An Efficient algrithm to find All 'Bidirectional' Edges of an Undirected Graph.

B. Mishra,

Department of Computer Science, Carnegie-Mellon University
Pittsburgh, PA 15213

ABSTRACT. *An efficient algorithm for the All-Bidirectional-Edges Problem is presented. The All-Bidirectional-Edges Problem is to find an edge-labelling of an undirected network, $G = (V, E)$, with a source and a sink, such that an edge $[u,v] \in E$ is labelled $\langle u,v \rangle$ or $\langle v,u \rangle$ (or both) depending on the existence of a (simple) path from the source to sink that visits the vertices u and v, in the order u,v or v,u, respectively. The algorithm presented works by partitioning the graph into a set of bridges and analyzing them recursively. The time complexity of the algorithm is shown to be $O(|E| \cdot |V|)$.*

The problem arises naturally in the context of the simulation of an MOS transistor network, in which a transistor may operate as a unilateral or a bilateral device, depending on the voltages at its source and drain nodes. For efficient simulation, it is required to detect the set of transistors that may operate as bilateral devices. Also, this algorithm can be used in order to detect all the sneak paths in a network of pass transistor.

Introduction

Let $G = (V, E)$ be a finite undirected graph with two distinguished vertices, the *source*, s, and the *sink*, t. We call an edge $e = [u, v]$ of G 'bidirectional', if there are two simple paths connecting s and t and traversing e in either order — u, v and v, u. Similarly, we call an edge $e = [u, v]$ of G 'unidirectional', if every simple path connecting s and t and containing e, traverses e only in one order, say u, v; additionally, e is labelled $\langle u, v \rangle$. The All-Bidirectional-Edges problem is to find all the 'bidirectional' and 'unidirectional' edges of G, together with the labellings of the 'unidirectional' edges.

From an alternative formulation of the problem it is easy to see that we can label each edge $[u, v] \in E$ by asking two questions: (i) Are there two disjoint paths $s \xrightarrow{*} u$ and $v \xrightarrow{*} t$? (ii) Are there two disjoint paths $s \xrightarrow{*} v$ and $u \xrightarrow{*} t$? There are $O(|E| \cdot |V|)$ time algorithms to find two vertex disjoint paths in an undirected graph, independently discovered by Ohtsuki(1980)[7], Seymour(1980)[9] and Shiloach(1980) [10]. The naïve way of solving the All-Bidirectional-Edges problem is to invoke an algorithm for Two-Disjoint-Path twice per each edge. This takes $O(|E|^2 \cdot |V|)$. On the other hand, existence of an algorithm for All-Bidirectional-Edges problem, with a time-complexity lower than $O(|E| \cdot |V|)$, readily, results in a similarly efficient algorithm for Two-Disjoint-Paths problem, which is unlikely.

This work was supported in part by NSF grant MCS-8216706.

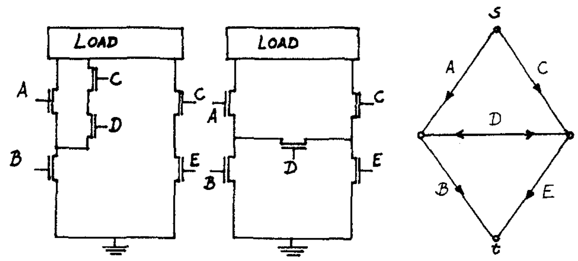

Figure 1: (a)) A circuit for $\neg\ ((A \wedge B) \vee (B \wedge C \wedge D) \vee (C \wedge E))$. (b)) The resulting subgraph after common subcircuit elimination. However, because of the *sneak path* through D, it is, in fact, a circuit for $\neg\ ((A \wedge B) \vee (B \wedge C \wedge D) \vee (C \wedge E) \vee (A \wedge D \wedge E))$. (c)) The corresponding graph in which D is 'bidirectional' and all other edges are 'unidirectional.'

In this paper, we describe an $O(|E| \cdot |V|)$ time algorithm for All-Bidirectional-Edges problem, using a completely new approach. The algorithm makes a novel use of *bridges* of a circuit in a *general* graph.

The problem of finding all 'bidirectional' edges arises naturally in the context of the simulation of an MOS transistor network, in which a transistor may operate as a unilateral or a bilateral device, depending on the voltages at its source and drain nodes. (Cf. Brand (1983)[2], Frank (1984)[5].) For efficient simulation, it is important to find the set of transistors that may operate as bilateral devices. Also, sometimes it is desired that information propagates in one direction only, and propagation in the wrong direction (resulting in a *sneak path*) can cause functional error. (See Figure 1) Our algorithm can be used to detect all the sneak paths.

1. Preliminaries

This section introduces some useful graph theoretic terms. The term 'bridge' is taken from Tutte(1977)[13].

0272-5428/84/0000/0207$01.00 © 1984 IEEE

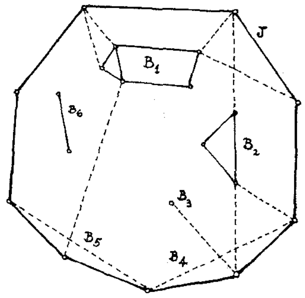

Figure 2: Bridges of J. Bridges B_1, B_2, B_3 and B_6 are proper bridges, and B_4 and B_5 are degenerate bridges. The bridges B_1 and B_2 interlace.

Definition 1.1: A *vertex of attachment* of a subgraph H of G is a vertex of H that is incident in G with some edge not belonging to H. Let J be any circuit of G. We define a *bridge* of J in G as any subgraph B that satisfies the following three conditions:

1. Each vertex of attachment of B is a vertex of J.
2. *B is not a subgraph of J.*
3. *No proper subgraph of B satisfies the above conditions.* □

Definition 1.2: (Cf. Figure 2.) An edge $e = [u, v]$ of G not belonging to J but having both ends in J is referred to as a *degenerate* bridge. Let G^- be the graph derived from G by deleting the vertices of J and all their incident edges. Let C be any component of G^-. Let B be the subgraph of G obtained from C by adjoining to it each edge of G having one end in C and one in J, and adjoining also the ends in J of all such edges. The subgraph B satisfies the conditions to be a bridge. Such a bridge is called *proper.* The component C of G^- is the *nucleus* of B. □

Definition 1.3: Let the vertices of attachment of a bridge B of J be $W(G, B) = \{v_0, v_1, \ldots, v_{k-1}\}$, where v_0, v_1, ..., v_{k-1} is their enumeration in the cyclic order on J. The vertices of attachment dissect J into k subpaths L_0, L_1, ..., L_{k-1} such that $L_j = J[\![v_j; v_{j+1 \pmod k}]\!]$. These subpaths are called the *residual paths* of B in J. $w(G, B)$ will stand for $|W(G, B)|$. □

Definition 1.4: Let B_1 and B_2 be two distinct bridges of a circuit J of G.

• We say B_1 *avoids* B_2 *if and only if* one of the following two conditions is satisfied:

1. $w(G, B_1) \leq 1$ or $w(G, B_2) \leq 1$.
2. *All the vertices of attachment of B_1 are contained in a single residual path L of B_2.*

• If B_1 and B_2 do not avoid one another we say that they *overlap.*

• If there exist two vertices of attachment x_1 and x_2 of B_1 and two vertices of attachment y_1 and y_2 of B_2, all four distinct, such that x_1 and x_2 separate y_1 and y_2 in the circuit J, then we say that they *interlace.*

• If B_1 and B_2 have exactly the same set of vertices of attachment we say that they are *equivalent.* □

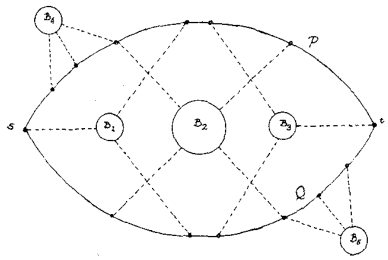

Figure 3: B^{PQ}-, B^P- and B^Q-bridges of P and Q. Bridges B_1, B_2 and B_3 are B^{PQ}-bridges; B_4 is a B^P-bridge and B_5, a B^Q-bridge.

