[9] N.P. Jouppi. Derivation of Signal Flow Direction in MOS VLSI. *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems*, CAD-6(3):480–490, May 1987.

[10] K.J. Lee, R. Gupta, and M.A. Breuer. An Algorithmic Method for Assigning Signal Flow Directions to MOS Transistors. University of Southern California, June 1990.

[11] B. Mishra. An Efficient Algorithm to find All 'Bidirectional' Edges of an Undirected Graph. In *25th Annual Symposium on Foundations of Computer Science*, pages 207–216, 1984.

[12] B. Mishra. *Some Graph Theoretic Issues in VLSI Design.* PhD thesis, Carnegie-Mellon University, September 1985.

[13] B. Mishra. Bidirectional Edges Problem: Part II, An Eficient Algorithm. Tech. Report, Courant Institute of Mathematical Sciences, September 1993.

[14] B. Mishra and R.E. Tarjan. A Linear-Time Algorithm for Finding an Ambitus. *Algorithmica*, 7:521–554, 1992.

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

[16] O. Ore. *The Four-Color Problem.* Academic Press, New York, London, 1967.

[17] H. Sachs. *Einführung in die Theorie der endlichen Graphen.* Teil H.B.G. Teubner, Leipzig, 1972.

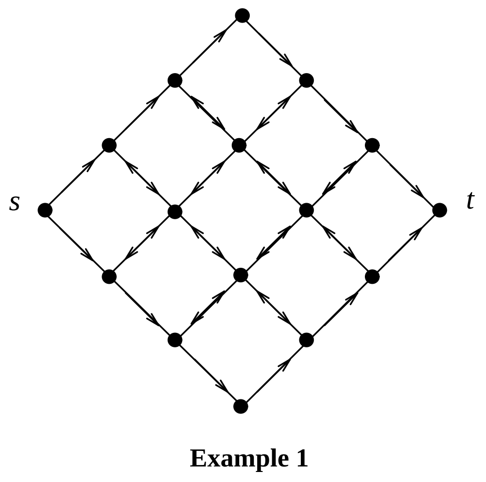
[18] P.D. Seymour. Disjoint paths in graphs. *Discrete Mathematics*, 29:293–309, 1980.

[19] Y. Shiloach. A Polynomial Solution to the Undirected Two Paths Problem. *J. Association for Computing Machinery*, 27(3):445–456, July 1980.

[20] W.T. Tutte. *Connectivity in Graphs.* University of Toronto Press, Toronto, 1966.

[21] W.T. Tutte. Bridges and Hamiltonian Circuits in Planar Graphs. *Aequationes Mathematicae*, 15, 1977.

[22] W.T. Tutte. *Graph Theory.* Addison-Wesley Publishing Co., Menlo Park, California, 1984.



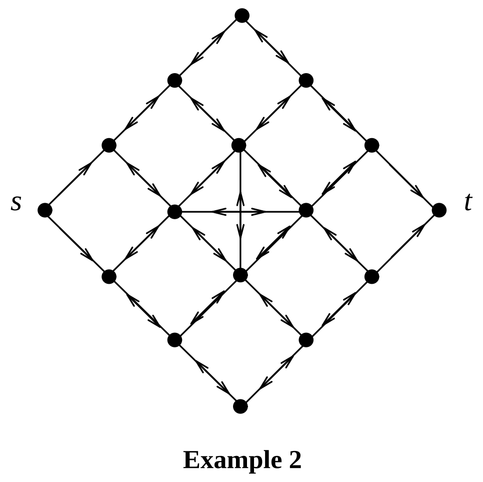


Figure 16: Labeling of the two example graphs.

Acknowledgement

I wish to acknowledge with gratitude the considerable help and advice I have received from Profs. E.M. Clarke, M. Foster, E. Frank, A. Frieze, M. Furst, R. Kannan, L. Rudolph, D. Sleator and R. Statman. Special thanks go to Prof. R.E. Tarjan without whose help this work would not have been possible. Thanks are also due to Prof. J.T. Schwartz for his encouragement to publish these results.

References

[1] Z. Barzilai, D.K. Breece, L.M. Huisman, V.S. Iyengar, and G.M. Silberman. SLS—A Fast Switch Level Simulator. *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems*, CAD-7(8):838–849, August 1988.

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

[3] H.H. Chen, R.G. Mathews, and J.A. Newkirk. An Algorithm to Generate Tests for MOS Circuits at the Switch Level. In *Proc. International Test Conference*, pages 304–312, 1985.

[4] M.A. Cirit. Switch Level Random Pattern Testability Analysis. In *Proc. Design Automation Conference*, pages 587–590, 1988.

[5] E.A. Dinic. Algorithm for solution of a problem of maximum flow in a network with power estimation. *Soviet Math. Docl.*, 11:1277–1280, 1970.

[6] S. Even. *Graph Algorithms*. Computer Science Press, Maryland, 1979.

[7] E. Frank. *A Data Driven Multiprocessor for Switch Level Simulation of VLSI Circuits.* PhD thesis, Carnegie-Mellon University, Pitssburgh, PA, 1985.

[8] J. Hopcroft and R. Tarjan. Efficient Planarity Testing. *Journal of Association for Computing Machinery*, 21, October 1974.

algorithm is

$$\begin{aligned} & O\left((|E|\cdot|V|)+\sum_{j=1}^{n} T(|E(G_j)|\,,|V(G_j)|)\right) \\ = \; & O\left((|E|\cdot|V|)+\sum_{j=1}^{n} T(|E(G_j)|\,,|V|)\right) \\ = \; & O\left((|E|\cdot|V|)+T\left((9\;|E|),|V|\right)\right) \\ = \; & O\left(T(|E|\,,|V|)\right). \quad \square \end{aligned}$$

Theorem 8.2 *Let G be a* Type.IV *graph and G', the nonseparable subgraph derived from G by deleting the vertices s and t together with their incident edges. Let B be the B^{PQ}-bridge of G. Then every edge $e=[u,v]$ of B not incident on s or t is bidirectional.*

PROOF.
First note 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. Now, since the two paths $P[s;x]*\ N[x;y]*\ Q[y;t]$ and $Q[s;y]*\ N[y;x]*\ P[x;t]$ traverse e in either directions, e is bidirectional. $\square$

Algorithm Label-Type-IV(G):

step1. Every edge $[s,u]$ incident s is labeled $\ell([s,u])=\langle s,u\rangle$ and every edge $[u,t]$ incident on t, labeled $\ell([u,t])=\langle u,t\rangle$.

step2. Every edge e of the B^{PQ}-bridge not incident on s or t is labeled bidirectional.

step3. Label the edges $[u,v]$ of $P[s_P^*;t_P^*]$ and $Q[s_Q^*;t_Q^*]$ by using the Algorithm for the Two-Vertex-Disjoint-Paths, *i.e.*,$\langle u,v\rangle \in \ell([u,v])$, if uvp2$(s,u;\ v,t;\ G)$ is true, and $\langle v,u\rangle \in \ell([u,v])$, if uvp2$(s,v;\ u,t;\ G)$ is true. Since every such edge can be labeled in $O(|E|\cdot|V|)$ time since there are at most $|V|$ edges, this step can be done in $O(|E|\cdot|V|^2)$ time.

end{Label-Type-IV.} $\square$

Since steps 1 and 2 can be done in $O(|E|)$ time, a Type.IV graph can be labeled in $O(|E|\cdot|V|^2)$ time. By Theorem 8.1, we get an $O(|E|\cdot|V|^2)$ time labeling algorithm for the general graph. Also it is easy to see that if the step.3 of the algorithm can be modified to run in time $O(|E|\cdot|V|)$ then the algorithm Label-Type-IV will also run in time $O(|E|\cdot|V|)$, and so will the algorithm Label-Graph.

Example 8.1 Both the graphs in figure 1 are Type.IV graphs. Their labelings are shown in figure 16. $\square$

Hence, the set of mutually recursive algorithms reduces the graph G to a set of TYPE.IV graphs after following amount of time:

$$O\left((|E| \cdot |V|) + \sum_{i=1}^{O(|V|)} |pE_i|\right),$$

where $pE_i = \bigcup_{j=1}^{n} pE(G_j)$.

We first demonstrate that $|pE_i| = O(|E|)$, for any stage i. Let H be a graph in stage $i-1$ with at least one graph edge, and $\{H_1, \ldots, H_m\}$ be the graphs of stage i, obtained from H by the application of the appropriate reduction algorithm. Following invariant properties of the reduction algorithms can be verified by straight forward examination:

> If $e \in pE(H_j)$ then either (i) e is an edge of the cycle J and it shares one of its ends with a vertex of attachment of a B^{PQ}-bridge of J, or (ii) e has one end at s or t, and it shares the other end with a graph edge.

Let the pseudo-edges, $pE(H_j)$, of H_j be partitioned as follows:

$$\begin{aligned} pE^{(1)}(H_j) &= \{e \in pE(H_j) : e \text{ is an edge incident at } s \text{ or } t\} \\ pE^{(2)}(H_j) &= \{e \in pE(H_j) : e \in P]s;t[\text{ or } Q]s;t[\text{ in } H_j\}. \end{aligned}$$

(a) Thus there is a function that maps a pseudo-edge, $e' = [u, v] \in pE^{(1)}(H_j)$ to a graph edge, $e \in gE(H_j)$. Further, this function maps at most four distinct pseudo-edges of $pE^{(1)}(H_j)$ to one graph edge of $gE(H_j)$ and hence,

$$\left|pE^{(1)}(H_j)\right| \leq 4\ |gE(H_j)|.$$

(b) Similarly, there is a function that maps a pseudo-edge, $e' = [u, v] \in pE^{(2)}(H_j)$ to a graph edge, $e \in gE(H_j)$, which also belongs to a B^{PQ}-bridge of J. Again, this function maps at most four distinct pseudo-edges of $pE^{(2)}(H_j)$ to one graph edge of $gE(H_j)$ and hence,

$$\left|pE^{(2)}(H_j)\right| \leq 4\ |gE(H_j)|.$$

Combining,

$$\sum_{j=1}^{m} |pE(H_j)| \leq 8\ gE(H).$$

Thus,

$$\sum_{i=1}^{O(|V|)} |pE_i| = \sum_{i=1}^{O(|V|)} 8\ |E| = O\left(|E| \cdot |V|\right).$$

Let $\{G_1, \ldots, G_n\}$ be the set of TYPE.IV graph obtained by the reduction algorithms. The total number of edges in a TYPE.IV graph is $\leq 9\ |gE(G_j)|$. Hence the time complexity of the

G_{C_1} has at least one B^{PQ}-bridge but no B^P- or B^Q-bridge. The rest of the argument proceeds as before. □

Lemma 7.2 *Let C be a nonseparable component of G' whose extremal vertices are labeled 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*

1. *Label of a is σ; and label of b is τ.*
2. *There is a simple path R' from a to b in C; traversing e in the order u and v.*

PROOF.
Straightforward. □

Theorem 7.3 *If* 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.III *graph.*
PROOF.
Follows from the Lemma 7.2. □

8 Final Algorithm and Its Complexity

Theorem 8.1 *Suppose we have an Algorithm* LABEL-TYPE-IV *that correctly labels the edges of a* TYPE.IV *graph $G = (E, V)$ in time $O(T(|E|, |V|)) \geq O(|E| \cdot |V|)$, where $T(\cdot, \cdot)$ is a monotonically-nondecreasing convex function in both its arguments,* i.e.,

$$T(x, \cdot) + T(y, \cdot) \leq T(x + y, \cdot) \qquad \textit{and} \qquad T(\cdot, x) + T(\cdot, y) \leq T(\cdot, x + y),$$

where $x \geq 0$ and $y \geq 0$.

Then the set of mutually recursive algorithms formed by LABEL-GRAPH, LABEL-TYPE-I, LABEL-TYPE-II *and* LABEL-TYPE-III, *correctly labels the edges of an undirected connected strict graph $G = (E, V)$ in time $O(T(|E|, |V|))$.*
PROOF.
The correctness of the algorithm follows immediately from the theorems 4.1, 5.4, 6.3 and 7.3.

