Subgraph Isomorphism

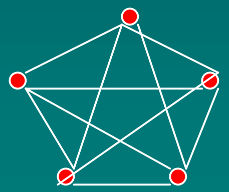
Graphs

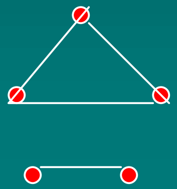
Ordered Pair $G = (V,E)$ where

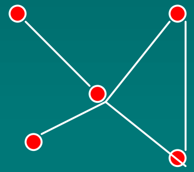
- V is a set of vertices
- $E \subseteq V \times V$ set of edges

Common Terminology

- Complete Graph:- A graph is called complete when every pair of vertices is an edge.
- Connected Graph:- A graph is connected if there exists a path between any two vertices.


Complete Graph


Disconnected Graph


Connected Graph

Terminology (contd..)

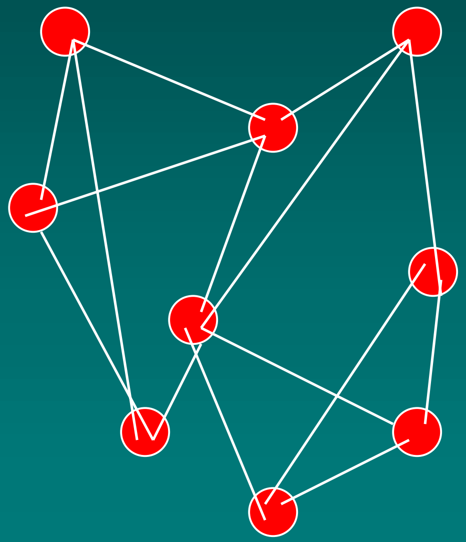
- Stable Set :- A subset of vertices with no edge between any two of them.
- Degree :- The degree of a node is the number of edges incident to it.
- Clique:- A subset of vertices in which every pair is an edge.

Types of Graphs

- K Connected Graphs
- K Partite Graphs
- Wheel Graphs
- Interval Graphs
- Grid Graph

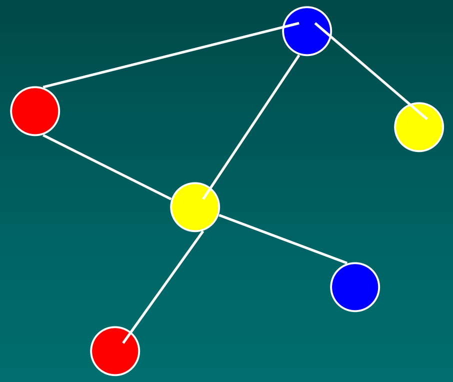
K-Connected Graph

A graph G is said to be k-connected if there does not exist a set of (k-1) edges whose removal disconnects the graphs, i.e. the vertex connectivity of G is k.



K-Partite Graphs

- A graph whose vertices can be partitioned into disjoint sets so that no two vertices within the same set are adjacent.
- Coverable by k independent sets.

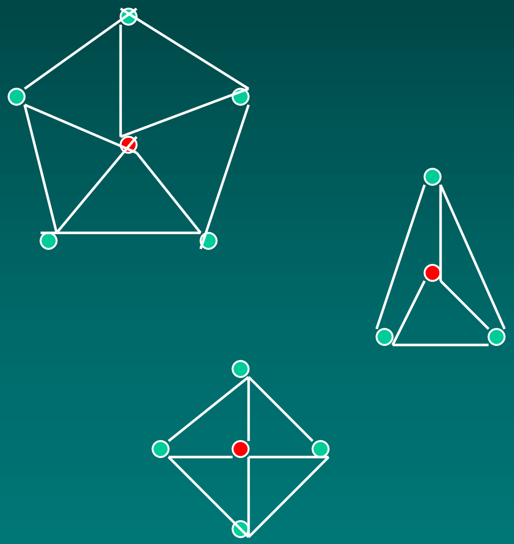


Tripartite Graph

Wheel Graphs

A wheel graph of order n has a cycle of order n-1, and for which every graph vertex in the cycle is connected to one other graph vertex called the hub.