Definition 1.5: Let J be a circuit of the graph G. A path N in G avoiding J but having its two ends x and y in J is called a *cross-cut* of J from x to y. □

Definition 1.6: (Cf. Figure 3.) Let J be a circuit of the graph, G such that s and t lie on J. Let the subpath $J[\![s; t]\!]$ be P and its complementary subpath in J be Q. The bridges of J are classified as follows:

•B^{PQ}-BRIDGES: *The set of bridges with at least one vertex of attachment on $P]\!]s; t[\![$ and at least one vertex of attachment on $Q]\!]s; t[\![$.*

•B^P-BRIDGES: *The set of bridges with at least one vertex of attachment on $P]\!]s; t[\![$ and no vertex of attachment on $Q]\!]s; t[\![$.*

•B^Q-BRIDGES: *The set of bridges with no vertex of attachment on $P]\!]s; t[\![$ and at least one vertex of attachment on $Q]\!]s; t[\![$.* □

Definition 1.7: Let J, P and Q be as in the previous definition. Then J is called an *ambitus* if every B^P- or B^Q-bridge avoids every B^{PQ}-bridge. □

Definition 1.8: Let J, P and Q be as before and $\mathcal{B} = B_1, \ldots, B_k$ be the B^P-bridges with respect to P. A non-empty subset of bridges $\mathbf{B}_n \subseteq \mathcal{B}$ is called a *block* of B^P-bridges if it satisfies the following two conditions:

1. *If $B_i \in \mathbf{B}_n$ and B_i and B_j overlap, then $B_j \in \mathbf{B}_n$.*
2. No non-empty proper subset of $\mathbf{B}_n$ satisfies the preceding condition. □

Definition 1.9: Let $G = (V, E)$ be a graph. The *tree* $T = (V', E')$ of the graph G is an undirected graph such that: The set of vertices V' correspond with the nonseparable components (*green*) and the separation vertices (*blue*); and the edges of T connect green vertices to blue vertices *if and only if* the nonseparable component contains the separation vertex. □

Definition 1.10: A graph $G = (V, E)$ with distinguished vertices s and t, $(G; s, t)$ is said to be a *chain graph,* if the tree of the graph G is a path P from C_s to C_t consisting of alternating green and blue vertices, where C_s and C_t are the nonseparable components of G containing the vertices s and t, respectively. □

We present the following propositions that will be used quite often later on.

Proposition 1.11: (Even[3]) *Let x, y and z be three distinct vertices of attachment of a bridge B of J in G. Then there is a vertex v belonging to the nucleus of B for which there are three internally vertex disjoint paths in B: $Y_1[\![x;v]\!]$, $Y_2[\![y;v]\!]$ and $Y_3[\![z;v]\!]$.* □

Following Tutte[14], we define a *Y-graph* as the union Y of three paths Y_1, Y_2 and Y_3 which have one end v in common but are otherwise mutually disjoint. We call v the *center* and the paths Y_i the *arms* of Y.

Proposition 1.12: *Let x, y and z be three distinct vertices of attachment of bridge B of J in G and let e be an edge of B such that there is a cross-cut of J between x and y containing e. Then at least one of the two following conditions is satisfied: (1) there is a cross-cut of J between x and z containing e (2) there is a cross-cut of J between y and z containing e.*

PROOF. The proof is similar to that of the proposition 1.11 and hence, omitted. □

Proposition 1.13: (Tutte[13]) *Let B_1 and B_2 be distinct overlapping bridges of a circuit J of G. Then either B_1 and B_2 interlace or they are equivalent 3-bridges.* □

2. Overview

For the ease of exposition, the algorithm is presented in two parts: The first part is simple but provides an algorithm of time-complexity of $O(|E| \cdot |V|^2)$ (to be improved later); it also identifies a graph of suitable structure (called a TYPE.IV graph) such that an efficient algorithm for such graph yields an efficient algorithm for general graphs. The second part, on the other hand, is more complicated but provides an $O(|E| \cdot |V|)$ time algorithm for a TYPE.IV graph, which, in turn, implies an $O(|E| \cdot |V|)$ time algorithm for the general graphs.

In the first part (Section 3), we introduce four different classes of graphs: TYPE.I, TYPE.II, TYPE.III and TYPE.IV graphs, of successively simpler structure and show that we can label the graphs of a particular type efficiently, on the assumption of existence of efficient algorithms for the subsequent types. In the process we give a set of four *mutually*† recursive algorithms such that these, together with an algorithm for TYPE.IV graph of time complexity $O(T(|E|, |V|)) \geq O(|E| \cdot |V|)$, will result in an $O(T(|E|, |V|))$ algorithm for general graphs. The first part concludes with an $O(|E| \cdot |V|^2)$ algorithm for TYPE.IV graphs.

In the second part (Section 4) we sketch an $O(|E| \cdot |V|)$ time algorithm for TYPE.IV graph. This is a Divide-and-Conquer algorithm and relies on an important concept, called a *U-Fragment.* Intuitively, a U-Fragment can be thought of as a *super node* which can be entered or exited through its end vertices (s' and t') or its upper and lower external attachments. The task is to find suitable vertex disjoint paths in this super node, connecting end vertices and external attachments. These paths can be extended to appropriate vertex-disjoint paths in the outer U-Fragment (recursively) or in a TYPE.IV graph. Such paths are guaranteed to exist *if and only if* the U-Fragment is *feasible* (Definition 4.2) and has certain disjoint pairs of Cross-Cuts. The main idea of the algorithms is to find such paths and to use them to label the edges.

†. Analysis of a TYPE.III graph may result in a call to the main algorithm.

3. A Simple Algorithm

3.1. Assumptions and Classification

Assumption 3.1: The graph, G, is a finite connected undirected strict graph. □

Assumption 3.2: The graph, G, is a chain graph. □

Justification for the Assumption 3.2. Let P be a path in the tree of G, $T(G)$ from C_s to C_t consisting of alternating green and blue vertices. Let $C_s = C_1, C_2, \ldots, C_m = C_t$ be the nonseparable components (*green*) and $a_1, a_2, \ldots, a_{m-1}$ the separation vertices (*blue*) on P. Let $s_1 = s$, $t_m = t$ and $s_i = a_{i-1} (1 < i \leq m)$ and $t_i = a_i (1 \leq i < m)$. Let $E_1 \subseteq E$ be the edges of the C_i's and $E_2 = E \setminus E_1$.

Theorem 3.3: *Let $G = (V, E)$ be an undirected network with a source s and sink t and let E_1 and E_2 be the partition of the edges, E as described. For each edge $e \in E$, $e \in E_2$ iff there is no simple path from s to t containing e.* □

In a pre-processing step, we find and delete the edges, E_2 to obtain the graph G', and present each non-separable component $(C_i; s_i, t_i)$ of G' as input to the main algorithm. The preprocessing step takes $O(|E| + |V|)$ time.

Definition 3.4: We introduce a classification of graphs as follows: (i)A nonseparable graph is said to be of *type I*, if it has a circuit J containing the vertices s and t and all its bridges are B^P-, B^Q- or B^{PQ}-bridges; (ii) *type II*, if it is of TYPE.I and all its bridges are of B^{PQ}-bridges; (iii) *type III*, if it is of TYPE.II and has only a single B^{PQ}-bridge; and (iv) *type IV*, if it is of TYPE.III and the subgraph, derived by deleting the vertices s and t together with their incident edges, is nonseparable. □

3.2. Labelling a Chain Graph

Assume that we have an algorithm, called LABEL-TYPE-I to label the edges of a TYPE.I graph.

Algorithm LABEL-GRAPH. (Cf. Figure 4.)

•STEP1. *Find the nonseparable components of the graph.* Let $C_1, C_2, \ldots, C_m$ be the chain of nonseparable components and let s_i and t_i be the vertices associated with C_i (Cf. Assumption 3.2). For each $(C_i;\ s_i,\ t_i)$, where $1 \leq i \leq m$, do the following steps.

•STEP2. *Find a circuit J containing the vertices s_i and t_i in C_i.* Since C_i is nonseparable such a circuit exists and can be found in $O(|E|)$ time, using depth-first search

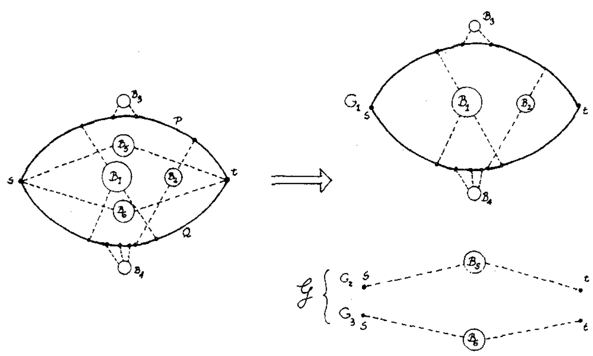

Figure 4: Steps of the Algorithm LABEL-GRAPH.

technique. We call the subpath $J[\![s_i;t_i]\!]$, P; and its complementary residual path in J, Q.

