アルゴリズム 1－5
(1988. 5. 23)

平面グラフでスタイナー林を求めるアルゴリズム

鈴木均　赤間長浩　西関隆夫

東北大学 工学部 通信工学科

VLSIのチップ上の一層配線等の配線問題は，平面(格子)グラフでスタイナー林を求める問題に帰着される．無向グラフGとG上にネットの集合が与えられたときに，各々がG上で一つのネットの全ての端子を結び互いに交わらない(点素な)木の集合をスタイナー林と呼ぶ．本報告では，Gが平面グラフでありその二つの面の周上に全てのネットがある場合について，スタイナー林を求める効率のよいアルゴリズムを与える．アルゴリズムの計算時間は$O(n\log n)$である．ここでnはグラフGの点数である．

Algorithms for Finding Steiner Forests in Planar Graphs

Hitoshi SUZUKI, Takehiro AKAMA, and Takao Nishizeki

Department of Electrical Communications, Faculty of Engineering
Tohoku University, Sendai 980, Japan

Several routing problems such as VLSI river routing and single-layer routing can be formulated as a problem for finding a Steiner forest in a planar (grid) graph. Given an unweighted planar graph G together with nets of terminals, our problem is to find a Steiner forest, i.e., vertex-disjoint trees, each of which interconnects all the terminals of a net with each other. This paper gives an efficient algorithm to solve the problem for the case all terminals lie on two face boundaries of G. The algorithm runs in $O(n\log n)$ time if G has n vertices.

1. Introduction

Given an unweighted planar graph G together with nets of terminals, our problem is to find a Steiner forest, that is, vertex-disjoint trees, each of which interconnects all the terminals of a net with each other. Since the "disjoint path problem" is NP-hard even for planar graphs [Lyn] or plane grids [KL, Ric], so is our problem if there is no restriction on the location of terminals. This paper presents a very efficient algorithm to solve the problem for the case all the terminals lie only on two face boundaries B_1 and B_2 of a planar graph G. Fig. 1 depicts a problem instance and a solution, where all the terminals of ten nets lie on the outer face boundary B_1 and an inner face boundary B_2, and a found Steiner forest of ten disjoint trees is drawn in thick lines.

Our algorithm runs in $O(n \log n)$ time or more precisely in $O(\mathrm{MIN}\{(k_{12}+1)n, n\log n\})$ time, where n is the number of vertices of G and k_{12} is the number of nets having both a terminal on B_1 and a terminal on B_2. In the example above $k_{12} = 3$. Our algorithm runs in $O(n)$ time especially when all the terminals lie on a single face boundary or $k_{12} = O(1)$.

Robertson and Seymour [RS] have studied the problem from a graph theoretic standpoint of view in an extended series of papers on the topic of "graph minors," and proved that the problem can be solved in polynomial time. The proof yields a polynomial-time algorithm, but the straightforward implementation requires $O(n^3)$ time.

On the other hand Baker and Pinter [BP] have studied a similar problem from a standpoint of VLSI river-routing, and given an algorithm for finding a Steiner forest in a plane grid. However there are several restrictions: a grid must have a rectangular outer boundary and exactly one nontrivial hole; and every net must consist of exactly two terminals, one on the outer boundary and the other on the boundary of the hole. The algorithm finds a Steiner forest, that is, vertex-disjoint paths in this case, in $O(k^2 + m^2)$ time where k is the number of nets and m is the number of segments on the boundary of the nontrivial hole.

Our algorithm is much faster than Robertson and Seymour's, and can find a Steiner forest in general planar graphs unlike Baker and Pinter's. Thus it yields a more practical and flexible routing algorithm applicable even to the case wires of 45° or unroutable barriers are admitted in grids. In order to improve the time complexity, we need several new ideas and careful treatment of planar graphs.

2. Preliminaries

Let $G = (V, E)$ be an undirected planar graph with vertex set V and edge set E. We sometimes write $V = V(G)$ and $E = E(G)$. Let n be the number of vertices in G, that is, $n = |V|$. Throughout the paper we assume that G is connected and embedded in the plane R^2. The image of G on R^2 is denoted by $Image(G)$. A *face* f of planar graph G is a connected component of $R^2 - Image(G)$. Denote by $V(f)$ the set of vertices on the boundary of f, and denote by $E(f)$ the set of edges on the boundary. For two graphs G and G', $G + G'$ means a graph $(V(G) \cup V(G'), E(G) \cup E(G'))$. $G - V'$ means a graph obtained from G by deleting all vertices in $V' \subset V$, while $G - E'$ means a graph obtained from G by deleting all edges in $E' \subset E$. Let f_1 be the outer face of G and let f_2 be any inner face of G. The boundary of face f_l is denoted by B_l. Throughout this paper $l = 1$ or 2, and we assume that a set of vertices in $V(f_1) \cup V(f_2)$ are designated as *terminals*. A *net* N is a set of terminals that are all to be interconnected with each other. A *net set* $S = \{N_1, N_2, ..., N_k\}$ is a partition of the set of terminals. Then a *network* $\mathcal{N} = (G, S)$ is a pair of a planar graph G and a net set S. A *Steiner forest of network* $\mathcal{N}$ is a forest $F = T_1 + T_2 + ... + T_k$ in G such that $N_i \subset V(T_i)$ for each tree T_i in F. For simplicity we often call F a forest of $\mathcal{N}$, and write $T_i \in F$, and say that tree T_i *spans* net N_i.

The net set S is partitioned into three subsets S_1, S_2 and S_{12} so that

(1) $N \in S_1 \Rightarrow N \subset V(f_1)$,

(2) $N \in S_2 \Rightarrow N \subset V(f_2)$, and

(3) $N \in S_{12} \Rightarrow N \cap V(f_1) \neq \phi$ and $N \cap V(f_2) \neq \phi$.