Let the pseudo-edges of a graph G be denoted by $pE(G)$ and the graph-edges of G, by $gE(G)$. We define a *set of graphs of a stage* of the algorithm as follows: graphs of stage $1 = \{G\}$. Let graphs of stage i be $\{G_1, \ldots, G_n\}$. If G_j $(1 \leq j \leq n)$ is a TYPE.IV graph then G_j does not contribute any graph to the next stage; otherwise, we apply the appropriate reduction algorithm (one of LABEL-GRAPH, LABEL-TYPE-I, LABEL-TYPE-II and LABEL-TYPE-III) to G_j to reduce it to a set of graphs $\{G_{j,1}, \ldots, G_{j,m_j}\}$ such that $G_{j,k} \prec G_j$.

It is easy to see that the total number of stages is $\leq 10 \cdot |V(G)|$. Since each graph edge occurs only one graph G_j of stage i, in stage i, the algorithms spend the following amount of time per stage:

$$O\left(|E| + \sum_{j=1}^{n} |pE(G_j)|\right).$$

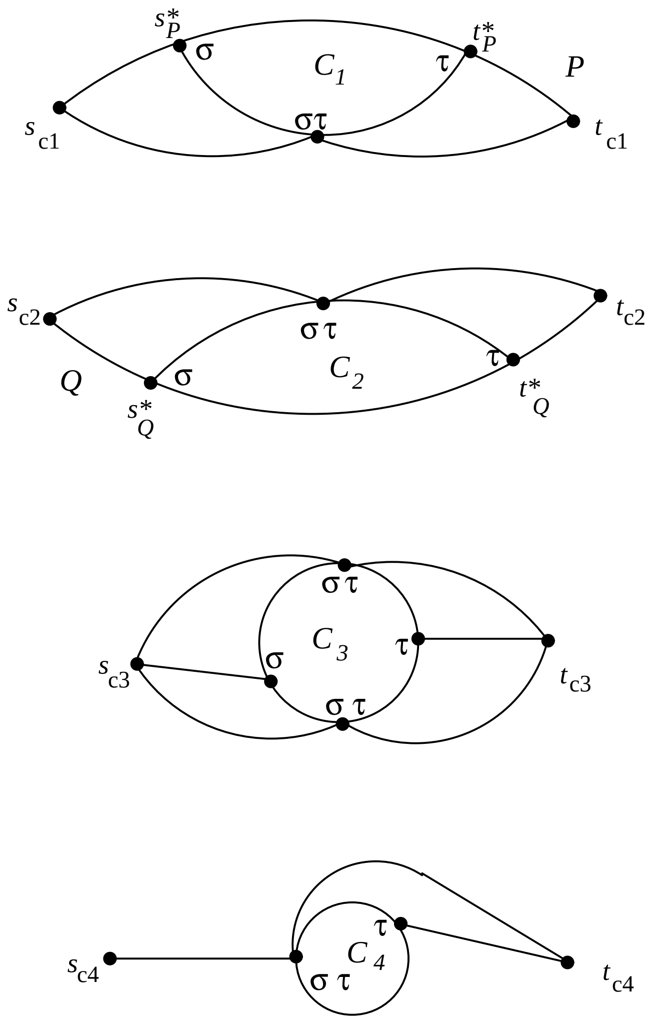


Figure 15: *The graph after* STEP3 *of algorithm* LABEL-TYPE-III.

Lemma 7.1 *Let C_1 and C_2 be the nonseparable components of the subgraph G', containing the subpaths $P[s^*_P; t^*_P]$ and $Q[s^*_Q; t^*_Q]$, respectively. Let G_{C_1} and G_{C_2} be the graphs derived from C_1 and C_2 as in the step3 of the algorithm. Then G_{C_1} and G_{C_2} are well-defined and* TYPE.II *graph.*
PROOF.
First note that, every edge of $P[s^*_P; t^*_P]$ belongs to one nonseparable component of G'.

Now, consider the case when C_1 is the nonseparable component of G' containing $P[s^*_P; t^*_P]$ but avoiding $Q[s^*_Q; t^*_Q]$; then C_1 contains an extremal vertex, x, distinct from s^*_P and t^*_P, and labeled $\{\sigma, \tau\}$. Thus G_{C_1} must be nonseparable. Further if B' is a bridge of J_{C_1} in G_{C_1}, with a vertex of attachment at a on $P]s_{C_1}; t_{C_1}[= P[s^*_P; t^*_P]$ then B' has a vertex of attachment at x, and thus, it is a B^{PQ}-bridge. Hence, G_{C_1} has no B^P-bridge. Nor does it have a B^Q-bridge, since C_1 by assumption is nonseparable. Thus G_{C_1} is a well-defined TYPE.II graph.

On the other hand, if $C_1 = C_2$ then if B' is a bridge of J_{C_1} in G_{C_1} with a vertex of attachment at $a \in P]s_{C_1}; t_{C_1}[$ then it has a vertex of attachment at $b \in Q]s_{C_1}; t_{C_1}[$ and *vice versa*, and hence step2. The vertices of G' adjacent to s in G are labeled σ; and the ones adjacent to t in G, τ. Such vertices will be called the extremal vertices of G'.

Find the nonseparable components, C_1, C_2, ..., C_n, of G', where C_1 and C_2 are the nonseparable components of G' containing the subpaths $P[s^*_P; t^*_P]$ and $Q[s^*_Q; t^*_Q]$, respectively. The separation vertices of a nonseparable component C_i of G' together with the vertices of C_i, which are also extremal vertices of G', are called its *extremal* vertices.

Label each extremal vertex v of each nonseparable component C_i as follows: if v is also an extremal vertex of G' then v has the same label as that in G'; otherwise, v is a separation vertex of C_i and has the label σ (respectively, τ), if there is a path from v to an extremal vertex u of G', already labeled σ (respectively, τ), and the path avoids the edges of C_i.

step3. For each nonseparable component C_i with its extremal vertices labeled σ and τ, do the following: Introduce two new vertices s_{C_i} and t_{C_i}; and join s_{C_i} (respectively, t_{C_i}) to all the extremal vertices of C_i labeled σ (respectively, τ). We call the resulting graph G_{C_i}, with the new edges as pseudo-edges.

1. If $C_1 = C_2$ then let the paths P_{C_1} and Q_{C_1} of G_{C_1} be $[s_{C_1}, s^*_P]* P[s^*_P; t^*_P]* [t^*_P, t_{C_1}]$ and $[s_{C_1}, s^*_Q]* Q[s^*_Q; t^*_Q]* [t^*_Q, t_{C_1}]$, respectively. Otherwise, C_1 has an extremal vertex x (distinct from s^*_P and t^*_P) labeled $\{\sigma, \tau\}$; let the paths P_{C_1} and Q_{C_1} be $[s_{C_1}, s^*_P]* P[s^*_P; t^*_P]* [t^*_P, t_{C_1}]$ and $[s_{C_1}, x] * [x, t_{C_1}]$, respectively. Find the bridges of J_{C_1} in G_{C_1}. The paths P_{C_2}, Q_{C_2} and the bridges of J_{C_2} in G_{C_2} are found in a similar manner. The graphs G_{C_1} and G_{C_2} are well defined and are TYPE.II graphs.

   Label each edge of C_1 and C_2 by recursively calling LABEL-TYPE-II with $(G_{C_1}, s_{C_1}, t_{C_1})$ and $(G_{C_2}, s_{C_2}, t_{C_2})$, respectively.
2. Label each edge of C_i (for $2 < i \leq n$) by recursively calling the main algorithm with $(G_{C_i}, s_{C_i}, t_{C_i})$.

step4. Each edge $[s, u]$ incident on s is labeled $\ell([s, u]) = \langle s, u\rangle$ and each edge $[u, t]$ incident on t is labeled $\ell([u, t]) = \langle u, t\rangle$.

end{LABEL-TYPE-III.} □

Remark 7.1 1. The relation between the nonseparable components of G' and their extremal vertices, is indicated by means of a tree of size no larger than $O(|E(G')|)$. Hence, the extremal vertices of each of the nonseparable components of G' can be labeled, according to the step2 in time $O(\text{SIZE}(\textit{tree}(G'))) = O(|E|)$ time.

2. Since all other steps take $O(|E|)$ time, the algorithm reduces the graph G to the graph G'' in step1, or to graphs G_{C_1}, G_{C_2}, ..., G_{C_n} in step2, in $O(|E|)$ time.
3. Since $G'' = G$, and since G is a TYPE.III graph, whereas G'' is a TYPE.IV graph, $G'' \prec G$. For each nonseparable component C_i of G', $|V(C_i)| < |V(G')|$ and hence for each graph G_{C_i}, $|V(G_{C_i})| < |V(G)|$. Thus, $G_{C_i} \prec G$. □

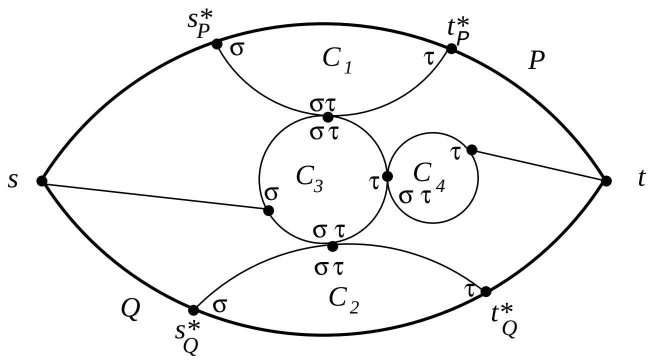


Figure 14: *A* TYPE.III *graph.*

$B_j \in$ Class.j such that e is bidirectional by Lemma 6.2. Hence $\mathbf{B}$ = Class.3. But since $\mathbf{B}$ is proper and not a block of equivalent 3-bridges there are two interlacing B^{PQ}-bridges B_i and $B_j \in$ Class.3 such that e is bidirectional by Lemma 6.2.

2. *There is no proper block of B^{PQ}-bridges, $\mathbf{B}$, such that $\mathbf{B}$ is not a block of equivalent 3-bridges, and e is an edge of the subpath $P[s^*_P(\mathbf{B}); t^*_P(\mathbf{B})]$.*
Notice that no bridge of Class.i interlaces with a bridge of Class.j, (where $i, j = 1, 2, 3$ and $i \neq j$), and if Class.3 is nonempty, then it is either a singleton set or a block of equivalent 3-bridges. If Class.3 is empty or if Class.3 contains two or more equivalent 3-bridges then $\ell(e) = \langle u, v \rangle$. On the other hand, if Class.3 $= \{B\}$ is a singleton then the labeling of e is completely determined by the bridge fragment $J \cup B$. $\square$

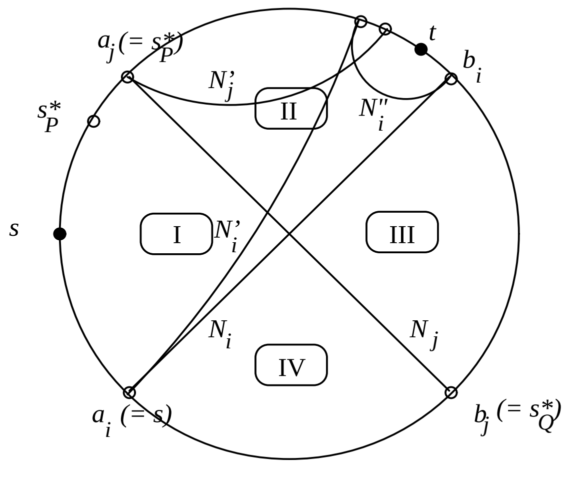
7 Labeling the edges of a Type.III Graph

As in the earlier sections, we assume that there is an algorithm called LABEL-TYPE-IV, to label the edges of a TYPE.IV graph; we present an algorithm to label the edges of a TYPE.III graph $(G; s, t)$ using LABEL-TYPE-IV.