•STEP3. *Find all the bridges of C_i with respect to the circuit J. Let $\mathcal{G} = \{G_2, G_3, \ldots, G_k\}$ be the set of bridges with only attachments s_i and t_i; and let $\mathcal{B}$ be the set of B^P-, B^Q- and B^{PQ}-bridges of J. Let G_1 be the graph derived from C_i by deleting the bridges of $\mathcal{G}$. Since G_1 is a* TYPE.I *graph, by assumption we can label the edges of G_1 using the algorithm* LABEL-TYPE-I. *Moreover, since $G_i \in \mathcal{G}$ is a chain graph, the labelling for G_i can be done by recursively calling the main algorithm.* We can find the bridges in $O(|E|)$ time as in the planarity testing algorithm. □

Theorem 3.5: *Assume that* LABEL-TYPE-I *correctly labels the edges of a* TYPE.I *graph. Then the algorithm* LABEL-GRAPH *correctly labels the edges of a nonseparable graph.* □

3.3. Labelling a Type. I Graph

Assume that we have an algorithm, called LABEL-TYPE-II, to label the edges of a TYPE.II graph.

Algorithm LABEL-TYPE-I.(Cf. Figure 5.)

•STEP1. *Modify J to obtain a circuit J' such that J' is an ambitus. Let $\mathbf{B}_1$, $\mathbf{B}_2$, ..., $\mathbf{B}_n$ be its blocks of B^P- and B^Q-bridges; and s_i and t_i, the left- and the right-most*

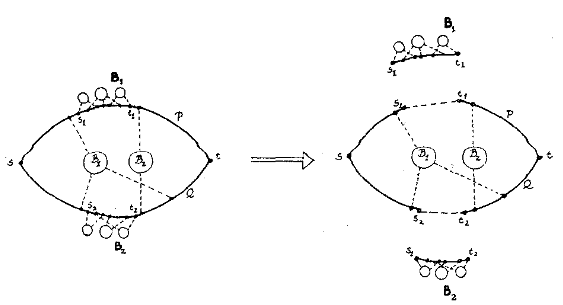

Figure 5: Steps of the Algorithm LABEL-TYPE-I.

vertices of attachments of the bridges of $\mathbf{B}_i$, respectively. Let G'' be the graph derived from G by first deleting all the blocks $\mathbf{B}_i$ and then replacing $P[\![s_i;t_i]\!]$ or $Q[\![s_i;t_i]\!]$ (depending on whether $\mathbf{B}_i$ is a block of B^P- or B^Q-bridges, respectively) by a single link $[s_i,t_i]$. we refer to $[s_i,t_i]$ as a pseudo-edge(I) of the block $\mathbf{B}_i$. This step can be done in time $O(|E|+|V|)$. A linear time algorithm for finding the ambitus can be found in the full paper.

•STEP2. *Since graph G'' is a* TYPE.II *graph, by assumption, we can label the edges of G'' using the algorithm* LABEL-TYPE-II.

•STEP3. *For each block $\mathbf{B}_i$, where $1 \le i \le n$, do the following: If the pseudo-edge of $\mathbf{B}_i$ is labeled 'bidirectional' by the Step. 2, then all edges of $\mathbf{B}_i$ are 'bidirectional'; otherwise, label the edges of $\mathbf{B}_i$ by calling* LABEL-GRAPH *with the argument $(\mathbf{B}_i; s_i, t_i)$.* □

Let G be a TYPE.I graph with circuit $J = P \cup Q$. Let $\mathbf{B}_i$ be a block of B^P-bridges of G and let its left- and right-most vertices of attachment be s_i and t_i.

Lemma 3.6: *Let G' be the graph derived by first deleting $\mathbf{B}_i$ from G and then replacing the subpath $P[\![s_i;t_i]\!]$ by a single link $[s_i,t_i]$, called the pseudo-edge of $\mathbf{B}_i$. (i) If in a legal labelling of G', $[s_i,t_i]$ is labelled 'bidirectional' then all edges of $\mathbf{B}_i$ are 'bidirectional'. (ii) If in a legal labelling of G', $[s_i,t_i]$ is labelled (s_i,t_i) then the labelling of the edges of $\mathbf{B}_i$ are determined by that of $(\mathbf{B}_i;s_i,t_i)$.* □

Lemma 3.7: *Let G'' be the graph derived from G by replacing every block $\mathbf{B}_j$ of B^P- and B^Q-bridges by the corresponding pseudo-edge $[s_j,t_j]$. Then $\ell([s_i,t_i])$ in $G' = \ell([s_i,t_i])$ in G''.* □

Theorem 3.8: *Suppose* LABEL-TYPE-II *correctly labels the edges of a* TYPE.II *graph. Then the algorithm* LABEL-TYPE-I *correctly labels the edges of a* TYPE.I *graph.*

PROOF. Follows from the previous two Lemmas. □

3.4. Labelling a Type.II Graph

We assume that we have an algorithm, called LABEL-TYPE-III to label the edges of a TYPE.III graph.

Algorithm LABEL-TYPE-II. (Cf. Figure 6.)

•STEP1. *Let B_1, B_2, ..., B_n be B^{PQ}-bridges of J in G. We modify each bridge fragment $G_i = B_i \cup J$ as follows: Let s_P and t_P be the left- and right-most vertices of attachment of B_i on P; and, similarly, s_Q and t_Q, on Q. Let L_0, L_1, ..., L_k be its residual paths not containing s or t. G'_i is derived from G_i by contracting the subpaths $P[\![s;s_P]\!]$, $Q[\![s;s_Q]\!]$, $P[\![t_P;t]\!]$, $Q[\![t_Q;t]\!]$, L_0, L_1, ..., L_k to single links, called a pseudo-edge(II) of the subpath. J' is the circuit in G'_i derived from J by the contraction. Since, each such G'_i is a* TYPE.III *graph, by assumption, we can label the edges of G'_i using the algorithm* TYPE.III.

•STEP2. *For each edge e of J, let $\{e'_1, e'_2, \ldots, e'_n\}$ be the set of edges of G'_i such that e'_i is the contraction of a subpath containing e. Then if any e'_i is 'bidirectional' mark e 'bidirectional'.* This step can be done in time $O(|E|)$,

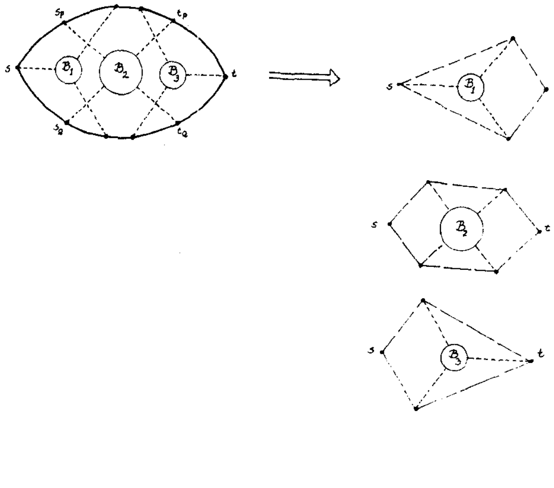

Figure 6: Steps of the Algorithm LABEL-TYPE-II.

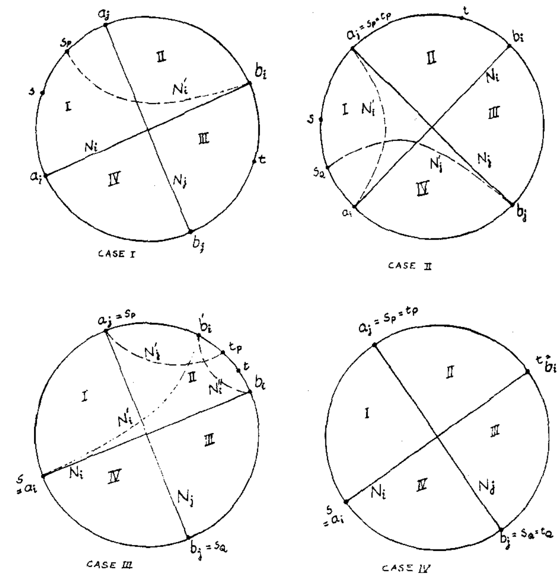

Figure 7: Four Cases of the Lemma.3.10

since, as it will be shown in Lemma 3.14 there are at most $4 \cdot |E|$ pseudo-edges(II).

