like U_1, U_2 ... V_1, V_2 ... U_k.

CASE 2: V is an endpoint Consider a path H in G. H looks like U_1, U_2 ... V_2 ... V_1, V A path H* in G* will look like U_1, U_2 ... V_2 ... V_1 Path in G* => Path in G

CASE 3: V1-V2 is in the Path Consider a path H* in G*. H* looks like U_1, U_2 ... V_1, V_2 ... U_k. A path H in G will look like U_1, U_2 ... V_1, V, V_2 ... U_k.

CASE 4: V1-V2 is not in the Path Consider a path H* in G*. H* looks like U_1, U_2 ... V_2 ... V_1 A path H in G will look like U_1, U_2 ... V_2 ... V_1, V Thus every degree 2 vertex who has a neighbour as a degree 2 vertex can be compressed without losing the Hamiltonian path property. This means that all degree 2 nodes in a cycle can be compressed into just 1 node.

4 Novel Probabilistic Algorithm for Finding a Hamiltonian Path/Cycle in a Graph:

To describe our algorithm, we need the following definitions related to graphs [2].

• *Definition: A Spanning Tree of a graph, is a tree connected subgraph of the graph, whose vertex set consists of all the nodes of the original graph. In the case of weighted graph, the weight of spanning tree is the sum of its edge weights.*

• *Definition: Line connected Spanning Tree of a graph is a spanning tree all of whose vertices, except root and leaf nodes have a vertex degree of exactly 2 (i.e. spanning tree topology is a LINE). It is also a spanning tree which is also a PATH in the graph.*

• *Definition: Minimum Spanning Tree of a weighted graph is the spanning tree of weighted graph whose weight is the smallest among (among all such spanning trees of the graph) all possible spanning trees of the graph.*

Note: It is well known that computing the Minimum Spanning Tree (MST) of a weighted graph is an important problem. Various efficient algorithms are designed for such a problem. For instance, Kruskal's greedy algorithm is an efficient example algorithm.

We now consider an undirected, weighted / unweighted graph.

4.1 GOAL

Our goal is to determine whether a Hamiltonian path/cycle exists or not in such a graph. If such paths exist in the graph, we would like to find atleast one of them.

• Essential Idea for Probabilistic algorithm:

Consider the graph under consideration. If it is weighted graph, set all the edge weights to one (or a constant value). In such a graph, using Minimum Spanning Tree (MST) algorithm, compute the Spanning Tree of the graph which is a PATH i.e. Line Connected Spanning Tree (defined above) if one exists. There is a positive probability that such a spanning tree is found. If such a spanning tree is found, the graph has a Hamitonian Path. Also, if the root and leaf vertices of such Line Connected Spanning tree are adjacent, the graph has a Hamiltonianm cycle.

Note: It can be readily reasoned that for a graph to contain a Hamitonian path, a necessary and sufficient condition is that it has atleast one line connected sparsest spanning tree.

Description of Algorithm

Step 1: Consider the graph under consideration. If it is a weighted graph, set all the edge weights to one. Compute Minimum Spanning Tree using, say Modified "Probabilistic" Kruskal's / Prim's algorithm (may be there are more than one such spanning trees).

Step 2: Determine if the spanning tree is a line connected spanning tree by checking the degree of vertices of such tree connected sub-graph (of original graph). If yes, the original graph has a Hamiltonian path. If the root and leaf vertices of such spanning tree are adjacent, the original graph has a Hamiltonian cycle.

Note: The number of spanning trees is super-polynomial in the number of vertices of the graph.

Note: Any Minimum Spanning Time (MST) algorithm (Such as Kruskal/Prim's algorithm) finds a line connected spanning tree (i.e. Hamiltonian Path) with positive probability (however small). Thus the algorithm is in the Probabilistic Polynomial Time (PP) class. The idea is to choose weights on the edges (assuming unweighted graph) such that the MST algorithm finds a Hamiltonian path with a high probability.

We now provide details about the probabilistic algorithm. The ultimate goal is to arrive at such an algorithm which maximizes the probability of finding a Hamiltonian path. We first consider the type of greedy algorithm for computing a Minimum Spanning Tree (MST) in a graph. With such an algorithm, if the weights on all edges is set to one, greedy choice of an edge (leading to a spanning tree) can be assumed to pick one among such edges incident at a vertex with equal probability. We now compute interesting quantities associated with such a probabilistic algorithm to find a Hamiltonian path. We first consider structured unweighted graphs:

4.2 CLIQUE (Fully Connected Graph without self loops at every vertex)

It is well known that the total number of spanning trees in such a graph is $n^{(n-2)}$. Hence a naïve estimate of probability of finding a Hamiltonian path is $1/n^{(n-2)}$ (if all spanning trees are equally likely to be Hamitonian paths). But we realize that an intelligently designed probabilistic greedy algorithm will lead to a better probability of finding a Hamiltonian path.

Simple reasoning shows that (with such a greedy algorithm) the number of possible Hamiltonian paths is

$$(n-1)(n-2)....(3)(2) = (n-1)!$$

If all these paths are chosen with equal probability, then Probability of Finding a Hamiltonian Path

$$1/(n-1)!$$

Note: We would like to consider more general graphs and evaluate the performance of probabilistic Greedy MST algorithm. Let v_{i_1} denote the vertex i_1. Also, let $d(v_{i_1})$ denote the degree of such vertex. We find the probability of finding a Hamiltonian path starting at vertex i_1 and ending at vertex i_n.

