Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 11, November 2014, pg.21 – 24

RESEARCH ARTICLE

A Polynomial Time Algorithm for Hamilton Cycle

Mamta[1], Siddhartha Sankar Biswas[2]

[1]Computer Engineering; Gurgaon Institute of Technology and Management, India

[2] Computer Science and Engineering; Gurgaon Institute of Technology and Management, India

[1] mamta100492@gmail.com; [2] mailtobiswas@gmail.com

Abstract— This paper has a random polynomial time algorithm for finding Hamilton Cycle. As Hamilton Cycle problem is a strong NP complete problem and there is no polynomial time algorithm has been developed till now with proof. I have tried to prove it theoretically.

Keywords— Hamilton cycle, NP completeness, Strong NP complete, polynomial time, random algorithm

I. INTRODUCTION

The Hamilton Circuit problem is a well-known NP-complete problem. This famous problem can be described as follows: A Hamiltonian path is a path in an undirected or directed graph that visits each vertex exactly once. A Hamiltonian circuit is a Hamiltonian path that is a cycle. Finding Hamilton cycles (paths) in simple graphs is a classical NP Complete problem, known to be difficult both theoretically and computationally. In spite of recent advances, the problem presents an imminent scientific challenge. Yet no general polynomial time algorithms have been developed for Hamilton cycles or paths. I develop a polynomial time algorithm for finding Hamilton cycle (path) in undirected graphs. This algorithm works very smooth for all kinds of undirected graphs, i.e., for graphs with the same vertex size, except a little cases which very quick, their run time are very close to one another. So using my algorithm, there is no graph to cost much more time than others, i.e., there is no "dead angle" for my algorithm. Over the past decades, Hamilton cycles and paths have been widely studied.

II. PRINCIPLE AND METHOD

This algorithm uses the principle of "divide" and "conquer", does its best to utterly make use of the relative information among all parts. First, using the idea from the Greedy Algorithm, for each step, the algorithm tries to get the final result as quickly as possible. In order to limit its time to polynomial, the algorithm draws references from the State-Space Method. The state-space contains at least one final result (if exist). For each element in the state-space, the calculation time is polynomial and all elements number is polynomial, so the algorithm is polynomial. In order to limit the number of elements in the state-space to polynomial, the algorithm draws references from the Genetic Algorithm. In each step, optimize the elements in the state-space, only keeps and calculates the optimized elements. See Table I. The biggest specialty of this algorithm is: dynamically to

combine some parts, dynamically to transform them, by comparing different combinations' results, to deduce the relative information in as less as possible time, so as to make all the algorithm polynomial.

Technique Name	**Use**
Divide and Conquer	To divide the whole problems into parts, solve and then again combine
Greedy Method	To get the final result as quickly as possible
State-space Method	To limit the execution time to polynomial
Genetic Algorithm	To limit the number of elements in state-space to polynomial

Table I Techniques used in HCALGO

III. POLYNOMIAL TIME ALGORITHM

I have developed a random algorithm which finds a Hamilton Cycle present in the given graph G, in polynomial amount of time. Here are the steps involved in the algorithm.

3.1 Algorithm HCALGO (G, n)

Assume the graph G has at least one Hamilton cycle.

(1) First get a cycle which includes all n vertices in any way.

(2) In this cycle, some two neighbor vertices may not be adjacent; I call this point "break point".

(3) For each break point, I add an edge between the two neighbor vertices except for one, i.e. only one break point to be left(remember the added edges, later, each time delete one, then do the algorithm from a NEW start). I call the one break point being left "main break point".

(4) Now my algorithm only needs to handle this one break point. Each time handle one break point, at most n times because the number of the added edges at most n. So it does not affect the polynomial.

(5) At each step, cut a segment from the main break point, insert the segment in some place of the cycle. How to cut and insert? The rule is: make the number of new break points the least, and one new break point must be different from all former main break points(this new one as the new main break point ,I use a symmetric matrix to record all main break points, this guarantees my algorithm's polynomial). Notes: when calculating the number of new break points for getting the least, if more than one case has the same least number, compare their next step(only one time "next step", do not need to continue to do so);also, avoid inserting the same segment in different places consecutively.

(6) Then with the new main break point, do the same job until getting a Hamilton Cycle. If one step does not get a new break point and there are some break points at other place, get one of them as the new main break point and continue.

Flow chart of HCALGO is given in Figure 1.