•STEP3. *Let B_i and B_j be two interlacing B^{PQ}-bridges of J in G and s_P and t_P (respectively, s_Q and t_Q), their left- and right-most vertices of attachment on P (respectively, Q), distinct from s and t. Label the edges of the subpaths $P[\![s_P; t_P]\!]$ and $Q[\![s_Q; t_Q]\!]$ 'bidirectional'.* This step can be done in time $O(|E|)$. □

Lemma 3.9: *Let B be a B^{PQ}-bridge of $J = P \cup Q$ in* TYPE.II *graph G and $e = [u, v]$, an edge of B. Let G' be a subgraph of G derived from G by deleting all bridges except B and contracting the residual subpaths as in the Algorithm* LABEL-TYPE-II. *Then there is a path from s to t in G traversing e in the order u, v if and only if there is a path from s to t in G' traversing e in the same order.*

PROOF. ($\Leftarrow$) Trivially true, since G' is a minor of G. ($\Rightarrow$) If $R[\![s; t]\!]$ traverse e in the order u and v then we can write $R[\![s; t]\!]$ as $R[\![s; x]\!] * R[\![x; y]\!] * R[\![y; t]\!]$ such that $R[\![x; y]\!]$ is a cross-cut of J, belonging to B and traversing e in the order u, v. If $x \in P[\![s; t]\!]$ and $y \in Q[\![s; t]\!]$ then the path $P[\![s; x]\!] * R[\![x; y]\!] * Q[\![y; t]\!]$ in G' traverses e in the same order. Hence, assume that $x, y \in P]\!]s; t[\![$. Since B is B^{PQ}-bridge it has a vertex of attachment $z \in Q]\!]s; t[\![$. Using proposition 1.12, we can find simple paths from s to t such that they traverse e in either direction. □

Lemma 3.10: *Let B_i and B_j be two interlacing B^{PQ}-bridges of $J = P \cup Q$. Let s_P and t_P be the left- and right-most vertices of attachment of B_i and B_j on P distinct from s and t; and similarly s_Q and t_Q on Q. Then the edges of the residual paths $P[\![s_P; t_P]\!]$ and $Q[\![s_Q; t_Q]\!]$ are 'bidirectional'.*

PROOF. (Cf. Figure 7.) Since B_i and B_j interlace, there exist two vertices of attachment a_i and b_i of B_i and two vertices of attachment a_j and b_j of B_j, all four distinct, such that a_i and b_i separate a_j and b_j in the circuit J. Let N_i be a cross-cut of J in G between a_i and b_i belonging to B_i; and N_j between a_j and b_j belonging to B_j. The four residual paths $J[\![a_i; a_j]\!]$, $J[\![a_j; b_i]\!]$, $J[\![b_i; b_j]\!]$ and $J[\![b_j; a_i]\!]$ will be referred to as quadrants I, II, III and IV, respectively. The distinguished vertices s and t lie on J. Observe that if two B^{PQ}-bridges B_i and B_j interlace then it is always possible to choose four distinct vertices a_i, b_i, a_j and b_j such that s and t do not lie in the same quadrant.

Hence there are four cases to consider:

•CASE.1 *s is an internal vertex of quadrant I and t is an internal vertex of quadrant III.* Let $J[\![s; t]\!]$ in quadrants I, II and III be $P[\![s; t]\!]$ and $Q[\![s; t]\!]$ its complement in J. Let a_i, b_i, a_j and b_j be modified such that a_i and a_j are the left-most vertices of attachment of B_i and B_j, distinct from s, on Q and P respectively; and similarly, b_i and b_j, distinct from t. Accordingly, N_i and N_j are modified. Observe that s_P (respectively, s_Q) is distinct from a_j (respectively, a_i) *if and only if* s_P is the left-most vertex of attachment B_i on P (respectively, s_Q, the left-most vertex of attachment B_j on Q), distinct from s. A similar observation can be made about t_P and t_Q.

Every edge of the residual paths $P[\![a_j; b_i]\!]$ and $Q[\![a_i; b_j]\!]$ is 'bidirectional'. The paths $P[\![s; t]\!]$ and $Q[\![s; a_i]\!] * N_i[\![a_i; b_i]\!] * P[\![b_i; a_j]\!] * N_j[\![a_j; b_j]\!] * Q[\![b_j; t]\!]$ traverse the edges of $P[\![a_j; b_i]\!]$ in either direction. Similarly, for $Q[\![a_i; b_j]\!]$.

If s_P and a_j are distinct then every edge of the path $P[\![s_P; a_j]\!]$ is 'bidirectional'. Since s_P is a vertex of attachment of B_i, there is a cross-cut between s_P and b_i, say N_i'. then the paths $P[\![s; t]\!]$ and $Q[\![s; b_j]\!] * N[\![b_j; a_j]\!] * P[\![a_j; s_P]\!] * N_i'[\![s_P; b_i]\!] * P[\![b_i; t]\!]$ traverse the edges of $P[\![s_p; a_j]\!]$ in either directions. Similarly, for $P[\![b_i; t_P]\!]$, $Q[\![s_Q; a_i]\!]$ and $Q[\![b_j; t_Q]\!]$.

•CASE.2 *s is an internal vertex of quadrant I and t, of quadrant II.* Let $J[\![s; t]\!]$ in quadrants I and II be $P[\![s; t]\!]$ and $Q[\![s; t]\!]$ its complement in J. We may assume that $a_j = s_P = t_P$. Since, otherwise, we can reduce this case to an instance of the case(1).

Since B_i is a B^{PQ}-bridge, a_j is also an attachment of B_i; and there are cross-cuts between a_j and a_i (N_i') and between a_j

and b_i (N_i''). Let a_i and b_i be modified to be the left- and right-most vertices of attachment of B_i on Q, distinct from s and t, respectively. Accordingly N_i is modified. Observe that s_Q is distinct from a_i *if and only if* s_Q is the left-most vertex of attachment of B_j on Q distinct from s. A similar relation holds between t_Q and b_i.

Every edge of the residual path $Q[\![a_i; b_j]\!]$ and $Q[\![b_j; b_i]\!]$ is 'bidirectional'. The paths $Q[\![s; t]\!]$ and $P[\![s; a_j]\!] * N_j[\![a_j; b_j]\!] * Q[\![b_j; a_i]\!] * N_i[\![a_i; b_i]\!] * Q[\![b_i; t]\!]$ traverse edges of $Q[\![a_i; b_j]\!]$ in either directions. Similarly, for $Q[\![b_j; b_i]\!]$.

If s_Q and a_i are distinct, then every edge of $Q[\![s_Q; a_i]\!]$ is 'bidirectional'. Since s_Q is a vertex of attachment of B_j, there is a cross-cut between s_Q and b_j, say N_j'. The paths $Q[\![s; t]\!]$ and $P[\![s; a_j]\!] * N_i'[\![a_j; a_i]\!] * Q[\![a_i; s_Q]\!] * N_j'[\![s_Q; b_j]\!] * Q[\![b_j; t]\!]$ traverse the edges of $Q[\![s_Q; a_i]\!]$ in either directions. The subpath $Q[\![b_i; t_Q]\!]$ is treated in a similar manner.

•CASE.3 *$s = a_i$ and t is an internal vertex of the quadrant II.* Let $J[\![s; t]\!]$ in quadrants I and II be $P[\![s; t]\!]$ and $Q[\![s; t]\!]$ be its complement in J. We may assume that $s_P = a_j$ and $s_Q = b_j$. Since, otherwise, we can reduce this case to an instance of the case(1) or case(2).

Since B_i is a B^{PQ}-bridge, it has a vertex of attachment b_i' on $P[\![s_P; t_P]\!]$. Let b_i and b_i' be modified to be the right-most vertices of attachment of B_i on P and Q, distinct from t. Accordingly N_i is modified. Moreover, there are cross-cuts between a_i and b_i' (say, N_i') and between b_i and b_i' (say, N_i''). Observe that t_Q is distinct from b_i *if and only if* t_Q is the right-most vertex of attachment of B_j distinct from t. A similar relation holds between t_P and b_i'.

Every edge of the residual path $Q[\![b_j; b_i]\!]$ is 'bidirectional'. The paths $Q[\![s; t]\!]$ and $N_i[\![s; b_i]\!] * Q[\![b_i; b_j]\!] * N_j[\![b_j; a_j]\!] * Q[\![a_j; t]\!]$ traverse the edges of $Q[\![b_j; b_i]\!]$ in either directions. If a_j and b_i' are distinct then, similarly, every edge of $P[\![a_j; b_i']\!]$ is 'bidirectional'.

