ISOMORPHISM TESTING FOR GRAPHS OF BOUNDED GENUS[†]

Gary Miller
Department of Mathematics
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

I. Introduction

We present an algorithm which determines isomorphism of graphs in $v^{O(g)}$ steps where v is the number of vertices and g is the genus of the graphs. In [FMR 79] an algorithm was presented for embedding graph on surfaces of genus g in $v^{O(g)}$ steps. Here we show how to extend this algorithm to isomorphism testing for graphs of small genus. This result is noteworthy for at least two reasons. First, this extends the polynomial time isomorphism results for the plane [HT 72] and also the projective plane [L 80] to arbitrary surfaces. Second, this gives one of the few known natural decompositions of the isomorphism problem into an infinite hierarchy of problems $P_0, P_1, \ldots$ such that isomorphism testing of problems in P_i is decidable in time $v^{O(i)}$.

The computational complexity of isomorphism testing is one of the classical unresolved questions in theory of computation. Few computational problems have such a wide appeal and also have the property that we know so little about their computational complexity. Karp [Ka 72] presented three problems, Primality Testing, Linear Programming, and Graphs Isomorphism which are in NP but had not been shown either to be in P or to be NP-complete. These problems are not a complete list, but they are fundamental problems, which form examples for large areas of research. Recently Khachian [Kh 79] has given a polynomial time algorithm for linear programming, see [GL 79]. In [M 76] some evidence was given in favor of primality also being polynomial time decidable. This in some sense leaves only, of the three, graph isomorphism still unresolved. Graph isomorphism and factoring integers are the two outstanding problems.

[†]Research supported in part by the National Science Foundation under grant MCS 78-05853.

Embedding graphs on surfaces seems to be as old as graph theory itself, going back to the father of graph theory, Euler. Other than embedding graphs on the plane, research in embedding graphs has focused mostly on embedding very regular graphs, e.g. the complete graph. The other main focus has been on extending Kurotowski's forbidden subgraph theorem to surface of higher genus. The author feels one possible explanation for the lack of research in embedding arbitrary graphs is the divergence in the interests of graph theorist and topologist which has created a void in the amalgam. At the present time there seems to be no simple source of good notation and, or simple results. The notation used is simply an arbitrary scheme that the author has found convenient during the research of this paper.

In this paper we explicitly only handle the orientable case. Most of the ideas are presented to handle the unorientable. We leave a formal discussion of the unorientable case until the final version of this paper.

Similar work has been done by J.N. Mayer and I.S. Filotti [MF 80].

II. Codes and Canonical Forms

Before we prove the main results it is worthwhile pointing out (possibly) harder isomorphism problems which this and most other isomorphism algorithms solve.

Besides determining isomorphism between graphs we may want a unique code or a canonical form for graphs. We first make these two notations precise. Let C denote a class of presentation for graphs, say, incidence matrices. A function f from C to strings or the natural numbers is a code if for all $G, G' \in C$, $G \simeq G'$ iff $f(G) = f(G')$. The function f is succinct if length $(f(x)) = O[\text{length } (x)]^k$ for some constant k. We shall say a function f from C to C is a canonical form or a canonical labeling if 1) $G \simeq f(G)$ and 2) $G \simeq G'$ iff $f(G) = f(G')$. Note that canonical forms are codes. In order to clarify the definition given in [M 79] we shall say a code f is a certificate if the set $\{(G, f(G)) | G \in C\}$ is recognizable in polynomial time. Note that for "natural" presentations of graphs isomorphic graphs have the same size presentations. So, canonical forms are succinct.

Using standard reducibility technique it is easy to see that succinct codes and canonical forms are polynomial time equivalent. By minor modification most known isomorphism algorithms can be transformed into canonical form algorithms. The algorithm in this paper for isomorphism of graphs of bounded genus will be viewed as a procedure to produce a succinct code for these graphs.

III. Notations and Definitions

The definitions and theorems from [FMR 79] will be used extensively in this paper. For the sake of conciseness it is assumed that the reader is familiar with the paper. We present some of the definitions from the paper which are nonstandard or which make a technical distinction of commonly used words. The following definition of a graph which has more of a topological form and seems much easier to work with will be used.

Definition: A graph G is a triple (P,V,R) where

(1) P is a finite set where elements are called points.

(2) V is a subset of P whose elements are called vertices.

(3) R is an antireflexive and symmetric binary relation on P such that:

(3.1) No two vertices are related.

(3.2) Points in P-V are related to at most two other points.

(3.3) The connected components of (P-V,R) are called the edges of G. The edges are acyclic.

A graph is closed if every point in P-V is related to exactly two other points. An embedding I of a closed graph G is simply a cyclic orientation of the edges associated with each vertex of G. A pair (G,I) consisting of a closed graph and an embedding will be called an embedded graph often denoted G_I. In [FMR 79] we allowed split embedding. For this presentation they do not seem to be necessary. Thus, we shall assume that all embeddings are simple. A standard graph (V,E) consisting of a collection of vertices and a binary relation E on V can be transformed into the above definition of a graph as follows:

1) $P = V \cup E$

2) $V = V$

3) $(v,e),(e,v) \in R$ if $e \in E$, $v \in V$ and e contains v.

If U is a subset of vertices of G the star of U denoted S(U) is the subgraph of G consisting of the vertices U plus the edges common to at least one vertex in U.

