Progress in Nonlinear Dynamics and Chaos
Vol. 1, 2013, 23-26
ISSN: 2321 – 9238 *(online)*
Published on 9 May 2013
www.researchmathsci.org

Progress in
Nonlinear
Dynamics and Chaos

Random Graphs and its Application to NP Complete Problems

M. Thiyagarajan
Professor Emirates and Dean Research
Nehru Group of Institution,
Kuniamuthur, Coimbatore- 641008, Tamilnadu, India
Email: m_thiyagarajan@yahoo.com

Abstract. Most NP- Complete problems have linear solutions when restricted to random graphs [2]. Random graph models provide probabilistic setting for studying such problems. Finding a Hamiltonian cycle in a graph is a NP - hard problem. In this paper, we give a polynomial time solution by suitably selecting random graphs, even though it may be hard to solve in general.

Keywords: Random graphs, NP Complete Problems, Coupon collector problems, Modified Hamilton cycle graphs, Randomized algorithms, Non – linear solutions.

1. Introduction

Bollobas [2] has introduced random graphs and explained the various applications to NP Complete Problems. Many graph theoretical algorithm are studied polynomial reductions of any NP Complete Problems. This forms the area of the complexity in theory of computation. Once such NP hard problem it's a generation of Hamilton Cycle. Here we present random graph theory solution the generation cycle. We give the algorithm and iC++ programs

1.1. Basic Definitions

Definition 1. Hamiltonian Cycle Problem

A Hamiltonian cycle in a graph is a cycle that visits each vertex exactly once. Given A directed graph G = (V,E) To Find If the graph contains a Hamiltonian cycle. Given a directed graph G= (V, E) and a certificate containing an ordered list of vertices on a Hamiltonian Cycle. It can be verified in polynomial time that the list contains each vertex exactly once and that each consecutive pair in the ordering is joined by an edge. We begin with an arbitrary instance of 3- SAT having variables $x_1,...,x_n$ and clauses $C_1,\ldots,C_k$. We model one by one, the 2^n different ways in which variables can assume assignments, and the constraints imposed by clauses.

Definition 2. Coupon Collector's Problem

In probability theory, the Coupon collector's problem describes the "collect all coupons and win" contests. It asks the following question: Suppose that there are *n* different

coupons, equally likely, from which coupons are being collected with replacement. What is the probability that more than *t* sample trials are needed to collect all *n* coupons? An alternative statement is: Given *n* coupons, how many coupons do you expect you need to draw with replacement before having drawn each coupon at least once? The mathematical analysis of the problem reveals that the expected number of trials needed grows as $\Theta(n \log n)$ [1]. For example, when $n = 50$ it takes about 225 [2] trials to collect all 50 coupons.

Definition 3. NP-complete

P is the class of languages that are decidable in polynomial time on a deterministic single-tape. A verifier for a language A is an algorithm V, where

A={w/V accepts <w,c> for some string c}.

We measure the time of a verifier only in terms of the length of w, so a polynomial time verifier runs in polynomial time in the length of w. A language A is polynomially verifiable if it has a polynomial time verifier. NP is the class of language that has polynomial time verifiers. Certain problems in NP whose individual complexity is related to that of the entire class. If a polynomial time algorithm exists for any of these problems, all problems in NP would be polynomial time solvable. These problems are called NP-complete. The phenomenon of NP-completeness is important for both theoretical and practical reasons.

D**efinition 4. SAT problem**

The satisfiability problem is to test whether a Boolean formula is satisfiable (if some assignment of 0s and 1s to the variables makes the formula evaluate to 1).

Definition 5. Randomized algorithm

In certain problems called optimization problems we seek the best solution among a collection of possible solutions. A probabilistic algorithm or Randomized algorithm is an algorithm designed to use the outcome of a random process.

2. Modified Hamiltonian cycle graph algorithm [7]

Input: A graph G = (V, E) with n vertices and associated edge lists.
Output: A Hamiltonian cycle or failure

Return a Hamiltonian cycle if one was found or failure if no cycle was found.

Hamiltonian cycle reduces to coupon collector problem:

Given a graph, the starting node and the ending node is there a path through the graph beginning and ending at the specified nodes such that every node in the graph is visited exactly once.

Coupon collector problem states that you want to collect the entire set of n different coupons by randomly drawing a coupon each trails. You can expect to make O(*n log n*) drawings before you collect the entire set.

Let *n* objects be picked repeatedly with probability P_i that object *i* is picked on a given try, with $\sum Pi = 1$.

Find the earliest time at which all n objects have been picked at least once.

It was first noted in Linial and Linial (1995) [5] that Hamiltonian cycle reduces to coupon collector's problem. Bach, Condon, Glaser and Tanguay (1996) [1] showed that Lipton's (1996) [6] algorithm for random solution reduces to coupon collector's problem.

The above modified algorithm of finding a Hamiltonian path looks exactly like the coupon collector's problem: the probability of finding a new vertex to add to the path when there are k vertices left to be added is k/n. once all the vertices are on the path, the probability

That a cycle is closed in each rotation is 1/n. Hence if no kist of unused edges is exhausted then we can expect a Hamiltonian path to be formed in about O(*n log n*) rotations, with about O(*n log n*) rotations to close the path to form a Hamiltonian cycle.

We initially analyze the algorithm assuming a specific model for the initial unused edges lists. We subsequently relate this model to the G_n, p model for random graphs. Assume that each of the n-1 possible edges connected to a vertex v is initially on the unused edges list for vertex v independently with some probability q. we also assume these edges are in a random order. One way to think of this is that, before beginning the algorithm, we create the unused edges list for each vertex v by inserting each possible edge(v,u) with probability q; we think of the corresponding graph G as being the graph including all edges that were inserted on some unused edges list. This means an edge(v,u) could initially be on the unused-edges list for v but not for u. also when an edge(v,u) is first used in the algorithm, if v is the head then it is removed just form the unused-edges list of v, if the edge is on the unused-edges list for u, it remains in the list.

By choosing the rotation edge from either the used edges list of the unused edges list with appropriate probabilities and then reversing the path with some small probability in each step, we modify the rotation process so that the next head of the list is chosen uniformly at random from among all vertices of the graph. Once we establish this property, the progress of the algorithm can be analysed through a straightforward application of our analysis of the coupon collector's problem which is also called the classical occupancy problem.

3. Conclusion

With high probability, a random graph chosen in this way has a Hamiltonian cycle. Can we use the modified Hamiltonian algorithm in the case where p is not known in advance, so that the edge lists must be initialized without knowledge of *p*?

REFERENCES

1. Bach, E., Condon, A., Glaser, E. and Tanguay, C., DNA Models and Algorithms for NP-complete Problems, New York, IEEE Computer Society Press, (1996), 290-299.
2. Bollobas B., Random graphs, Second Edition, Academic Press, Orlando FL (1999).

3. Bollobas B., Graph theory: An Introductory Course, Grad Texts in Math. 63, Springer-Verlag, 1979.
4. Bollobas B., Modern Graph Theory, Grad Texts Math 184, Springer, (1998).
5. Linial, M. and Linial, N., Letters to Science, Science, 268 (1995).
6. Lipton, R,J., DNA solution of hard computational problems, Science, 268 (1995), 542-545
7. Michael Mitzenmacher and Eli Upfal, Probability and Computing-Randomized Algorithms a Probabilistic Analysis, Cambridge University Press, (2005).
8. Michael Sipser, Introduction to the Theory of Computation, Second Edition, Thomson USA, (2007).