Subgraph Isomorphism: Spanning Trees in Graphs

Author: Thomas Schuler
tschule2@nebrwesleyan.edu

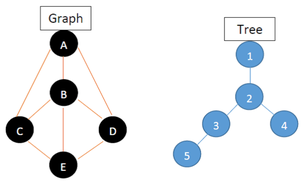
Advisor: Dr. Austin Mohr
Nebraska Wesleyan University

Objective

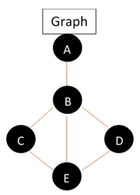
Given a tree T and a graph G, efficiently and quickly determine whether T is a spanning tree of G.

Definitions

- Spanning Tree
  - A spanning tree is a tree with a relationship to a graph in that the tree has the same number of vertices as the graph and the tree can be placed within the graph.



- Cut Vertex
  - A cut vertex is a vertex within a graph that if removed will create new graphs from the original graph.
  - The following example shows that B is a cut vertex.



- Component list
  - A component list is a list that contains the number of vertices in a block when a cut vertex is removed and stored with the index of the vertex.

Subgraph Isomorphism

- The problem of Subgraph Isomorphism is to find if some graphs exist within a graph G.
- Our approach to this problem is to limit the subgraphs to spanning trees and to minimize the time spent seeing if the graph is a subgraph of the graph G.

Our Approach

Our approach is to use the facts about cut vertices to reduce the run time on the algorithm to discover subgraphs. By using a slightly modified algorithm to detect blocks, that runs in linear time, the algorithm would detect the size of the components and update the component list with the cut vertices. Our approach:

- assign cut vertices by component list
- run through the possibilities of cut vertices first
- run through all remaining possibilites.

Examples

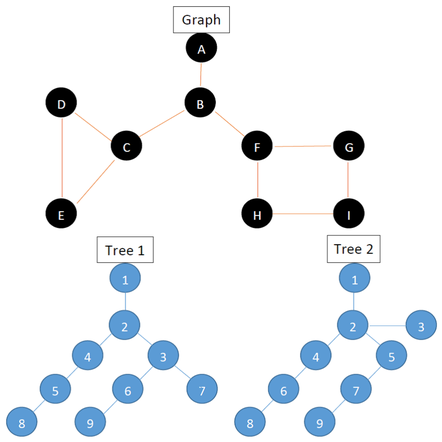


Figure 1: Graph and Trees

The example is a graph with 3 cut vertices B, C, and F and all of the vertices in the tree are cut vertices since it is a tree. To demonstrate the algorithm, there are two examples Tree 1 and Tree 2.

Tree 1

1. Run the algorithm to find the cut vertices in the graph which are B, C, and F and their block sizes.
   - $B : \{1, 3, 4\}$, $C : \{2, 6\}$, $F : \{3, 5\}$
2. Now we build the component list for the trees.
   - $1 : \{8\}$, $2 : \{1, 3, 4\}$, $3 : \{1, 2, 5\}$, $4 : \{2, 6\}$, $5 : \{1, 7\}$, $6 : \{1, 7\}$, $7 : \{8\}$, $8 : \{8\}$, $9 : \{8\}$
3. Now build a candidate list from the tree onto the cut vertices.
   - $B : \{2\}$, $C : \{2, 3, 4\}$, $F : \{2, 3\}$
4. Now $B \to 2$, $F \to 3$, and $C \to 4$.
5. Set candidate lists for other vertices in the graph based on their blocks.
   - $A : \{1\}$, $D : \{5, 8\}$, $E : \{5, 8\}$, $G : \{6, 7, 9\}$, $H : \{6, 7, 9\}$, $I : \{6, 7, 9\}$
6. And $A \to 1$.
7. Run through the possibilities of the tree onto the graph and we find that tree 1 exists in the graph.

Tree 2

1. Run the algorithm to find the cut vertices in the graph which are B, C, and F and their block sizes.
   - $B : \{1, 3, 4\}$, $C : \{2, 6\}$, $F : \{3, 5\}$
2. Now we build the list of components for the trees.
   - $1 : \{8\}$, $2 : \{1, 1, 3, 3\}$, $3 : \{8\}$, $4 : \{2, 6\}$, $5 : \{2, 6\}$, $6 : \{1, 7\}$, $7 : \{1, 7\}$, $8 : \{8\}$, $9 : \{8\}$
3. Then build a candidate list for the tree onto the graph.
   - $B : \{2\}$, $C : \{2, 4, 5\}$, $F : \{2\}$
4. Now $B \to 2$ and F has no candidate.
5. So since F has no candidate tree 2 is not a subgraph.

Further Research

Our approach seems to be very efficient for graphs with many cut vertices but we would like to use it and generate data for our approach. We would also like to possibly implement a recursive algorithm for the cut vertices (that is to look for cut vertices in the components created after breaking the graph into smaller components). We believe that this has a potential to greatly reduce the run time.

References

[1] Douglas West.
Introduction to Graph Theory.
Prentice-Hall, 2nd edition, 2001.

Acknowledgements

- Mathematical Association of America
- NWU Student-Faculty Collaborative Research Committee
- NWU Department of Mathematics
- NWU Student Affairs Senate

NEBRASKA WESLEYAN UNIVERSITY