# A SIMPLE TEST FOR PLANAR GRAPHS

**Wei-Kuan Shih[1] and Wen-Lian Hsu[2]**

[1] Department of Computer Science, University of Illinois, Urbana, Illinois, USA. The work of this author was done while employed as a summer research assistant in Academia Sinica.

[2] Institute of Information Science, Academia Sinica, Taipei, Taiwan, ROC

## ABSTRACT

Given an undirected graph, the planarity testing problem is to determine whether the graph can be drawn in the plane without any crossing edges. Linear time planarity testing algorithms have previously been designed by Hopcroft and Tarjan, and by Booth and Lueker. However, their approaches are quite involved. We develop a very simple linear time testing algorithm based only on a depth-first search tree. Our algorithm uses Kuratowski's theorem very explicitly. A graph-reduction technique is adopted so that the embeddings for the planar biconnected components constructed at each iteration never have to be changed.

## 1. Introduction

Given an undirected graph, the planarity testing problem is to determine whether there exists a clockwise edge ordering around each vertex such that the graph can be drawn in the plane without any crossing edges. Planarity testing is a very useful tool in many disciplines of computer science. Linear time planarity testing algorithm was first established by Hopcroft and Tarjan [1974] based on a "**path addition** approach". A "**vertex addition** approach", originally developed by Lempel, Even and Cederbaum [1967], was later improved by Booth and Lueker [1976] to run in linear time using a data structure called a "PQ-tree". Both of these approaches are quite involved. Furthermore, the partial planar subgraph constructed at each iteration needs to be continually modified to obtain a final planar embedding.

We develop a very simple linear time testing algorithm based only on a depth-first search tree. A graph-reduction technique is adopted so that the embeddings for the planar biconnected components constructed at each iteration never has to be

modified. The key to our approach is to add vertices according to a postordering obtained from a depth-first search tree.

## 2. Our Modified Vertex Addition Approach

In the two previous approaches, the partial subgraph constructed at each iteration is always connected. The st-numbering of Lempel et al's approach further requires that those vertices not added induce a connected subgraph. We adopted a "vertex addition" approach which only require that those vertices "not added" induce a connected subgraph. Thus a simple postordering of a depth-first search tree of G suffices.

To better illustrate the difference between our approach and those of others an analogy can be drawn as follows. In writing an article, it is difficult to proceed in a contiguous fashion without ever having to reorganize what has been written. However, if each individual idea is written down in a separate piece of paper, then it is relatively easy to put them together in the final article. Let $\mathbf{G_i}$ be the subgraph at the i-th iteration consisting of the first i vertices and those edges among them. In our approach $G_i$ may be disconnected, but the embedding for each biconnected component of $G_i$, once determined, is never changed.

Assume the given graph G is biconnected and the degree of each vertex is at least 3. To simplify the discussion we use a ***generalized forest representation*** $\boldsymbol{F_i}$ for each $G_i$. Each node in the forest $F_i$ represents either

(a) an original vertex of G (denoted by a ***v-node***) adjacent to vertices not in $G_i$.

(b) a biconnected component of $G_i$ whose planar embedding has already been determined (denoted by a ***c-node***).

Let $T^*$ be a rooted depth-first search tree of G with all edges directed away from the root. Assume the vertices of G are labeled by a postordering of $T^*$. Thus, the label of a parent is always larger than those of its children. All edges in $T^*$ are called ***tree edges*** and the other edges of G are called ***back edges***. A depth-first search tree of an undirected graph satisfies that all back edges must direct from a node to one of its ancestors.

The vertices are added one by one according to their ascending label. Consider the beginning of the i-th iteration of the algorithm at which vertex i is to be added to $G_{i-1}$. We need to determine an ordering of edges emanating from vertices in $G_{i-1}$ to i. If a vertex in the current forest has a back edge pointing to i, then there must exist a tree edge directed from i to its root. List the trees whose roots are adjacent to i in the current forest as $T_1, T_2, ..., T_r$. Let S be a subset of

vertices. Denote by ***G[S]*** the subgraph of G induced on S. Since i is an articulation vertex of the induced subgraph $H_i = G[\{i\} \cup T_1 \cup ... \cup T_r]$, $H_i$ is planar if and only if each $G[\{i\} \cup T_j]$ is planar, j = 1, ..., r. Hence, consider the embedding problem of each $G[\{i\} \cup T_j]$. We shall find an embedding of $G[\{i\} \cup T_j]$ that corresponds to a partial embedding of G. Denote the root of $T_j$ by $r_j$.

Define the ***external degree*** of a vertex at iteration i to be the the number of its neighbors among {i+1,...,n}. its root. To simplify our description we need to apply a **vertex contraction** procedure to eliminate some vertices in $G_{i-1}$ with external degree 0. This procedure is applied at each iteration so that at the end of that iteration only vertices with non-zero external degree are shown. Let $T_j'$ denote the contracted tree after the vertex contraction from $T_j$.

The contraction procedure starts by marking all vertices of $T_j$ that are adjacent to i (assume, by induction, all vertices with external degree 0 before the i-th iteration have already been eliminated). Note that a vertex which is not marked at this stage can become marked later when it is made adjacent to i through a contracted edge. Scan all marked vertices in ascending order (the neighbors of every vertex in G can be presorted in O(*m*) time). If a marked vertex u (_ $r_j$) is a leaf with external degree 0, then delete u and mark parent(u). The edge from parent(u) to i now represents the path: **parent(u)-u-i**.

During the contraction process, if an internal vertex u of $T_j$ becomes a leaf, then all of its descendants must have external degree 0. If several (but not all) children of a vertex u have been contracted when u is scanned, then the edge from u to i represents the connection between i and every vertex in those children subtrees. Each back edge from a vertex u in $T_j'$ to i actually represents the edge connection between i and all contracted vertices in a partial subtree of $T_j$ rooted at u. A pictorial description is shown in Figure 2.1.