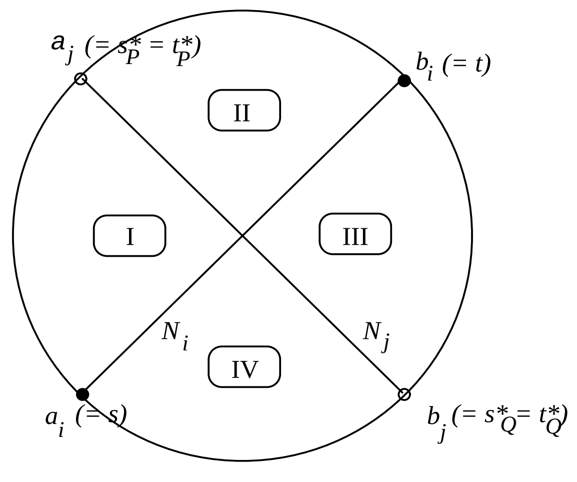
The cycle J in graph G consists of the path $P[s; t]$ and $Q[s; t]$; and has a single B^{PQ}-bridge, B. Without loss of generality we may assume that s^*_P and s^*_Q are adjacent to s and t^*_P and t^*_Q, adjacent to t. If not, G may be modified by contracting the subpaths $P[s; s^*_P]$, $Q[s; s^*_Q]$, $P[t^*_P; t]$ and $Q[t^*_Q; t]$ to single edges $[s, s^*_P]$, $[s, s^*_Q]$, $[t^*_P, t]$ and $[t^*_Q, t]$, respectively, and the labeling of the edges of these subpaths are uniquely determined by that of the corresponding edges, as in the preceding section.

Algorithm LABEL-TYPE-III(G):

step1. Let G' be the subgraph derived from G by deleting the vertices s and t together with the edges incident on s and t. If G' is nonseparable then G is a TYPE.IV graph and by assumption, we can label $G'' = G$ using the algorithm LABEL-TYPE-IV. Otherwise, go to the next step.



- CASE.3 -



- CASE.4 -

Figure 13: CASES.3 AND .4 *of the Lemma.*

If t^*_P and b'_i are distinct then every edge of $P[b'_i; t^*_P]$ is bidirectional. Note that t^*_P must be a vertex of attachment of B_j. Thus, there is a cross-cut between a_j and t^*_P (say, N'_j). Then 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. The subpath $Q[b_i;t^*_Q]$ is treated in a similar manner.

•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, this case could be reduced to one of the previous cases. But, then the theorem is trivially true in this case. □

Theorem 6.3 *If* 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 $e = [u, v]$ be an edge of a TYPE.III graph G. If e is an edge of a B^{PQ}-bridge then e is correctly labeled by lemma 6.1. Thus assume that e is an edge of the path P and that u is to the left of v on P.

Let us divide the set of B^{PQ}-bridges, $\mathcal{B}$, into following three disjoint subsets:

- Class.1= $\{B \in \mathcal{B} : B$ has no vertex of attachment on $P[v;t].\}$
- Class.2= $\{B \in \mathcal{B} : B$ has no vertex of attachment on $P[s;u].\}$
- Class.3 = $\mathcal{B} \setminus \{$ Class.1 $\cup$ Class.2 $\}$
  = $\{B \in \mathcal{B} : B$ has a vertex of attachment on $P[s;u]$ and on $P[v;t].\}$

Consider the following two cases:

1. *There is a proper block of B^{PQ}-bridges, **B**, such that **B** is not a block of equivalent 3-bridges, and e is an edge of the subpath $P[s^*_P(\mathbf{B}); t^*_P(\mathbf{B})]$.*
   We may assume that no bridge of Class.i interlaces with a bridge of Class.j, (where $i, j = 1, 2, 3$ and $i \neq j$), since, otherwise, there are two interlacing B^{PQ}-bridges $B_i \in$ Class.i and

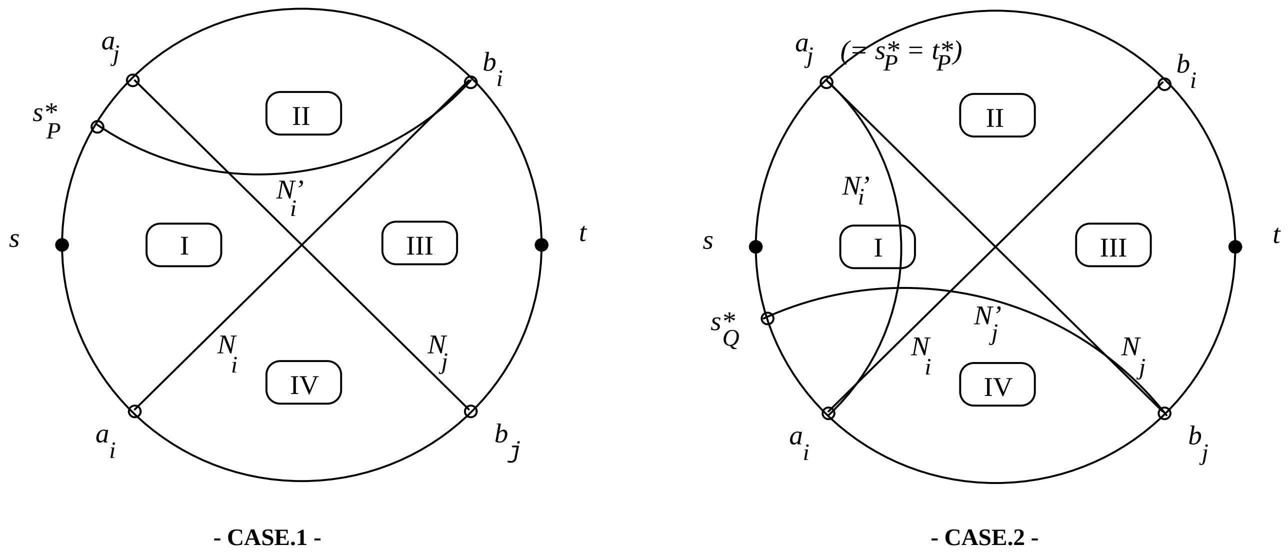


Figure 12: CASES.1 AND .2 *of the Lemma.*

edges of $P[s^*_P; a_j]$ in either directions. The cases for $P[b_i; t^*_P]$, $Q[s^*_Q; a_i]$ and $Q[b_j; t^*_Q]$ are treated similarly.

•CASE.2 *s is an internal vertex of quadrant I and t, of quadrant II.*
We may assume that $a_j = s^*_P = t^*_P$. Since, otherwise, this case could be reduced to the previous case.

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.

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.

If s^*_Q and a_i are distinct, then every edge of $Q[s^*_Q; a_i]$ is bidirectional. Note that s^*_Q must be a vertex of attachment of B_j. Thus, there is a cross-cut between s^*_Q and b_j, say N'_j. Then 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 order. 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.*
We may assume that $s^*_P = a_j$ and $s^*_Q = b_j$. Since, otherwise, this case could be reduced to one of the two previous cases.

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).

Every edge of the residual path $Q[b_j; b_i]$ is bidirectional. The paths $Q[s; t]$ and $N_i[s; b_i]*\ Q[b_i; b_j]*\ N_j[b_j; a_j]*\ P[a_j; t]$ 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.

PROOF.
The forward direction is trivial as G' is a minor of G. The converse: Let $R[s;t] \in G$ be a path from s to t that traverse e in the order u and v in G. Using Proposition 2.3, we can write $R[s;t]$ as $R[s;x]*\ R[x;y]*\ R[y;t]$ such that x and y are vertices of attachment of B and $R[x;y]$ belongs completely to B, is a a cross-cut of J and traverses e in the order u, v.

We need to show that there is a path in G' that traverses e in the same order. There are two cases to consider: First, if $x \in P[s;t]$ and $y \in Q[s;t]$ (or vice versa) then the desired path is $P'[s;x]*\ R[x;y]*\ Q'[y;t]$ in G'. If, on the other hand, x and y are both in $P]s;t[$ (symmetrically, x, $y \in Q]s;t[$), we proceed as follows: Since B is B^{PQ}-bridge it has another attachment $z \in Q]s;t[$. By Proposition 2.5, there is a cross-cut either between x and z or between y and z that contains e. Assume the former, *i.e.*,the cross-cut is $R'[x;z]$. Then the paths $P'[s;x]*\ R'[x;z]*\ Q'[z;t]$ and $Q'[s;z]*\ R'[z;x]*\ P'[x;t]$ are both in G' and one of them traverses e in the order u, v. $\square$

Lemma 6.2 *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 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.
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 cycle 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.

Without loss of generality, we may assume that s and t do not lie in the same quadrant. Suppose not, *i.e.*,s and t both belong to the same quadrant, say I, which contains $P[s;t]$. Since B_j is a B^{PQ}-bridge it has an attachment on $P]s;t[$; call it a'_j. Let the path from a'_j to b_j be the new cross-cut N'_j. Now s and t lie in different quadrants defined by a_i, b_i, a'_j and b_j.

Now we prove the theorem, separately in each of the following four cases:

1. *s and t are internal vertices of diametrically opposite quadrants.*
2. *s and t are internal vertices of adjacent quadrants.*
3. *s is one of a_i, b_i, a_j or b_j; and t is an internal vertex of an adjacent quadrant.*
4. *$s = a_k$ and $t = b_k$, where $k = i$ or j.*

•CASE.1 *s is an internal vertex of quadrant I and t is an internal vertex of quadrant III.*
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.

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. Note that s^*_P must be a vertex of attachment of B_i. Thus, 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

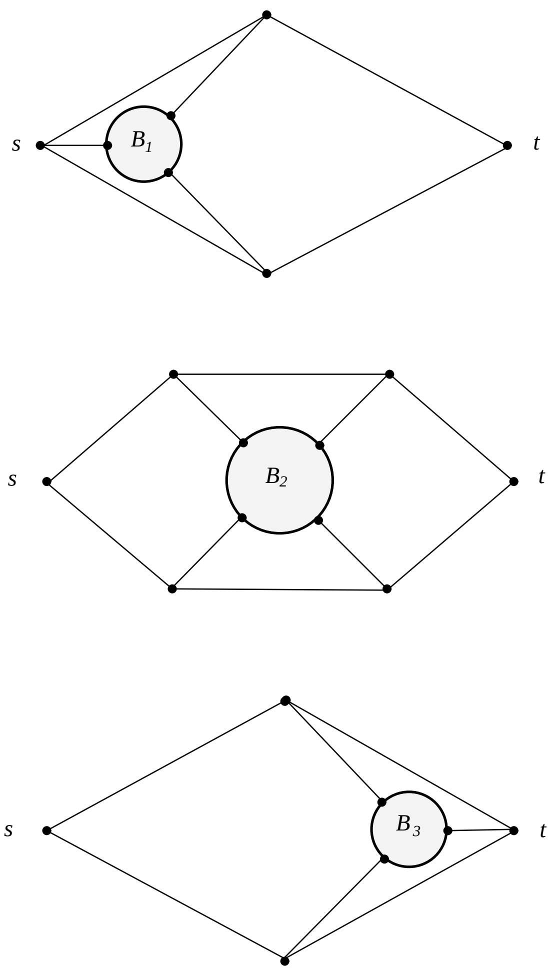


Figure 11: *The graph after* STEP1 *of algorithm* LABEL-TYPE-II.

2. Since each step takes $O(|E|)$ time, the algorithm reduces the graph G to graphs G'_{B_1}, G'_{B_2}, ..., G'_{B_n} in $O(|E|)$ time.

3. For all $1 \leq i \leq n$, since $\left|V(G'_{B_i})\right| \leq |V(G)|$, and since the algorithm introduces no new edge with ends at s and t in G'_{B_i}, the first and second elements of the signature of G'_{B_i} are no larger than those of G. But since G is a TYPE.II graph, whereas G'_{B_i} is a TYPE.III graph, $G'_{B_i} \prec G$.
   □

Lemma 6.1 *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.

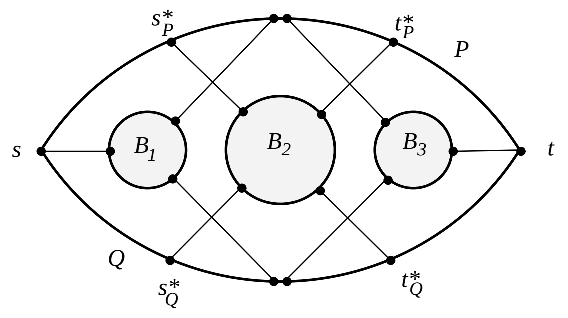


Figure 10: *A* TYPE.II *graph.*

Algorithm LABEL-TYPE-II(G):

step1. Let G be a TYPE.II graph with the set of B^{PQ}-bridges, $\mathcal{B} = \{B_1, B_2, \ldots, B_n\}$ $(n > 0)$. Let $\mathbf{B}_1, \ldots, \mathbf{B}_l$ be a partition of B, where $\mathbf{B}_i$ is a block of B^{PQ}-bridges.