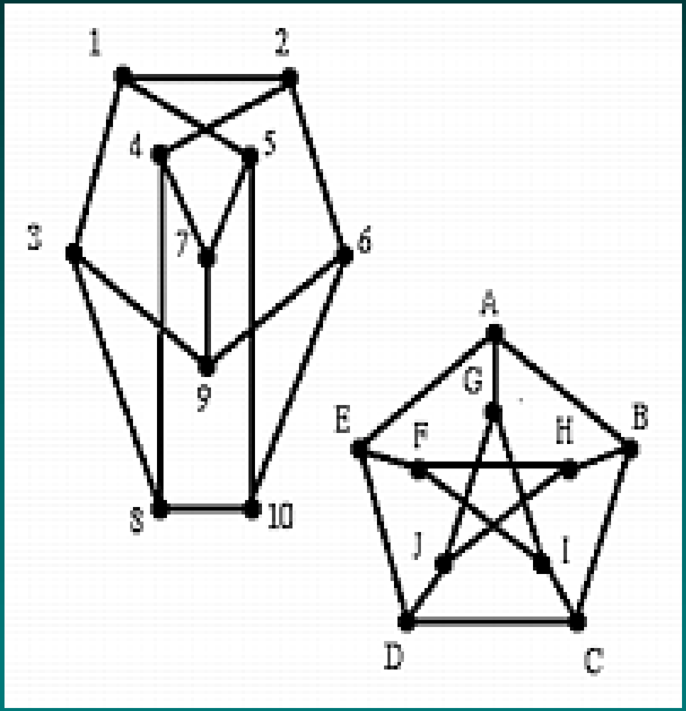
The central point of a wheel graph is called hub and it has a degree n-1.

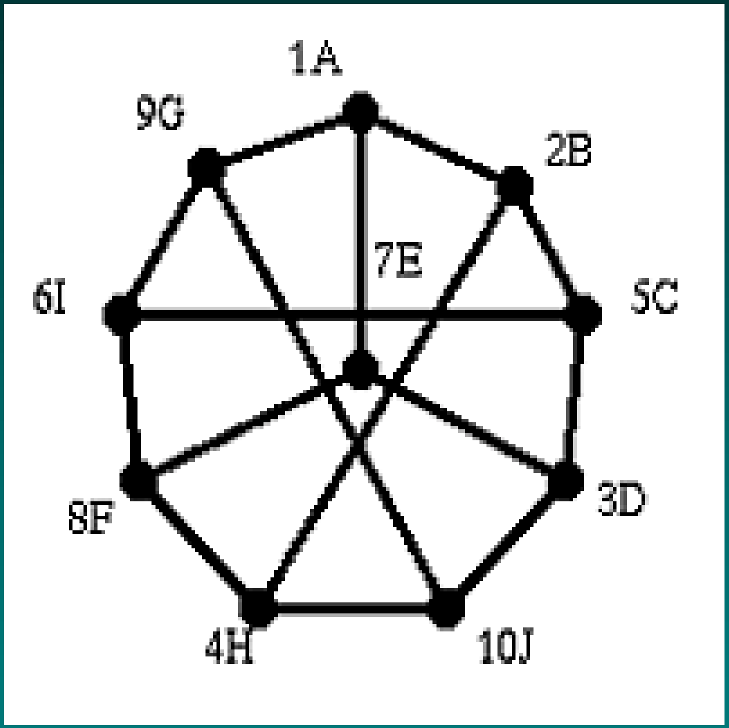


Interval Graphs

An undirected graph G whose vertices can be put into one-to-one correspondence with a set of intervals of a linearly ordered set(like the real line) such that two vertices are connected by an edge of G if their corresponding intervals have non-empty intersection.

What is graph isomorphism ?