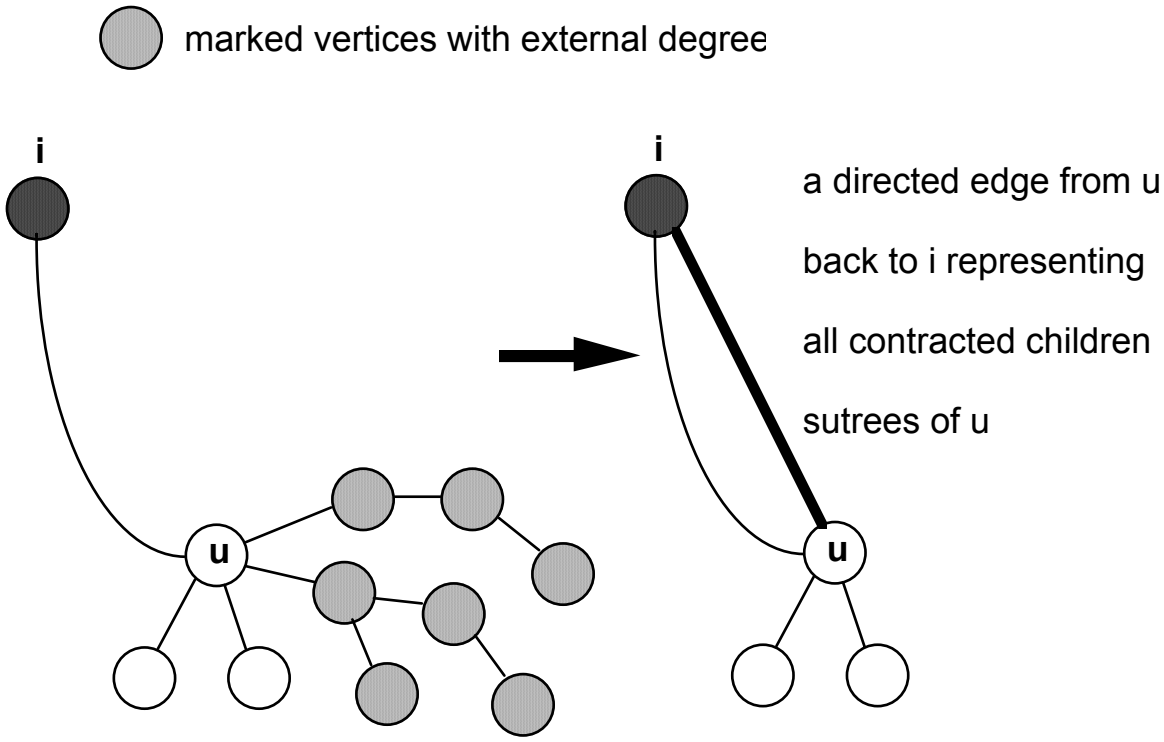
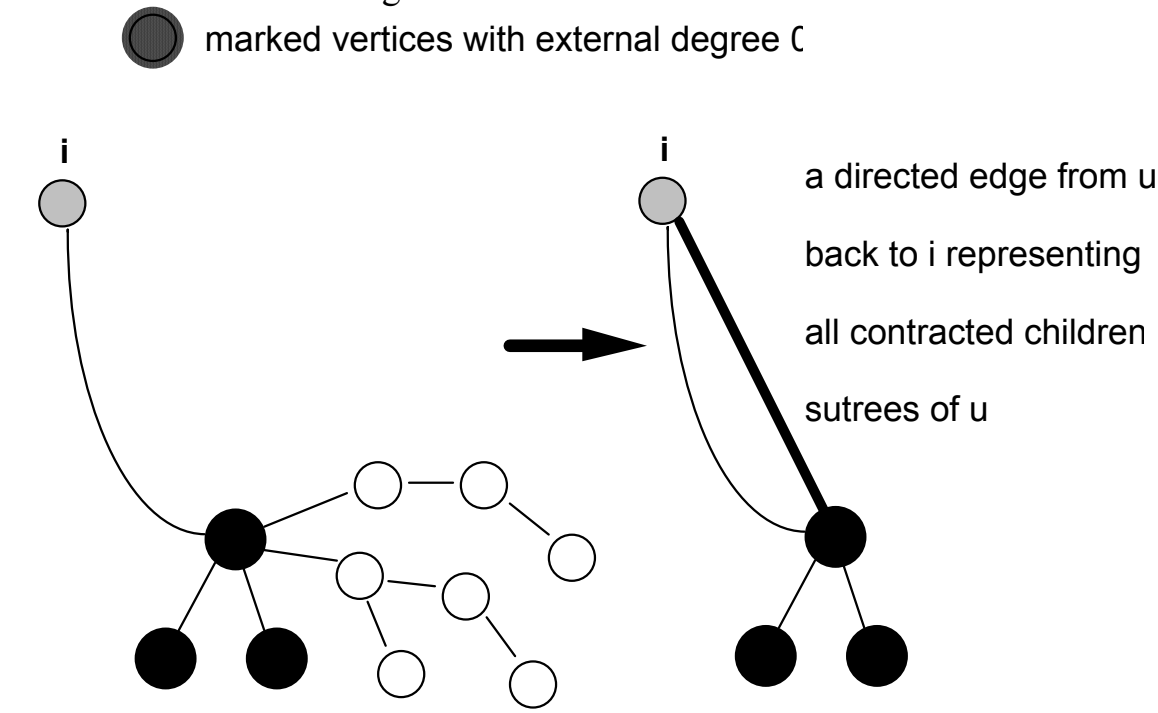



Figure 2.1 Vertex Contraction




Define a ***maximal adjacent path*** to be a path in $T_j'$ containing no c-nodes with as many nodes as possible from $r_j$ to a marked vertex. The following theorem illustrates the simplicity of the embedding algorithm after vertex contraction.