In Fig. 1 $S_1 = \{N_2, N_5, N_6, N_7, N_9\}$, $S_2 = \{N_3, N_{10}\}$, and $S_{12} = \{N_1, N_4, N_8\}$. Let $k_{12} = |S_{12}|$. We consider the following three cases (i), (ii) and (iii) separately:

(i) all the terminals lie on a single face boundary, that is, either $S = S_1$ or $S = S_2$;

(ii) there is no net intersecting with $V(f_1)$ and $V(f_2)$, that is, $S_{12} = \phi$; and

(iii) there is a net intersecting with $V(f_1)$ and $V(f_2)$, that is, $S_{12} \neq \phi$.

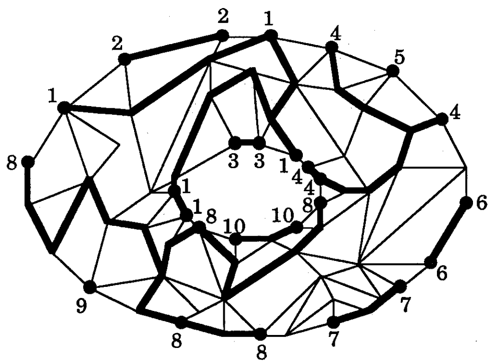

Fig. 1 A planar graph G and a forest F.

In the succeeding three sections we present algorithms for these cases: an $O(n)$ algorithm FOREST1 for case (i) in Section 3, an $O(n)$ algorithm FOREST2 for case (ii) in Section 4, and an $O(\mathrm{MIN}\{k_{12}n, n\log n\})$ algorithm FOREST3 for case (iii) in Section 5. These algorithms necessarily find a Steiner forest whenever there exists. It is easy to modify them so that they can also check the existence of a Steiner forest. Hereafter we assume that there exists a Steiner forest in a given network $\mathcal{N}$.

3. Case in which all the terminals lie on a single face boundary

This section deals with the easiest case in which all the terminals of a network $\mathcal{N} = (G, S)$ lie on a single boundary. We may assume that $S = S_1$. The outer face boundary B_1 is not necessarily a simple cycle, but is a closed walk. If such a network $\mathcal{N}$ has a Steiner forest, then there exist no two nets $N, N' \in S$ such that $t_1, t_3 \in N$, $t_2, t_4 \in N'$, and the four terminals t_1, t_2, t_3 and t_4 appear clockwise on B_1 in this order, and consequently B_1 has a subwalk W_1 which contains all the terminals of a single net, say $N_1 \in S$, but does not contain any other terminals. (An example of such walks for the network in Fig. 2(a) is a single edge joining the two terminals in net 1.) Clearly such a walk W_1 includes a tree T_1 spanning N_1. A new network $\mathcal{N}' = (G - V(W_1), S - N_1)$ has a Steiner forest F'; otherwise, $\mathcal{N}$ would not have a Steiner forest. (A similar argument was used by a classic flow algorithm, called the uppermost path algorithm, for finding a maximum flow in a planar graph with the source and sink both on the outer face boundary [FF].) Clearly $F = T_1 + F'$ is a Steiner forest of $\mathcal{N}$. These observations immediately yield an iterative algorithm to find a Steiner forest of $\mathcal{N}$ in $O(n^2)$ time. Note that W_1 and N_1 above can be found in $O(n)$ time.

The main result of this section is to improve the complexity to $O(n)$. We define some more terms before presenting a refined algorithm. Let $J \subset R^2$ be a simple closed curve J passing through all vertices in $V(f_1)$ such that the closure of J's inside includes $Image(G)$. In Fig. 2(a) J is drawn in dotted lines. Let v_0 be a vertex on B_1, then *the starting terminal* $s(N)$ *of a net* $N \in S$ *(with respect to* v_0*)* is the terminal of N appearing first on J clockwise going from v_0, while *the ending terminal* $t(N)$ is the terminal appearing last. The starting and ending terminals of nets are all represented as double circles in Fig. 2. We assume that a planar graph G is represented by embedding lists: a set of adjacency lists $\{L(v) | v \in V\}$; all edges incident to v appear in $L(v)$ in clockwise order with respect to a plane embedding of G. Especially if $v \in V(f_1)$, then edges in $L(v)$ are ordered based on J. That is, if $v' \in V(f_1)$ is the vertex clockwise next to v on J, then the edge embedded around v clockwise next to the v-v' segment of J first appears in list $L(v)$. We denote by $W(N_i)$ the walk which goes clockwise on B_1 from $s(N_i)$ to $t(N_i)$, starting with the first edge in list $L(s(N_i))$ and ending with the last edge in $L(t(N_i))$.

Our idea is very simple: let v_0 be an arbitrary vertex on B_1, and number the nets N_1, N_2, ..., N_k in S so that the ending terminals $t(N_1)$, $t(N_2)$, ..., $t(N_k)$ appear in this order on J clockwise going from v_0, as depicted in Fig. 2(a). Then walk $W(N_1)$ satisfies the desired property as W_1 for $\mathcal{N}$, and $W(N_2)$ does for network $(G - V(W(N_1)), S - N_1)$, and so on. This fact can be proved by an easy induction. Therefore the following procedure FOREST1($\mathcal{N}$,v_0) finds a Steiner forest $F = T_1 + T_2 + ... + T_k$ of $\mathcal{N}$.