If t_P and b_i' are distinct then every edge of $P[\![b_i'; t_P]\!]$ is 'bidirectional'. Since t_P is a vertex of attachment of B_j, there is a cross-cut between a_j and t_P (say, N_j'). The paths $P[\![s; t]\!]$ and $P[\![s; a_j]\!] * N_j'[\![a_j; t_P]\!] * P[\![t_P; b_i']\!] * N_i''[\![b_i'; b_i]\!] * Q[\![b_i; t]\!]$ traverse edges of $P[\![b_i'; t_P]\!]$ in either directions. Similarly, for $Q[\![b_i; t_Q]\!]$.

•CASE.4 *$s = a_i$ and $t = b_i$.* We may assume that $s_P = t_P = a_j$ and $s_Q = t_Q = b_j$. Since, otherwise, we can reduce this case to an instance of the case(3). Since the paths are empty, the theorem is trivially true in this case. □

Theorem 3.11: *Assume* LABEL-TYPE-III *correctly labels the edges of a* TYPE.III *graph. Then the algorithm* LABEL-TYPE-II *correctly labels a* TYPE.II *graph.*

PROOF. Let e be an arbitrary edge of the TYPE.III graph G. Without loss of generality we may assume that e is an edge of the path P, and that there are no two interlacing bridges such that $e \in P[\![s_P; t_P]\!]$, as in the previous lemma. Otherwise the proof is immediate from the previous two lemmas.

We partition the set of B^{PQ}-bridges, $\mathcal{B}$ of J into following three disjoint subsets: $\mathcal{B}_1$ = the set B^{PQ}-bridges whose vertices of attachment on P are to the *left* of e; $\mathcal{B}_2$ = the set B^{PQ}-bridges whose vertices of attachment on P are to the *right* of e; $\mathcal{B}_3 = \mathcal{B} \backslash \{\mathcal{B}_1 \cup \mathcal{B}_2\}$. Notice that no bridge of $\mathcal{B}_i$ interlaces with a bridge of $\mathcal{B}_j$, (where $i, j = 1, 2, 3$ and $i \neq j$); and $\mathcal{B}_3$ does not contain a pair of interlacing bridges. If $\mathcal{B}_3$ is empty or if $\mathcal{B}_3$ contains two or more equivalent B^{PQ} 3-bridges (Cf. Proposition 1.13) then it can be shown that $\ell(e) = \langle u, v \rangle$, if u is to the left of v on P. On the other hand, if $\mathcal{B}_3 = \{B_3\}$ is a singleton then the labelling of e is completely determined by the bridge fragment $J \cup B_3$. □

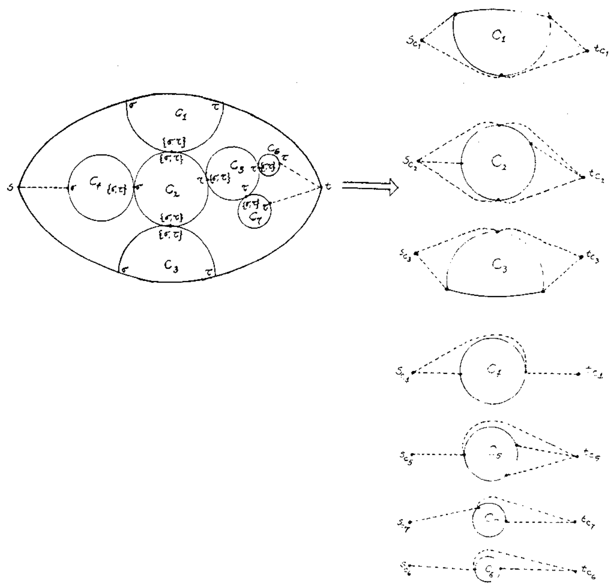

Figure 8: Steps of the Algorithm LABEL-TYPE-III.

3.5. Labelling a Type.III Graph

We assume that there is an algorithm called LABEL-TYPE-IV, to label the edges of a TYPE.IV graph.

Algorithm LABEL-TYPE-III. (Cf. Figure 8.)

•STEP1. *Let s_P and t_P be the left- and right-most vertices of attachment of B on P; and similarly, s_Q and t_Q, on Q. Let G' be the subgraph derived from G by deleting the vertices s and t together with the edges incident on s and t and the subpaths $P[\![s; s_P]\!]$, $Q[\![s; s_Q]\!]$, $P[\![t_P; t]\!]$ and $Q[\![t_Q; t]\!]$. The vertices s_P, s_Q and the vertices of G' adjacent to s in G are labelled σ; and the vertices t_P, t_Q and the vertices of G' adjacent to t in G are labelled τ.*

•STEP2. *Find the nonseparable components of G' and label each separation vertex v of each nonseparable component C as follows: v is labelled σ (respectively, τ), if there is a path from v to a vertex u, already labelled σ (respectively, τ), in the tree $T(G')$ of G' and the path avoids C.* This step can be done in time $O(size(T(G'))) = O(|E|)$ using a depth first search.

•STEP3. *For each nonseparable component C, introduce two new vertices s_C and t_C; and join s_C (respectively, t_C) to all the vertices of C labelled σ (respectively, τ). We call the graph derived from the nonseparable component C, G_C; and the new edges, pseudo-edges (III). If G' is* *nonseparable then the graph derived from the nonseparable component G', called G'', is a* TYPE.IV *graph and by assumption, we can label G'' using the algorithm* LABEL-TYPE-IV. *Otherwise, label the edges of the components by recursively calling the main algorithm with $(G_C; s_C, t_C)$.*

•STEP4. *Edge $e = [u, v]$ on $P[\![s; s_P]\!]$, $Q[\![s; s_Q]\!]$, $P[\![t_P; t]\!]$ and $Q[\![t_Q; t]\!]$ is labelled $\ell(e) = \langle u, v\rangle$, if u is to the left of v on P or Q. Edge $[s, u]$ incident on s is labelled $\ell([s, u]) = \langle s, u\rangle$ and edge $[u, t]$ incident on t is labelled $\ell([u, t]) = \langle u, t\rangle$.* □

Lemma 3.12: *Let C be a nonseparable component of G' whose separation vertices are labelled σ or τ (or both), and let e be an arbitrary edge of C. Then there is a simple path R from s to t in G traversing e in the order u, v if and only if there are two distinct separation vertices a and b in C such that (i) Label of a is σ; and label of b is τ. (ii) There is a simple path R' joining a and b in C; and R' traverses e in the same order.*

PROOF. ($\Leftarrow$) By labelling of the separation vertices in the tree $T(G')$ of G', there are simple path $N_1[\![s; a]\!]$ and $N_2[\![b; t]\!]$ in G such that N_1 and N_2 are vertex disjoint and do not belong to C. The path $R[\![s; t]\!] = N_1[\![s; a]\!] * R'[\![a; b]\!] * N_2[\![b; t]\!]$ in G, is simple and traverses in the same order as R'. ($\Rightarrow$) Let $R[\![s; t]\!]$ traverse e in the order u and v and let e belong to the nonseparable component C. Then $R[\![s; t]\!]$ can be written as $R[\![s; a]\!] * R[\![a; b]\!] * R[\![b; t]\!]$, where a and b are two distinct separation vertices of C. Let R' in C be the subpath $R[\![a; b]\!]$. R' traverses the edge e in the same order as R in G. Moreover, the path $R[\![s; a]\!]$ induces a path in the tree $T(G')$ such that it joins a to some vertex labelled σ and the path avoids C. Hence the separation vertex is labelled σ. Similarly the separation vertex b is labelled τ. □

Theorem 3.13: *Assume that* LABEL-TYPE-IV *correctly labels the edges of a* TYPE.IV *graph. Then the algorithm* LABEL-TYPE-III *correctly labels the edges of a* TYPE.II *graph.*

PROOF. By induction on the size of the graph and the Lemma 3.12. □

3.6. Labelling a Type.IV Graph

Theorem 3.14: *Suppose we have an Algorithm* LABEL-TYPE-IV *that correctly labels the edges of a* TYPE.IV *graph in time $O(T(|E|, |V|)) \geq O(|E| \cdot |V|)$, where T is a monotonic nondecreasing function in $|E|$ and $|V|$. Then the set of mutually recursive algorithms,* LABEL-GRAPH, LABEL-TYPE-I, LABEL-TYPE-II *and* LABEL-TYPE-III, *correctly labels the edges of an undirected connected strict graph in time $O(T(|E|, |V|) + |E| \cdot |V|)$*

PROOF. (1) Follows immediately from the Theorems 3.5, 3.8, 3.11 and 3.13.

(2) The set of mutually recursive algorithms works as a Divide-and-Conquer algorithm; and each divide step and conquer step takes $O(|E| + |pE|)$, where $|pE|$ = the number of pseudo-edges introduced at each stage. Since each 'divide' step reduces the number of vertices of the subgraph by at least one, there can be at most $O(|V|)$ stages of 'divide' stages before the graph is divided into a set of TYPE.IV graphs.