**Theorem 2.1.** *There exist at most two maximal adjacent paths in $T_j'$ which contain no c-nodes.*

**Proof.** If there exist more than two maximal adjacent paths, then pick any three of them. Denote the three end vertices (other than $r_j$) of these paths by 1, 2 and 3, respectively. Then each of them has a descendant (could be themselves) with a neighbor larger than i. Choose any three neighbors among {i+1,...,n} for 1, 2 and 3, respectively and let the medium vertex be t. Let the smallest of the three least common ancestors of {1,2}, {1,3} and {2,3} be r. Then a subgraph homeomorphic to $K_{3,3}$ can be found as shown in Figure 3.1. ¤

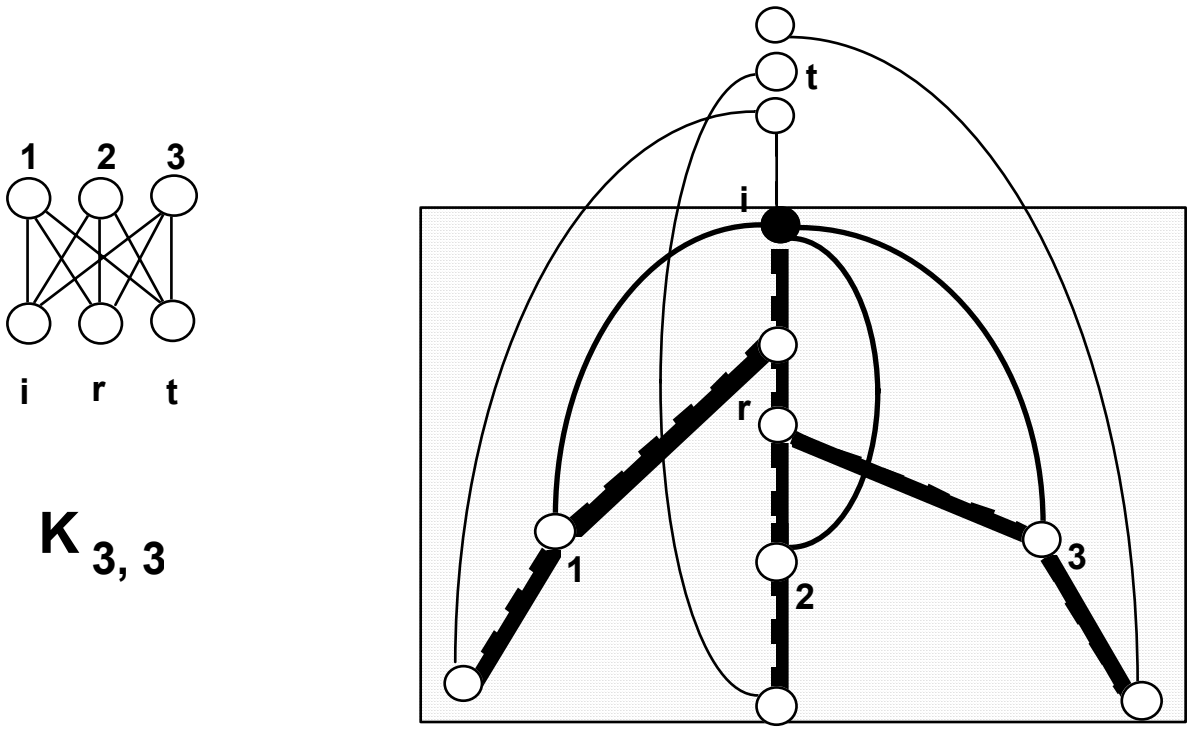



Figure 3.1. A Forbidden Structure

Now consider the embedding of a maximal adjacent path P in $T_j'$ from $r_j$ to a marked vertex u. We shall describe the embedding procedure in the next two sections.

## 3. A Special Case of the Embedding

Consider the special case that path P contains no c-nodes. The back edge (u,i) together with the tree edge $(i,r_j)$ and edges in P form a cycle **C**. Let $u_0$ (= u), $u_1$, ..., $u_t$ be the marked nodes in P.

**Lemma 3.1.** *Suppose G is a planar graph. Then there exists an embedding of G such that all vertices and edges contracted to the back edges $(u_1,i)$, ..., $(u_t,i)$ lie on the same side of C and all other vertices lie on the other side.*

**Proof.** Consider an arbitrary embedding of G. Cycle C separates the other vertices and edges of G into two parts. Since G[{i+1, ..., n}] is connected in G-C, they must lie on one side of C, say side 1. Similarly, those contracted vertices represented by an edge $(u_s,i)$ form a partial subtree of $T_j$ together with $u_s$. Hence, they must also lie on the same side. Contract those vertices as before into $(u_1,i)$, ..., $(u_t,i)$. Partition $\{(u_1,i), ..., (u_t,i)\}$ into $E_1$ and $E_2$, where $E_1$ is the set of edges on side 1 and $E_2$ are those on side 2.

Since those vertices of $T_j'$ not in P must be connected to descendants which are adjacent to vertices in G[{i+1,...,n}], they must also lie on side 1. Hence, $E_2$ is the only set of edges embedded in side 2. Thus, we can move all edges of $E_1$ in the current embedding and place them on side 2 without affecting planarity. □

Lemma 3.1 indicates a way to construct a planar embedding of the biconnected component **H** composed from C and those contracted vertices represented by $(u_1,i)$, ..., $(u_t,i)$. Namely, first embed the contracted edges $(u_1,i)$, ..., $(u_t,i)$ on the same side of C. Next, convert each edge $(u_s,i)$ back to its uncontracted partial subtree in $T_j$ and connect all vertices in the subtree to i. There are many different ways to achieve this. One way is to lay the subtree down on the plane (namely, specify a left-right children relationships) and to arrange a connecting order based on a preorder traversal.

To maintain the forest structure when more than one $T_j$ is considered (see, for example, Figure 3.2, when r = 4), make a copy $\mathbf{i_j}$ of i in each $T_j$ and change all back edges (u,i) from vertices in $T_j$ to $(u,i_j)$. Connect $i_j$ with i using a **virtual tree edge.** Hence, $i_j$ (instead of i) is placed into the cycle C. A tree edge of $T_j$, once becomes an edge in H, is no longer considered as a tree edge.

