AD-A102 231 ILLINOIS UNIV AT URBANA DEPT OF COMPUTER SCIENCE F/G 12/1
EXAMINATION OF A BOUNDARY BETWEEN P AND NP-COMPLETE PROBLEM,(U)
1980 J R EGAN

UNCLASSIFIED NL

END
DATE FILMED
8 81
DTIC

Examination of a Boundary Between P and NP-Complete Problems

J. R. Egan

1. Introduction

Since the introduction of the notion of NP-completeness in the study of complexity theory about ten years ago, there has been an explosion of research activities in the area. In particular, the collective effort of many researchers has led to the discovery of a very large number of problems that are certified to be NP-complete. However, there is an interesting question about which very little is known, namely, what makes a problem NP-complete? In order to gain futher insight into the characterization of NP-complete problems, we studied several problem areas in which there is a spectrum of problems whose complexities vary with some "small" variations on the assumptions. We hope that a further understanding of the "boundary" that separates those problems in P and those problems that are NP-complete will lead to a further understanding of the characterization of NP-complete problems.

2. The Subgraph Homeomorphism Problem

A problem that we studied is the Subgraph Homeomorphism Problem (SHP). Given two graphs G and H, the question is to determine whether there exists a mapping f from the set of vertices of H onto the set of vertices of G such that for every edge (v_i, v_j) in H there is a corresponding path between $f(v_i)$ and $f(v_j)$ in G. There are several obvious variations of the problem, for example, the graphs G and H can both be directed or undirected, the paths in G can be either vertex-disjoint or edge-disjoint, the pattern graph H can be either fixed or varied for different G.

Previous work upon these problems has resulted in the following findings: the general Subgraph Homeomorphism Problem is NP-complete, the fixed SHP for directed graphs has been completely solved by Fortune, Hopcraft, and Wyllie, and two polynomial time algorithms have been found for simple pattern graphs - two disjoint paths by Y. Shiloach and triangle by LaPaugh and Rivest - within the framework of the fixed SHP for undirected graphs.

The vertex-disjoint version of the SHP is NP-complete because it includes as a subproblem the Hamiltonian Circuit Problem. As a fixed SHP, though, pattern graph H is no longer part of the input; thus, a reasonable question to ask in the exploration of the boundary between those problems having polynomial time algorithms and those problems which are NP-complete can be stated as follows:

Instance: Let pattern graph H be an elementary cycle containing K vertices for a fixed value of K.

Input: Graph G = (V,E) and a set of K vertices of graph G onto which the vertices of H are to be mapped.

Question: Can this instance of the fixed (vertex-disjoint) SHP be solved in polynomial time for all values of K or is there some value of K for which this problem is NP-complete?

Solving this problem will help to solve other problems which have one of their parameters fixed, due to the frequency with which the Hamiltonian Circuit Problem has been used to prove that other problems are NP-complete.

Presented here are a variant of LaPaugh and Rivest's linear time algorithm for k = 3 (triangle algorithm) and a linear time algorithm k = 4 which was previously unknown. Both of these algorithms depend upon the

technique of generating a sufficient number of paths; so that, if an elementary cycle containing the K specified vertices exists, then it is necessary that one of these paths belongs to some elementary cycle containing the K specified vertices. Having found the image of one or more of the edges in H, the problem can be reduced to working on a homeomorphic mapping of a simpler pattern graph.

Variant of LaPaugh and Rivest's Triangle Algorithm

This problem was first solved by LaPaugh and Rivest, but their algorithm was more involved and included many cases. Their problem was stated as follows: given an undirected graph G, and three vertices of G, determine whether the three vertices lie on a common elementary cycle and construct that cycle if it exists. The variant of the algorithm here avoids all the case analysis by observing at one point that if such a cycle is to exist then one of the paths already generated must be one of the sides of the desired triangle. This observation reduces the pattern graph to two vertex-disjoint paths sharing a common endpoint which can easily be solved using a network flow algorithm.

Algorithm #1:

let the three specified vertices of G be A, B, and C.

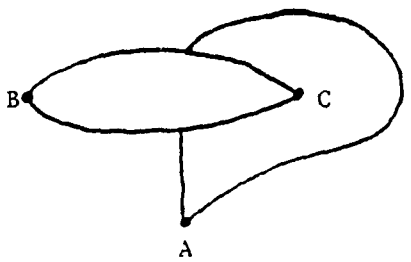
Step 1. Find an elementary cycle which contains vertices B and C. If no elementary cycle exists then the triangle can not exist.