IV. Isomorphic Embedded Graphs

Let G_I and G'_I be two embedded graphs; we shall say f is an orientation preserving isomorphism if f is an isomorphism of G onto G' and f preserves orientation, i.e., $I(x) = \langle e_1, \ldots, e_r \rangle$ implies $I'(f(x)) = \langle f(e_1), \ldots, f(e_r) \rangle$. Note that there are at most 2e orientation-preserving isomorphisms since any two such maps which agree on a chain xey must be the same map. We shall say that G_I is isomorphic to $G'_{I'}$ if there is an orientation-preserving isomorphism from G_I onto $G'_{I'}$. By the above remark we can test isomorphism of embedded graphs in $O(e^2)$ steps.

We can also quickly generate succinct codes for embedded graphs. Let list $(G_I, \vec{e})$ be some fixed systematic list of the edges and vertices of an embedded graph G_I starting from the edge e in G which is oriented; e.g. List $(G_I, \vec{e})$ is a depth first search starting from e where the priority for searching edges from some vertex is determined by I and the edge with which we first encountered x. To get a succinct code for G_I we simply take the minimum over all edges of G, i.e. Code $(G_I) = \min\{\text{List}(G_I, \vec{e}) | e \in G\}$. By a previous remark Code is a polynomial time succinct code for isomorphism of embedded graphs.

Hopcroft and Wong [HW 74] have shown how to construct a succinct code for embedded 3-connected planar graphs in linear time. It is open whether there are linear time constructable codes for non-planar embeddings.

The basic approach of this paper for checking isomorphism or generating codes is to simply find all minimal embeddings I of G and take minimum $\{\mathrm{Code}(G_I)\}$. This approach fails even in the planar case when the graph is not 3-connected for the number of embedding may be exponential in the number of vertices, but if the graph is simple and 3-connected then it can have at most 2 embedding in the plane. This fact is known as Whitney's theorem:

Theorem I: (Whitney) [W 33] A simple planar 3-connected graph has exactly 2 embedding in the plane.

This theorem of Whitney's is widely referenced in the literature but the author knows of no simple proof in print. We present a simple proof due to Edmonds [E pc].

Proof of Theorem: Let G_I be a planar embedding of 3-connected graph G. We need only give a characterization of the faces of G_I independent of I. But the faces of G_I are simply those cycles F of G such that G-F is connected. It is clear that if F is a cycle which is not a face then G-F is not connected by the planarity of G_I. Suppose F is a face of G_I and x,y are two vertices of G-F. Since G is 3-connected there exist 3 vertex disjoint paths from x to y. Now, one of these 3 paths must be disjoint from F since F is a face and G_I is planar. We need only consider the case when some component of G-F is an open edge e. Let x and y be the attachments of e on F. Since G is simple these two vertices cannot be consecutive elements of F. Thus x,y separate the vertices of F. Therefore, no component of G-F is an open edge.

If the graphs are not 3-connected then the number of embedding in the plane may be exponential in the number of vertices. A simple example is the n-bond (i.e. a graph on two vertices with n edges between these two vertices). We shall always identify multiple edges when we count the number of embedding (except in extension problems). So an n-bond will have but one embedding, namely as a 1-bond. In general a graph is decomposed into its 3-connected components.

Hopcroft and Tarjan were able to resolve the planar graph isomorphism problem by using the following facts:

1) The 3-connected components are unique and quickly computable.
2) The 3-connected components form a tree.
3) Tree isomorphism is quickly decidable.
4) 3-connected components have at most 2 embedding in the plane.

When the graph is not planar then 3-connectivity is not sufficient to force the graph to have only a "small" number of minimal embedding. Lichtenstein has exhibited 3-connected nonplanar graphs which have an exponential number of embeddings on the projective plane. We shall present a graph which has an exponential number of embeddings on the torus.

Note that, if a graph has only d minimal embeddings then the size of the automorphism group must be $\leq 2d \cdot e$ where e is the number of edges. So by simply exhibiting a graph with an exponential large automorphism group we have exhibited a graph with an exponential number of minimal embeddings.

Consider a graph on 4n vertices which we shall call the n-nest. As an example, consider a 3-nest in Figure 1.

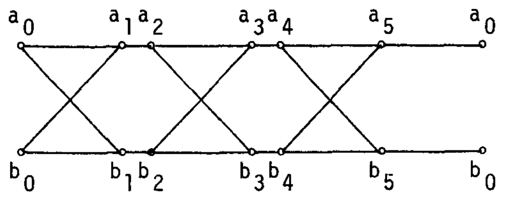

Figure 1

Now an n-nest is a cubic graph with a vertex transitive automorphism group. The vertex stabilizer is an elementary 2-group of size 2^n, $n \geq 3$. The size of the full automorphism group is $n\ 2^{n+2}$. Consider the embedding of a 4-nest on the torus in Figure 2.

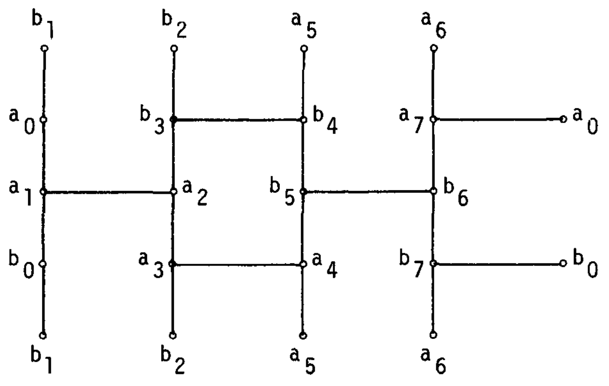

Figure 2

An important fact to notice about the embeddings of n-nest on the torus is that by removing only 2 vertices from the n-nest it becomes embeddable on a cylinder and hence the plane.