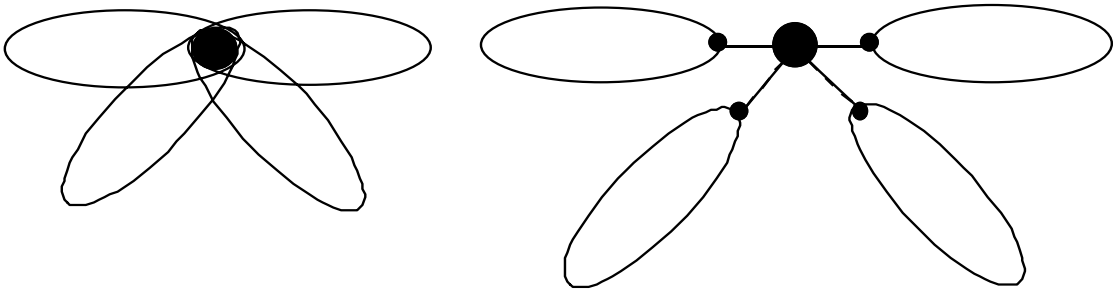


Figure 3.1. Virtual Tree Edges

To represent the biconnected component H in the forest after the embedding of path P, let $k_1$, ..., $k_s$ be vertices (ordered along the cycle) of C which are adjacent to vertices outside H (namely, either incident to a tree edge or adjacent to a vertex in {i+1,...,n}). Store $k_1$, ..., $k_s$ in a circular linked list as the ***representative cycle*** for H. H will be denoted by a c-node in the forest.

## 4. The General Embedding

Now, consider the general case in which path P contains some c-nodes. A node in P which is not the end node of P is called an **intermediate node** of P. In a path P the representative cycle of each intermediate c-node H contains exactly two vertices (called ***high P-vertex $h_2$*** and ***low P-vertex $h_1$***, respectively, according to their labels) adjacent to the two tree edges of P. Each v-node in P and each high, low P-vertex of a c-node in P is referred to as a P-vertex. For every non-P-vertex u in the representative cycle define the ***upward*** (resp. ***downward***) ***direction*** as the direction which leads u first to the high (resp. low) P-vertex along the cycle.

Obviously, only "one-half" of this cycle can be embedded inside the cycle formed by path P and the edge from the last marked vertex of P to i. It is quite easy to determine which half should be embedded inside. To simplify our description, we need to further contract vertices in the c-node . In the uncontracted $T_j$, a marked vertex u in a c-node is called ***saturated*** if (a) it has external degree 0; (b) all of its descendants outside the c-node have external degree 0.

We now generalize our vertex contraction to saturated vertices in c-nodes as follows. Let u be a saturated vertex in a c-node H of $T_j$. Let $u_1$, $u_2$ be the two neighbors of u in the representative cycle of H. If each of $u_1$ and $u_2$ is either a P-vertex or, a marked vertex with external degree 0, then delete u and connect $u_1$, $u_2$ by an edge. This edge now represents the path: $u_1$-u-$u_2$. Mark the neighbor of u in the downward direction. Let $H_1$, $H_2$ be the two boundary paths in H connecting $h_1$ and $h_2$. . It is possible that both of $H_1$ and $H_2$ are contracted to edges (when all non-P vertices in them are saturated). In that case, we allow temporarily two parallel edges. Let ***$T_j'$*** be the forest after the vertex contraction at the i-th iteration. With the presence of c-nodes, our ***maximal adjacent path*** is generalized to be a path in $T_j'$ with as many nodes as possible from $r_j$ to either a marked v-node or a c-node that contains a marked vertex.

The following characterization of c-nodes motivates our contraction strategy for vertices in the c-nodes.

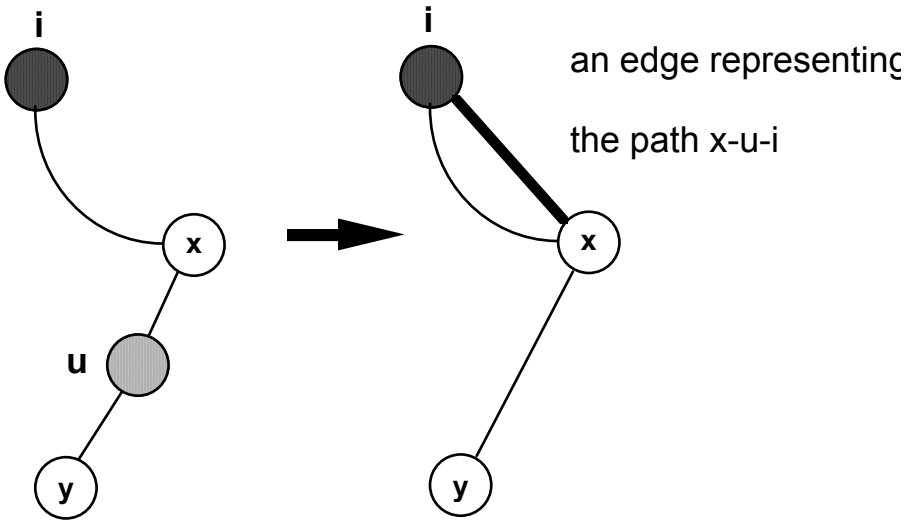


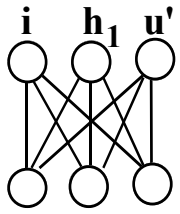

Figure 4.1. A Contracted Vertex in a C-node

**Lemma 4.1.** *Consider the uncontracted $T_j$. Let P be a maximal adjacent path. Let u be a marked vertex in a c-node H of a path P which is not a P-vertex. Suppose the low P-vertex in h has a descendant with non-zero external degree. Then every vertex of the representative cycle encountered by traversing in the upward direction from u to (and including) the high P-vertex is saturated.*