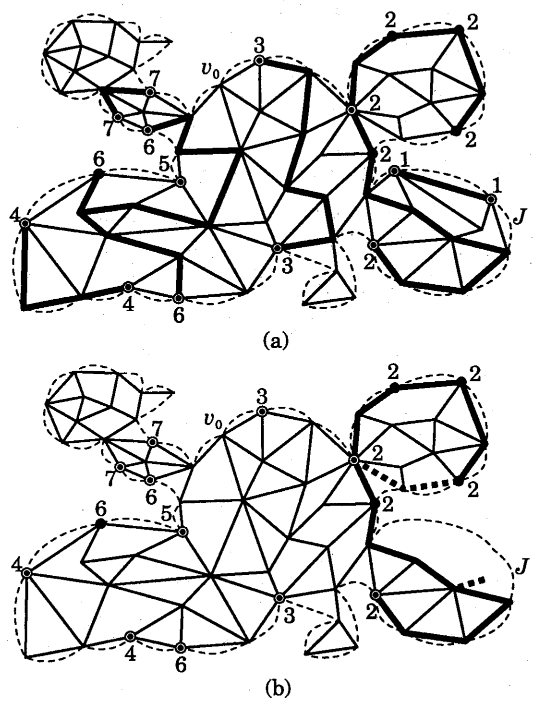

Fig. 2(a) Forest F of network $\mathcal{N} = (G, S)$, and (b) network $(G - V(W(N_1)), S - N_1)$.

procedure FOREST1($\mathcal{N}$,v_0);
begin
number the nets $N_1, N_2, ..., N_k$ in S with respect to v_0 as above;
for $i := 1$ to k **do**
begin
let $W(N_i)$ be the walk on the outer boundary B_1 of G going clockwise from $s(N_i)$ to $t(N_i)$;
$G := G - W(N_i)$;
delete some redundant edges and vertices from $W(N_i)$ to obtain a tree T_i spanning N_i
end
end;

Fig. 2(b) depicts a graph obtained from graph G in Fig. 2(a) by deleting all vertices in $W(N_1)$. The walk $W(N_2)$ on a new graph is drawn in thick solid and dotted lines. Thus $W(N_2)$ is not necessarily a tree and may have a non-terminal vertex of degree one. Tree T_2, drawn in thick solid lines in Fig. 2(b), is obtained from $W(N_2)$ by deleting three edges drawn in thick dotted lines.

Clearly the desired numbering of nets can be found in $O(n)$ time. Since a planar graph G is represented by embedding lists, one can execute the **for** loop in time proportional to the number of deleted edges. Since a planar graph G has $O(n)$ edges, the **for** loop spends $O(n)$ time in total. Therefore procedure FOREST1 runs in $O(n)$ time in total. Thus we have the following theorem.

Theorem 1. A Steiner forest of network $\mathcal{N} = (G, S)$ can be found in $O(n)$ time for the case all the terminals lie on a single face boundary of a planar graph G. ■

4. Case in which there is no net intersecting with $V(f_1)$ and $V(f_2)$

This section deals with the most difficult case in which $S_{12} = \phi$. Our algorithm FOREST2 for this case is completely different from one suggested in [RS]. In subsection 4.1 we define some more terms. Then in subsection 4.2 we present a sequence of lemmas on which FOREST2 is based. Finally in subsection 4.3 we present FOREST2, and show that it finds a Steiner forest in $O(n)$ time.

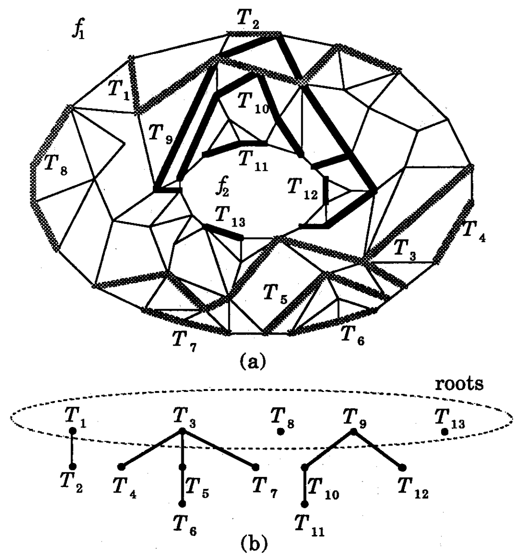

Fig. 3(a) $F = T_1 + T_2 + ... + T_{13}$, and (b) the genealogy.

4.1 Definitions. Since $S_{12} = \phi$, $S = S_1 \cup S_2$. Let $\mathcal{N}_1 = (G, S_1)$ and $\mathcal{N}_2 = (G, S_2)$. Let F_1 be a Steiner forest of $\mathcal{N}_1$, and F_2 a Steiner forest of $\mathcal{N}_2$. F_1 is drawn in gray thick lines and F_2 in black thick lines in Fig. 3(a). Note that F_1 and F_2 can be found by procedure FOREST1 in Section 3. Let $F = F_1 + F_2$. Since F_1 may cross with F_2 as in Fig. 3(a), F is not necessarily a Steiner forest of $\mathcal{N}$. So we must modify F_1 and F_2 in order to obtain a Steiner forest of $\mathcal{N}$.