In order to solve the isomorphism problem we must introduce one more trick. We shall prove that a 3-connected graph having more than $v^{O(g)}$ minimal embeddings must have two vertices such that the graph minus these two vertices has strictly smaller genus.

Definition: An embedded graph G_I is k-stable if for all subsets U of k vertices of G

1) G-S(U) is connected

2) genus $(S-S(U)_I)$ = genus (G_I).

We shall say G_I is k-critical if it is not k-stable.

Using this notation and using the E-P formula for graphs one can prove the following lemmas:

Lemma 1: The embedded graph G_I is 1-critical if and only if there exists a vertex x of G which appears at least twice on some face of G_I.

Lemma 2: Let G_I be 1-stable; then G_I is 2-critical if and only if there exist two vertices x and y such that

1) x and y share an edge e implies x and y share a face not common to e

2) x and y share no edge but they share 2 faces.

We find the following definition convenient:

Definition: A graph G is k-stable if G is 3-connected and for all minimal embedding I the embedded graph G_I is k-stable. We shall say G is k-critical if it is not k-stable.

In the simple extension section we shall show that 2-stability is testable in time $v^{O(g)}$.

V. Constructing the 3-Connected Components

Throughout this paper we shall use the notation and definition of Hopcroft and Tarjan [HT 73] for 3-connected components of a graph. Hopcroft and Tarjan gave a linear time algorithm for uniquely decomposing a graph into its 3-connected components.

We shall need to attach labels to vertices and edges of our graph. So, we shall assume that a graph is partially edge labelled, partially vertex labelled and some edges are directed.

The algorithm (HT 73] divides a graph into a tree of components with the following properties:

1) The components of the tree T are either 3-connected homeomorphic subgraphs, cycles, or vertices.
2) Two adjacent components of T are related via a common edge or a common vertex.
3) The total number of new edges or vertices is linear in V.
4) If any vertex or edge of G is labelled and this vertex or edge is duplicated then all copies have this label.

We shall denote this procedure by 3-connected components (G).

We shall need a procedure which removes the star of a vertex x from a graph G. This procedure should modify the labels such that (1) the original graph is "quickly" reconstructable and (2) the construction is unique up to isomorphism. This can be done in many ways, so let Remove (G,x) be some fixed procedure which given a labeled graph G and a vertex x outputs the graph G-S(x) plus the appropriate labels satisfying the above conditions.

Let L be a leaf of a tree of components T with attachment vertex or edge p. We shall need a procedure which given L and p assigns a distinguishing label to p in L. In the case when p is a vertex, we simply assign a distinguishing label to p. If p is an edge we must keep track of the orientation of p. In the edge case we shall try both orientations of p and take the lexicographic minimal of the two codes of L. We fix some procedure $\text{label}_1(L,p)$ which satisfies the above conditions. Let $\text{label}_2(L,p)$ be the procedure which agrees with $\text{label}_1(L,p)$ except that it assigns the opposite orientation to p when p is an edge.

VI. Simple Extension Problems

In [FMR 79] it was shown that the embeddings of G of genus g can all be obtained by one of $v^{O(g)}$ simple extension problems. An extension problem is a pair (H_I,G) where H_I is an embedded subgraph of G. The extension problem was simple if H_I was quasiplanar and every component of G-H could be embedded in H_I in at most two ways.

In [FMR 79] we allowed some of the simple extension problems to be obtained from vertex splitting in H. If H_I has a proper vertex split at x and I is extendable to G then G-S(x) has genus at most g-1. Therefore G is 1-critical. If any of the simple extension problems for G are split embedding then we can simply guess a vertex of G whose removal decreases the genus of G. These remarks prove the following lemma:

Lemma 3: G is 1-critical if one of the $v^{O(g)}$ simple extension problems (H_I,G) is extended and I is a proper split embedding of H.

Testing 1-stability requires at most $v^{(g)}$ steps. We shall now characterize the 2-critical graph G and again get a $v^{O(g)}$ algorithm for testing 2-stability.

Lemma 4: A 3-connected graph G which is 1-stable is 2-critical if and only if one of the simple extension problems (H_I,G) has two vertices x and y satisfies lemma 2 and after adding two edges from x to y in H_I then I is still extendable to G.

This gives the following theorem:

Theorem: 2-stability is decidable in $v^{O(g)}$ time.

We shall say (H_I,G) is a split free extension problem if H_I has no split vertices. For 2-critical graphs we may guess two vertices whose removal decreases the genus of G. So we need only show how to handle the case when some 3-connected component of G is 2-stable. For these graphs we shall show that the number of embedding is not too large. Since there are at most $v^{O(g)}$ simple extension problems we need only show that these are at most $v^{O(g)}$ extensions per extension problem. For 3-connected graphs the distinct extension of a simple extension problem are precisely the instantiations of its 2-CNF formula. We state this as a lemma:

Lemma 5: If (H_I,G) is a quasiplanar extension problem and G is 3-connected then there is at most one embedding of a component of G-H in any face of H_I.

Proof: Let F be a face of H_I in which some component C is embeddable. Since H_I is quasiplanar F is a simple cycle. We can easily construct an embedded planar 3-connected graph L_I' which contains F as a face.