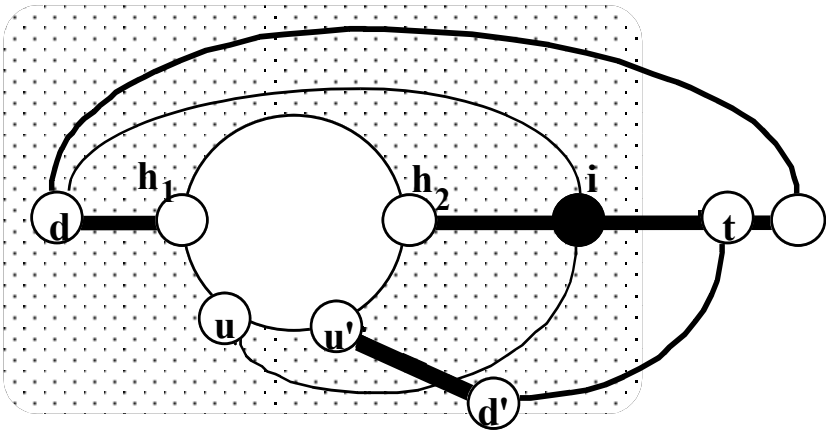
**Proof.** Let d be a marked descendant of $h_1$ (d could coincide with $h_1$) with non-zero external degree. Suppose u' is the lowest neighbor of u in the upward direction which has a descendant d' outside H with non-zero external degree. Let the low, high P-vertices be $h_1$ and $h_2$, respectively. In case u' _ $h_2$, arbitrarily choose two neighbors of u' and d that are greater than i, and let t be the smaller one. Then a subgraph homeomorphic to $K_{3,3}$ can be found as shown in Figure 4.1(a).

Hence assume u' = $h_2$. Arbitrarily choose three neighbors that are higher than i one for each of $h_1$, $h_2$ and u. List them from low to high as $i_1$, $i_2$ and $i_3$ (not necessarily distinct). In case d _ $h_1$, let t be $i_2$, then we have a $K_{3,3}$ as shown in Figure 4.1(b). Now, assume d = $h_2$. If the three vertices $i_1$, $i_2$ and $i_3$ coincide, let it be t, then we have a $K_5$ as shown in Figure 4.1(c) Otherwise, without loss of generality, assume $i_1$ _ $i_2$. Then we have a $K_{3,3}$ as shown in Figure 4.1(d) and (e) (a $K_{3,3}$ in every other case can be identified similarly).

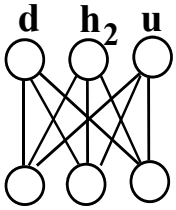
(a)



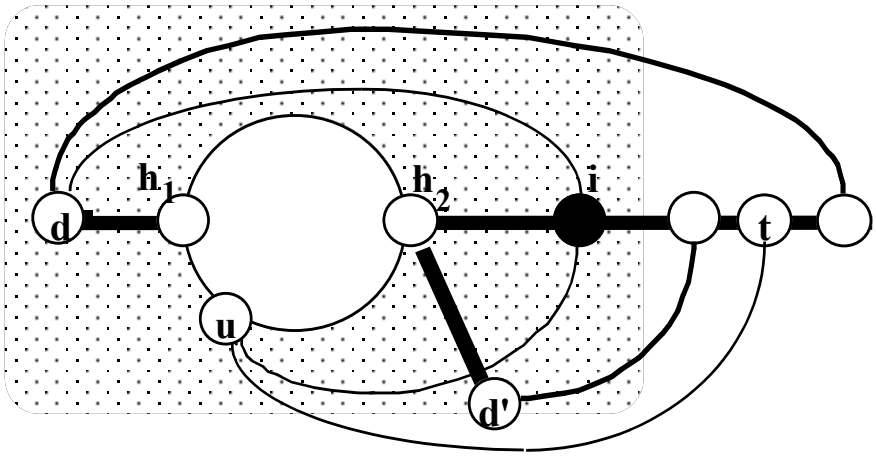

$K_{3,3}$




(b)



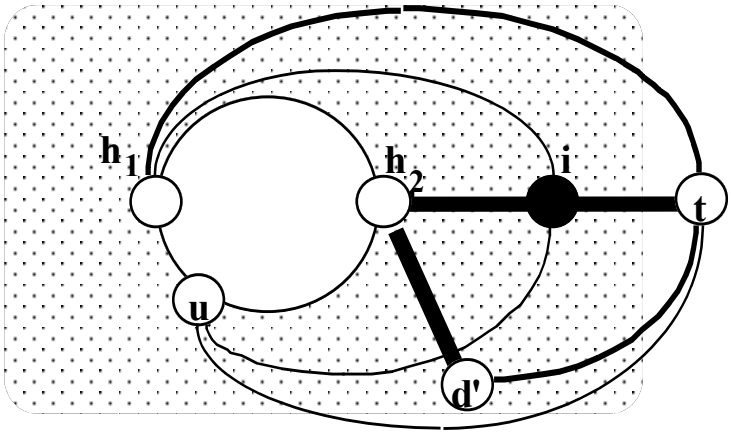

$K_{3,3}$




(c)