Graph Isomorphism

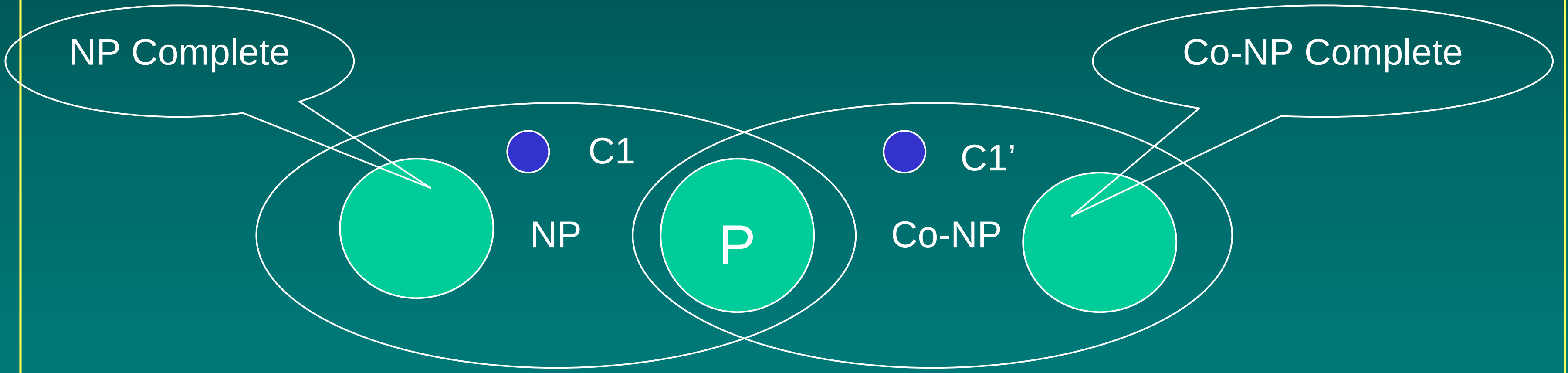
A graph $G = (V, E)$ is said to be isomorphic to another graph $G' = (V',E')$ if there exists a bijective function $f:V\rightarrow V'$ such that

For any edge $e = (v_i, vj) \in E$, there exists an edge $e' = (f(v_1),f(v2)) \in E'$ and for any $e'=(v_1',v_2') \in E'$ there exists an edge $e = (f^{-1}(v1),f^{-1}(v2)) \in E$

The notation $G \; = G'$ is used to indicate that G is isomorphic to G'.

Class of the problem

- Its in a class of its own, neither into NP complete, co-NP or into P.
- Its classified in the "Isomorphism Complete" class.



Applications

- Computer Chip Intellectual Property rights control
- Identification of similar molecules
- Pattern Recognition and Computer Vision.

Possible Approaches

- Brute Force
- Ulmanns backtracking algorithm (1976)
- Nauty by Brendan McKay (1981)
- VF algorithm by Cordella et all. (1998)

Subgraph Isomorphism

Given a graph G = (V,E) a sub-graph G’ = (V’,E’) is defined such that

- V’ $\subseteq$ V
- E’= E$\cap$(V’ X V’)

The notation G $\subseteq$ G’ is used to indicate that G is a sub-graph of G’

A graph G = (V,E) is said to be subgraph isomorphic to another graph G’=(V’,E’) if $\exists$ S such that S $\subseteq$ G’ and S = G.

G’ is called the input graph and G is called the model graph.

Applications

- Pattern Recognition in Bio-Informatics & Bio-Computing.
- Image Processing & Computer Vision
- Identification of sub-compound molecules of a given molecule.
- Recognition of distorted shapes.

Complexity

General graphs NP complete (Manning (1990))

Special planar graphs Polynomial

• Embedded planar	(Manning (1990))
• 3-connected	(Hong, McKay, Eades (2002))
• Outerplanar	(Manning, Attalah (1992))
• Series-Parallel	(Hong, Eades, Li (2000))
• Trees	(Manning, Attalah (1989))

Aim of our project

It is well known that subgraph isomorphism is NP complete for general graphs. The aim of this project is to study the various algorithms pertaining to the problem, and based on our study to come up with a heuristic for the problem of subgraph isomorphism, which runs with high probability of success000000000

What is meant by heuristic algorithm

- A problem-solving technique in which the most appropriate solution is selected at successive stages of a program for use in the next step of the program. (Dictionary Definition)
- A rule of the thumb, simplification or educated guess that reduces or limits the search for solutions. Heuristics do not guarantee optimal or even feasible solutions and are often used with no theoretical guarantee.

Major Algorithms we studied

- Brute-force Enumeration technique
- Ullmann’s Algorithm
- Messmer’s Algorithm
- Eppstein’s Algorithm

Enumeration

The enumeration algorithm aims at finding isomorphism between the model graph and subgraph of input graph.

M’ is defined as a matrix of order m x n where m in the number of nodes in the input graph and n is the number of nodes in the model graph.

Enumeration (contd..)

M’ is a matrix which is generated by systematically assigning 0’s and 1’s each row contains exactly one 1 and no column contains more than one 1. This matrix M' = [m'ij] can be used to permute rows and columns of B to produce a further matrix C. Specifically, we define $C = [c_{ij}] = M'(M'B)^T$.