We first show that $F \cup C$ reduced is 3-connected. Let v,w be two arbitrary vertices of the reduced form of $F \cup C$. Note that all vertices of $F \cup C$ reduced are contained in the closer of C. We need only show that for any two vertices x,y distinct from v and w, x and y are in the same connected component of $(F \cup C) - \{v,w\}$. If v and w are both on F then x and y are connected via C. So we may assume that it is not the case that both v and w are points of F. Since G is 3-connected x and y are each 3-connected to F. Using these facts we have:

1) x is connected to $F - \{v,w\}$ in $F \cup C - \{v,w\}$
2) $F - \{v,w\}$ is connected
3) y is connected to $F - \{v,w\}$ in $F \cup C - \{v,w\}$.

So x and y are connected. We have shown that F C is 3-connected. By the previously mentioned theorem of Whitney's [W 33] 3-connected simple graphs have at most 2 embedding in the plane. After fixing F, $F \cup C$ must have only one embedding. This proves the result.

Since the only information about the components of G-H we shall use is their attachments to H we shall assume that the components are star-shaped.

Definition: A component of G-H is star-shaped or simply a star if it consists of an edge or the star of a vertex. We shall also require the attachments to be distinct.

VII. Algorithm

Using the procedures defined in the previous section we can now present an algorithm (in pigeon ALGOL) which when input a graph G outputs a succinct code for G.

```
Procedure: Code(G):
While G is 3-connected do
```

```
begin
  if G is 1-critical then guess a critical vertex x;
    Code(Remove(G,x))
  If G is 2-critical then
    1) guess a critical pair (x,y) in G;
    2) Code(Remove(Remove(G,x),y)
  else
    Code(G) ← Min {list($G_I$)|I minimal genus-embedding}
end
T←3-connected components(G);
While T is not a single component do
Comment Let $L_1,\ldots,L_k$ be the leaves of T with attachments $p_1 \ldots p_k$.
  begin
   for each $L_i$ do
    If Code($label_1(L_i,p_i)$) = Code($label_2(L_i,p_i)$)
    Then
      T←T minus $L_i$ [where Code(label($L_i,p_i$)) is
      added to the label of $p_i$ without orientation.].
   else
     1) pick j such that Code($Label_j(L_i,p_i)$) is
        the minimal of the two codes;
     2) T←T minus $L_i$ where Code($Label_j(L_i,p_i)$)
        added with orientation to the label of $p_i$.
  end
If T is a single component H then Code(G)←Code(H)
```

Roughly speaking, the algorithm decomposes a graph into 3-connected components for each component which is not 2-stable, it decomposes the component into a tree of 3-connected component. The algorithm continues in the manner until we have a nested set of trees such that the final components are 2-stable. Since each component on a level is a homeomorphic subgraph of the graph on the level above the genus of the component must be less than or equal to the genus of the graph it was derived from. On the other hand we only reconstructed 3-connected components when we have removed a vertex which strictly decreased the genus. So the nested trees of trees is at most g deep. Therefore if we show that step A) will only require $v^{O(g)}$ steps in the case when G is 2-stable we will have proved the main theorem:

Theorem 2: Isomorphism testing of graph of genus g can be performed in $v^{O(g)}$ steps.

So to prove the theorem we need only prove the following "structure" theorem.

Theorem 3: A graph which is 2-stable has at most $v^{O(g)}$ minimal embedding.

It is also important to observe that the embedding algorithm generates all embeddings in $v^{O(g)} + kn^{O(1)}$ steps where k is the number of embedding of genus g. The rest of the paper will prove Theorem 3.

VIII. Logical Components and a Theorem on 2-CNF

Let (H_I,G) be a simple extension problem and P be its corresponding 2-CNF formula. Let #(P) be the number of distinct satisfying truth assignments to P. By the last section we know that the number of extensions of (H_I,G) is precisely #(P). In this section we analyze #(P) in terms of "independent" variables of P. Note that #(P) is #P-complete [V 79] so we cannot hope, at the present time, to explicitly characterize #(P) but we can get good enough estimates on #(P).

Let A,B,C either be viewed as variables of P or the corresponding components of G-H. We shall say two variables A and B conflict if as components they conflict in H_I, i.e. they appear in some clause of P. The transitive closure of P, denoted $\overline{P}$, where the variables of P are $A_1,\ldots,A_n$ is:

$$\overline{P} = \bigcap\{(X\ Y) \mid P \mid (X \cup Y) \text{ and } X,Y \text{ are literals in the variables } A_1 \ldots A_n\}.$$

We can view P as a labeled graph over the variables $A_1 \ldots A_n$ as follows: For each pair of variables A,B in $A_1 \ldots A_n$ (1) add a directed edge from A to B if P contains $(\neg A \neg B)$ (2) add an edge labeled T from A to B if P contains $(\neg A \cup \neg B)$ (3) add an edge labeled F from A to B if P contains $A \cup B$. Let G(P) denote this graph. In general we shall say that a graph is a logical graph if its edges are either directed edges or undirected edges with labels T and F.

The transitive closure of a logical graph G is minimal graph $\overline{G}$ containing G which is closed under the following rules:

1) If $A \to B \to C$ is in $\overline{G}$ then $A \to C$ is in $\overline{G}$
2) If $A \to B \overset{T}{—} C$ is in $\overline{G}$ then $A \overset{T}{—} C$ is in $\overline{G}$
3) If $A \overset{F}{—} B \to C$ is in $\overline{G}$ then $A \overset{F}{—} C$ is in $\overline{G}$
4) If $A \overset{T}{—} B \overset{F}{—} C$ is in $\overline{G}$ then $A \to C$ is in $\overline{G}$.