Modify each bridge fragment $G_{B_i} = B_i \cup J$ as follows: Let $v_1, \ldots, v_{p-1}$ and $w_1, \ldots, w_{q-1}$ be the vertices of attachment of B_i on $P]s;t[$ and $Q]s;t[$, ordered in their left-to-right order, respectively. Let $\mathcal{L} = \{P[v_0(=s);v_1], \ldots, P[v_{p-1};v_p(=t)], Q[w_0(=s);w_1], \ldots, Q[w_{q-1};w_q(=t)]\}$ be the set of residual paths of J.

G'_{B_i} is derived from G_{B_i} by contracting the subpaths $P[v_{i-1};v_i]$ and, $Q[w_{j-1};w_j]$, $(1 \le i \le p$ and $1 \le j \le q)$ to single links $[v_{i-1}, v_i]$ and $[w_{j-1}, w_j]$ respectively, the "pseudo-edges" of the subpath. $J'_i = \{P'_i\} \cup \{Q'_i\}$ is the cycle in G'_i derived from J by the contraction. Since, each such G'_{B_i} is a TYPE.III graph, by assumption, we can label the edges of $(G'_{B_i}; s, t)$ using the algorithm TYPE.III.

step2. Let $\mathcal{E}_i = \{e'_{i,1}, e'_{i,2}, \ldots, e'_{i,l_i}\} \subseteq E(P'_i)$ such that $e'_{i,1}, e'_{i,2}, \ldots, e'_{i,l_i}$ are bidirectional in G'_{B_i}. Label the edges of P corresponding to the ones in $\mathcal{E}_i$ of P bidirectional. Label every other edge $e = [u, v]$ of P unidirectional, with the label $\langle u, v\rangle$, if u is to the left of v on P. Label the edges of Q, in a similar manner.

step3. Let $\mathbf{B} = \mathbf{B}_i$ be a block of B^{PQ}-bridges, where $1 \le i \le l$. If $\mathbf{B}$ is a proper block but not a block of equivalent 3-bridges, label the edges of the subpaths $P[s^*_P(\mathbf{B}); t^*_P(\mathbf{B})]$ and $Q[s^*_Q(\mathbf{B}); t^*_Q(\mathbf{B})]$ bidirectional.

end{LABEL-TYPE-II.} □

Remark 6.1 1. It is easily seen that the step2. can be done in time $O(|pE + E| + |V|)$, where $|pE| =$ is the total number of pseudo-edges introduced by the algorithm. We will show that $|pE| = O(|E|)$. Moreover, the blocks of B^{PQ}-bridges bridges can be found in linear time.

must hold. But from the assumption it follows that, UVP2$(v_{i-1}, u; v, v_i; G_i)$ must be false. But then UVP2$(v_{i-1}, v; u, v_i; G_i)$ holds, and again from our assumption it follows that UVP2$(s, v_i; v_{i-1}, t; G^i)$ must be false.

For the second part, the forward direction is obvious. In the other direction we proceed as follows:

Let $R_1[s; u]$ and $R_2[v; t]$ be two vertex-disjoint paths in G^{i-1}. If R_1 and R_2 avoid the subgraph G_i then the lemma holds trivially.

Hence assume that R_1 contains a vertex $a \in V(G_i)$. By the Lemma 5.1, it must contain the vertices v_{i-1} and v_i. Since R_2 is vertex disjoint with R_1, R_2 must avoid the subgraph G_i. Let R_1' be the path in G^i derived from path R_1, by contracting the subpath $R_1[v_{i-1}; v_i]$ to the pseudo-edge $[v_{i-1}, v_i]$. The paths R_1' and R_2 are simple and vertex-disjoint in G^i. $\square$

The following is a direct consequence of the preceding lemma.

Corollary 5.3 *Let $e = [u, v] \in E(G)$.*

1. *If $e \in E(G_i)$, for some $1 \le i \le p$, then* UVP2$(s, u; v, t; G)$ *if and only if (i)* UVP2$(v_{i-1}, u; v, v_i; G_i)$ *or (ii)* UVP2$(s, v_i; v_{i-1}, t; G')$.
   Similarly, if $e \in E(G_i)$, for some $p + 1 \le i \le p + q$, then UVP2$(s, u; v, t; G)$ *if and only if (i)* UVP2$(w_{i-p-1}, u; v, w_{i-p}; G_i)$ *or (ii)* UVP2$(s, w_{i-p}; w_{i-p-1}, t; G')$.
2. *If $e \notin E(G_i)$, for all $1 \le i \le p+q$, then* UVP2$(s, u; v, t; G)$ *if and only if* UVP2$(s, u; v, t; G')$.
   $\square$

Theorem 5.4 *If* 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.

If G has no B^{PQ}-bridge then, since the subgraphs G_1 and G_2 meet each other only in the vertices s and t, and otherwise avoid each other, the correctness of the algorithm, for this case follows immediately.

If G has one or more B^{PQ}-bridge, then the correctness of the algorithm follows from the previous corollary. $\square$

6 Labeling the edges of a Type.II Graph

As before, we assume that we have an algorithm, called LABEL-TYPE-III to label the edges of a TYPE.III graph; we present an algorithm to label the edges of a TYPE.II graph $(G; s, t)$ using LABEL-TYPE-III. The algorithm follows:

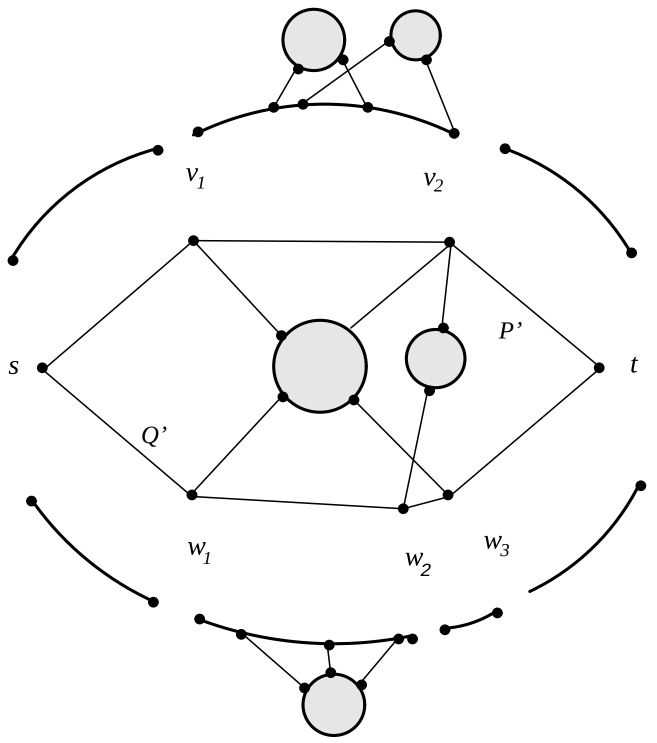


Figure 9: *The graph after* STEP1 *of algorithm* LABEL-TYPE-I.

Lemma 5.1 *Consider the subgraph G_i of G^{i-1} defined earlier.*

If $1 \leq i \leq p$ then every path in G^{i-1}, connecting s or t to a vertex of G_i must contain v_{i-1} or v_i. Similarly, if $p+1 \leq i \leq p+q$ then every path in G^{i-1}, connecting s or t to a vertex of G_i must contain w_{i-p-1} or w_{i-p}.

PROOF.
This is a direct consequence of the fact that J^{i-1} is an ambitus of G^{i-1}. $\square$

Lemma 5.2 *Let $1 \leq i \leq p+q$, and $e = [u, v] \in E(G^{i-1})$.*

1. *If $1 \leq i \leq p$ and $e \in E(G_i)$ then* UVP2$(s, u; v, t; G^{i-1})$ *if and only if*
   (i) UVP2$(v_{i-1}, u; v, v_i; G_i)$ *or (ii)* UVP2$(s, v_i; v_{i-1}, t; G^i)$.
   Similarly, if $p+1 \leq i \leq p+q$ and $e \in E(G_i)$ then UVP2$(s, u; v, t; G^{i-1})$ *if and only if*
   (i) UVP2$(w_{i-p-1}, u; v, w_{i-p}; G_i)$ *or (ii)* UVP2$(s, w_{i-p}; w_{i-p-1}, t; G^i)$.
2. *If $e \notin E(G_i)$ then* UVP2$(s, u; v, t; G^{i-1})$ *if and only if* UVP2$(s, u; v, t; G^i)$.

PROOF.
Consider the case when $1 \leq i \leq p$. For the first part:
($\Rightarrow$) Assume that UVP2$(s, u; v, t; G^{i-1})$ holds. Since by the Lemma 5.1, the vertex disjoint paths of G^{i-1} must contain v_{i-1} and v_i, either UVP2$(v_{i-1}, u; v, v_i; G_i)$ or UVP2$(s, v_i; v_{i-1}, t; G^i)$ (or both) must hold.
($\Leftarrow$) Assume that UVP2$(s, u; v, t; G^{i-1})$ does not hold. Since G^{i-1} is nonseparable, UVP2$(s, v; u, t; G^{i-1})$ must hold. By the Lemma 5.1, it follows that either UVP2$(v_{i-1}, u; v, v_i; G_i)$ or UVP2$(v_{i-1}, v; u, v_i; G_i)$

All the subgraphs G_1, G_2, ..., G_{p+q} are well-defined. Simply, note that since J is an ambitus, every B^P-bridge (respectively, B^Q-bridge) of J avoids every B^{PQ}-bridge and has all its vertices of attachment on some residual path $P[v_{i-1}; v_i]$ (respectively, $Q[w_{i-p-1}, w_{i-p}]$). Moreover, the graphs G', G_1, ..., G_{p+q} partition the edges of the graph G.

1. The graph G' is a Type.II graph; label the edges of G' using the algorithm Label-Type-II.
2. For each subgraph G_i $(1 \leq i \leq p)$ repeat the following: If the pseudo-edge, $[v_{i-1}, v_i]$ is labeled bidirectional by the first substep, label all edges of G_i bidirectional; otherwise, label the edges of G_i by recursively calling the main algorithm with the argument (G_i, v_{i-1}, v_i).
3. For each subgraph G_i $(p+1 \leq i \leq p+q)$ repeat the following: If the pseudo-edge, $[w_{i-p-1}, w_{i-p}]$ is labeled bidirectional by the first substep, label all edges of G_i bidirectional; otherwise, label the edges of G_i by recursively calling the main algorithm with the argument $(G_i, w_{i-p-1}, w_{i-p})$.

end{Label-Type-I.} □

Remark 5.1 1. See Mishra and Tarjan[14] for an $O(|E|)$ algorithm to find an ambitus.

2. Since all other steps take $O(|E|)$ time, the algorithm reduces the graph G to G_1 and G_2 in step2, or to graphs G', G_1, ..., G_{p+q} in step3, in $O(|E|)$ time.
3. If G_1 and G_2 are the graphs generated in step2, then $|V(G_i)| \leq |V(G)|$ $(i = 1, 2)$. Moreover, if $|V(G_1)| = |V(G)|$ then G_2 must be link-graph, and the second element of G_1's signature is strictly smaller than that of G. Similarly for G_2. Hence $G_i \prec G$ $(i = 1, 2)$.
4. If G', G_1, ..., G_{p+q} are the graphs generated in step3, then we see that
   (a) Since $|V(G')| \leq |V(G)|$, and since the algorithm introduces no new edge with ends at s and t in G', the first and second elements of G' are no larger than those of G. But since G is a Type.I graph, whereas G' is a Type.II graph, $G' \prec G$.
   (b) For each subgraph G_i $(1 \leq i \leq p+q)$, since $|V(G_i)| < |V(G)|$, $G_i \prec G$. □

Let G be a Type.II graph with an ambitus J such that G has one or more B^{PQ}-bridge. We define a sequence of minors of G as follows:

1. $G^0 = G$, and $J^0 = J$.
2. (a) $1 \leq i \leq p$. Then $G_i = \langle E(P[v_{i-1}; v_i]) \cup E(\mathcal{B}^P(v_{i-1}, v_i)\rangle$ is a subgraph of G^{i-1}. Let G^i be a minor of G^{i-1} obtained by first deleting the edges of $\mathcal{B}^P(v_{i-1}, v_i)$ and then contracting the subpath $P[v_{i-1}, v_i]$ to a single edge $[v_{i-1}, v_i]$. J^i is the ambitus of G^i obtained from J^{i-1} by the contraction.
   (b) $p+1 \leq i \leq p+q$. Then $G_i = \langle E(Q[w_{i-p-1}; w_{i-p}]) \cup E(\mathcal{B}^Q(w_{i-p-1}, w_{i-p})\rangle$ is a subgraph of G^{i-1}. Let G^i be a minor of G^{i-1} obtained by first deleting the edges of $\mathcal{B}^Q(w_{i-p-1}, w_{i-p})$ and then contracting the subpath $Q[w_{i-p-1}, w_{i-p}]$ to a single edge $[w_{i-p-1}, w_{i-p}]$. J^i is the ambitus of G^i obtained from J^{i-1} by the contraction.