We define some more terms. Let $N \in S_l$ be a net of m terminals, and let T be a tree in F_l spanning N, where $l = 1$ or 2. Assume that the terminals $t_1, t_2, ..., t_m$ in N appears in this order clockwise on the boundary B_l of face f_l, as in Fig. 4. For a terminal pair $p_i = (t_i, t_{i+1})$, let Q_i be a walk on B_l clockwise going from t_i to t_{i+1}, and let P_i be a path on T connecting t_i and t_{i+1}, where $1 \leq i \leq k$ and $t_{m+1} = t_1$. We denote by $in(T, p_i)$ the inside of a cycle $Q_i + P_i$ (including the boundary). Exactly one of the m insides $in(T, p_i)$, $1 \leq i \leq m$, includes face f_2. Without loss of generality we may assume $f_2 \subset in(T, p_m)$. Then the path P_m, drawn in thick lines in Fig. 4, is called the *trunk of tree T* and denoted by $trunk(T)$. The *inside* $in(T)$ *of tree T*, hatched in Fig. 4, is $\bigcup_{i=1}^{m-1} in(T, p_i)$. The *inside* $in(F_l)$ *of forest* F_l is $\bigcup_{T \in F_l} in(T)$. We call t_1 the *starting terminal* $s(T)$ *of tree T*, and t_m the *ending terminal* $t(T)$ *of tree T*. The ordered pair $p_m = (t_m, t_1)$ is called the *outer (terminal) pair* of T, and denoted by $p(T)$, while the ordered pairs $p_i = (t_i, t_{i+1}), 1 \leq i \leq m-1$, are called *inner (terminal) pairs*. The set of all outer terminal pairs of trees in F_l is denoted by $Z(F_l)$: $Z(F_l) = \{p(T) | T \in F_l\}$. Let $u_1, u_2, ..., u_r$ be the vertices on $trunk(T)$ which are either terminals or have degree three or more in T, and assume that these vertices appear in this order on $trunk(T)$ going from $s(T)$ to $t(T)$. Note that $r \leq m$, $u_1 = s(T)$ and $u_r = t(T)$. We call a subpath of $trunk(T)$ between u_j and u_{j+1} an *interval of* $trunk(T)$, where $1 \leq j \leq r-1$. Each interval U of $trunk(T)$ lies on one of paths $P_1, P_2, ..., P_{m-1}$. If U lies on P_i, then p_i is called an *inner (terminal) pair* $p(T, U)$ *of interval* U. If T and T' are two distinct trees in a forest F_l and $in(T') \subset in(T)$, then T is an *ancestor* of T' and T' is a *descendant* of T (See Fig. 3(b)). We denote the set of ancestors of T by $ANC(F_l, T)$, and denote the set of descendants of T by $DES(F_l, T)$. A *son* of T is a descendant of T whose inside is maximal. We denote by $SON(F_l, T)$ the set of sons of T. A tree $T' \in DES(F_l, T)$ is a *descendant of an inner pair* p_i (or *an interval* U) if $in(T') \subset in(T, p_i)$. The set of descendants of p_i is denoted by $DES(F_l, T, p_i)$. A *root* of forest F_l is a tree $T \in F_l$ which has no ancestors. The set of roots of F_l is denoted by $root(F_l)$. We define the *roots of* $F = F_1 + F_2$ by $root(F) = root(F_1) \cup root(F_2)$. Denote by $Z(F)$ the set of outer terminal pairs of F: $Z(F) = Z(F_1) \cup Z(F_2)$.

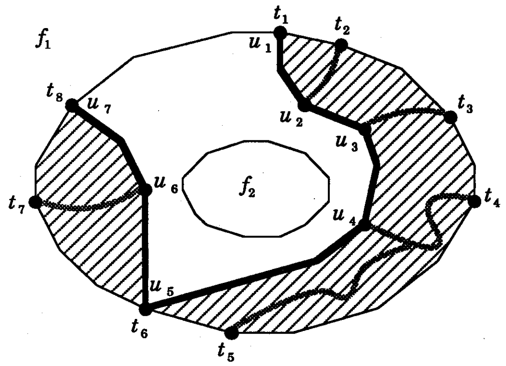

Fig. 4 Tree T, $trunk(T)$, and $in(T)$.

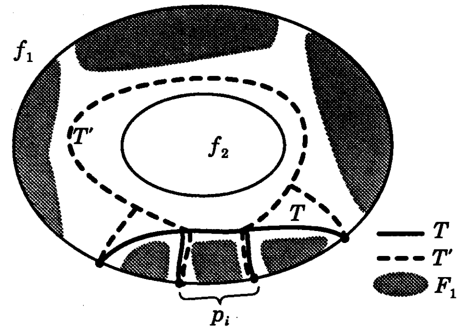

Fig. 5 Tree T, and tree T' obtained from T by opening p_i.

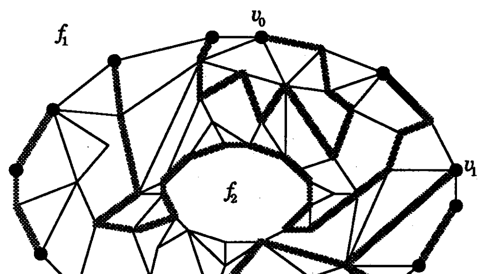

Fig. 6 Non-tight forest F_l' of $\mathcal{N}_1$ obtained by FOREST1($\mathcal{N}_1, v_0$).

4.2 Lemmas. Clearly the following lemma holds.

Lemma 1. Let F_l and F_l' be two distinct forests of network $\mathcal{N}_l = (G, S_l)$, where $l = 1$ or 2. Let $T \in F_l$ correspond to $T' \in F_l'$, and let $T_d \in F_l - T$ correspond to $T_d' \in F_l' - T'$. Let p_i be an inner pair of T. Then (a)-(c) below hold.
(a) If p_i is also an inner pair of T' and $T_d \in DES(F_l, T, p_i)$, then $T_d' \in DES(F_l', T', p_i)$ and $p(T_d) = p(T_d')$.
(b) If p_i is the outer pair of T' and $T_d \in F_l - T - DES(F_l, T, p_i) - ANC(F_l, T)$, then $T_d' \in DES(F_l', T')$ and $p(T_d) = p(T_d')$ (see Fig. 5).
(c) If $T \in root(F_l)$, $T' \in root(F_l')$ and $p(T) = p(T')$, then $Z(F_l) = Z(F_l')$. ■

We say that a tree T *intersects with a face* f if $V(f) \cap V(T) = \phi$. Informally a forest F_l of $\mathcal{N}_l$ is tight if it is compacted as close to B_l as possible. Formally F_l is *tight* if every tree $T \in F_l$ satisfies the following condition.