Step 2. If A is not on this cycle, then designate A to be the source and all vertices on cycle BC to be sinks. Assign all vertices other than A to have capacity equal to one. If a flow of three exists then so does the triangle since

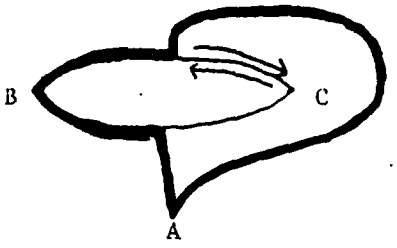
two of the three paths must intercept the same path linking B and C. If a flow of less than two occurs then the triangle can not exist since A, B, and C are not all in the same biconnected component.

Step 3. In this case a flow of two was found from A to cycle BC. Either the two intercept vertices (i.e., those vertices common to cycle BC and one of the paths from A) are on the same path linking B and C or else the intercept vertices are on different paths. In the first case an elementary cycle containg A, B, and C obviously exists so consider the latter situation.

Claim One of the six paths shown below between A and B, B and C, or A and C must exist in some elementary cycle containing A, B and C if indeed any such cycle does exist.



Proof Given the set of paths shown above which includes no triangle, one observes that if any one of the six subpaths is allowed to be traversed once in each direction then a triangle exists containing A, B, and C.

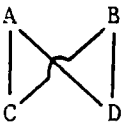


Each of these six subpaths creates a bottleneck preventing a triangle from existing among the paths generated so far. Only if a pair of vertex-disjoint paths having the effect of breaking the bottleneck exist, can triangle ABC exist in graph G. These two paths will intercept the same side of a cycle containing two of the specified vertices, thus leaving the other side of the cycle intact. Thus to find triangle ABC, assuming one exists, repeat the following for each of the six paths between A, B, and C. Assign all the interior vertices of one of the six paths capacity zero, thus effectively deleting these vertices. Designate the two endpoints of the chosen path to be sinks, and the remaining vertex of the trio A, B, and C to be the source. Assign capacity one to all vertices other than the source and those vertices which were "deleted". If a flow of two exists, then these two paths along with the "deleted" path form an elementary cycle containing A, B, and C.

Algorithm #2

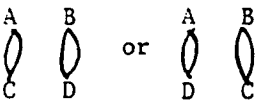
Given an undirected graph G, can an elementary cycle be found which includes four specified vertices A, B, C and D. The order of these vertices around the cycle is not specified.

Step 1. Assign A, B, C, and D to have capcity two, all other vertices have capacity one. Designate A and B as sources, and C and D as sinks. Find the maximum flow. Three possibilities exists:

a) 

flow of one from A to C, B to C, A to D, B to D

These four paths form the desired cycle.

b) 

The desired cycle may exist, go to step two.

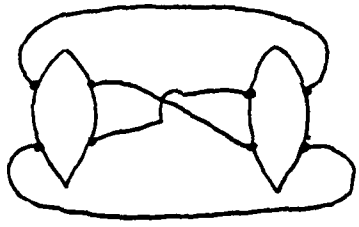
c) A flow of four is not possible. If the flow is less than two then the cycle can not exist. If the flow is two or three then go to step four.

Step 2. Case (b) from step one has occurred. Designate the vertices of one of the two cycles as the sources and the vertices of the other cycle as the sinks. Assign every vertex capacity to be one.

Find a flow of four if possible. If the maximum flow is less than two then the cycle can not exist. If the flow is two or three then go to step four.

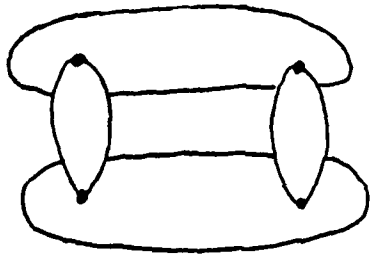
Step 3. A flow of four was found in step two. In step two, four vertices on each cycle were found which serve as endpoints of the four vertex-disjoint paths that connect the two cycles. If any of the four paths found in step one have three or four intercept vertices then a cycle exists; otherwise, each path has two intercept vertices. Three possible forms:

a)