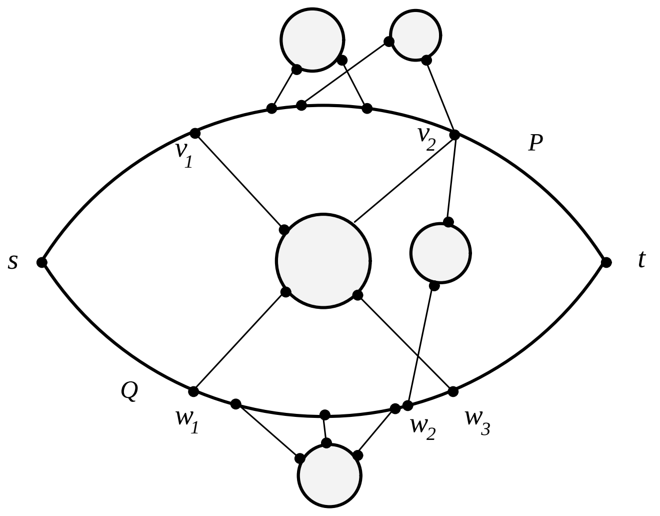


Figure 8: *A* TYPE.I *graph.*

Algorithm LABEL-TYPE-I(G):

step1. Modify the cycle J to an ambitus.

step2. If G has no B^{PQ}-bridge then divide G into subgraphs

$$G_1 = \langle E(P) \cup E(\mathcal{B}^P)\rangle, \quad \text{and} \quad G_2 = \langle E(Q) \cup E(\mathcal{B}^Q)\rangle,$$

where $\mathcal{B}^P$ and $\mathcal{B}^Q$ are the (possibly, empty) set of B^P- and B^Q-bridges of J in G. Label (G_1, s, t) and (G_2, s, t), separately. Otherwise, go to the next step.

step3. Let $v_1, \ldots, v_{p-1}$ and $w_1, \ldots, w_{q-1}$ be the vertices of attachment of the set of B^{PQ}-bridges on $P]s;t[$ and $Q]s;t[$, ordered in their left-to-right order, respectively. Let $\mathcal{L} = \{P[v_0(= s); v_1], \ldots, P[v_{p-1}; v_p(= t)], Q[w_0(= s); w_1], \ldots, Q[w_{q-1}; w_q(= t)]\}$ be the set of residual paths of J.

Let G' be the minor of G obtained by first deleting every B^P- and B^Q-bridge and then contracting the subpaths $P[v_{i-1}; v_i]$ and, $Q[w_{j-1}; w_j]$, $(1 \le i \le p$ and $1 \le j \le q)$ to single links $[v_{i-1}, v_i]$ and $[w_{j-1}, w_j]$ respectively, the "pseudo-edges" of the subpath.

Let $G_1, \ldots, G_p, G_{p+1}, \ldots, G_{p+q}$ be the subgraph of G, where

$$G_i = \langle E(P[v_{i-1}; v_i]) \cup E(\mathcal{B}^P(v_{i-1}, v_i))\rangle, \quad (1 \le i \le p),$$

and

$$G_i = \langle E(Q[w_{i-p-1}; w_{i-p}]) \cup E(\mathcal{B}^Q(w_{i-p-1}, w_{i-p}))\rangle, \quad (p+1 \le i \le p+q),$$

and $\mathcal{B}^P(v_{i-1}, v_i)$ (respectively, $\mathcal{B}^Q(w_{i-p-1}, w_{i-p})$) are the set of B^P-bridges (respectively, B^Q-bridges) with all the vertices of attachment on $P[v_{i-1}; v_i]$ (respectively, $Q[w_{i-p-1}, w_{i-p}]$.)

2. The linear time complexity of step.3 follows from the Remark 2.3.
3. Since all other steps take $O(|E|)$ time, the algorithm reduces the graph G to graphs G_1, G_3, …, G_k in $O(|E|)$ time.
4. Since $|V(G_1)| \leq |V(G)|$, and since the algorithm introduces no new edge in G_1, the first and second elements of G_1's signature are no larger than those of G. But since G is an arbitrary graph, whereas G_1 is a TYPE.I graph, $G_1 \prec G$.
   For each subgraph G_i $(3 \leq i \leq k)$, since $|V(G_i)| < |V(G)|$, $G_i \prec G$. $\square$

Theorem 4.1 *If* 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.*
PROOF.
Since G_1 is derived by deleting the bridges $G_2, \ldots, G_k$ of a nonseparable component C_i, by Proposition 2.8, G_1 is nonseparable. Moreover, the bridges of G_1 with respect to J are B^P-, B^Q- and B^{PQ}-bridges. Hence, G_1 is of TYPE.I. The rest follows from the following claim.
Claim: If $e = [u, v]$ is an edge of G_j (for $1 \leq j \leq k$), then there is simple path from s_i to t_i in C_i traversing e in the order u, v if and only if there is a simple path from s_i to t_i in G_j traversing e in the same order.
Proof of claim:
$(\Leftarrow)$ True, since G_j is a subgraph of C_i.
$(\Rightarrow)$ First note that if there is a simple path R from s_i to t_i containing x, a vertex in the nucleus $N(G_j)$ of a bridge G_j (for $2 \leq j \leq k$), then the edges of R belong to G_j. (Follows from proposition 2.3.) Now, assume that R is a path in C_i traversing e in the order u and v. Then there are few cases to consider: If $e = [u, v] \in G_j$, $(2 \leq j \leq k)$ and G_j is degenerate then the path $s_i = u \rightarrow v = t_i$ is the required path in G_j. If, on the other hand, G_j is proper then either u or v (or both) belongs to $N(G_j)$ and the path R lies completely in G_j and traverses e in the same order. Finally, if $e = [u, v] \in G_1$ then the claim holds easily, since otherwise R has an edge or vertex in some G_j (for $2 \leq j \leq k$) and then all the edges of R belong to G_j. This contradicts the premise that $e \in G_1$.

As a result of this claim and the proposition 3.1, we see that the edge labeling given by the Algorithm LABEL-GRAPH is, in fact, correct. $\square$

5 Labeling the Edges of a Type.I Graph

We assume that there is an algorithm, called LABEL-TYPE-II, to label the edges of a TYPE.II graph; we present an algorithm to label the edges of a TYPE.I graph $(G; s, t)$ using LABEL-TYPE-II. The algorithm follows:

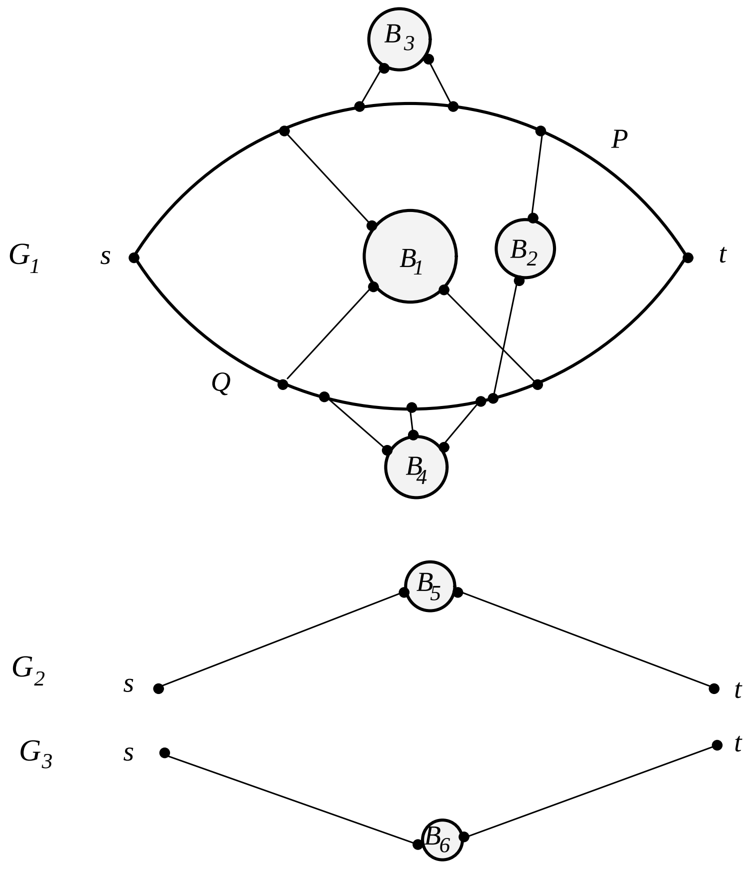


Figure 7: *The graph after* STEP3 *of algorithm* LABEL-GRAPH.

1. G_1 is a TYPE.I graph; and by assumption, the edges of G_1 can be labeled using the algorithm LABEL-TYPE-I.
2. G_2 is a link graph with ends at s and t; label $\ell([s,t]) = \langle s,t\rangle$.
3. Label the edges of $G_j \in \mathcal{G}$, (for $3 \le j \le k$) by recursively calling the algorithm LABEL-GRAPH with the argument (G_j, s, t).

end{LABEL-GRAPH.} $\square$

Remark 4.1 1. The time complexity of step.2 of the algorithm is $O(|E|)$. Assume that all edges of the network have capacity one. We observe that in Dinic's algorithm for maximal flow in a layered network, each time we find a path all the edges on it become blocked, and in case the last edge leads to a dead end, we back-track on this edge and it becomes blocked. Thus the total number of edge traversals is bounded by $|E|$, per *phase*. Since we are interested in finding only two paths, we need at most *two* phases, and hence, $O(|E|)$ time.

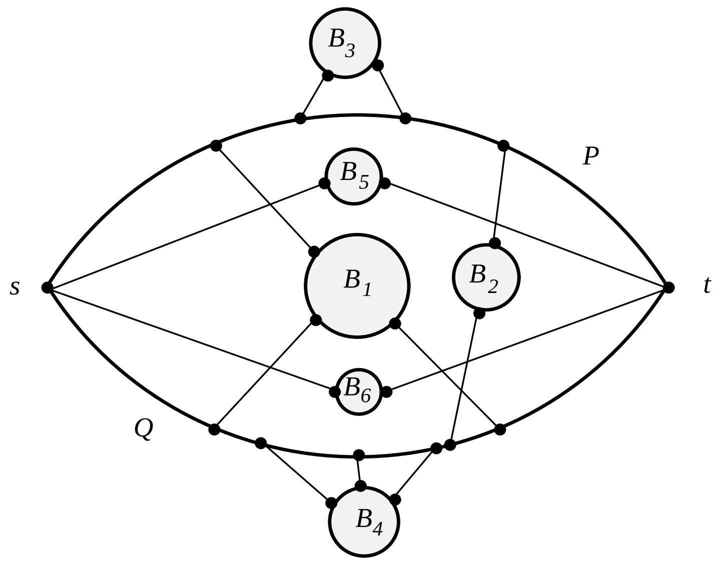


Figure 6: *A non-separable graph with the bridges of J.*

Algorithm LABEL-GRAPH(G):

step1. Find the nonseparable components of the graph. Let $C_1, C_2, \ldots, C_m$ be the chain of nonseparable components G with respect to s and t. 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. If $(C_i;\ s_i,\ t_i)$ is a link-graph (*i.e.*,$[s_i, t_i]$ is an isthmus of G) then $\ell([s_i, t_i]) = \langle s_i, t_i \rangle$. Otherwise, find two internally vertex-disjoint paths connecting s_i and t_i in C_i by applying Dinic's Algorithm for MaxFlow problem.(Dinic[5]) For every nonseparable graph, such paths exist. We call these two paths P and Q.

step3. Find the set of bridges of the cycle $J = \{P\} \cup \{Q\}$ in C_i. Let $\mathcal{B}$ stand for the B^P-, B^Q- and B^{PQ}-bridges of J and $\mathcal{G}$ for the remaining set of bridges of J. Notice that every bridge in the set $\mathcal{G}$ has exactly two vertices of attachment s and t, and since G is a strict graph at most one of such bridges is degenerate.