It follows that $\overline{G(P)} = G(\overline{P})$. Since a 2-CNF formula and its graph are essentially the same we shall confine our attention to the graph.

Certain transformations of P or G(P) do not change #(P). We consider a few transformations we shall need. A switch of the variable A is the procedure which replaces every occurrence of A with $\neg A$ and every occurrence of $\neg A$ with A. For G this corresponds to simultaneously replacing subgraphs $A\rightarrow$, $A\leftarrow$, $A\overset{T}{-}$, and $A\overset{F}{-}$ with $A\overset{F}{-}$, $A\overset{T}{-}$, A , and A respectively. We shall say two logical graphs or two propositional formulas are equivalent if one is derived from the other by switching a collection of variables.

We shall say that two variables A and B are equivalent if $\overline{G(P)}$ contains $A\leftrightarrow B$ or $A\overset{T,F}{-}B$ (i.e. $\neg A\leftrightarrow B$). The number #(P) is independent of the size of these equivalence classes. By an appropriate switch of variables we may assume that A is equivalent to B if and only if $A\leftrightarrow B$. Once we have made this change of variables we can identify equivalent variables of P or G(P). Note that #(P) is unchanged. Some variables of $\overline{G(P)}$ may have selfloops, i.e. $A\overset{T}{-}A$ or $A\overset{F}{-}A$. In a natural way we can evaluate these variables. We shall ignore edges $A\rightarrow A$. After evaluating selfloops and identifying equivalent variables the new graph $\overline{G(P)}$ is simple, i.e. no multiple edges or selfloops.

A set of variables X are independent if they form an independent set in $\overline{G(P)}$. A variable with a selfloop is not independent. The independence number of P is the size of the largest independent set in P. Note that if P has independence number k then $\#(P) \geq 2^k$. We next show that the upper bound is not much worse.

Theorem 4 (Miller-Reid) If P is a 2-CNF in n variables with independence number $k \geq 1$ then $\#(P) \leq n^k+1$.

This bound does not seem to be tight. By taking k "chains" of size n/k we get a lower bound of $(\frac{n+k}{k})^k$. We make the following conjecture:

Conjecture: If P is a 2-CNF in n variables with independence number $k \geq 1$ then $\#(P) \leq (\frac{n+k}{k})^k$.

We prove Theorem 4 via a collection of lemmas.

Lemma: Every simple logical graph $\overline{G}$ is equivalent to a graph $\overline{G}'$ with only directed edges and edges labeled T.

Proof: The proof is by induction on the number of edges labeled F in $\overline{G}$. Suppose $A\overset{F}{-}B$ is contained in $\overline{G}$. Let X be the following variables of G:

$$X = \{C|A\rightarrow C\}\cup\{A\}.$$

We note two facts about X:

1) The variable $B \notin X$ since $B \in X$ implies $B\overset{F}{-}B$ but this contradicts the fact that $\overline{G}$ is simple.
2) If the variables $C, D \in X$ then $\overline{G}$ does not contain $C\overset{T}{-}D$ since this would simply the self-loop $A\overset{F}{-}A$. We need only prove the following claim.

Claim By switching the variables in X no new F's are created.

To prove the claim, we need only consider those edges common to a variable in X. Let $C,D \in X$ and $E \notin X$ then the following cases are transformed by the switch on X as follows:

$C \rightarrow D$ goes to $C \leftarrow D$
$C\overset{F}{-}D$ goes to $C\overset{T}{-}D$
$C \leftarrow E$ goes to $C\overset{T}{-}E$
$C\overset{F}{-}E$ goes to $C \rightarrow E$
$C\overset{T}{-}E$ goes to $C \leftarrow E$.

The other two cases were eliminated by properties of X and 2). The pair $A\overset{F}{-}B$ goes to $A \rightarrow B$ by 1).

-X-

Lemma If $\overline{G}$ is simple with no edge labeled F then there exists a variable A with indegree zero, i.e., $\rightarrow A$ is not contained in G.

Proof Pick a vertex A of G if A does not have indegree zero then pick a predecessor i.e., B such that $B \rightarrow A$ in $\overline{G}$. This process either cycles or it finds a variable of indegree zero. Since $\overline{G}$ is simple it cannot have a cycle.

-X-

Let $T_k(n)$ be the maximum number of satisfying instances over all formulas on n variable and independence number k. For simplicity let $T_0(n) = 1$.

Lemma $T_k(n)$ for $k \geq 1$ and $n \geq k$ satisfies $T_k(n) \leq T_k(n-1) + T_{k-1}(n-1)$.

Proof Let $\overline{G}$ be a simple logical graph with n variables and independence number k. By previous lemma we may assume that the labeled edges of $\overline{G}$ are label T. By the last lemma $\overline{G}$ contains a variable A with indegree zero. Let $\overline{G}(A/F)$ and $\overline{G}(A/T)$ be the simple graphs obtained by evaluating A to F and T respectively and evaluating all "induced" selfloops. Note that $\#(\overline{G}) = \#(\overline{G}(A/F))+\#(\overline{G}(A/T))$. Now, $\overline{G}(A/F)$ has independence number at most k and has at most n-1 variables. So, $\#(\overline{G}(A/F)) \leq T_k(n-1)$.

By assigning T to A, and the fact that all edges common to A are either of the form $A \rightarrow B$ or $A \overset{T}{—} B$, all variables which share an edge with A are forced to be evaluated to T or F. Thus the variables of $\overline{G}(A/T)$ have no edge in common with A and so the independence number k-1. This gives $\#\overline{G}(A/T) \leq T_{k-1}(n-1)$. This proves the lemma.