Claim. The total number of pseudo-edges in all the subgraphs that are produced at the end of each stage, $|pE| = O(|E|)$.

Proof of the Claim. Let G' be a subgraph produced at some stage. We define three functions f_1, f_2 and f_3 such that f_2 maps at most four distinct pseudo-edges(II) of G' to exactly one graph-edge, f_3 maps at most four distinct pseudo-edges(III) to exactly one graph-edge, and f_1 is an injective function mapping a pseudo-edge(I) to a graph-edge or a pseudo-edge(III). Since the graph-edges of the subgraphs are disjoint, the claim follows.

If e' is a pseudo-edge, incident on a vertex v of G' then there must be B^{PQ}-bridge of G' with a vertex of attachment at v. Let e be the edge of the B^{PQ}-bridge, incident at v. Define $f_2(e') = e$. Since f_2 maps at most *four* distinct pseudo-edges(II) of G' to one graph-edge of G', and since the graph-edges of the subgraphs are disjoint, the total number of pseudo-edges(II), $|pE^{\mathrm{II}}| \leq 4 \cdot |E|$. If e' is a pseudo-edge(III), incident at a vertex v of G', then since v is one of s_P, t_P, s_Q, t_Q, a vertex adjacent to s or t, or a separation vertex of a component then there must be graph-edge e that is also incident on v. Define $f_3(e') = e$. Since f_3 maps at most *four* pseudo-edges(III) of G' to exactly one graph-edge, the total number of pseudo-edges(III), $|pE^{\mathrm{III}}| \leq 4 \cdot |E|$. We define a *injective* function f_1 that maps a pseudo-edge(I) to a graph-edge or a pseudo-edge(III). Let $\mathbf{B}_i$ be a block with the associated vertices s_i and t_i and J_i, the ambitus of $\mathbf{B}_i$, containing both s_i and t_i. Let $e' = [s_i, t_i]$ be the pseudo-edge(I) and let e be an edge of J_i incident on t_i but not belonging to J. Define $f_1(e') = e$. It is easy to show that f_1 is one-one and hence, the total number of pseudo-edges(I), $|pE^{\mathrm{I}}| \leq 5 \cdot |E|$. Summing the number of pseudo-edges, we obtain $|pE^{\mathrm{I}}| + |pE^{\mathrm{II}}| + |pE^{\mathrm{III}}| \leq 13 \cdot |E|$. *(End of the Claim.)*

Hence, the algorithms spend $O(|E| \cdot |V|)$ time to reduce the graph G into set of TYPE.IV subgraphs. Since each TYPE.IV subgraph has less than $|V|$ vertices, and since the total number of edges is $O(|E|)$, the theorem follows. □

Theorem 3.15: *Let G be a* TYPE.IV *graph and G', the subgraph derived from G by deleting s and t together with their incident edges. Let B be the B^{PQ}-bridge of G. Then every edge e of B not incident on s or t is 'bidirectional.'*

PROOF. Observe that there exist vertices of attachment of B, x on $P[\![s; t]\!]$ and y on $Q[\![s; t]\!]$ such that there is a cross-cut N between x and y containing e. (This is a consequence of proposition 1.12 and the nonseparability property of G'.) The two paths $P[\![s; x]\!] * N[\![x; y]\!] * Q[\![y; t]\!]$ and $Q[\![s; y]\!] * (N[\![x; y]\!])^{\mathrm{R}} * P[\![x; t]\!]$ traverse e in either directions. □

Every edge $[s, u]$ incident on s is labelled $\langle s, u\rangle$ and every edge $[u, t]$ incident on t, labelled $\langle u, t\rangle$. All other edges of the B^{PQ}-bridge are labelled 'bidirectional.' Hence, we can label all but the edges on P and Q of a TYPE.IV graph in $O(|E|)$ time. But, each edge of $P[\![s_P; t_P]\!]$ and $Q[\![s_Q; t_Q]\!]$ can be labelled in $O(|E| \cdot |V|)$ time, using an algorithm for Two-Disjoint-Path problem. Since there are at most $|V|$ such edges, this step can be done in $O(|E| \cdot |V|^2)$ time, thus, yielding an $O(|E| \cdot |V|^2)$ labelling algorithm for the general graph.

4. An Efficient Algorithm.

We sketch an algorithm to label the edges of the subpath $P\llbracket s_P;t_P\rrbracket$ and $Q\llbracket s_Q;t_Q\rrbracket$ of a TYPE.IV graph, G, in $O(|E|\cdot|V|)$ time. Recall that this will provide an $O(|E|\cdot|V|)$ time algorithm for a general graph. (Cf. Theorems 3.14.) The algorithm makes use of many special properties of a bridge and its proofs of correctness are rather complicated. These will be supplied in the author's thesis.

4.1. U-Fragment

Before we describe the main algorithm, we introduce the notion of a U-Fragment and sketch an algorithm to find certain pairs of disjoint cross-cuts in it.

Definition 4.1: A *U-Fragment* and *Ū-Fragment* are defined inductively on the structure of a TYPE.II graph as follows:

•A TYPE.II graph is called a U-Fragment, with $\mathcal{B}_2 =$ its set of B^{PQ}-bridges. Its *upper* and *lower external vertices of attachment* are empty sets.

•Let U be a U-Fragment consisting of the circuit $J = P\cup Q$ and let B be a B^{PQ}-bridge of U. Let the left-most and the right-most vertices of attachment of B on $Q\llbracket s;t\rrbracket$ be the distinct vertices s' and t'. Let R be a path in B, connecting s' and t' and decomposing B into following sets of bridges with respect to $\{P\}\cup\{Q\}\cup\{R\}$: $\mathcal{B}_1$ =the set of bridges with vertices of attachment on $P\rrbracket s;t\llbracket$ and on $R\rrbracket s';t'\llbracket$; $\mathcal{B}_2 =$ the set of bridges with vertices of attachment on $R\rrbracket s';t'\llbracket$ and $Q\rrbracket s';t'\llbracket$; and the set of bridges with vertices of attachment solely on R and avoiding $\mathcal{B}_1$ and $\mathcal{B}_2$.

The subgraph $U' = \{R\llbracket s';t'\rrbracket\}\cup\{Q\llbracket s';t'\rrbracket\}\cup\mathcal{B}_2$ of U, is called a *U-Fragment of U on Q*. The vertices of attachment of $\mathcal{B}_1$ on $R\rrbracket s';t'\llbracket$ are called its *upper external vertices of attachment, UA* and the lower external vertices of attachment of U lying on $Q\rrbracket s';t'\llbracket$, its *lower external vertices of attachment, LA.*

The subgraph $\overline{U'} = \{R\llbracket s';t'\rrbracket\}\cup\{P\llbracket s;t\rrbracket\}\cup\mathcal{B}_1$ of U, is called a *complementary U-Fragment* (simply Ū-Fragment) *of U on Q.* The external vertices of attachment of U lying on $P\rrbracket s;t\llbracket$ are called its *upper external vertices of attachment, UA*, and the vertices of attachment of $\mathcal{B}_2$ on $R\rrbracket s';t'\llbracket$ together with the vertex s' (if s' is distinct from s) and the vertex t' (if t' is distinct from t), its *lower external vertices of attachment, LA.*

The U- and Ū-Fragments of an Ū-Fragment are defined in an identical manner. □

NOTATION: Let U be a U- or a Ū-Fragment. Let the left- and right-most upper attachments on P (respectively, Q) be s_U and t_U (respectively, s_L and t_L.) Let B be a B^{PQ}-bridge of U and let the left- and right-most vertices of attachment of B on $P\llbracket s;t\rrbracket$ be s_P^B, t_P^B, and those on $Q\llbracket s;t\rrbracket$ be s_Q^B and t_Q^B. Similarly, let the left- and right-most vertices of attachment of B on $P\rrbracket s;t\llbracket$ be s_P and t_P, and those on $Q\rrbracket s;t\llbracket$ be s_Q and t_Q. We consider the bridge B augmented with the paths as follows: Let the modified bridge be $B\cup P\llbracket s_P^B;t_P^B\rrbracket\cup Q\llbracket s_Q^B;t_Q^B\rrbracket$. If $s_P^B = s_Q^B = s$ then label the vertex with 's', and if $t_P^B = t_Q^B = t$ then label the vertex with 't.' By an abuse of notation, we also refer to the modified bridge as the *bridge*, B. □