$AK_5$ on

$h_1$ , $h_2$ , t,u and i



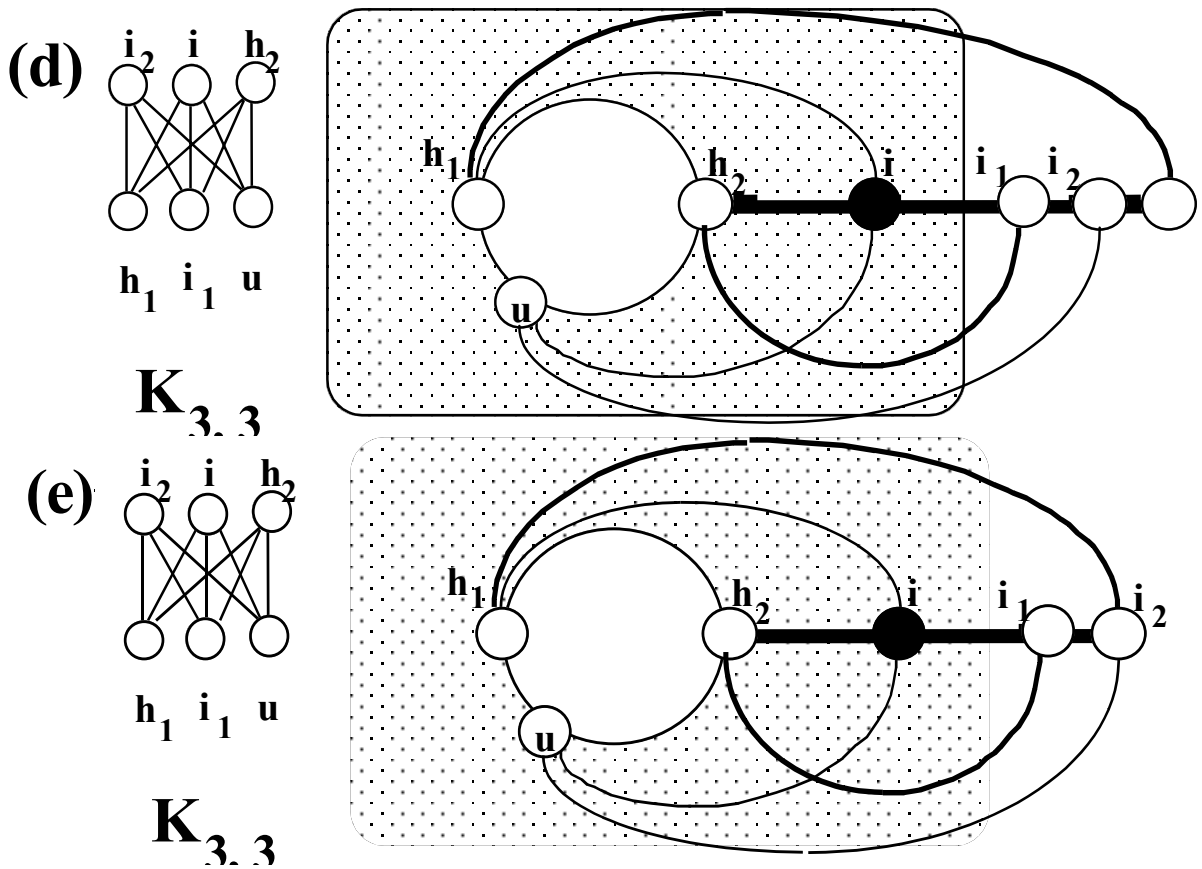



Figure 4.1. A Forbidden Structure

**Theorem 4.2.** *In the contracted $T_j'$, each intermediate c-node H in a maximal adjacent path P must satisfy that the two P-vertices are adjacent in its representative cycle. Furthermore, there exist at most two maximal adjacent paths.*

**Proof.** Suppose there is an intermediate c-node in a maximal adjacent path P whose P-vertices $h_1$, $h_2$ are not adjacent in the representative cycle of the c-node. Then there must exist two non-saturated vertices u, u' arranged as $h_1uh_2u'$ in the representative cycle of H. Hence, each of u and u' has a descendant adjacent to i. Let v (respectively, v') be the smallest descendent of u (respectively, u') on the uncontracted boundary of H which has a descendent adjacent to i. Let d be a descendant of $h_1$ adjacent to i. Let d' and d'' be descendants of v and v', respectively, adjacent to vertices greater than i. Arbitrarily choose two neighbors of d', d'' that are greater than i, and let t be the smaller one. Then a subgraph homeomorphic to $K_{3,3}$ can be found as shown in Figure 4.2.

If there exist more than two maximal paths, then a similar forbidden structure can be found as in the proof of Theorem 2.1. ¤

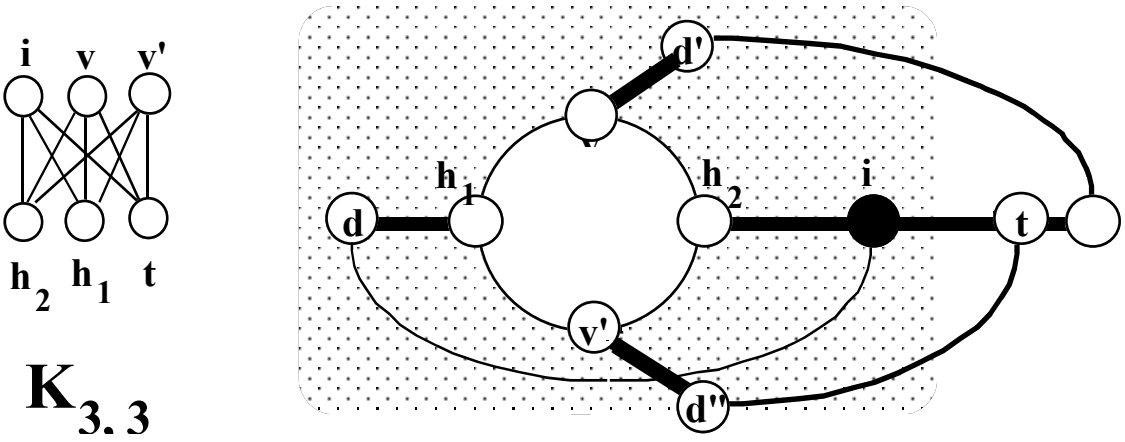



Figure 4.2. A Forbidden Structure

**Corollary 4.3.** *There exist at most two marked non-P vertices in the contracted* $T_j'$.

**Proof.** Suppose there exist three such vertices. ----

We now describe the embedding of a maximal adjacent path P in $T_j'$ from $r_j$ to a node k. Refer to the "outside face" as the one containing all vertices in {i+1,...,n}. The reason that these vertices must be contained in one face is that they form a connected induced subgraph of G. Let H be a c-node in P. Denote the representative cycle of H by **C(H)**. Let $\mathbf{h_1}$, $\mathbf{h_2}$ be the low, high P-vertices ($h_1 < h_2$) in C(h). By Theorem 4.2, $h_1$, $h_2$ must be adjacent. Denote the path formed by deleting the edge ($h_1$,$h_2$) from C(h) by $\mathbf{P_2(h)}$.

Unlike the special case that P contains no c-node (where a unique cycle is formed), we now form two cycles based on P. The **inner cycle** is used to construct an embedding of the resulting biconnected subgraph; the **outer cycle** is used to construct the representative cycle for the corresponding c-node. Consider two subcases:

**Case 1.** k is a v-node. Let the inner cycle be the union of (k,i), (i,$r_j$), the collection of tree edges of P together with the edge ($h_1$,$h_2$) from each c-node h of P. Let the outer cycle be the union of (k,i), (i,$r_j$), the collection of tree edges of P together with the path $P_2$(h) from each c-node h of P. The embedding of the inner cycle corresponds to case (i) of Figure 4.3.