$(\forall i \forall j)(a_{ij} = 1) \Rightarrow (c_{ij} = 1)$ [Condition 1]

Enumeration Algorithm

- Step1 : $M = M^0$, $d = 1$; $H_1 = 0$; for all i = 1, .. , m, , set $F_i = 0$;
- Step2 : If there is no value of j such that m_{dj} = l a n d $F_j = 0$ then go to step 7; $M_d = M$, i f $d = 1$ then $k = H_1$ else $k = 0$,
- Step3 : $k = k + 1$, if $m_{dk} = 0$ or $F_k = 1$ then go to step 3; for all $j \mathrel{!}= k$ set $m_{dj} = 0$,
- Step4 : If $d < m$ then go to step 6 else use condition (1) and give output if an isomorphism is found;

Enum. Algorithm(contd..)

- Step5 : If there is no j >k such that m_{dj} = l and F_j = 0 then go to step 7; M = M_d , go to step 3;
- Step6 : H_d=k, F_k=1; d=d+1; go to step 2,
- Step7 : If d=1 then terminate algorithm, F_k=0 ; d=d-1, M = M_d, k=H_d go to step 5

Ullmann’s Algorithm

- Ullmann algorithm is basically a modification of enumeration technique.
- It follows the branch and bound paradigm.
- A constraint is devised which inhibits the generation of those matrices M’ which gives unsuccessful mappings.

Ullmann's Algorithm

- Step1 : $M = M^0$, $d = 1$; $H_1 = 0$; for all i = 1, .. , m, , set $F_i = 0$; Refine M if exit FAIL then terminate algorithm.
- Step2 : If there is no value of j such that $m_{dj} = l$ a n d $F_j = 0$ then go to step 7; $M_d = M$, i f $d = 1$ then $k = H_1$ else $k = 0$,
- Step3 : $k = k + 1$, if $m_{dk} = 0$ or $F_k = 1$ then go to step 3; for all j != k set $m_{dj} = 0$ if exit FAIL then go to step 5

Ullmann Algorithm(contd..)

- Step4 : If $d < m$ then go to step 6 else use condition (1) and give output if an isomorphism is found;
- Step5 : If there is no $j > k$ such that $m_{dj} = l$ and $F_j = 0$ then go to step 7; $M = M_a$, go to step 3;
- Step6 : $H_d=k$, $F_k=1$; $d=d+1$; go to step 2,
- Step7 : If $d=1$ then terminate algorithm, $F_k=0$; $d=d-1$, $M = M_d$, $k=H_d$ go to step 5

Decomposition Heuristic

- Optimal decomposition of the of a graph takes exponential time.
- An heuristic approach using decomposition was suggested by Messmer[1].
- It does not guarantee the optimal solution but finds a nearly optimal decomposition in much less time.
- This heuristic gives best results for multiple model graphs.

Messmer's Algorithm

- This is an algorithm which uses the dynamic programming paradigm.
- A graphs is decomposed into two graphs S_{max} and $G – S_{max}$ where S_{max} is the largest subgraph of G already memoized.
- If no S_{max} exists then G is randomly divided into two graphs each consisting of equal vertices.

Polynomial Algorithm

- Enumerating all sub-graphs of a given graph takes exponential time.
- Thus for a general graph it is not possible do subgraph isomorphism in polynomial time.
- Some constraints needs to be evolved in order to solve subgraph isomorphism in polynomial time even if it is only for some restricted class of graphs.

Eppstein's Algorithm

- Constraints:
  - Input graph is planar or has a wheel symmetry.
  - Model graph is K_3 or K_4
- Methodology
  - Decomposition into bounded trees of fixed width

What is Bounded Tree

- A tree decomposition of a graph G consists of a tree T, in which each node N$\in$T has a label L(N) $\subset$V(G) , such that the set of tree nodes whose label contain any particular vertex of G forms a contiguous subtree of T, such that any edge of G connects two vertices belonging to the same label L(N) for at least one node N of T.Width of tree is one less than minimum width of any label.