Let $\mathcal{G}_1 = \{G_1 = P \cup Q \cup \mathcal{B}\}$, $\mathcal{G}_2 = \{G_2\}$ = the set of degenerate bridges with vertices of attachment at s and t, and $\mathcal{G}_3 = \{G_3, \ldots, G_k\}$ = the set of proper bridges with vertices of attachment at s and t.

•TYPE II GRAPHS: A graph is said to be of *type II* if it is of TYPE.I and has at least one B^{PQ}-bridge, but no B^P- or B^Q-bridges.
•TYPE III GRAPHS: A graph is said to be of *type III* if it is of TYPE.II and has only one single B^{PQ}-bridge.
•TYPE IV GRAPHS: A graph is said to be of *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. □

Definition 3.4 SIGNATURE OF A GRAPH. Let G be a graph with two distinguished vertices s and t. To G, we assign a triple of positive integers $\langle i_i, i_2, i_3\rangle$, called its *signature*, where

- $i_1 = |V(G)|$
- $i_2 =$ the number of edges of the form $[s,t]$
- $i_3 = 0$, 1, 2, 3 and 4, depending on whether G is a TYPE.IV, TYPE.III, TYPE.II, TYPE.I or an arbitrary graph.

We say graphs $G_1 \prec G_2$, if

$$\textit{signature}(G_1) <_{\text{lex}} \textit{signature}(G_2).$$

This defines a well-ordering among the graphs. Let $G_0 \succ G_1 \succ \cdots \succ G_n$ be a decreasing chain of graphs ordered by the above ordering. If G_0, G_1, ..., G_n are strict graphs then the second elements of their signature can have the values 0 or 1, and hence $n \le 10 \cdot |V(G_0)|$. □

Intuitively, these four classes of graphs: TYPE.I, TYPE.II, TYPE.III and TYPE.IV graphs, form a hierarchy of graphs of successively simpler structure. In the following sections we present four algorithms: LABEL-GRAPH, LABEL-TYPE-I, LABEL-TYPE-II and LABEL-TYPE-III, which take as input an arbitrary graph, a TYPE.I graph, a TYPE.II graph and a TYPE.III graph, respectively, and reduce the graph into graphs with a strictly smaller signature. Each of these algorithms labels the graphs of the appropriate type correctly, on the assumption of existence of correct algorithms for the subsequent types. Moreover, this set of four mutually recursive algorithms, together with an algorithm for TYPE.IV graph of time complexity $O(T(|E|,|V|)) \ge O(|E| \cdot |V|)$, provide an $O(T(|E|,|V|))$ algorithm for labeling a general graph. This chapter concludes with an $O(|E| \cdot |V|^2)$ algorithm for TYPE.IV graphs, and hence a similarly efficient algorithm for general graphs.

However, each of these algorithms may introduce additional edges into the resulting smaller graphs; these edges will be referred to as *pseudo-edges*, as distinguished from the original edges of the graph, the later being referred to as *graph-edges*. The algorithms achieve their efficiency by introducing these pseudo-edges in a controlled manner such that at any stage *not too many* pseudo-edges are introduced.

4 Labeling the Edges of a General Graph

In this section we assume that we have an algorithm, called LABEL-TYPE-I to label the edges of a TYPE.I graph; we provide an algorithm to label the edges of a general graph $(G; s, t)$ using LABEL-TYPE-I. The algorithm follows:

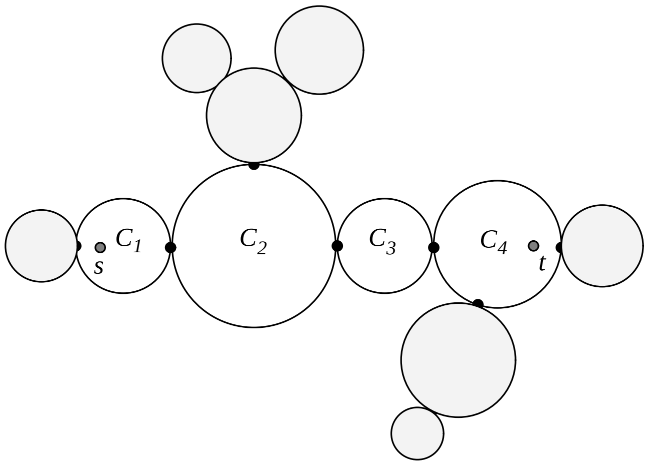


Figure 5: *The chain graph.*

Assumption 3.2 G is a chain graph. $\square$

Let $G = (V, E)$ be a graph with source s and sink t and let $G' \subseteq G$ be a minimal subgraph of G containing s and t such that every nonseparable component of G' is a nonseparable component of G and G' is a chain graph.

Let $C_s = C_1, C_2, \ldots, C_k = C_t$ be the nonseparable components of G', where $s \in V(C_s)$ and $t \in V(C_t)$. Let $a_1, a_2, \ldots, a_{k-1}$ be the separation vertices of G', where $a_i \in V(C_i) \cap V(C_{i+1})$.

The nonseparable components $C_1, C_2, \ldots, C_k$ are the *chain of nonseparable components* of G with respect to s and t.

With each C_i, we associate two vertices s_i and t_i, where $s_1 = s$, $t_k = t$ and $s_i = a_{i-1}(1 < i \leq k)$ and $t_i = a_i(1 \leq i < k)$. Let $E_1 \subseteq E$ be the edges of the nonseparable components C_i's on the path P and $E_2 = E \setminus E_1$. (See Figure 5.) Note that:

Proposition 3.1 (Mishra[12]) *Let $G = (V, E)$ be an undirected graph with a source s and sink t and let E_1 and E_2 be the partition of edges, E as described. For each edge $e \in E$,*

$$e \in E_2 \text{ if and only if } \Big(\exists \text{ no simple path from } s \text{ to } t \text{ containing } e\Big). \quad \square$$

Thus each edge $[u, v] \in E_2$ has the labeling: $\ell([u, v]) = \emptyset$. Similarly, the labeling of each edge in E_1 is determined by analyzing the subgraph G', described earlier. Further note that, given any graph $G = (V, E)$ with source s and sink t, we can find its nonseparable components and the separation vertices in time $O(|E|)$. Hence, the chain graph $G' \subseteq G$, and ultimately, the set of edges E_2 can all be found in linear time.

§3.2 A Classifications of Graphs

Definition 3.3 Classification of Graphs. We introduce a classification of graphs as follows:
•Type I Graphs: A nonseparable graph G is said to be of *type I* if it has a cycle J containing the vertices s and t and all its bridges are B^P-, B^Q- or B^{PQ}-bridges.

Remark 2.16 Following Tutte[22], 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's, the *arms* of Y.

Proposition 2.5 *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.* $\square$

Proposition 2.6 *Let B_1 and B_2 be distinct overlapping bridges of a cycle J of G. Then either B_1 and B_2 interlace or they are equivalent 3-bridges.* $\square$

Proposition 2.7 *Let G be a nonseparable graph; $e = [u, v] \in E(G)$, an edge of G; and a and $b \in V(G)$, two distinct vertices of G. Then there is a simple path $N[a; b]$ in G from a to b containing e.* $\square$

Proposition 2.8 *Let G be a nonseparable graph with a cycle J and let $B_1, \ldots, B_k$ the bridges of J in G. Let G' be the graph derived from G by deleting a bridge B_i.*

1. *$w(G, B_i) \geq 2$ for $1 \leq i \leq k$.*
2. *G' is nonseparable.* $\square$

3 Overview of the Algorithm

The main technique of our algorithm is to use a sequence of reductions, showing that a general class of graphs can be properly labeled *efficiently* if and only if certain progressively simpler class of graphs can be properly labeled with a similar time complexity. The underlying classification of the graphs is given in terms of the bridges it contains and the nonseparability property of certain subgraph; these classes are referred to as TYPE.I, TYPE.II, TYPE.III and TYPE.IV. More concretely, we show that efficient algorithms for graphs of each type can be found, provided that we have good algorithms for graphs of subsequent types. The rôles of each type of graph will be clear in the following sections. In the final section, we present an $O(|E| \cdot |V|^2)$ algorithm to label the edges of a TYPE.IV graph and show that this implies an $O(|E| \cdot |V|^2)$ algorithm for the general graph.

§**3.1** First, we introduce some simplifying assumptions about the graph G under consideration.

Assumption 3.1 G is a finite connected undirected strict graph. $\square$

Note that if G is not connected and the vertices s and t belong to two distinct connected components of G, then every edge $e = [u, v]$ of G, $\ell([u, v]) = \emptyset$. On the other hand, if s and t belong to the same connected component, then we only need to analyze that particular component. For every edge $e = [u, v]$ of all other the labeling is $\ell([u, v]) = \emptyset$.

If G has loops and multi-links then for every loop at v, $\ell([v, v]) = \emptyset$, and for every edge in a multi-link, the labeling is determined by any single edge.

Definition 2.12 AMBITUS. 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. □

See [14] for a linear-time algorithm to compute an ambitus.

Definition 2.13 A BLOCK OF BRIDGES. Let J, P and Q be as before and $\mathcal{B} = B_1, \ldots, B_k$ be the bridges of J in G. A non-empty subset of bridges $\mathbf{B} \subseteq \mathcal{B}$ is called a *block* of bridges if it satisfies the following two conditions:

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

If all the bridges of $\mathbf{B}$ are B^P-, B^Q- or B^{PQ}-bridges, then it is referred to as a *block of B^P-, B^Q- or B^{PQ}-bridges*, respectively. □

Definition 2.14 PROPER AND DEGENERATE BLOCK OF BRIDGES. Let J, P and Q be as before and $\mathbf{B}$ a block of bridges of J in G. We say $\mathbf{B}$ is *proper*, if it contains more than one bridge of J in G, otherwise, it is *degenerate*. □

Notation 2.15 Let G, s, t, $P[s;t]$, $Q[s;t]$ and $J = \{P\} \cup \{Q\}$ be as before. If B is a bridge of the cycle J (and similarly, for $\mathbf{B}$, a block of bridges) with at least one vertex of attachment on $P]s;t[$, then the left- and the right-most vertices of attachment of B on $P[s;t]$ are referred to by $s_P(B)$ and $t_P(B)$ (and, in case of a block of bridges, $\mathbf{B}$, $s_P(\mathbf{B})$ and $t_P(\mathbf{B})$), and the left-most and the right-most vertices of attachment of B on $P]s;t[$ are referred to by $s_P^*(B)$ and $t_P^*(B)$ (and, in case of a block of bridges, $\mathbf{B}$, $s_P^*(\mathbf{B})$ and $t_P^*(\mathbf{B})$).

If, on the other hand, B is a bridge of the cycle J with at least one vertex of attachment on $Q]s;t[$, then $s_Q(B)$, $t_Q(B)$, $s_Q(\mathbf{B})$, $t_Q(\mathbf{B})$, etc. are defined in an identical manner.

If the bridge or the block of bridges under consideration is clear from the context then we simply write s_P, s_Q, s_P^*, s_Q^*, t_P, t_Q, t_P^* and t_Q^*. □

§**2.3** In this subsection we present some technical propositions that will be used quite often later on. Their profs may be found in Even[6], Mishra[12] and Tutte[21].