**Case 2.** k is a c-node. If $k_2$ is the only marked vertex in C(h), then follow the procedure in Case 1 with k replaced by $k_2$. If $k_1$ and $k_2$ are the only marked vertices (then C(h) must be the end node of P by Corollary 4.3), let $c_1$ be the last vertex contracted into edge $(k_1,k_2)$ from $k_2$ in the downward direction that are adjacent to i ($c_1$ could be $k_1$). Then follow the procedure in Case 1 with k replaced by $c_1$.

Otherwise, some non-P-vertex in $P_2(k)$ is marked. Hence, all vertices of $T_j$ not in the subtree with root k must be adjacent to i with external degree 0. This means that they will all be contracted except for those on path P. It is easy to argue that P would be the only maximal adjacent path in this case. By Lemma 4.1, the set of marked vertices with zero external degree in C(k) must be arranged consecutively. Let $c_1$, $c_2$ be the two end vertices of this consecutive sequence. Let $P_1(h)$ be the path containing $k_2$ between $c_1$ and $c_2$ in C(k); let $P_2(h)$ be the path not containing $k_2$ between $c_1$ and $c_2$ in C(k). Let the inner cycle be the union of $P_1(h)$ and $(c_1,i)$, $(c_2,i)$; let the outer cycle be the union of $P_1(h)$ and $(c_1,i)$, $(c_2,i)$. The embedding of $C_1$ corresponds to case (ii) of Figure 4.3.

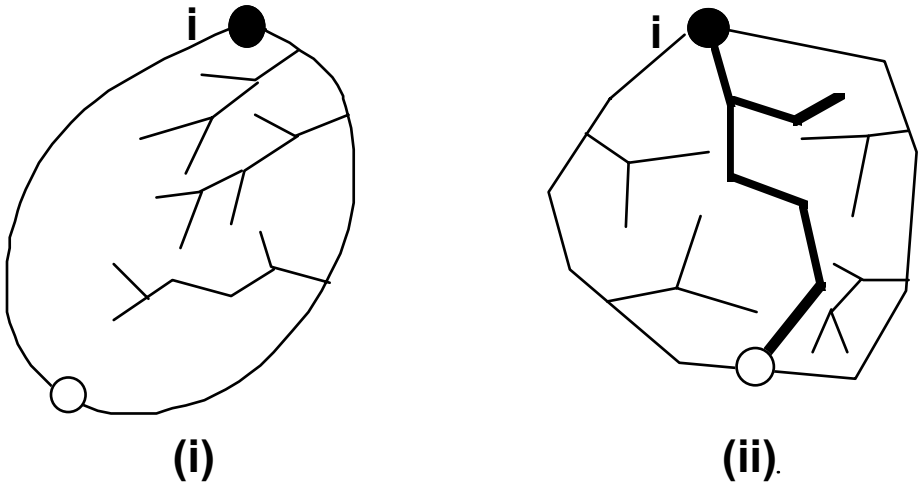


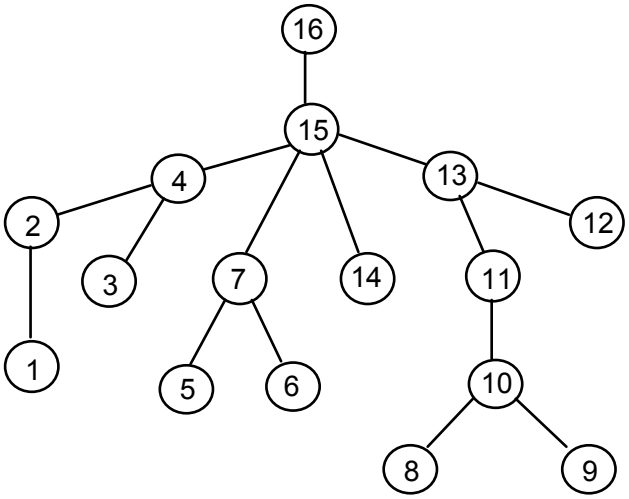

Figure 4.3. Two Possible Internal Embeddings

## 5. Complexity Analysis

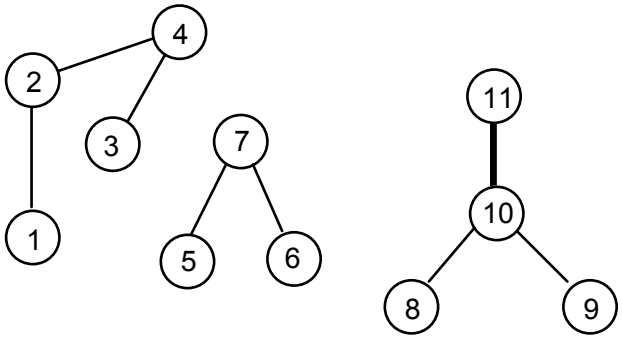
The construction of a depth-first search tree and a postorder traversal takes $O(m+n)$ time. At each iteration, Let the degree of i in $G_{i-1}$ be denoted by $\mathbf{deg_{Gi}(i)}$. Vertex contraction takes time proportional to $deg_{Gi}(i)$. The end node of a maximal adjacent path can be identified by checking the smallest marked vertex in $T_j'$. In the embedding of a maximal adjacent path P, each tree edge of P will become a non-tree edge. Summing over all iterations, the number of times a tree edge becomes a non-tree edge is at most $O(n)$.

The key saving in our algorithm is that, in each c-node h of a maximal adjacent path, the set of marked vertices with zero external degree in C(h) must be arranged consecutively and hence, can be easily determined. Furthermore, vertices in $P_2(h)$ never need to be traversed in composing the outer cycle. It suffices to find its two end vertices in the linked list. Therefore, besides traversing the tree edges in maximal adjacent paths (which totaled to be $O(n)$), the amount of work involved in each iteration is proportional to $deg_{Gi}(i)$. Hence, the complexity of the algorithm is $O(m+n)$.