Eppstein An Example

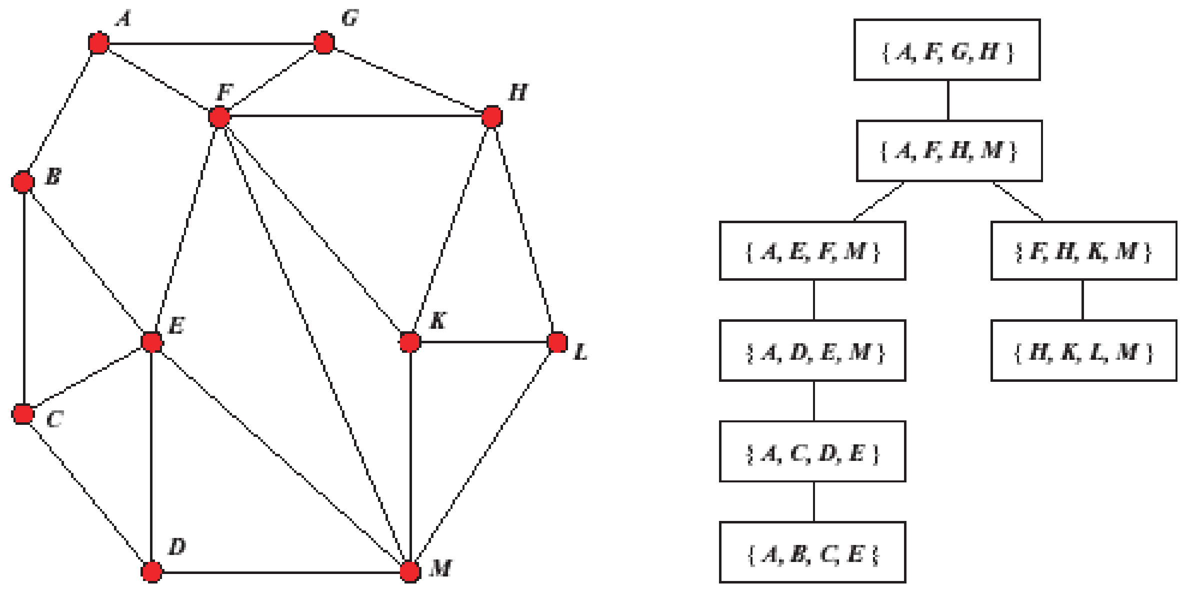


Figure 1: Tree decomposition of a planar graph.

Eppstein(contd…)

- Lemma 1
  - Let planar Graph G have a rooted snapping tree T in which the longest path has length l.Then a tree decomposition of G with width at most 3l can be found in time O(ln).
- Theorem
  - We can count the isomorphs of any Wheel W_k in a planar text graph G with n verticies in time $O(nk^2)$.By applying a dynamic programming approach on the generated trees to lookup for the isomorphism.

Original Work

Naming Scheme

- This approach is motivated by the fact that rate of information retrieval is influenced by the representation of data.
- The matrix representation which is currently the norm for graphs contains a lot of redundant information is which is not relevant to the problem under consideration.

Naming Scheme Possible ?

- A naming scheme is possible for trees and planar graphs.
- It is not possible to come up encoding which maps a string to a general graph in such a manner that the implicit details regarding the graph is coded into the string.

Better Base Case (BBC)

- In all the approaches that are presently being used the basis of comparison are two nodes with an edge in between.
- A better base case will lead to a asymptotically better algorithm.

Problems with BBC

- For a effective utilization of this technique for subgraph isomorphism domain knowledge is required.
- For example in the case of K_3 all the possible graphs containing 3 nodes maybe used as a base case.
- Thus this method cannot be used as a general paradigm but can be used in order to improve the performance of a a given algorithm.

Problems with Current Methods

- Finally going through all the above mentioned algorithms we found, that the bottleneck was the number of edges in the model graph.
- Every algorithm tries to overcome this bottleneck by decomposing the model graph into smaller instances and finally recombining them in order to form the final solution.

Bottleneck in Subgraph

- This process though ingenious, does not prove effective in most cases and can be efficiently used only in specific domains.
- So we propose to remodel the given model graph into a using a new data structure, called a constrained BFS_C tree.