Using probabilistic greedy algorithm, we have probability of finding a Hamiltonian Path = q, where

$$q = \frac{1}{d(v_{i_1})}\frac{1}{d(v_{i_2})-1}...\frac{1}{d(v_{i_k})-m_{i_k}}...\frac{1}{d(v_{i_n})-m_{i_n}}$$

where m_{i_k} is the number of edges in the path so far that are one hop neighbours of vertex i_k.

4.3 Bounds on Probability of Finding a Hamiltonian Path :

We now compute the maximum and minimum attainable values of utilizing a probabilistic greedy Minimum Spanning Tree algorithm. To arrive at such probabilities, we need to order the vertex degrees (i.e. number of vertices directly connected to the vertices). We let

$$d(v_{i_1}) \geq d(v_{i_1}) \geq ...d(v_{i_1}) \geq ...d(v_{i_1})$$

Thus,

1. Maximum attainable probability of finding a Hamiltonian Path, P_H starting at vertex i_n and ending at vertex i_1.

$$P_H = \frac{1}{d(v_{i_n})}\frac{1}{d(v_{i_n})-1}...\frac{1}{d(v_{i_k})-m_{i_k}}...\frac{1}{d(v_{i_1})-m_{i_1}}$$

2. Maximum attainable probability of finding a Hamiltonian Path, P_H starting at vertex i_1 and ending at vertex i_n,P_L.

$$P_H = \frac{1}{d(v_{i_1})}\frac{1}{d(v_{i_2})-1}...\frac{1}{d(v_{i_k})-m_{i_k}}...\frac{1}{d(v_{i_n})-m_{i_n}}$$

It should be realised that the m_{i_k} values in (1) (i.e. P_H computation) are different from those in (2) (i.e. P_L computation).

Note: From the above probabilities, we can also estimate bounds on Missing detection of a Hamiltonian path. Given the above discussion, we have the following broader goal by an intelligent choice of weights on the edges of unweighted graph.

4.4 BROADER GOAL

By a suitable choice of weights (on the Unweighted Graph), design a Minimum Spanning Tree (MST) algorithm which maximizes the probability of finding a Hamiltonian path if one exists. In effect, we choose the weights on edges which will maximize the probability of finding a line connected spanning tree using MST algorithm.

Towards achieving that goal , we first attempt to maximize the probability P_L by a suitable choice of weights on the edges of an unweighted graph.

(A) Let w_{i_k} be the weight of the edge from vertex i_k to i_{k-1}. Choose those weights in such a way that

$$w_{i_1} \geq_{i_2} \geq ...w_{i_k} \geq ...w_{i_n}$$

(B) Set all other weights to ONE.

4.5 Calculation of Time Complexity of the Algorithm

It is well known that Kruskal's algorithm has a time complexity of O(E log V) time, where E is the number of edges in the graph and V is the number of vertices. Since the number of edges in the graph can atmost be (for fully connected graph), a *PROBABILISTIC POLYNOMIAL TIME* algorithm is designed for performing step 1. It can be readily seen that step 2 also requires Polynomial number of computations in the vertices. i.e. the algorithm belongs in the PP class of algorithms.

Note: The above algorithm can be used to compute all possible Hamiltonian paths/ cycles in a graph (using all possible line connected spanning trees in a graph).

Note: Suppose, we consider the adjacency matrix of a line connected spanning tree. It is proved in [4, Rama] that when the number of vertices of line connected graph is even, its adjacency matrix is unimodular and when the number of vertices is odd, the matrix is singular. Further, it is also proved that the determinant of adjacency matrix of any spanning tree other than line connected spanning tree is zero. Thus, to determine if a Hamiltonian path exists in a graph with even number of vertices, N , we need to determine if an N x N Jacobi sub-matrix exists in the N x N adjacency matrix of the graph (with a suitable labeling/ordering of the vertices).

Note: A more general problem is the so called "sub-graph isomorphism problem". In such a problem, a query graph is checked to be isomorphic to any sub-graph of a given graph. In this paper, we consider the query graph to be a Hamiltonian path/cycle [3, Rama].

5 CONCLUSION

In this research paper, a novel probabilistic algorithm for the problem of finding a Hamiltonian path/cycle in a graph is designed. The essential idea deals with computing line connected spanning tree if one exists.

References

[1] Garey, M. R. and Johnson , D. S. Computers and Intractability: A Guide to the Theory of NP-Completeness. In *New York: W. H. Freeman, p. 199*, 1983.

[2] Angluin, D. and Valiant, L. "Probabilistic Algorithms for Hamiltonian Circuits and Matchings." *J. Comput. Sys. Sci. 18, 155-190*, 1979.

[3] G. Rama Murthy. "Novel Shannon Graph Entropy, Capacity: Spectral Graph Theory: Polynomial Time Algorithm for Graph Isomorphism," *Technical Report IIIT/TR/2015/61, IIIT. Hyderabad, India, October*, 2015.

[4] G. Rama Murthy. "NP Hard Problems: Many Linear Objective Function Optimization Problem," *Technical Report IIIT/TR/2016/23, IIIT. Hyderabad, India, May*, 2016.