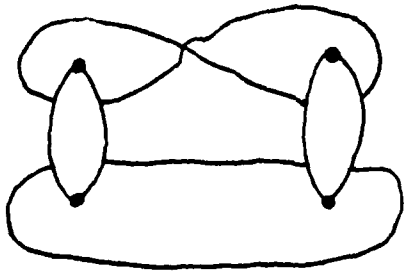
cycle exists

b)



cycle exists

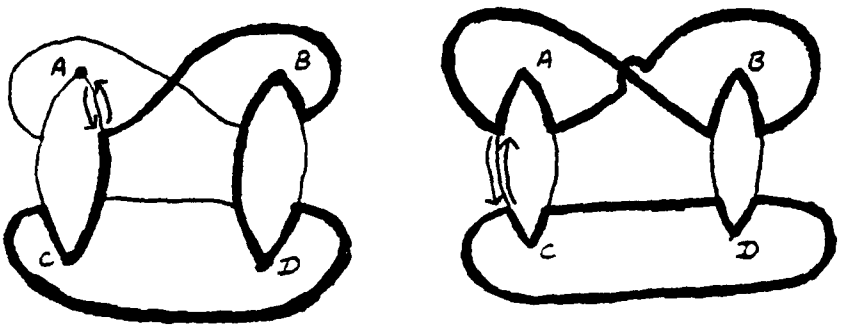
c)



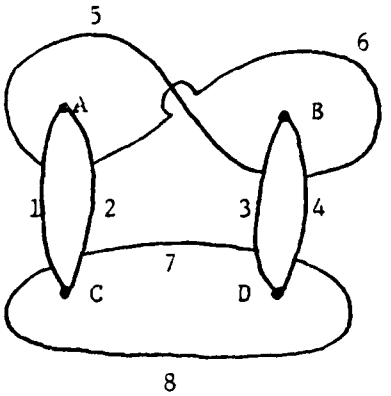
no elementary cycle exists among this set of eight paths

Claim If an elementary cycle which includes the vertices A, B, C and D exists in graph G, then an elementary cycle exists which includes two of the eight paths which have been generated between A, B, C, & D.

Proof Obviously, only case (c) from step three need be considered here since in all other cases a cycle exists among the eight generated paths. This set of paths is comprised of 16 subpaths, each of which acts as a bottleneck preventing the desired elementary cycle from existing among the set of generated paths. If any one of these subpaths is allowed to be traversed twice, once in each direction, then a cycle exists. Due to the symmetry involved the 16 subpaths belong to only two classes.



Only if a pair of vertex-disjoint paths having the effect of breaking the bottleneck exist, can an elementary cycle ABCD exist in graph G. In the above example, these two paths must leave intact either path CD and BD of the triangle or one path from each distinct cycle.



To find the desired elementary cycle, repeat for each of the following pairs of paths: 1+3, 1+4, 2+3, 2+4

5+7, 5+8, 6+7, 6+8

1+6, 1+7, 2+5, 2+8

3+6, 3+8, 4+5, 4+7

If the pair of paths form one path containing three of the four specified vertices, then "delete" the interior vertices of this path by assigning them capacity equal to zero. Assign the remaining specified vertex (source) capacity two, and all other vertices capacity one. The two endpoints of the "deleted" path are the sinks. If a flow of two exists then the desired elementary cycle exists which includes the "deleted" path and the two flow paths. If the pair of paths are vertex-disjoint, then "delete" the interior vertices of each path by assigning them capacity zero. Assign all other vertices capacity equal to one. Designate the endpoints of one of the two paths as the sources, and the endpoints of the other path as the sinks. If a flow of two exists then these two paths along with the "deleted" paths form an elementary cycle containing A, B, C, and D. Note that in place of the above procedure, the triangle algorithm could be used six times since at least one of the paths 1 through 6 must appear in the desired elementary cycle. This is true because at least one of these paths appears in each of the 16 pairs which are listed above.

Step 4. At this point, either two or three cut-vertices have been found which separate A, B from C, D or A, C from B, D. First, the case of two cut vertices, call them X and Y, these cut vertices split G into two or more components.

Suppose that the cut vertices separate A, B from C, D as the other case is treated similarly.

If A and B are in different components then no elementary cycle is possible; likewise, if C and D are in different components then again no elementary cycle is possible. Add a copy of X and Y to each component, and the edges incident to that component from X or Y. If a vertex-disjoint path from X to Y which includes A and B exists, and also a second vertex-disjoint path from X to Y which includes C and D then an elementary cycle containing A, B, C, and D exists. If either of these paths fails to exist then the desired cycle must fail to exist also. A slight variant of the triangle algorithm can be used to check for the existence of the two paths.

Secondly, the case of three cut-vertices, call them X, Y, and Z. There are three possibilities: X and Y, X and Z, or Y and Z are the vertices used to pass from one component to another. For each possibility, the third cut-vertex can also be on the elementary cycle. So, proceed as above for each possibility except now, the third cut vertex must be allowed to be on one of the two vertex-disjoint paths that might be found, but not both. Again suppose that the cut-vertices separate A, B from C, D.

a) if A, B, C, and D are all in different components then no cycle can exist.

b) if only one of these pairs (A and B or C and D) are in different components, then add the third cut-vertex and those edges incident to this vertex which are not incident to the component containing the pair of specified vertices. Next proceed as with two cut vertices.

c) If A and B are in the same component and so are C and D then there are two cases that must be allowed for:

1) the third cut-vertex is on the path from one cut-vertex to another which includes A and B

2) the third cut-vertex is on the path from one cut-vertex to another which includes C and D.