Constrained BFS Tree

- This is fundamentally a BFS (Breadth First Search) tree with a added constraint that for each pair of nodes in the resulting tree we allow for only for one (cross/forward/backward) edge which will henceforth be referred to as special edge(SE). So from its definition it is quite evident that for a BFS_C T with n nodes we can have only have a maximum of [n/2] such SE edges

Lemma 1

- Given a k–connected graph G , with set of vertices V and edges E such that |V|=n , we can decompose such a graph into maximum of (K-1) such unique BFS_C trees, or in other words these (k-1) trees would cover the graph
- $\Leftrightarrow G(E,V) \equiv U^{(k-1)}BFS_C$

Proof

- – Given a k-connected graph G with |V|=n , so by the graph property we have maximum of [Kn/2] number of edges
- – For a tree with n vertices has maximum of (n-1) tree edges.
- – Now applying the constraint property we can have maximum of [n/2] special SE edges in our BFS_C tree which are unique.

Proof (contd..)

- So in the worst case of a completely k-connected graph, we will need to have (k-1) such BFS_C trees each having maximum of unique[n/2] SE edges .
- As (n-1)+(k-1)*[n/2]≡[Kn/2]

• Hence proved.

Completeness

- We can show that such a construction of BFS_C tree will cover the complete graph.
  - This follows directly from the proof of lemma 1 , as during the construction of the BFS_C trees we have shown (k-1) such trees will cover all the edges of the graph G uniquely , ie there would be no duplication of SE edges in two different BFS_C trees.
  - So we are not missing my edge of the graph if we consider all such (k-1) trees.

Lemma 2

- If a subgraph is present in a input graph G_i, then it equivalent to say it exists at the point where all of these (k-1) BFS_C tress map to.
- Proof
  - Again from lemma 1 we get that (K-1) trees have covered all the edges of the Model Graph G_m , so the place where each of these BFS_C trees map , would mean the union of all the SE and tree edges of all the tress which is finally our model graph.

Better Performance

- This technique of decomposition works faster than an implementation of Ullmann Algorithm, because the time required for our algorithm during its comparison phase would be lesser than its counterpart as it has to look for more sparse structure.
- The time required to arrive at the successful node where the pattern exists would also be lesser due to the faster rejection of unsuccessful nodes

- In the comparison module of Ullmann, has to modified by the virtue of which it would not be checking for the internal connections.This further enhances the performance.
- The model graph will be compared using the relaxed standards of comparisons used for BFS_c

New Algorithm

- Inputs:
  - A Input graph G_i
  - A Model graph G_m
- Output
  - The location of the subgraph isomorphic structure in the input Graph if it exists
- Step 1:
  - Perform vertex testing on the two graphs, if it fails then exit.

NA (contd1..)

- Step 2:
  - Generate the set of BFS_C tress of model Graph G_m as described in lemma 1. The generation is to be done in lexicographic order.
- Step 3:
  - Perform modified NA on the first BFS_C tree and the input graph, and keep the track of the vertex correspondence of the structure if any found in form of a matrix.

NA (contd2..)

- Step 4:
  - The found pattern is fed into a different running thread which checks for the remaining edges , with the (k-2) left BFS_C tree, with the pattern discarded if it is not matched with even one of the BFS_C tree. While the other execution continues in the main thread.
- Step 5
  - If the pattern found is matched with all the (k-2) BFS_C trees, then both the thread exits , and a Exact Sub graph isomorphicstructure is found. Otherwise it will continue the main thread till all the possible cases are exausted.

References

[1] Bruno T. Messmer and Horst Bunke. Efficient Subgraph Isomorphism Detection: A Decomposition Approach, IEEE Transactions on Knowledge and Data Engineering. March 2000. isomorph\iam-95-003.ps

[2] Ron Shamir and Dekel Tsur. Faster Subtree Isomorphism. isomorph\shamir97faster.pdf

[3] Arvind Gupta and Naomi Nishimura. Complexity of Subgraph Isomorphism: Duality results for graphs of bounded path and tree width. isomorph\gupta95complexity.ps

[4] Combinatorial Algorithms.

[5] Max Peysakhov and William Regli. Genetic Algorithms for the Sub-Graph Isomorphism Problem isomorph\genetic.pdf.

[6] J. R. Ullmann. An Algorithm for Subgraph Isomorphism. JACM 1976. isomorph\p31-ullmann.pdf

[7] David Eppstein. Subgraph Isomorphism in Planar Graphs and Related Problems. isomorph\eppstein99.3.3.pdf

A heartfelt thanks to Prof. R.K.Ghosh

A special mention for Mr. B. V Raghavendra Rao.