Condition A. For each edge e on $trunk(T)$, either e lies on the walk on B_l clockwise going from $s(T)$ to $t(T)$, or a son of T intersects with the face (in $in(T)$) adjoining e. ■

As shown later in Corollary 1, if F_l is tight, then the inside $in(F_l)$ is minimal among all forests of $\mathcal{N}_l$ having the same set of outer pairs. $F = F_1 + F_2$ is *tight* if F_1 and F_2 are tight forests of $\mathcal{N}_1 = (G, S_1)$ and $\mathcal{N}_2 = (G, S_2)$, respectively. F in Fig. 3(a) is tight. One can easily observe the following lemma.

Lemma 2. If network $\mathcal{N}$ has a forest F, then $\mathcal{N}$ has a tight forest F' with $Z(F) = Z(F')$. ■

One can find a tight forest of $\mathcal{N}_l$ by executing FOREST1 twice, as follows. First choose an arbitrary vertex v_0 on B_l, and execute FOREST1($\mathcal{N}_l, v_0$) to find a forest F_l' of $\mathcal{N}_l$. Here F_l' is not necessarily tight, as depicted in Fig. 6. Then choose the starting terminal v_1 of an arbitrary root of F_l', and execute FOREST1($\mathcal{N}_l, v_1$) to find a Steiner forest F_l of $\mathcal{N}_l$. One can easily observe that F_l is necessarily tight, because every tree in F_l is constructed after all the descendants are constructed. The tight forest F_1 obtained in this way is drawn by thick lines in Fig. 3(a). Thus we have:

Lemma 3. A tight Steiner forest F_l of network $\mathcal{N}_l$ can be found in $O(n)$ time. ■

Using Lemma 1, one can prove the following lemma.

Lemma 4. Let F_l be a tight forest of $\mathcal{N}_l$, and let F_l' be an arbitrary forest of $\mathcal{N}_l$. Let $T \in F_l$ correspond to $T' \in F_l'$, and let U be an interval of T. If $p_i = p(T, U)$ is an inner pair of T', then U is in $in(T', p_i)$.

Proof. We shall show that every edge e on U is in $in(T', p_i)$. If e lies on the walk on B_l clockwise going from $s(T)$ to $t(T)$, then e lies on the walk on B_l clockwise going from t_i to t_{i+1} where $p_i = (t_i, t_{i+1})$, and hence e is $in(T', p_i)$ since p_i is also an inner pair of T'. Thus we may assume that e is not on the walk from $s(T)$ to $t(T)$ and that the claim holds true for all descendants of T. Since F_l is tight, a son T_c of T intersects with the face f (in $in(T)$) adjoining e at a vertex v on $trunk(T_c)$. By Lemma 1(a), the tree $T_c' \in F_l'$ corresponding to T_c is a descendant of T', and $p(T_c) = p(T_c')$. Therefore by the assumption $trunk(T_c)$ is in $in(T_c')$. Therefore v is in $in(T_c')$, and hence f is in $in(T', p_i)$. Consequently e is in $in(T', p_i)$. Q.E.D.

Lemma 4 immediately yields the following corollary.

Corollary 1. Let F_l and F_l' be two Steiner forests of $\mathcal{N}_l$.
(a) If F_l is tight and $p(T) = p(T')$ for two corresponding trees $T \in F_l$ and $T' \in F_l'$, then $in(T) \subset in(T')$.
(b) If F_l is tight and $Z(F_l) = Z(F_l')$, then $in(F_l) \subset in(F_l')$.
(c) If both F_l and F_l' are tight and $p(T) = p(T')$ for $T \in F_l$ and $T' \in F_l'$, then $trunk(T) = trunk(T')$.

Proof. (a) By Lemma 4, $trunk(T)$ is in $in(T')$. Therefore $in(T) \subset in(T')$.
(b) immediate from (a) above.
(c) Since $in(T) \subset in(T')$ and $in(T') \subset in(T)$, $trunk(T) = trunk(T')$. Q.E.D.

The elementary operation to modify F_l is "opening an interval." Assume that $T \in root(F_l)$ spans $N \in S_l$, U is an interval of T, and $p_i = p(T, U)$. Suppose further that G has a tree T' with $p(T') = p_i$ which spans N and does not cross with $F_l - T$, as depicted in Fig. 5. Replace the root T in F_l with T', and let F_l' be the resulting forest of $\mathcal{N}_l$, that is, let $F_l' = F_l - T + T'$. Then $Z(F_l') = Z(F_l) - p(T) + p_i$. Let $G' = G - V(F_l - T - DES(F_l, T, p_i))$, and let f_l' be the face of G' such that $f_l \subset f_l'$. Let $p_i = (t_i, t_{i+1})$, and let W be the walk on the boundary of f_l' clockwise going from t_{i+1} to t_i. Let Q_i be the walk on B_l clockwise going from t_i to t_{i+1}. Then clearly the necessary and sufficient condition for G to have T' above is

Condition B. Face f_2 is included in the inside of cycle $Q_i + W$. ■

We say that *interval U (or inner pair p_i) can be opened* if Condition B holds. A tree T' which spans N and is contained in W is said to be obtained from T by *opening* U or p_i. Note that if F_l is tight then $F_l' = F_l - T - T'$ is also tight. One can easily check Condition B and find T' in $O(n)$ time. Furthermore one can find T' without deleting the vertices in $V(F_l - T - DES(F_l, T, p_i))$ from G if each vertex on a tree is marked with the net number of the tree. That is, one can find T' simply by traversing edges in $in(T') - \bigcup_{T_c \in SON(F_l', T')} in(T_c)$. Denote by $m(F_l', T')$ the number of these edges, and by $time(F_l', T')$ the time needed to construct T' from T, then we have:

Lemma 5. $time(F_l', T') = O(m(F_l', T'))$. ■

Using Lemmas 1, 2 and 4 and Corollary 1, one can prove the following lemma.