The following lemma gives theorem 4:

Lemma $T_1(n) \leq n+1$ and $T_k(n) \leq n^k$ for $k \geq 2$, $n \geq k$.

Proof Since a formula on n variables has at most 2^n truth assignments we have $T_k(k) \leq 2^k \leq k^k$ for $k \geq 2$. This gives one set of initial conditions. We first prove the cases k = 1 and k = 2. Now $T_1(n) \leq T_1(n-1) + 1$ by the previous lemma. So by induction $T_1(n) \leq n+1$. Now $T_2(n) \leq T_2(n-1) + T_1(n-1) \leq T_2(n-1) + n$. By induction $T_2(n-1) \leq (n-1)^2$ so $T_2(n) \leq n^2 - n$. This proves the case k = 2. To show inductively the general case, assume the lemma true for $T_k(n-1)$ and $T_{k-1}(n-1)$. Now $T_k(n) \leq T_k(n-1) + T_{k-1}(n-1) \leq (n-1)^k + (n-1)^{k-1} = n(n-1)^{k-1} \leq n^k$. This proves the lemma and hence the theorem.

A strong component of P or (H_I,G) will be a class of equivalent variables of P. While the weak components will be connected components of $G(\overline{P})$.

Lemma 7: If $\{S_1 \ldots S_k\}$ is a independent set of strong components of (H_I,G) then there is a partial extension such that $\{S_1 \ldots S_k\}$ are precisely the strong and weak components.

Proof: We simply evaluate all variables not in $S \ldots S_k$ such that they do not force variables in $S \ldots S_k$ using the fact that S_i are independent.

IX. Planar Strong Components

By the previous section we know that we can restrict our attention to simple extendable extension problem (H_I,G) such that the strong components are also weak components, and where each component of G-H is star shaped. By adding edges to H we can assume that the strong components span distinct faces of H_I. We shall call this a canonical extension problem.

If X is a set of components of (H_I,G) we can construct an embedded graph from X and (H_I,G) as follows:

1) For each $C \epsilon X$ construct two copies of C.
2) Embed these two copies in the two possible ways. If a pair of copies conflict or share a face then identify their vertices in that face.
3) Let 2X be the two copies of components after identification plus the attachment points of X on H. Let $(2X)_I$ be the above embedding restricted to 2X.

The graph 2X is bipartite. Those vertices not in H we shall call face vertices. While the vertices on H will be called attachment vertices. The faces spanned by X are the faces of H_I used by $(2X)_I$.

Lemma 8: If X is an independent strong component of (H_I,G) and $(2X)_I$ is planar then X only spans two faces and G has a critical pair.

Proof: By lemma 7 we can assume that G-H is X. (Since $2X_I$ is planar no two cycles of 2X can "crossover" so X must span only 2 faces used by X.) Let A be one of the two faces. Let $X_1 \ldots X_k$ be the attachments of X as they appear on A. For each pair (X_i,X_{i+1}) there is a unique cycle in 2X which passes through these two points. Since $(2X)_I$ is planar these cycles are precisely the faces of $(2X)_I$. If any of these cycles are not homologous to zero then we can simply remove the two corresponding vertices. In the case when all these cycles are homologous to zero they must partition the graph G-C. The graph G cannot be planar by Whitney's Theorem. So some pair (X_i,X_{i+1}) must be a separation pair for G but this contradicts the fact that G is 3-connected.

X. Counting the Number of Nonplanar Independent Strong Components

In the last section we showed that if G is 2-stable then no strong component of (H_I,G) can be planar. We shall now show that the number of nonplanar independent strong components of (H_I,G) is bounded by O(g).

Throughout this section let (H_I,G) be a simple extension problem of genus g where the components consist of k independent strong nonplanar components $X_1 \ldots X_k$.

We next prove a lemma which will extract a very simple embedded graph from each strong component $2X_i$. Let B_I, B_I' and B_I' be the following embedded graphs

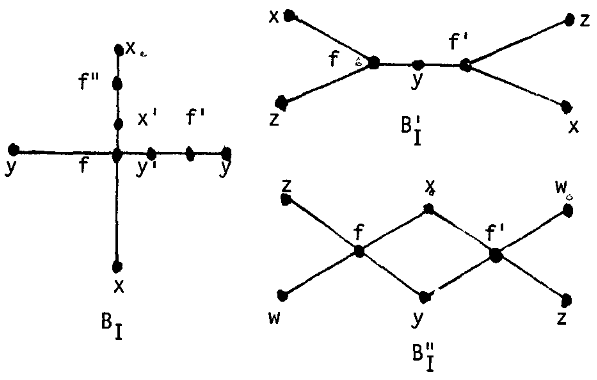

Figure 3

Lemma 9: If X is a strong component then $2X_I$ contains either an isomorphic copy of B_I, B_I' or B_I' where f, f' are face vertices.

Proof We consider two cases depending on the number of faces spanned by X.

Case I X spans 3 or more faces.

We form an equivalent relation on the component of X by relating all components which span the same two faces. Since X spans more than 2 faces X has more then one equivalence classes. Since X is a strong component there must be two equivalence class which each contain a component say D_1 and D_2 such that they conflict. Now the graph $2(D_1 \cup D_2)$ must contain two cycles C_1 and C_2 one in $2D_1$ and the other in $2D_2$ which conflict. The embedded graph $(C_1 \cup C_2)_I$ is isomorphic to B_I.

Case 2: X spans 2 faces.