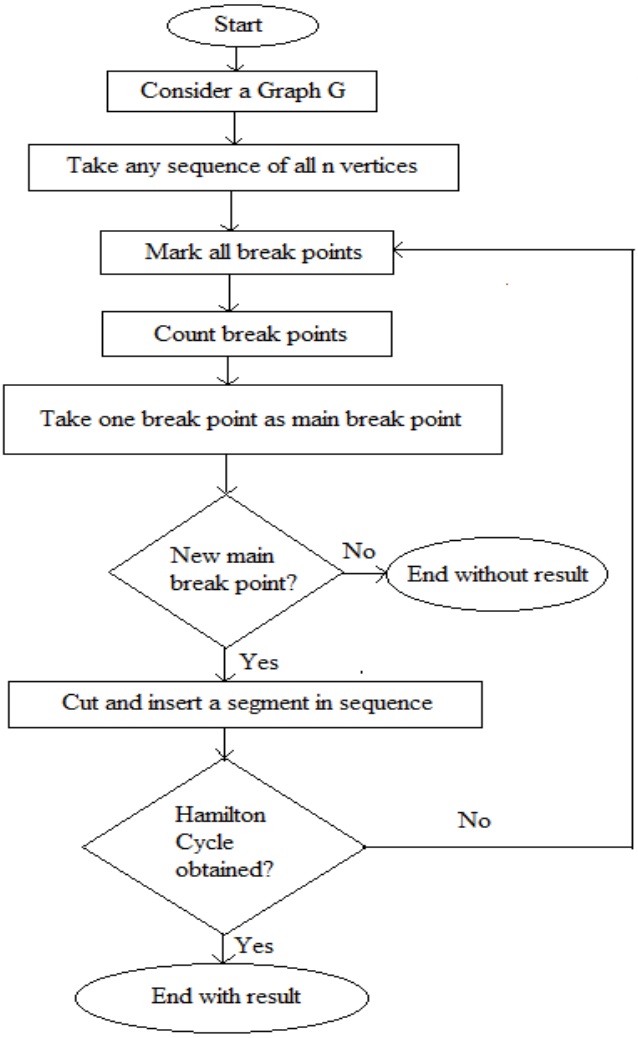

Figure 1 Flow chart of HCALGO

For each step, the algorithm tries to get the final result as quickly as possible. In order to limit its time to polynomial, the algorithm draws references from the State-Space Method. The state-space contains at least one final result (if exist). For each element in the state-space, the calculation time is polynomial and all elements number is polynomial, so the algorithm is polynomial. In order to limit the number of elements in the state-space to polynomial, the algorithm draws references from the Genetic Algorithm. In each step, optimize the elements in the state-space, only keeps and calculates the optimized elements. The biggest specialty of this algorithm is: dynamically to combine some parts, dynamically to transform them, by comparing different combinations' results, to deduce the relative information in as less as possible time, so as to make all the algorithm polynomial.

IV. CONCLUSIONS

HCALGO is a fixed algorithm that fits all graphs, and is proved theoretically. This algorithm is mixture of various known n widely used techniques. I have successfully tested this algorithm on 100s of graphs with no failures. So based on my research, I can conclude that polynomial time algorithm for Hamilton cycle problem is possible. And hence all the known NP complete problems can be solved using similar methods. Possibilities are there that P is equal to NP. My algorithm can be taken as basis for future work in this area and I hope soon someone will prove this most open problem of computer science field as P equals to NP.

REFERENCES

[1] Michael R. Garey and David S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, ISBN 0-7167-10455A1.3: GT37–39, pp. 199–200, 1997.

[2] Yuri Gurevich and Saharon Shelah, Expected computation time for Hamiltonian Path Problem, SIAM Journal on Computing, volume 16, Issue 3, pp 486—502, 1987.

[3] D.Johnson, The NP-completeness column-an ongoing guide, Journal of Algorithms, pp. 284-299, 1984.

[4] Yuri Gurevich, Complete and Incomplete Randomized NP Problems, 28th Annual Symposium on Foundations of Computer Science, Los Angeles, 12-14 October 1987, pp. 111-117.

[5] L. Du, "A polynomial time algorithm for hamiltonian cycle _Path_," in Proceedings of the International MultiConference of Engineers and Computer Scientists (IMECS '10), vol. 1, pp. 17–19, Hong Kong, 2010.

[6] M. S. Rahman and M. Kaykobad, "On Hamiltonian cycles and Hamiltonian paths," Information Procesing Letters, vol. 94, no. 1, pp. 37–41, 2005.

[7] R. Diestel, Graph Theory, vol. 173, Springer, New York, NY, USA, 2nd edition, 2000.