Lemma 6. Let F_l be a Steiner forest of network $\mathcal{N}_l$, and let p_i be an inner pair of $T \in root(F_l)$. If $\mathcal{N}_l$ has a Steiner forest F_l' with $p_i \in Z(F_l')$, then p_i can be opened and $\mathcal{N}_l$ has a Steiner forest F_l'' with $Z(F_l'') = Z(F_l) - p(T) + p_i$. (See Fig. 5.)

Proof. We may assume without loss of generality that $l = 1$, trees $T \in F_1$ and $T' \in F_1'$ span a net $N \in S_1$, and $p(T') = p_i$. Furthermore by Lemma 2 we may assume that F_1 is tight. Let P_i be the path on T connecting t_i and t_{i+1}. We first claim that T has an interval U such that $p_i = p(T, U)$. Suppose otherwise, then $E(P_i) \cap E(trunk(T)) = \phi$, and hence $p(T, U_j) \neq p_i$ for every interval U_j of T. Since $p(T, U_j)$ is an inner pair of T', by Lemma 4 U_j is in $in(T')$. Consequently $trunk(T)$ is in $in(T')$, contradicting to the assumption that $p_i = p(T')$.

Since $p_i = p(T')$, the inside of cycle $Q_i + trunk(T')$ includes f_2. Let T_c' be a son of $T' \in F_1'$, and let $T_c \in F_1$ correspond to T_c'. Then by Lemma 1 $p(T_c') = p(T_c)$, and by Corollary 1(a) $in(T_c) \subset in(T_c')$. Therefore Condition B above holds. Q.E.D.

We sometimes call a vertex set $X \subset V$ a *cut*. The *capacity* of a cut X is $|X|$. For $X \subset V$ and net $N \in S$, let $d(X, N) = \mathrm{MIN}_T\, |V(T) \cap X|$, where T runs over all trees in G spanning N. The *demand* $d(X)$ of a cut X is $\sum_{N \in S} d(X, N)$. We say that network $\mathcal{N} = (G, S)$ *satisfies the cut condition* if $d(X) \leq |X|$ for every cut $X \subset V$. Clearly the following lemma holds.

Lemma 7. Network $\mathcal{N} = (G, S)$ satisfies the cut condition if $\mathcal{N}$ has a Steiner forest. ■

The converse of Lemma 7 does not necessarily hold, because there exists a network which satisfies the cut condition but has no Steiner forest. We are now ready to present the key lemmas on which our algorithm FOREST2 is based.

Lemma 8. Let F_1 be a tight forest of $\mathcal{N}_1$, and let F_2 be a tight forest of $\mathcal{N}_2$. If an interval U_1 of $T_1 \in root(F_1)$ crosses with an interval U_2 of $T_2 \in root(F_2)$, then (a)-(g) below hold true.

(a) If $T_a \in root(F_1) - T_1$ and $T_b \in root(F_2) - T_2$, then T_a and T_b do not intersect with the same face f of G.

(b) None of trees in $root(F_1) - T_1$ crosses with the path on B_2 clockwise going from $t(T_2)$ to $s(T_2)$.

(c) None of trees in $root(F_2) - T_2$ crosses with the path on B_1 clockwise going from $t(T_1)$ to $s(T_1)$.

(d) Either $p(T_1, U_1) \in Z(F_a)$ or $p(T_2, U_2) \in Z(F_a)$ for an arbitrary Steiner forest F_a of $\mathcal{N}$.

(e) Either $p(T_1, U_1)$ or $p(T_2, U_2)$ can be opened, and either $\mathcal{N}_1$ has a tight forest F_1' with $Z(F_1') = Z(F_1) - p(T_1) + p(T_1, U_1)$ or $\mathcal{N}_2$ has a tight forest F_2' with $Z(F_2') = Z(F_2) - p(T_2) + p(T_2, U_2)$.

(f) If $\mathcal{N}_1$ has F_1' above, then tree $T_1' \in F_1'$ obtained from T_1 by opening $p(T_1, U_1)$ crosses with none of trees in $root(F_2) - T_2$, that is, T_1' may cross only with T_2 or T_2's descendants. If $\mathcal{N}_2$ has F_2', then tree $T_2' \in F_2'$ obtained from T_2 by opening $p(T_2, U_2)$ crosses with none of trees in $root(F_1) - T_1$.

(g) Assume that $p(T_1, U_1)$ can be opened, and let F_1' and T_1' be as above. Let $\mathcal{U}_2$ be the set of inner terminal pairs of T_2 whose intervals cross with U_1. Let $\mathcal{U}_2'$ be the set of inner terminals pairs of T_2 whose intervals cross with $trunk(T_1')$. Then $|\mathcal{U}_2 \cap \mathcal{U}_2'| \leq 1$.