Proposition 2.1 *Let x be any edge or vertex of G not belonging to the cycle J. Then x belongs to exactly one bridge of J of G that is degenerate or proper.* □

Corollary 2.2 *Let x and y be distinct vertices of attachment of a bridge B of J in G. Then there is a cross-cut of J between x and y.* □

Proposition 2.3 *Let y be a vertex of G belonging to some bridge B of J and x be any vertex $x \in G \setminus B$ such that there is a path from x to y in G. Then there is a vertex z on this path that is also a vertex of attachment of B.* □

Proposition 2.4 *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 $Y_3[z;v]$.* □

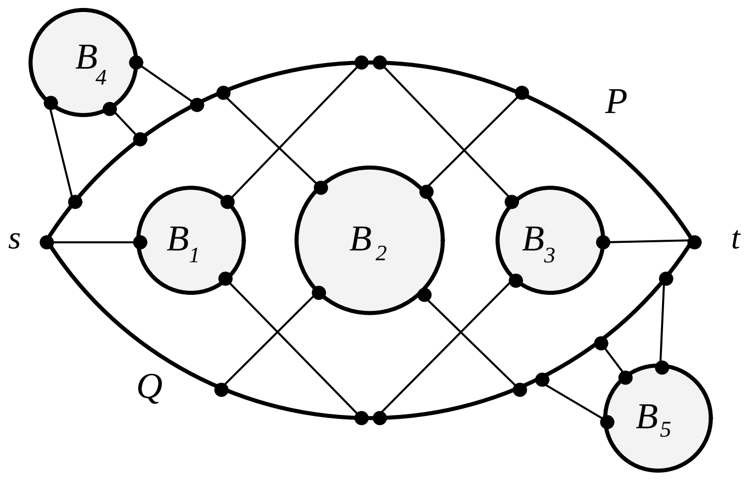


Figure 4: *B^{PQ}-, B^P- and B^Q-bridges of P and Q.*

The vertices, s and t dissect the cycle J into two internally vertex disjoint paths: $P[s;t]$, where $P = J[s;t]$ and its complementary subpath in J, $Q[s;t]$, where $Q^{\mathrm{R}} = J[t;s]$. Clearly P and Q are internally vertex disjoint.

The vertices of P are ordered according to the cyclic order, and vertices of Q, according to the reverse cyclic order. A vertex u of P is said to be *to the left* of a vertex v of P, if u precedes v in the cyclic order of J; and u is *strictly to the left* of v, if, in addition, u and v are distinct. On the other hand, a vertex u of Q is said to be *to the left* of a vertex v of Q, if v precedes u in the cyclic order of J; and u is *strictly to the left* of v, if, in addition, u and v are distinct. The relation '*to the right of*' is the inverse of the relation 'to the left of;' and the relation '*strictly to the right of*' is the relation 'to the right of' with additional irreflexivity property. □

Definition 2.10 BRIDGES WITH RESPECT TO THE PATHS.

Let G be an undirected graph with two distinguished vertices s and t with two internally vertex disjoint paths $P[s;t]$ and $Q[s;t]$, which meet each other only in their end vertices, s and t; $J = \{P\} \cup \{Q\}$ is a cycle in G. We consider three different classes of bridges with respect to J:

- 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 $Q]s;t[$.

If a bridge B of $J = P \cup Q$ in G is not a B^{PQ}-, B^P- or B^Q-bridge then it has only s or t as vertices of attachment. □

Example 2.11 In the figure 4, we show B^{PQ}-, B^P- and B^Q- bridges of the paths 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.

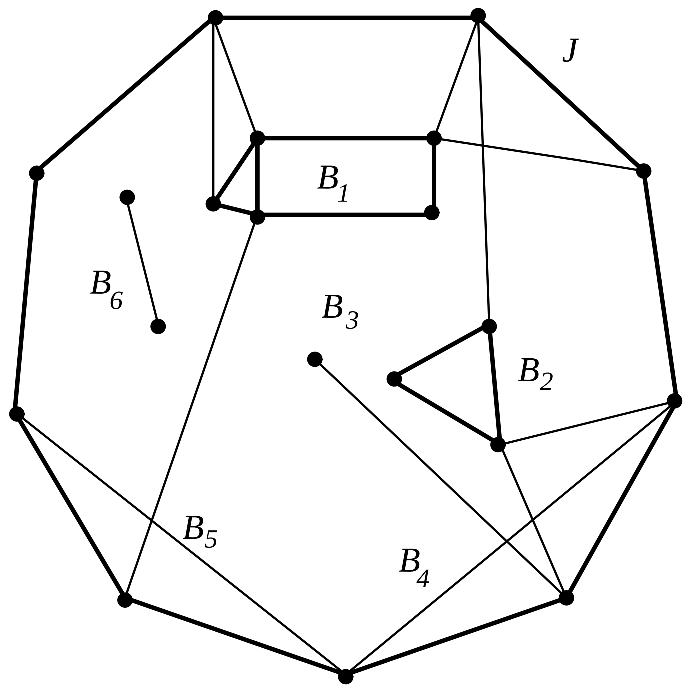


Figure 3: *Bridges of J.*

Definition 2.6 RELATIONS BETWEEN BRIDGES.

Let B_1 and B_2 be two distinct bridges of a cycle 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 cycle J, then we say that they *interlace.* □

Example 2.7 In the Figure 3, the bridges B_1 and B_2 interlace, where as, the bridge B_1 avoids B_4. Also notice that the bridge B_3 avoids every other bridge.

Definition 2.8 CROSS-CUTS.

Let J be a cycle 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 between x and y. □

Definition 2.9 PATHS P AND Q.

Let G be an undirected graph with two distinguished vertices s and t and let J be a cycle of the graph containing the vertices, s and t. Let the vertices of J be ordered in a clockwise cyclic order starting with the vertex s. A subpath $J[a; b] = (a =)u_0, u_1, \ldots, u_{k-1}, u_k(= b)$ denotes the *unique* subpath of J in which u_{i-1} precedes u_i in the clockwise cyclic order (for $1 \leq i \leq k$).

A connected graph is said to have a *separation vertex* v (also called an articulation point) if there exist vertices a and b, $a \neq v$ and $b \neq v$, such that all the paths connecting a and b pass through v. A graph which has a separation vertex is called *separable*, and one which has none is called *nonseparable* (also called biconnected). The maximal nonseparable subgraphs of G are its *nonseparable components* (also called biconnected components). A connected graph is said to be a *chain graph* if each of its nonseparable component contains no more than *two* separation vertices.

§**2.2** Now, we introduces the notion of bridges for cycles in general graphs. The term 'bridge' is taken from Tutte [21], which also contains a rather complete survey of bridge theory in both general and planar graphs. The equivalent terms for bridge, in some older literature, are 'component *mod* J', 'J-component' [20] and 'Gespinst'[2] [17].

Definition 2.1 BRIDGES.[Tutte]

Let J be a fixed subgraph of G. A subgraph G_1 of G is said to be J*-detached* in G, if all its vertices of attachment are in J. We define a *bridge* of J in G as any subgraph B that satisfies the following three conditions:

- B is not a subgraph of J.
- B is J-detached in G.
- No proper subgraph of B satisfies both (1) and (2).

The set of vertices of attachment of a bridge B of a subgraph J in G is denoted by $W(G, B) = \{v_0, v_1, \dots, v_{k-1}\}$. $\square$

Definition 2.2 DEGENERATE AND PROPER BRIDGES. NUCLEUS OF A BRIDGE.

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. $\square$

Remark 2.3 By a Theorem due to Tutte, if B is any bridge of a subgraph J of G then B is either degenerate or proper. Hence using the above definition of a bridge and the theorem, we can give a linear-time algorithm to find all bridges of a subgraph J of G. $\square$

Example 2.4 In the Figure 3, we give an example of bridges of a cycle J. In this example, bridges B_1, B_2, B_3 and B_6 are proper bridges, and B_4 and B_5 are degenerate bridges.

Definition 2.5 RESIDUAL PATHS.

Let the vertices of attachment of a bridge B of a cycle J in G, be $W(G, B) = \{v_0, v_1, \dots, v_{k-1}\}$ and let $v_0, v_1, \dots, v_{k-1}$ be their enumeration in their cyclic order on J. The vertices of attachment dissect J into k subpaths $L_0, L_1, \dots, L_{k-1}$ such that $L_j = J[v_j; v_{j+1(\bmod k)}]$. These subpaths are called the *residual paths* of B in J. $\square$

[2]Gespinst is the German word for 'cobweb'.

§2.1 Graph Theoretic Terminology

A *graph* $G = (V, E)$ is a finite set V of *vertices* and a set E of pairs of vertices, called *edges*. Either the edges are ordered pairs $\langle u, v\rangle$ of distinct vertices (the graph is *directed*) or the edges are unordered pairs $[u, v]$ of distinct vertices (the graph is *undirected*). If $[u, v]$ is an undirected edge, u and v are *adjacent*. If $\langle u, v\rangle$ is a directed edge, u is a *predecessor* of v (respectively, v is a *successor* of u), sometimes denoted by, $u \rightarrow v$ (respectively, $v \leftarrow u$). We call u and v, the *ends* of the edge.

A graph $G_1 = (V_1, E_1)$ is a *subgraph* of G, if $V_1 \subseteq V(G)$, $E_1 \subseteq E(G)$, and each edge of G_1 has the same ends in G_1 as in G. If G_1 is a subgraph of G, other than G itself, then G_1 is a *proper* subgraph of G. If $V_1 = V(G)$ then G_1 is said to be a *spanning subgraph* of G. A *vertex of attachment* of G_1 in G is a vertex of G_1 that is incident in G with some edge not belonging to G_1.

If $E_1 \subseteq E(G)$, let $V(E_1)$ be the set of all vertices v of G such that v is incident with a member of E_1, *i.e.*,

$$V(E_1) = \Big\{v \in V(G) : \Big(\exists\, u \in V(G)\Big)\; [u, v] \in E_1\Big\}.$$

Then the subgraph $\langle E_1\rangle = (V(E_1), E_1)$ is the *reduction* of G to E_1. Similarly, if $V_1 \subseteq V(G)$, let $E(V_1)$ be the set of all edges of G having both ends in V_1,*i.e.*,

$$E(V_1) = \Big\{[u, v] \in E(G) : u, v \in V_1\Big\}.$$

Then the subgraph $\langle V_1\rangle = (V_1, E(V_1))$ is the subgraph of G *induced* by V_1.

An undirected graph is *connected* if there is a path connecting every pair of vertices and *disconnected* otherwise. The maximal connected subgraphs of G are its *connected components* or simply, *components*.

A *path* of *length* k from u to v in G is a sequence of vertices $(u =)\ u_0, u_1, \ldots, u_k\ (= v)$ such that $\langle u_i, u_{i+1}\rangle \in E$ for $0 \le i < k$. (Sometimes denoted by $u \overset{*}{\longrightarrow} v$.) The path *contains* the edges $\langle u_i, u_{i+1}\rangle$ for $0 \le i < k$ as well as vertices u_i for $0 \le i \le k$. The vertices u and v are called the *ends* of the path P. All other vertices of the path (*i.e.*, u_i's for $0 < i < k$) are the *internal vertices* of the path.

If $0 \le i \le j \le k$, then the sequence of vertices, $u_i, u_{i+1}, \ldots, u_j$ is a *subpath* of the path from u to v. If P is a path from u to v, $u = u_0, u_1, \ldots, u_k = v$, and $0 \le i \le j \le k$ then the subpath from u_i to u_j, including both u_i and u_j is represented by $P[u_i; u_j]$; the subpath excluding u_i but including u_j, by $P]u_i; u_j]$; the subpath including u_i but excluding u_j, by $P[u_i; u_j[$ and the subpath excluding both u_i and u_j, by $P]u_i; u_j[$. If $P = u_0, u_1, \ldots, u_{k-1}, u_k$ is a path from u_0 to u_k, then the *reversal* of the path P is $P^{\mathrm{R}} = u_k, u_{k-1}, \ldots, u_1, u_0$. If $P_1 = u_0, u_1, \ldots, u_i$ and $P_2 = u_i, u_{i+1}, \ldots, u_k$ are two paths then the *concatenation* of P_1 and P_2 is $P_1 * P_2 = u_0, u_1, \ldots, u_i, u_{i+1}, \ldots, u_k$.

The path is *simple* if $u_0, \ldots, u_k$ are distinct (except possibly $u_0 = u_k$) and the path is a *cycle* if $u_0 = u_k$. By convention there is a path of no edges from every vertex to itself (*null path*), but a cycle must contain at least two edges. Two simple paths P_1 and P_2 are said to be *vertex disjoint*, if the vertices of P_1 and P_2 are mutually distinct; *internally vertex disjoint*, if the internal vertices of P_1 and P_2 are mutually distinct.