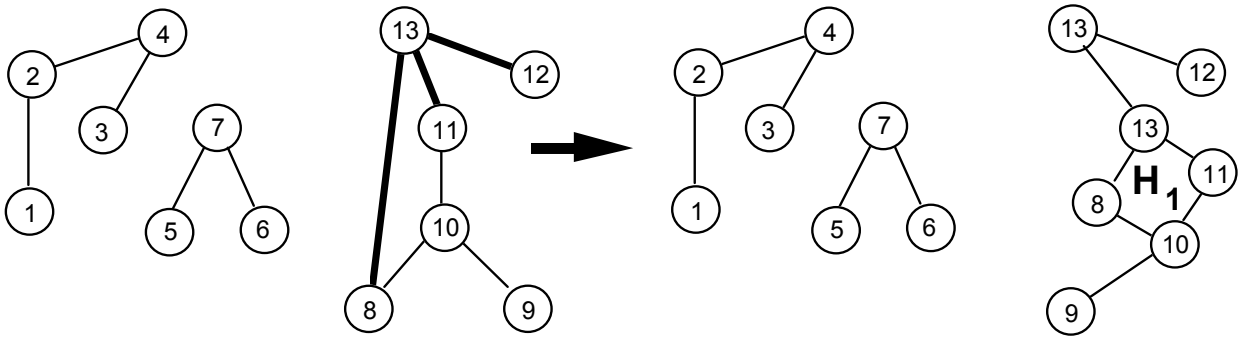
An example of the embedding algorithm is illustrated in Figure 5.1. Since all of the forbidden structure we have shown are $K_{3,3}$, the reader might be wondering when $K_5$ could arise. The answer is to apply the algorithm to $K_5$ directly. This forbidden structure arises when one tries to embed the second maximal adjacent path.




**A depth-first search tre**




**Iteration 11**




**Iteration 13**

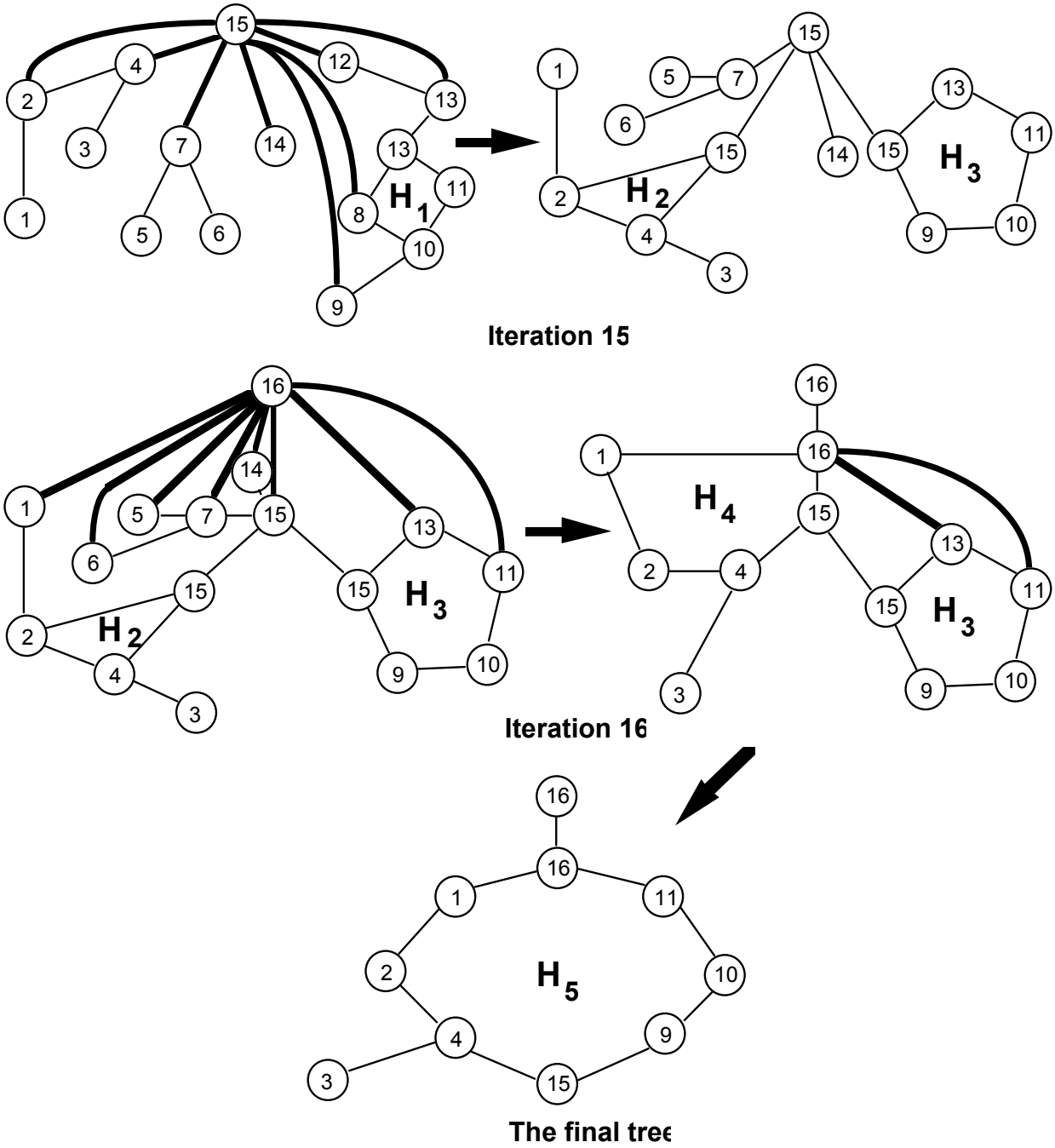



Figure 5.1. An Example for the Embedding Algorithm

## REFERENCES

1. K. S. Booth and G. S. Lueker [1976], *Testing the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms*, **J. Comput. Syst. Sci.** 13, 335-379.
2. J. E. Hopcroft and R. E. Tarjan [1974], *Efficient planarity testing*, **J. Assoc. Comput. Mach.** 21, 549-568.

3. A. Lempel, S. Even and I. Cederbaum [1967], *An algorithm for planarity testing of graphs*, **Theory of Graphs**, ed., P. Rosenstiehl, Gordon and Breach, New York, 215-232.