Finding all the Maximal Independent Sets in a graph

Gaganjeet Reen ICM2015003
Vinay Surya Prakash IHM2015004
Swapnesh Narayan ISM2015002
Leetesh Meena ISM2014008
Group - D

***Abstract*—A Maximal Independent Set is a basic building block in distributed computing. Some other problems pretty much follow directly from the Maximal Independent Set problem. In this paper we discuss what Maximal Independent Sets are and propose algorithms to find all the Maximal Independent Sets in a graph**

I. Introduction

An interconnection of points is known as Graphs , or more formally , A graph **G** is a set of **E** and **V** , where **E** denotes the set of edges and **V** denotes the set of Vertices . Here a vertex v represents a point or node and an edge e is a connection between two vertices. v_i and v_j are known as endpoints of that edge if an edge e_{ij} connects the vertices v_i and v_j. The problem of finding Maximal Independent Sets is a popular problem in Graph Theory and has widespread applications in the field of distributed systems.In the course of this paper we discuss two algorithms to calculate Maximal Independent Sets. We discuss a sequential algorithm which works but runs slow and then we discuss a corresponding randomised algorithm which works in logarithmic run time. We also discuss the applications of Maximal Independent Sets in the real world and explain the concept with examples.

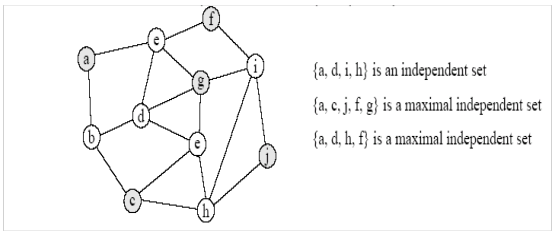

Fig. 1. Example of Independent Sets

II. Definitions

A) Adjacency Matrix
The adjacency matrix, sometimes also called the connection matrix, of a simple labeled graph is a matrix with rows and columns labeled by graph vertices.If there exists an edge from v_i to v_j ,the corresponding entry in the adjacency matrix is 1,otherwise the entry is 0. For a simple graph with no self-loops, the adjacency matrix must have 0s on the diagonal. For an undirected graph, the adjacency matrix is symmetric.

B) Independent Sets :-
Given an undirected Graph G = (V, E) an independent set is a subset of nodes U , such that no two nodes in U are adjacent. An independent set is maximal if no node can be added without violating independence. An independent set of maximum cardinality is called maximum.

C) Maximal Independent Sets :-
An independent set is maximal if no node can be added without violating independence.

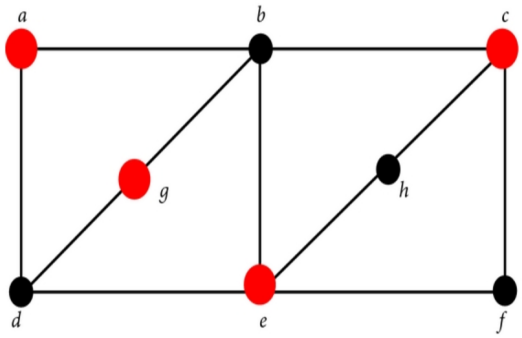

Fig. 2. Example of Maximal Independent Sets

D) Graph Coloring :-
In graph theory, graph coloring is a special case of graph labeling; it is an assignment of labels traditionally called ”colors” to elements of a graph subject to certain constraints. In

its simplest form, it is a way of coloring the vertices of a graph such that no two adjacent vertices are of the same color; this is called a vertex coloring. Similarly, an edge coloring assigns a color to each edge so that no two adjacent edges are of the same color, and a face coloring of a planar graph assigns a color to each face or region so that no two faces that share a boundary have the same color.