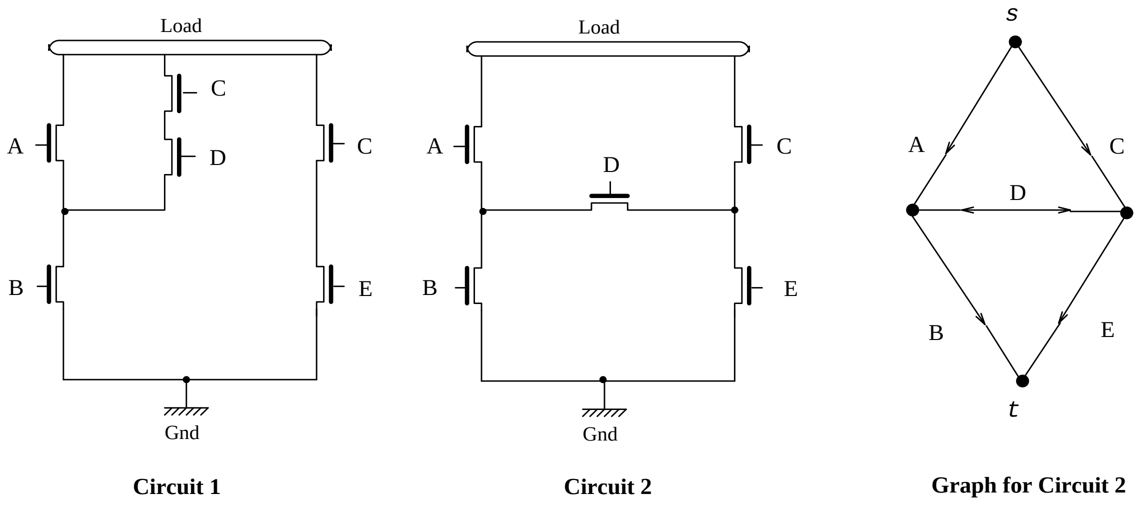


Figure 2: Example of a Sneak Path.

$B) \vee (B \wedge C \wedge D) \vee (C \wedge E))$ and the resulting circuit after common subcircuit elimination. However, because of the sneak path through D, it implements the Boolean formula $\neg((A \wedge B) \vee (B \wedge C \wedge D) \vee (C \wedge E) \vee (A \wedge D \wedge E))$, instead. Also in figure 2, we present the corresponding graph in which D is bidirectional, thus showing the existence of a sneak path. Our algorithm can be used to detect all the sneak paths.

For a discussion of other applications and the history of the problem, see the followings: Frank[7] posed the original problem in the context of switch level simulation; Barzilai, Breece, Huisman, Iyengar and Silberman [1] used the directionality information to speed up their simulation; Jouppi[9] used the transistor direction in determining critical paths in the circuit, which can then be used for timing simulation and design rule checking; Chen, Mathews and Newkirk[3] used the directionality information for generating the test sets for MOS circuits at the switch level. Additional applications may also be found in Brand[2], Lee, Gupta and Breuer[10], and Cirit[4].

The paper is organized as follows: Section 2 introduces the various graph theoretic terminology and concepts necessary for this process. Section 3 presents an overview of the algorithm and defines a classification of graphs, suggested by the reduction processes used by the algorithm. Sections 4, 5, 6 and 7 discuss the reduction processes and their complexity analysis in details. Section 8 concludes the paper with the timing analysis of the complete algorithm.

2 Preliminaries

Here, we define some graph theoretic terminology and introduce other key concepts required in the paper. Most important among these are the terms: *bridge*, *residual path*, *cross-cut* and *ambitus*. The definitions are similar to those used in the context of Tutte's Theorem on Hamiltonian circuits in 4-connected planar graphs given in Tutte [21], those in the planarity-testing algorithm of Hopcroft and Tarjan [8] or those in connection with the four-color problem as presented in Ore [16].

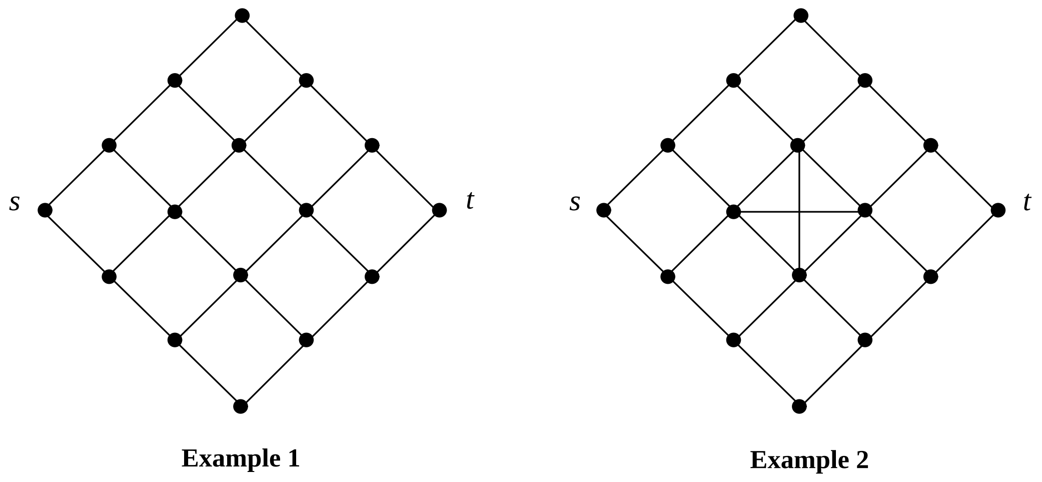


Figure 1: Two example graphs.

From the proposition above it is easy to see that we can label each edge $[u, v] \in E$ by asking the following two questions: 'Is UVP2(s, u; v, t; G) true?' and 'is UVP2(s, v; u, t; G) true?'

There are polynomial-time algorithms to find two vertex disjoint paths in an undirected graph. (Cf. Ohtsuki[15], Seymour[18] and Shiloach[19]; also see Mishra and Tarjan[14] in relation to Ohtsuki's algorithm.) The most efficient algorithms for this problem have a time complexity $O(|E| \cdot |V|)$. (Note, both Ohtsuki's and Shiloach's Algorithms have this complexity.) The naïve way of solving the bidirectional edges problem is to invoke an algorithm for two vertex disjoint paths problem twice per each edge. Altogether this takes $O(|E|^2 \cdot |V|)$.

Moreover, if we have an algorithm that correctly 'labels' the edges of a graph $G = (E, V)$ in time $O(T(|E|, |V|)) \geq O(|E| + |V|)$ then it is easy to see that this provides an $O(T(|E|, |V|))$ algorithm for two vertex disjoint paths problem, as well. The reasoning is as follows: Given two pairs of vertices $\{s_1, t_1\}$ and $\{s_2, t_2\}$ in a graph G, we modify the graph G by adding a new edge $[t_1, s_2]$. Let the modified graph be called G'. Now, using proposition 1.1, it is easily seen that UVP2$(s_1, t_1; s_2, t_2; G)$ holds if and only if $\langle t_1, s_2 \rangle \in \ell([t_1, s_2])$ in G'. Thus, an algorithm for bidirectional edges problem with a running time of $O(|E| \cdot |V|)$ is probably the best that one can hope for.

In this paper and its sequel[13], we devise an $O(|E| \cdot |V|)$ time algorithm for bidirectional edges problem; the algorithm makes a novel use of *bridges* of a circuit in a *general* graph. We begin this paper with a simple set of reduction processes which suggest an $O(|E| \cdot |V|^2)$ algorithm; subsequently, in the sequel[13], we introduce some additional machinery that further reduces the complexity to $O(|E| \cdot |V|)$. The algorithms described here first appeared in [11] and [12].

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[2].) 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.

For instance, in the Figure 2, we show a circuit implementing the Boolean formula $\neg((A \wedge$

1 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 labeled $\langle u, v\rangle$. The "bidirectional edges problem" is to find all the 'bidirectional' and 'unidirectional' edges of G, together with the labelings of the 'unidirectional' edges.

The notions of 'unidirectional' and 'bidirectional' edges can be formalized in terms of the *labeling function*, ℓ, that maps each undirected edge $[u, v]$ to a subset of $\{\langle u, v\rangle, \langle v, u\rangle\}$.

Definition 1.1 The *edge-labeling function*, ℓ, is defined as follows:

$$\ell([u,v]) \ni \begin{cases} \langle u,v\rangle, & \text{iff there is a simple path} \\ & \qquad (s =)\ w_0, \ldots, w_i, w_{i+1}, \ldots, w_n\ (= t) \\ & \qquad \text{such that } w_i = u \text{ and } w_{i+1} = v; \\ \langle v,u\rangle, & \text{iff there is a simple path} \\ & \qquad (s =)\ w_0, \ldots, w_i, w_{i+1}, \ldots, w_n\ (= t) \\ & \qquad \text{such that } w_i = v \text{ and } w_{i+1} = u. \end{cases}$$

Clearly, an edge $e = [u, v]$ is bidirectional, if $\ell([u, v]) = \{\langle u, v\rangle, \langle v, u\rangle\}$; and unidirectional, if $\ell([u, v]) = \{\langle u, v\rangle\}$ or $\{\langle v, u\rangle\}$. □

See figure 1 and 16 for two example graphs and their labeling, respectively. The relation between bidirectional edges problem and the classical two vertex disjoint paths problem is elucidated in proposition 1.1.

Definition 1.2 TWO-VERTEX-DISJOINT PATHS IN AN UNDIRECTED GRAPH: UVP2. Let G be an undirected graph with two pairs of distinct vertices $\{s_1, t_1\}$ and $\{s_2, t_2\}$. The predicate UVP2(s_1, t_1; s_2, t_2; G) is true if and only if there are two vertex-disjoint paths, $P_1 = s_1, \ldots, t_1$ and $P_2 = s_2, \ldots, t_2$ in the undirected graph G. □

Proposition 1.1 *Let G be a finite undirected graph with two distinguished vertices, the source, s and the sink, t. Let $e \in E(G)$ be any edge of G. If ℓ is the edge labeling function as defined earlier,*

$$\langle u,v\rangle \in \ell([u,v]) \quad \Leftrightarrow \quad \text{UVP2}(s,u;v,t;G).$$

PROOF.

$$\begin{aligned} &\langle u,v\rangle \in \ell([u,v]) \\ &\quad \Leftrightarrow \quad \Big(\exists \ \textit{a simple path } (s =)w_0, \ldots, w_i, w_{i+1}, \ldots, w_n(= t)\Big) \Big[w_i = u \textit{ and } w_{i+1} = v\Big]. \\ &\quad \Leftrightarrow \quad \Big(\exists \ \textit{two vertex disjoint paths} \\ &\qquad\qquad P_1 = (s =)w_0, \ldots, w_i(= u) \textit{ and } P_2 = (v =)w_{i+1}, \ldots, w_n(= t) \textit{ in } G \\ &\qquad\qquad \textit{and an edge } [u,v] \in G\Big). \\ &\quad \Leftrightarrow \quad \text{UVP2}(s,u;v,t;G). \quad \square \end{aligned}$$

Contents

Abstract

The "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. We provide several algorithms for this problem in the current paper and the accompanying sequel. In this paper, we show the relation between this problem and the classical two-vertex-disjoint-paths problem and then devise a simple algorithm with a time complexity of $O(|E| \cdot |V|^2)$. In the sequel, we improve the time complexity to $O(|E| \cdot |V|)$. The main technique exploits a clever partition of the graph into a set of paths and bridges which are then analyzed recursively.*

Bidirectional edges 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, sometimes it is intended to propagate information in one direction only, and propagation in the wrong direction (resulting in a sneak path*) can cause functional error. Our algorithms can be used to detect all the sneak paths.*

Key words.
Bridge, Cross-cut, Disjoint Paths, MOS Circuit, Pass Transistor, Sneak Path, complexity

Bidirectional Edges Problem: Part I

A Simple Algorithm

B. Mishra[1]

Department of Computer Science
Courant Institute of Mathematical Sciences
New York University
719 Broadway
New York, NY 10003

[1]Supported in parts by National Science Foundation Grants DMS-8703458 and CCR-9002819.