Definition 4.2: A U-Fragment, U', of U (a U- or a Ū-Fragment) on Q is said to be a *Feasible U-Fragment*, if it satisfies at least one of the following two conditions:

1. $|\mathrm{LA}| > 0$ *and* $|\mathrm{UA}| > 0$.
2. (i) $|\mathrm{LA}| = 0$; (ii) *not all the vertices of attachment of* $\mathcal{B}_2$ *on* P' *belong to* $P'\rrbracket s';s_U\rrbracket$ *or to* $P'\llbracket t_U;t'\llbracket$; *and* (iii) *there exist two vertex disjoint paths in* $\mathcal{B}_1$, $R_a\llbracket a';a\rrbracket$ *and* $R_b\llbracket b';b\rrbracket$, *where* R_a (R_b) *meets* $P\rrbracket s;t\llbracket$ *only in* a (b) *and meets* $P'\rrbracket s';t'\llbracket$ *only in* a' (b'). □

Definition 4.3: A U-Fragment, $\overline{U'}$, of U (a U- or a Ū-Fragment) on Q is said to be a *Feasible Ū-Fragment*, if $|\mathrm{LA}| > 0$ or $|\mathrm{UA}| > 0$. □

Henceforth, it will be implicitly assumed that U-Fragment and Ū-Fragment are found using the following conventions:

Convention 4.4: Let U' be a U- or Ū-Fragment with the B^{PQ}-bridge B.

•$s_P^B = t_P^B$ *and* $s_Q^B = t_Q^B$. Then B may be discarded.

•$s_P^B = t_P^B$ *and* s_Q^B *and* t_Q^B *are distinct.* If there is a lower vertex of attachment $b\in Q'\rrbracket s_Q^B;t_Q^B\llbracket$ then the U- and Ū-Fragment in B are on Q. Otherwise, B is discarded.

•s_P^B *and* t_P^B *as well as* s_Q^B *and* t_Q^B *are distinct.* (a) *(U' is a U-Fragment satisfying 1. of definition 4.2.)* If B has no lower attachment on $Q'\rrbracket s_Q^B;t_Q^B\llbracket$, but has an upper attachment on $P'\rrbracket s_P^B;t_P^B\llbracket$ then the U- and Ū-Fragment in B are on Q. Similarly, with P and Q interchanged. (b) *(U' is a U-Fragment of U on Q satisfying 2. of definition 4.2.)* The U- and Ū-Fragment in B are on Q. (c) *(U' is a Ū-Fragment of U on Q.)* The U- and Ū-Fragment in B are on Q. Otherwise, the U- and Ū-Fragment are on P or Q, the choice being arbitrary. □

Definition 4.5: (Cf. Figure 9.) Let U be a U-Fragment.

•Two vertex-disjoint cross-cuts $N_1\llbracket x_P;x_Q\rrbracket$ and $N_2\llbracket y_P;y_Q\rrbracket$ are said to be a *PQ-Cross-Cut Pair*, if: *(1)* x_P, $y_P\in P\rrbracket s;t\llbracket$ and x_P is to the left of y_P; *(2)* x_Q, $y_Q\in Q\rrbracket s;t\llbracket$ and x_Q is to the right of y_Q and *(3)* if $|\mathrm{LA}| = 0$ or if x_Q, $y_Q\in Q\rrbracket s;s_L\rrbracket$ then $|\mathrm{UA}| > 0$ and not both x_P and y_P belong to $P\rrbracket s;s_U\rrbracket$; and similarly, if $|\mathrm{LA}| = 0$ or if x_Q, $y_Q\in Q\llbracket t_L;t\llbracket$ then $|\mathrm{UA}| > 0$ and not both x_P and y_P belong to $P\llbracket t_U;t\llbracket$. (Or, with UA and LA interchanged.)

•Two vertex-disjoint cross-cuts $N_1\llbracket s;t\rrbracket$ and $N_2\llbracket x_P;x_Q\rrbracket$ are said to be an *ST-Cross-Cut Pair*, if: *(1)* $x_P\in P\rrbracket s;t\llbracket$ and $x_Q\in Q\rrbracket s;t\llbracket$; *(2)* there are an upper attachment on $P\rrbracket s;t\llbracket$ and a lower attachment on $Q\rrbracket s;t\llbracket$.

•Two vertex-disjoint cross-cuts $N_1\llbracket x'_P;x''_P\rrbracket$ and $N_2\llbracket y_P;y_Q\rrbracket$ are said to be a *P-Cross-Cut Pair*, if: *(1)* x'_P, $x''_P\in P\llbracket s;t\rrbracket$ and at least one of them is distinct from s and t; *(2)*

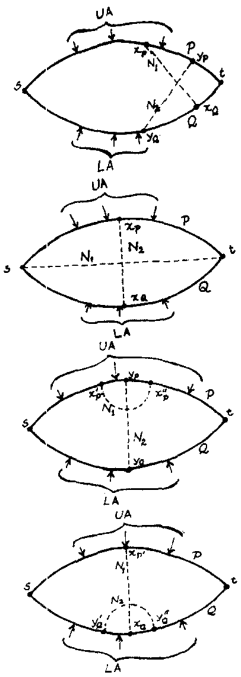

Figure 9: *PQ*-, *ST*-, *P*- and *Q*-Cross-Cut Pairs.

$y_P \in P\rrbracket x'_P; x''_P\llbracket$ and $y_Q \in Q\rrbracket s; t\llbracket$; and (*3*) there is an upper attachment $c' \in P\rrbracket x'_P; x''_P\llbracket$.

A *Q-Cross-Cut-Pair* is defined similarly. □

The Cross-Cut Pairs of $\bar{U}$-Fragments and bridges are defined similarly. Now, we present an algorithm to find a *P*-, *Q*-, *PQ*- or *ST*-Cross-Cut Pair in a U-Fragment.

Algorithm ANALYZE-U-FRAGMENT.

•STEP1. *First, check the interlacing bridges of $\mathcal{B}_2$ to determine if the required cross-cut pairs exist. If so, return* 'YES'. *In the next step, analyze each bridge of $\mathcal{B}_2$ for the cross-cut pairs. If the answer is* 'YES' *for any $\mathcal{B}_2$ bridge, return* 'YES'; *otherwise, return* 'NO'.

•STEP2. *Let $B \in \mathcal{B}_2$. If the vertices of attachment of B on P belong to $P\rrbracket s; s_U\rrbracket$ and those on Q belong to $Q\rrbracket s; s_L\rrbracket$ (or symmetrically, if those on P belong to $P\llbracket t_U; t\llbracket$ and those on Q belong to $Q\llbracket t_L; t\llbracket$) then B can not have the required cross-cut pairs and is discarded. Otherwise, analyze each single bridge using the algorithm* ANALYZE-BRIDGE.
□

Algorithm ANALYZE-BRIDGE(B) :
begin
DIVIDE : Find a path R in B following the convention 4.4. Modify R such that the bridges with vertices of attachment solely on R avoid other bridges. Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be as in the definition 4.1 and let $\mathcal{B}_3$ be the set of bridges with vertices attachment on $P\llbracket s_P^B; t_P^B\rrbracket$, $R\rrbracket s_Q^B; t_Q^B\llbracket$ and $Q\rrbracket s_Q^B; t_Q^B\llbracket$
if TEST-AND-MODIFY(B) returns 'YES' **then**
  if $|\mathcal{B}_3| \neq 0$ **then** return 'YES' **else goto** RECUR;
else goto UBAR;

RECUR :Let U be the U-Fragment formed by the paths P, R and the bridges $\mathcal{B}_2$.
if U is marked feasible **then**
  if ANALYZE-U-FRAGMENT($\bar{U}$) returns 'YES' **then**
    return 'YES';

UBAR :[Analyze the set of $\mathcal{B}_1$-bridges:]
Analyze bridges $\mathcal{B}_1$ to determine if it has the required Cross-Cut pairs. If so, return 'YES'.
[Analyze each bridge $B' \in \mathcal{B}_1$:]
Let the left- and right-most lower attachments of the $\bar{U}$-Fragment on R be s'_L and t'_L.
If the vertices of attachment of B' on P belong to $P\rrbracket s; s_U\rrbracket$ and those on R belong to $R\rrbracket s; s'_L\rrbracket$ (or symmetrically, if those on P belong to $P\llbracket t_U; t\llbracket$ and those on R belong to $R\llbracket t'_L; t\llbracket$) then discard B';
if ANALYZE-BRIDGE(B')returns 'YES' **then** return 'YES';

return 'NO';
end. □

Algorithm TEST-AND-MODIFY.