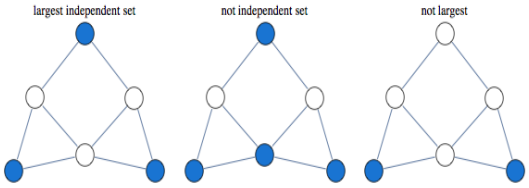

Fig. 3. Example of Independent Sets

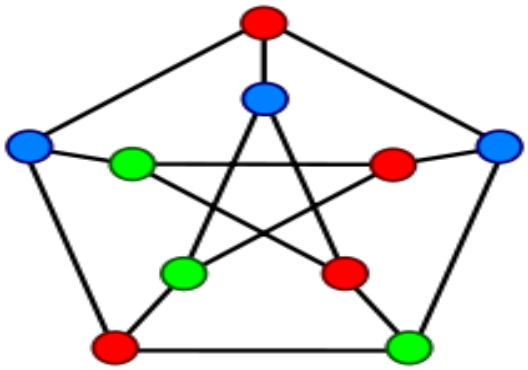

Fig. 4. Example of Graph Coloring

III. Motivation

Finding Maximal Independent Sets has a lot of application in the real world and we provide methods to tackle the problem in this paper.We propose both an easy and also an efficient solution for the problem. We do not find the Maximum Independent Set but the Maximal Independent Sets. Finding the Maximum Independent Set is an NP-Hard problem.Computing a MIS sequentially is trivial: Scan the nodes in arbitrary order. If a node u does not violate independence, add u to the MIS. If u violates independence, discard u . So the only question is how to compute a MIS in a distributed way.

IV. Algorithms

,

Algorithm 1 Slow MIS

1: assume node **IDs**
2: **for** each node v in Graph **do**
3: **if** all neighbors with larger IDs have decided not to join MIS **then**
4: **v** decided to join MIS

,

Algorithm 2 Fast MIS

1: Proceed in rounds consisting of phases
2: **for** each phases **do**
- each node chooses a random value r(v) $\in$ [0,1] and sends it to ist neighbors
- If r(v) $<$ r(w) for all neighbors w $\in$ N(v), node v enters the MIS and informs the neighbors
- If v or a neighbor of v entered the MIS, v terminates (and v and edges are removed), otherwise v enters next phase

Algorithm 3 Lubys Algorithm

1: Initialize **I** to an empty set.
2: **while** **V** is not empty **do**
- Choose a random set of vertices S $\subseteq$ V, by selecting each vertex v independently with probability $1/(2d(v))$, where d is the degree of v (the number of neighbours of v).
- For every edge in E, if both its endpoints are in the random set S, then remove from S the endpoint whose degree is lower (i.e. has fewer neighbours). Break ties arbitrarily, e.g. using a lexicographic order on the vertex names.
- Add the set **S** to **I**
- Remove from V the set S and all the neighbours of nodes in S.

3: return **I**

V. Methods

We have proposed the following methods to find the Maximal Independent Sets in a graph :-

A) **Method 1 :-**
The first method proposed is the the sequential method which goes to each vertex and checks if adding it violates the independence property of the set already formed.

B) **Method 2 :-**
The second method proposed is a randomised method which makes use of probablilistic analysis to efficiently compute the Maximal Independent Sets.

C) **Method 3 :-**
The third method proposed is the Luby Rack-off Method. It combines the concepts of graph coloring and maximal independent sets.
Each color in a valid coloring constitutes an independent set (but not necessarily a Maximal Independent Set).
Computing Maximal Independent Set from Colorings :-
Choose all nodes of first color . Then for any additional color, add in parallel as many nodes as possible.

D) **Method 4 :-**
A reasonable (but not very efficient for large graphs) method for obtaining all maximal independent sets in any graph can be developed using Boolean arithmetic on the vertices. Let each vertex in the graph be treated as a Boolean variable. Let the logical (or Boolean) sum a + b denote the operation of including vertex a or b or both; let the logical multiplication ab denote the operation of including both vertices a and b, and let the Boolean complement a! denote that vertex a is not included. For a given graph G we must find a maximal subset of vertices that does not include the two end vertices of any edge in G. Let us express an edge (x,y) as a Boolean product, xy, of its end vertices x and y, and let us sum all such products in G to get a Boolean expression.Let us further take the Boolean complement of this expression, and express it as a sum of Boolean products.A vertex set is a maximal independent set if and only if the complemented expression is true, which is possible if and only if at least one term making up the expression is 1, which is possible if and only if each vertex appearing in the term (in complemented form) is excluded from the vertex set of G.

VI. Implementation

The code has been implemented using python language.The graphs displayed in the results have been generated using the matplotlib library of python.The graph is taken as input from the user in the form of an edge list.

VII. Analysis of Algorithms

A) **Slow-MIS :-**
Let us assume that the the maximum degree that any vertex has is m. Let the total number of nodes in the graph be n. The upper bound for the time complexity in this scenario is $O(mn)$.

B) **Fast-MIS and Luby's Algorithm:-**
Given a coloring algorithm with runtime T that needs C colors, we can construct a MIS in time C+T.We can color trees in log* time and with 3 colors, so: There is a deterministic MIS on trees that runs in distributed time O(log n).