Proof. (a) Since F_1 and F_2 are tight, there exist two maximal sequences of vertices $X = \{x_1, x_2, ..., x_q\}$ and $Y = \{y_1, y_2, ..., y_r\}$ such that

(1) $x_1 = y_1 \in V(U_1) \cap V(U_2)$, $x_q \in V(f_1)$, and $y_r \in V(f_2)$;

(2) the tree in F_1 containing vertex x_i, $2 \leq i \leq q$, is a son of the tree in F_1 containing x_{i-1}, and both x_i and x_{i-1} lie on the same face boundary of G; and

(3) the tree in F_2 containing y_i, $2 \leq i \leq r$, is a son of the tree in F_2 containing y_{i-1}, and both y_i and y_{i-1} lie on the same face boundary of G.

Suppose for a contradiction that T_a and T_b intersect with face f. Then there exist vertices $v_a \in V(T_a) \cap V(f)$ and $v_b \in V(T_b) \cap V(f)$. We may assume that $v_a \in V(trunk(T_a))$ and $v_b \in V(trunk(T_b))$. Similarly as above, there exist two maximal sequences of vertices $X' = \{x_1', ..., x_{q'}'\}$ and $Y' = \{y_1', ..., y_{r'}'\}$ such that $x_1' = v_a$, $x_{q'}' \in V(f_1)$, $y_1' = v_b$, $y_{r'}' \in V(f_2)$ and X' and Y' satisfy conditions similar to (2) and (3). Let $|W| = X \cup Y \cup X' \cup Y'$, then $|W| < d(W)$, because

$$|W| \leq |X| + |Y| + |X'| + |Y'| - 1,$$

and

$$d(W) \geq |X| + |Y| + |X'| + |Y'|.$$

Thus $\mathcal{N}$ does not satisfy the cut condition, and hence by Lemma 7 $\mathcal{N}$ has no Steiner forest, a contradiction.

The proofs of (b) and (c) are similar to (a) above.

(d) Let $T_1' \in F_a$ correspond to $T_1 \in F_1$, and let $T_2' \in F_a$ correspond to $T_2 \in F_2$. Suppose that $p(T_1') \neq p(T_1, U_1)$ and $p(T_2') \neq p(T_2, U_2)$ although intervals U_1 and U_2 cross at vertex v. Then $p(T_1, U_1)$ and $p(T_2, U_2)$ are inner pairs of T_1' and T_2', respectively. Therefore by Lemma 4 $v \in in(T_1') \cap in(T_2')$, and hence trees T_1' and T_2' in forest F_a would cross each other, a contradiction.

(e) By (d) above and Lemma 6 either F_1' or F_2' exists.

(f) Using (a)-(c) above, one can easily verify the claim.

(g) Either $p(T_1, U_1) \in Z(F_a)$ or $p(T_2, U_2) \in Z(F_a)$ for an arbitrary forest F_a of $\mathcal{N}$. Therefore if $p(T_1, U_1) = p(T_1') \notin Z(F_a)$, then by (d) above $\mathcal{U}_2 \subset Z(F_a)$, and hence $|\mathcal{U}_2| = 1$. Similarly, if $p(T_1, U_1) = p(T_1') \in Z(F_a)$, then $|\mathcal{U}_2'| = 1$. Therefore $|\mathcal{U}_2 \cap \mathcal{U}_2'| \leq 1$ in either case. Q.E.D.

We denote by $\#(F_1)$ the number of trees in F_1 crossing with F_2, and by $\#(F_2)$ the number of trees in F_2 crossing with F_1. Let $\#(F_1, F_2) = \#(F_1) + \#(F_2)$. By Lemma 8(g) $|\mathcal{U}_2 \cap \mathcal{U}_2'|$ is either 1 or 0. For these two cases Lemmas 9 and 10 below show how to modify F_1 and F_2 to decrease $\#(F_1, F_2)$. One can prove the following two lemmas by using Lemmas 6, 7, and 8.

Lemma 9. Assume that $p(T_1, U_1)$ can be opened and $\mathcal{U}_2 \cap \mathcal{U}_2' = \{p'\}$, where $p' = p(T_2, U_2)$. Then $p' \in Z(F_a)$ for an arbitrary forest F_a of $\mathcal{N}$. Let T_2' be a tree obtained from T_2 by opening p', and let $F_2' = F_2 - T_2 + T_2'$. If T_2' does not cross with F_1, then $\#(F_1, F_2') < \#(F_1, F_2)$. On the other hand, if T_2' crosses with F_1, then (a)-(c) below hold:

(a) T_2' crosses only with tree T_1 in F_1;

(b) $trunk(T_2')$ crosses with exactly one interval U_1' of $trunk(T_1)$;

(c) let T_1'' be a tree obtained from T_1 by opening U_1', and let $F_1'' = F_1 - T_1 + T_1''$, then T_2' does not cross with F_1'', T_1'' does not cross with F_2', and hence $\#(F_1'', F_2') < \#(F_1, F_2)$. ■

Lemma 10. Assume that $p(T_1, U_1)$ can be opened and $\mathcal{U}_2 \cap \mathcal{U}_2' = \phi$. Let U_2 be an interval of T_2 with $p(T_2, U_2) \in \mathcal{U}_2$, and let U_2' be an interval of T_2 with $p(T_2, U_2') \in \mathcal{U}_2'$.

Case (a): *U_2' can be opened.* Let T_2'' be a tree obtained from T_2 by opening U_2', and let $F_2'' = F_2 - T_2 + T_2''$. If T_2'' does not cross with T_1', then T_2'' does not cross with F_1', T_1' does not cross with F_2'', and hence $\#(F_1', F_2'') < \#(F_1, F_2)$.