•STEP1. *Check if B has a lower vertex of attachment on $Q\rrbracket s_Q^B; t_Q^B\llbracket$. Then, if $|\mathcal{B}_3| \neq 0$ then return* 'YES'. *If $|\mathcal{B}_3| = 0$ then mark U 'feasible' and return* 'YES'. *Otherwise, go to the next step.*

•STEP2. *Let the left- and right-most vertices of attachment of $\mathcal{B}_1$ and $\mathcal{B}_3$ bridges on R be s'_R and t'_R (and if $|\mathcal{B}_3| = 0$ then those on $Q\rrbracket s_Q^B; t_Q^B\llbracket$ be s'_Q and t'_Q.) Let G_B be the subgraph of B obtained by deleting the followings: (i) the bridges $\mathcal{B}_2$, (ii) the vertices s_Q^B and t_Q^B together with the edges incident on them, (iii) the subpaths $R\llbracket s_Q^B; s'_R\rrbracket$ and $R\llbracket t'_R; t_Q^B\rrbracket$, (iv) the subpath $Q\llbracket s_Q^B; t_Q^B\rrbracket$, if $|\mathcal{B}_3| = 0$ (or the subpaths $Q\llbracket s_Q^B; s'_Q\rrbracket$ and $Q\llbracket t'_Q; t_Q^B\rrbracket$, if $|\mathcal{B}_3| \neq 0$.) The vertices of G_B adjacent to s_Q^B are labelled σ; those adjacent to t_Q^B, τ; the vertices s'_R and t'_R, ρ; and the vertices s'_Q and t'_Q, χ.*

•STEP3. *Find the nonseparable components of G_B and let the component containing the path $P\llbracket s_P^B; t_P^B\rrbracket$ be called C_π. Let each separation vertex of C_π be labelled as follows: the separation vertex v is labelled σ (respectively, τ, ρ or χ) if there is a path from v to a vertex u, already labelled σ (respectively, τ, ρ or χ) in the tree $T(G_B)$ of G_B and the path avoids C_π.*

•STEP4 *If $|\mathcal{B}_3| \neq 0$ and C_π has two distinct vertices one labelled ρ and the other labelled χ then return* 'YES'. *If $|\mathcal{B}_3| = 0$ and C_π has two distinct vertices labelled ρ then check if U satisfies 2(ii) of definition 4.2. If so, mark U 'feasible' and return* 'YES'.

Otherwise, modify the bridge B to form a $\bar{U}$-Fragment as follows: Delete all the edges of the bridge except the ones in C_π; Adjoin the separation vertex of C_π, labelled ρ, to s_Q^B and t_Q^B with new edges; And adjoin the separation vertices of C_π, labelled σ (respectively, τ) to s_Q^B (respectively, t_Q^B). Let the path from s_Q^B to t_Q^B, touching the vertex labelled ρ, be R and $\mathcal{B}_1'$, the set of bridges with vertices of attachment only on P and R. Return the modified graph as the $\bar{U}$-Fragment. □

4.2. Labelling the Path

We present an algorithm to label the path P of a TYPE.IV graph, G, in $O(|E| \cdot |V|)$ time. By using the algorithm twice (once for P and once for Q), it is possible to label the paths of G in $O(|E| \cdot |V|)$ time. In this section, we only sketch the algorithm for the case when the bridge B of G has its vertices of attachment s_Q^B and t_Q^B distinct from each other and from s and t. The other cases are similar, but slightly more complicated; and will appear in the full paper.

Algorithm LABEL-PATH(P, B) :
begin
DIVIDE : Find a path R in B joining s_Q^B and t_Q^B, with $\mathcal{B}_1$, $\mathcal{B}_2$ and $\mathcal{B}_3$ as in the algorithm ANALYZE-BRIDGE.
if TEST-AND-MODIFY(B) returns 'YES' **then**
  if $|\mathcal{B}_3| \neq 0$ **then** label the edges of $P[\![s_P; t_P]\!]$ 'bidirectional' and return 'YES';
  else goto RECUR;
else goto UBAR;

RECUR :Let U be the U-Fragment as in ANALYZE-BRIDGE
if U is marked feasible **then**
  if ANALYZE-U-FRAGMENT(U) returns 'YES' **then**
    label the edges of $P[\![s_P; t_P]\!]$ 'bidirectional' and return 'YES';

UBAR :[Analyze the set of $\mathcal{B}_1$-bridges:]
Analyze the blocks of interlacing bridges of $\mathcal{B}_1$ to label the edges of the path $P[\![s_P; t_P]\!]$, as in LABEL-TYPE-II;
Also, determine if they have the required Cross-Cut pairs.
If so, label the edges of $P[\![s_P; t_P]\!]$ 'bidirectional' and return 'YES';
[Analyze each bridge $B' \in \mathcal{B}_1$:]
-- We analyze each bridge B' to label the subpaths
-- $P[\![s_P^{B'}; t_P^{B'}]\!]$ (recursively), as well as to find if it has the
-- required cross-cut pairs, using LABEL-PATH;
if LABEL-PATH(P, B') returns 'YES' **then**
  label the edges of $P[\![s_P; t_P]\!]$ 'bidirectional' and return 'YES';

return 'NO';
end . □

Notice that the algorithm uses Divide-and-Conquer paradigm: Each 'divide' step involves finding a path in a bridge, where the path has additional properties that the bridges with vertices of attachment solely on the path avoid $\mathcal{B}_1$, $\mathcal{B}_2$ and $\mathcal{B}_3$. This can be done in $O(|E|+|pE|)$ time by combining DFS technique with the ambitus-finding algorithm. (pE = the pseudo-edges introduced in the step 4 of TEST-AND-MODIFY.) Each 'conquer' step involves analyzing interlacing bridges for appropriate cross-cut pairs, which can be done in linear time, and analyzing each individual bridge (recursively). However, each 'divide' step reduces the number of vertices of the subgraph by at least one, and hence the algorithm takes $O((|E| + |pE|) \cdot |V|)$ time. But, again, it can be shown that the number of pseudo-edges introduced is $O(|E|)$, thus, giving an $O(|E| \cdot |V|)$ time algorithm.

The proof of correctness involves two parts: In part 1, we show, by induction on the structure of the U- and Ū-Fragments, that when the algorithm claims the existence of certain disjoint paths, such paths, in fact, exist; In part 2, we show by using U-Fragments as gadgets, that when the algorithm fails to muster enough evidence for the existence of certain disjoint paths, it is only when no such paths exist.

REFERENCES

[1] A.V.AHO, J.E.HOPCROFT AND J.D.ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

[2] D.BRAND, "Detecting Sneak Paths in Transistor Networks", IBM Thomas J. Watson Research Center, Yorktown Heights, N. Y., 1983.

[3] S.EVEN, *Graph Algorithms*, Computer Science Press, Maryland 1979.

[4] S.EVEN AND R.E.TARJAN, "Network Flow and Testing Graph Connectivity", *SIAM Journal of Computing*, Vol. 4, No. 4, December 1975.

[5] E.FRANK, *A Data Driven Multiprocessor for Switch Level Simulation of VLSI Circuits.*, Ph. D. Thesis (in preparation), 1984.

[6] J.HOPCROFT AND R.TARJAN, "Efficient Planarity Testing", *Journal of the Association for Computing Machinery*, Vol. 21, No. 4, October 1978.

[7] T.OHTSUKI, "The Two Disjoint Path Problem and Wire Routing Design", in *Graph Theory and Algorithms* (Eds. N.Saito, T.Nishizeki), Springer 108, October 1980.

[8] Y. PERL AND Y.SHILOACH, "Finding Two Disjoint Paths Between Two Pairs of Vertices in a Graph", *Journal of the Association for Computing Machinery*, Vol. 25, No. 1, January 1978.

[9] P.D.SEYMOUR, "Disjoint Paths in Graphs", *Discrete Mathematics*, Vol. 29, No. 3, March 1980.

[10] Y.SHILOACH, "A Polynomial Solution to the Undirected Two Paths Problem", *Journal of the Association for Computing Machinery*, Vol. 27, No. 3, July 1980.

[11] R.TARJAN, "Depth-First Search and Linear Graph Algorithms", *SIAM Journal of Computing*, Vol. 1, No. 2, June 1972.

[12] R.E.TARJAN, *Data Structures and Network Algorithms*, Society for Industrial and Applied Mathematics, Philadelphia, Pa., 1983.

[13] W.T.TUTTE, "Bridges and Hamiltonian circuits in planar graphs", *Aequationes Mathematicae*, **15**, 1977.

[14] W.T.TUTTE, *Graph Theory*, Addison-Wesley Publishing Company, Menlo Park, California, 1984.