VIII. Applications

A. Graph Coloring

The Maximal Independent Set Algorithms can be used for graph coloring.
Clone each node v, d(v)+1 many times. Connect clones completely and edges from i-th clone to i-th clone.Run MIS: if i-th copy is in MIS, node gets color i.

B. Matching

The Independent Set algorithms can also be used for solving matching problems in the field of graph theory.

C. Communication and Data Exchange

Distributed maximal independent set algorithms are strongly influenced by algorithms on the PRAM model. The original work by Luby and Alon et al. has led to several distributed algorithms. In terms of exchange of bits, these algorithms had a message size lower bound per round of $O(logn)$ bits and would require additional characteristics of the graph. For example, the size of the graph

would need to be known or the maximum degree of neighboring vertices for a given vertex could be queried. In 2010, Métivier et al. reduced the required message size per round to $O(1)$, which is optimal and removed the need for any additional graph knowledge

.

IX. Results

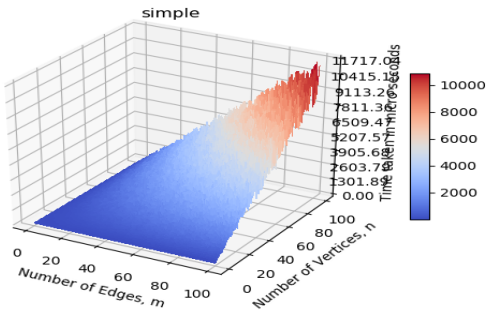

Fig. 5. Results of Simple Algorithm

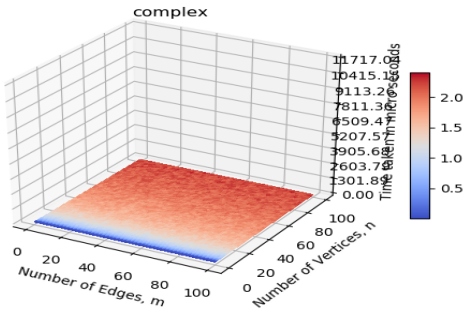

Fig. 6. Results of Fast-MIS Algorithm

The above images show the time taken by the Simple Algorithm and the Fast Algorithm to compute the number of Maximal Independent Subsets in a graph. It is evident that the fast algorithm gives much better results than the simple algorithm.

X. Code Snippets

```
MIS = []
for v in vertices:
  neighbors = []
  for edge in edgeList:
    if v == edge[0]:
      neighbors.append(edge[1])
    if v == edge[1]:
      neighbors.append(edge[2])
    for neighbor in neighbors:
        nonRepeatCond = (idList[neighbor] > idList[vertex])
        isNotJoined = ( joined[neighbor] != True)
        if(nonRepeatCond and isNotJoined):
           MIS.append(v)
```

Fig. 7. Simple Algorithm

```
def fastMIS(vertices,phases):
    MST
    for phase in phases:
       #randomising the vertices first
       for v in vertices:
           rand[v] = float(random.randInt(0,100))/100.
       #selecting one v from the vertices
       for v2 in vertices:
           isLessThanAll = True
           for v2 in vertices:
               if(rand[v2] < rand[v1]):
                   isLessThanAll = False
                   break
           if(isLessThanAll):
               MST.append(v)
               #terminating v by removing all the edges of v
               vertices.remove(v)
               removableList = []
               for edge in edgeList:
                   if(edge[0] == v or edge[1] == v):
                   removableList.append(edge)
               for edge in removableList:
                   edgeList.remove(edge)
    return MST
```

Fig. 8. Fast MIS

XI. Conclusion

We have discussed 4 methods which can be used to compute all the maximal independent sets.The most efficient method proposed is the Fast-MIS which takes $O(log(n))$ time to compute the maximal independent sets. The applications of Maximal Independent Sets have been explained as well with a detailed analysis of all the algorithms proposed.

REFERENCES

[1] Narsingh Deo, Graph Theory with Applications to Engineering and Computer Science.
[2] http://www.iitg.ac.in/gkd/aie/slide/MIS-PSM.pdf
[3] http://www.site.uottawa.ca/ lucia/courses/4105-02/a2.pdf
[4] https://www.net.t-labs.tu-berlin.de/ stefan/class02-mis.pdf
[5] https://en.wikipedia.org/wiki/Graph-coloring
[6] http://www.mate.unlp.edu.ar/ liliana/lawclique_2016/07.pdf
[7] https://en.wikipedia.org/wiki/Independent_set_$graph_theory$