Let A be one of the two faces spanned by X and let $x_1 \ldots x_t$ be the attachments vertices of X on A. Each pair (x_i, x_{i+1}) determines a cycle in $2X$ say C_i. Now the cycles of $C_1 \ldots C_{t-1}$ form a basis for the cycle space of $2X$. Since $2X$ is nonplanar we can pick two cycle C_i and C_j from $C_1 \ldots C_{t-1}$ such that C_i and C_j crossover i.e. $(C_i, C_j)=1$, see [Mta]. The embedded graph $(C_i \cup C_j)_I$ must be isomorphic to B_I' or B_I''.

Let $(B_1)_I, \ldots (B_k)_I$ be embedded graphs given by the last lemma for the strong components $X_1 \ldots X_k$. Let B be the union of B_1 to B_k. We shall assume that B is connected independently. Let g' = genus (B_I). These embedded subgraphs B_I, B_I', and B_I'' were chosen because they have no important properties:

1) The cycle space of the B's is generated by two cycles.
2) These two cycles cross over i.e., $(C_1, C_2) = 1$ for the two cycles C_1 and C_2 is B_i.

Thus, the 2k cycles $C_1, \ldots, C_{2k}$ where we choose pairs from each B_I form a basis for the cycle space of B. By basis arguments for vector spaces there must be a subset of $2g'$ cycles, say $C_1, \ldots, C_{2g'}$ such that genus $(C_1 \cup \ldots \cup C_{2g'}) = g'$. Since each C_i are contained in some B_j there must be at most $2g'$ B_j's which contain C_1 to $C_{2g'}$. Let L be the union of these B_j's. So $g(L_I) = g' \leq g$. We intend to view (L_I, B) as an extension problem but first we must enlarge L to a connected graph. Note that every cycle C_i not contained in L must "cross over" some cycle of L since otherwise using properties of interproduct spaces the genus of L could not be maximum. So each $B_i \epsilon B-L$ is connected to L. We can now construct a spanning tree over the B_i'sϵL using pairs of B_i's at a time. So by adding at most $2(2g'=1)$ more B_i's to L we can force L to be connected.

We can now view (L_I, B) as an "extension problem" where the B_i's & L are the components of $B-L$. We list all possible ways of embedding f of some $B_i \epsilon B-L'$ in L' where $L \subseteq L' \subseteq B$.

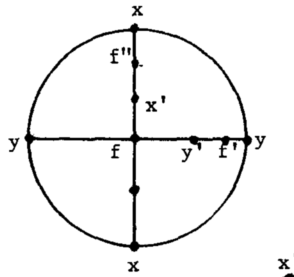

Fig. 4A

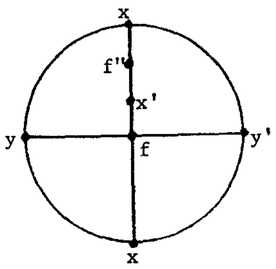

Fig. 4B

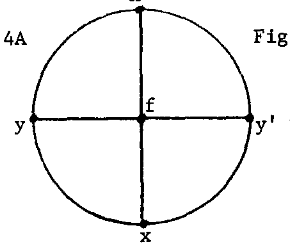

Fig. 4C

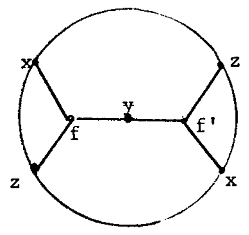

Fig. 4D

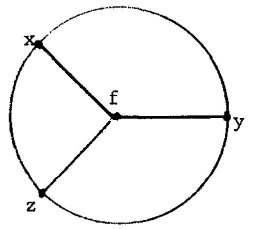

Fig. 4E

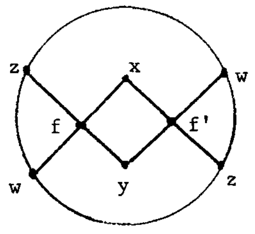

Fig. 4F

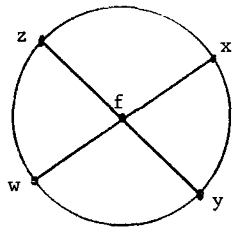

Fig. 4G

Let $B_i' \ldots B, B_t'$ be a new indexing of the old B_i's not in L. Define $L_i = \{L \cup B_1' \cup \ldots \cup B_i'\}$ for $0 \le i \le t$. Since the genus (L) = genus (B) we have genus $(L_0)_I = \ldots =$ genus $(L_t)_I$. The main technical fact which we still have to prove is that t=O(g'). We shall prove this fact using characters similar to the ones used in the analysis of the embedding algorithm [FMR 79]. Consider the following two characters not $L' = L_i$ for some $0 \le i \le t$:

1) $nf(L_I') = \sum_{nf(F) \ge 2} nf(F)-2$

2) $in(L_I') = \sum_{x \varepsilon F} (F(x)-1)$

where F varies over faces of L_I', x over non-face vertices of L_I', nf(F) = the number of non-face vertices of L' on F with multiplicity, and F(x) = the number of occurrences of x on F.

Since B is a bipartite graph with minimum cycle size 4 it must be the case that $nf(F) \ge 2$. For these graphs we can rewrite $nf(L_I') = e = 2f$ where e is the number of edges and f is the number of faces of L_I'. Let nf + 2in be the character $nf(L_I') + 2\,(in(L_I'))$.

We next prove a collection of lemmas which will give us the technical fact.