These two possibilities can be handled similarly to case (b).

case 1) Add the third cut vertex and the edges incident to any component but the one which contains C & D. Now proceed as with two cut-vertices.

case 2) Add the third cut vertex and the edges incident to any component but the one which contains A & B. Now proceed, again, as with two vertices. This gives a maximum of six cases, which means that the triangle algorithm might be performed as many as twelve times in step four.

3. The k-color path problem

A problem which we studied shall be referred to here as the k-color path problem, and can be stated as follows:

Instance - Given a network G as input, either directed or undirected with source s and sink t, K colors are used to "color" the remaining vertices such that each color is used for "coloring" at least one vertex

Question - Does there exist a vertex-disjoint path from s to t which includes at least one vertex of each color?

When K is not fixed, this problem is NP-complete, since one of the instances of the problem is $k = |V|$, which is the Hamiltonian Path Problem. On the other hand, for fixed $k = 1$ the problem is obviously polynomial. Somewhere in between there exists a distinct boundary separating the two classes.

A closely related problem is the Fixed Subgraph Homeomorphism Problem, where pattern graph H is a path with K vertices. This problem was completely solved by Fortune, Hopcroft, and Wyllie for directed graphs, and from their work it can be shown that the 2-color path problem for directed graphs is NP-complete. The problem of finding a vertex-disjoint directed path from source s to sink t which includes a specified vertex v, where neither edge (s,v) nor (v,t) exist, is one of the problems shown by Fortune, et al to be NP-complete. Applying the simple reduction of "coloring" the specified vertex red and all other vertices blue to any directed graph G given as input for the above problem. The existence of a polynomial-time solution for the directed 2-color path problem would contradict the fact that the above problem is NP-complete.

Is there a polynomial time algorithm for the 2-color path problem, if it is known that there are at least n vertices of each color in G for any fixed value of n? This problem can be shown to be NP-complete by reducing the directed 2-color path problem for n=1 to the same problem with any fixed value of n. Use the same instance as in the previous reduction where v is the only red vertex in directed graph G and the edges (s,v) and (v,t) do not exist. Then vertex v can be replaced by n copies of v such that each copy is adjacent to (from) the same set of vertices as v. If needed, blue vertices can also be added to graph G. Obviously, this problem can not be solved in polynomial time without also solving the directed 2-color path problem which is NP-complete.

Though, the directed 2-color path problem is NP-complete, there are instances of this problem which can be solved in polynomial-time. Observe that any directed 2-color path from source s to sink t must include at least one edge of E', where E' is the set of edges in G which have one red endpoint and one blue endpoint. Obviously, if E' is a cut-set which separates source s from sink t then the problem can be solved by using other breadth-first or depth-first search techniques, since any path from s to t is a 2-color path. The condition that E' is a cut-set is a sufficient condition for the solution of the 2-color path problem in polynomial time. In fact, we can also show the following sufficient condition:

Claim Given G = (V,E) whose vertices have been colored with two colors-red and blue. Assign capacity one to all edges. If a minimal cut-set of G is larger than that of a minimal cut-set for $G' = (V,E-E')$ then using max flow techniques a path can be forced to include at least one edge of E' in polynomial time.

The proof is not difficult. The Max Flow-Min Cut Theorem says that the number of edges in the minimum cut-set equals the value of the maximum flow; and, since the edges were defined to have capacity equal to one, the maximum flow value equals the number of edge-disjoint paths from source to sink. If there exists fewer edge-disjoint paths in G' than in G, then one of the edge-disjoint paths in G must contain an edge from set E'. If the two cut-sets have the same size, then possibly none of the paths include an edge from E'.

A restriction of the directed 2-color path problem which is solvable in polynomial time is to find a path from source s to sink t which contains only one edge from set E'. This restriction forces all the vertices of one color to appear along the path before any vertex of the other color; thus, it is impossible for the two monochromatic subpaths to share any vertices. To find a red-blue path (i.e., all red vertices precede any blue vertices) delete from G all arcs from the source to any blue vertices, from any red vertices to the sink, from the source to the sink, and finally all arcs from blue vertices to red vertices. Any path now from source to sink will be a red-blue path, and can be found using either breadth-first or depth-first search techniques. Blue-red paths can be found similarly.


DAT
FILM