Lemma 10: $nf + 2in(L_I) \le 144g'$

Proof By the above remarks L consists of at most $2(2g'-1) + 2g' = 6g'$ B_i's. Now, $nf(L_I) = e - 2f \le e$. Since each B_i's can have at most 8 edges we have $nf(L_I) \le 8 \cdot 6g' = 48g'$. By a similar argument in $(L_I) \le 48g'$. This proves the lemma.

Lemma 11: The character nf + 2in is strictly decreasing i.e. $nf + 2in(L_i)_I > nf + 2in(L_{i+1})_I$ for $0 \le i \le t$.

Proof We prove the lemma by considering each of the cases A,...,G separately. We shall explicitly handle the case A and leave its other 6 cases for the reader.

Suppose B_{i+1} is embedded in $(L_i)_I$ as in Figure 4A. Now $nf(L_{i+1})_I - nf(L_i)_I =$ (# edges of B_{i+1}) - 2 (# of new faces) = 8 - 6 = 2. Consider $in(L_{i+1})_I - in(L_i)_I$. The difference is $x \le -2$ since the sum $in(X) = \sum_{x \varepsilon F} F(X)-1$ will decrease by at least 1, similarly in(Y) will decrease by at least 1. We have $nf+2in(L_{i+1})_I - nf+2in(L_i)_I \le -2$.

Using arguments similar to case A we get the following table of values: the values are upper bounds on the difference $C(L_{i+1})_I - C(L_i)_I$ for C equals the characters nf, in, nf+2in:

Case	Δnf	Δin	Δ(nf+2in)
A	2	-2	-2
B	0	-1	-2
C	-2	0	-2
D	0	-2	-4
E	-2	0	-2
F	0	-2	-4
G	-4	0	-4

Since the Δ's are strictly negative the lemma is proved.

Lemma 12: If $nf + 2in(L_i)_I = 0$ the $L_i = B$.

Proof: Since nf and in are nonnegative we must have that $nf(L_i)_I = 0$ and $in(L_i)_I = 0$. Now, $nf(L_i)_I = 0$ implies that $(L_i)_I$ has at most 2 non-face vertices per face. But, to embed a B_{i+1} to $(L_i)_I$ we must have 3 or more non-face vertices.

Using the above 3 lemmas we see that $t \leq 144g'$. In fact we prove $t \leq 72g' \leq 72g$. Thus we have proved the following theorem:

Theorem 5: If (H_I,G) is a simple extension problem which is 2-stable then the independence number $\leq 78g'$.

Using the last theorem and theorem 4 we can get an explicit bound for theorem 3. Namely, if (H_I,G) is a simple extension problem which is 2-stable then (H_I,G) has at most n^{78g} extensions.

Acknowledgments

It is a pleasure to thank L. Babai, J. Edmonds, I. Fillioti, I. Gessel, D. Lichtenstein, M. Reid-Miller and J. Reif for all their help. Without them this paper would not have been possible.

References

[E pc] J. Edmonds, Private Communication.

[FMR 79] I.S. Filotti, G. L. Miller and J. Reif "On Determining the Genus of a Graph in $O(V^{O(g)})$ Steps," STOC 1979.

[GL 79] P. Gacs and L. Lovasz "Khachian's Algorithm for Linear Programming" Stanford Technical Report #STAN-CS-79-750.

[HT 72] J. E. Hopcroft and R. E. Tarjan "Isomorphism of Planar Graphs (working paper) Complexity of Computer Computations (R. E. Miller,; J. W. Thatcher, eds) Plenum, 1972, 131-152.

[HT 73] J. E. Hopcroft and R. E. Tarjan "Dividing A Graph into Triconnected Components" SIAM J. Comput., Vol. 2, No. 3, Sept. 1973, 294-303.

[HT 74] J. E. Hopcroft and R. E. Tarjan "Efficient Planarity Testing." J. ACM Vol. 21, No. 4, Oct. 1974, pp. 549-568.

[HW 74] J. E. Hopcroft and J. K. Wong "Linear Time Algorithm for Isomorphism of Planar Graphs"; Proc. Sixth Annual ACM Symp. or the Theory of Computing (1974) 172-184.

[Ka 72] R. M. Karp, "Reducibility Among Combinatorial Problems, Complexity of Computer Computations (R. E. Miller and J. W. Thatcher, ed) Plenum Press, 1972.

[Kaz 64] N. N. Kazarinoff "Analytic Inequality", Holt, Rinehart and Winston, (1964).

[Kh 79] L. G. Khachian "Polynomial Algorithm for Linear Programming" Doklady Adademii Nauk SSSR, 1979, Vol. 244, No. 5, 1093-1096.

[L 80] D. Lichtenstein "Isomorphism for Graphs Embeddable on the Projective Plane", this Proceedings.

[M 76] G. L. Miller "Riemann's Hypothesis and Tests for Primality", JCSS Vol. 13, No. 3, Dec. 1976, pp. 300-317.

[M 79] G. L. Miller, "Graph Isomorphism, General Remark" JCSS, Vol. 18, No. 2, April, 1979, pp. 128-141.

[M ta] G. L. Miller, Poincare Intersection Numbers and Some Application" (to appear).

[MF 80] J. N. Mayer and I. S. Filotti "A Polynomial-time Algorithm for Determining the Isomorphism of Graphs of Fixed Genus", this Proceddings.

[V 79] L. G. Valiant, "The Complexity of Enumeration and Reliability Problems" SIAM J Comput. Vol. 8 No. 3, August 1979, pp. 410-421.

[W 33] H. Whitney, "A Set of Topological Invariants for Graphs," American J. Math 55 (1933) pp. 321-235.