Case (b): *either U_2' cannot be opened or T_2'' crosses with T_1'.* In this case U_2 can be opened. Let T_2' be a tree obtained from T_2 by opening U_2, and let $F_2' = F_2 - T_2 + T_2'$. If T_2' does not cross with F_1, then $\#(F_1, F_2') < \#(F_1, F_2)$. If T_2' crosses with F_1, then $\#(F_1''', F_2') < \#(F_1, F_2)$, where U_1'' is the interval of $trunk(T_1)$ crossing with $trunk(T_2')$, T_1''' is a tree obtained from T_1 by opening U_1'', and $F_1''' = F_1 - T_1 + T_1'''$. ■

4.3 Algorithm. Lemmas 8-10 immediately yields the following iterative algorithm to find a Steiner forest of $\mathcal{N}$.

```
procedure FOREST2(N);
 begin
(1) find tight forests F1 of N1 and F2 of N2;
(2) while #(F1,F2) > 0 do
     begin
       let an interval U1 of T1 ∈ root(F1) cross with an interval U2
       of T2 ∈ root(F2);
(3)    open intervals U1 and U2, and assume w.l.o.g. that U1 can be
       opened;
       if T1' does not cross with F2 then
(4)      (F1,F2) := (F1',F2) {replace F1 with F1'}
       else {T1' crosses with T2 by Lemma 8(f)}
         if |U2 ∩ U2'| = 1 then {Lemma 9}
           if T2' does not cross with F1 then
(5)          (F1,F2) := (F1,F2')
           else
(6)          (F1,F2) := (F1'',F2')
         else {U2 ∩ U2' = φ. Lemma 10}
           if U2' can be opened and T2'' does not cross with T1' then
(7)          (F1,F2) := (F1',F2'')
           else {Case (b) of Lemma 10}
             if T2' does not cross with F1 then
(8)            (F1,F2) := (F1,F2')
             else
(9)            (F1,F2) := (F1''',F2')
     end
 end;
```

By Lemma 3 Statement (1) in FOREST2 can be done in $O(n)$ time. When the **while** loop (2) is executed once, $\#(F_1, F_2)$ necessarily decreases. Therefore the loop is executed at most $k = |S|$ times. One execution of Statement (3) can be done in $O(n)$ time. At most two intervals of T_l are opened during one execution of the loop, and each opening can be done in $O(n)$ time. Thus one can easily know that our algorithm runs in $O(kn)$ time. (On the contrary, Robertson and Seymour considered all possible $O(n^2)$ patterns of "homotopy" (or $Z(F)$), so the straightforward implementation of an algorithm whose existence was proved by them requires $O(n^3)$ time.)

We next give an $O(n)$ time implementation of FOREST2. Let F_{1s} be an initial tight forest of $\mathcal{N}_1$ and F_{2s} of $\mathcal{N}_2$, found by Statement (1). Let F_{1e} be a final one of $\mathcal{N}_1$ and F_{2e} of $\mathcal{N}_2$ obtained by FOREST2. When executing Statement (2) first, one need to traverse all the edges in $\bigcup_{T \in root(F_1) \cup root(F_2)} trunk(T)$ to check whether $\#(F_1, F_2) = 0$. However, when executing Statement (2) later, one need not traverse all these edges but the edges on trees which newly become roots. Therefore the total time needed to check whether $\#(F_1, F_2) = 0$ is

$$\sum_{T \in F_{1s} \cup F_{2s}} O(|E(trunk(T))|) = O(|E|) = O(n).$$

The straightforward execution of Statement (3) requires $O(kn)$ time in total as above. Our idea to improve this to $O(n)$ is simply to replace (3) with the following statement (3)′:

(3)′ open intervals U_1 and U_2 simultaneously, and when either U_1 or U_2, say U_1, has been opened, terminate the operation for opening U_2;

Statement (3)′ constructs F_1' and F_2' from F_1 and F_2 simultaneously. By Lemma 5 $time(F_1', T_1') = O(m(F_1', T_1'))$ and $time(F_2', T_2') = O(m(F_2', T_2'))$. We are assuming that $time(F_1', T_1') \le time(F_2', T_2')$. Tree $T_1' \in F_1'$ found in Statement (3)′ does not necessarily belong to F_{1e}. However if $T_1' \notin F_{1e}$, then $T_2' \in F_{2e}$. Therefore, let $T_i' \in F_{1e} \cup F_{2e}, i = 1$ or 2, then the execution time of Statement (3)′ is at most $2time(F_{ie}, T_i')$. Thus the total execution time of Statement (3)′ is bounded above by the time needed to modify trees.

Each execution of the **while** loop does either Statement (4), (5), (6), (7), (8), or (9). In each case, the execution time of the loop, excluding the time needed to check whether $\#(F_1, F_2) = 0$, is bounded by the following amount.

Case of (4). $2time(F_1', T_1') = O(m(F_1', T_1')) = O(m(F_{1e}, T_1'))$. (Note that $T_1' \in F_{1e}$ since every tree modified in an execution of the loop (2) will never be modified in any later execution.)

Case of (5). $time(F_1', T_1') + time(F_2', T_2')$
$= O(m(F_2', T_2')) = O(m(F_{2e}, T_2'))$.

Case of (6). $time(F_1', T_1') + time(F_2', T_2') + time(F_1'', T_1'')$
$= O(m(F_{2e}, T_2') + m(F_{1e}, T_1''))$.

Case of (7). $2time(F_1', T_1') + time(F_2'', T_2'')$
$= O(m(F_1', T_1') + m(F_2'', T_2''))$
$= O(m(F_{1e}, T_1') + m(F_{2e}, T_2''))$.

Case of (8). Since the outer pairs of T_2'', T_2', and T_2 are all distinct, by Lemma 4 $in(T_2'') \subset in(T_2') \cup in(T_2)$. Therefore

$$m(F_2'', T_2'') \le m(F_2', T_2') + m(F_2, T_2).$$

Hence,

$$\begin{aligned} time(F_1', T_1') + time(F_2'', T_2'') + time(F_2', T_2') \\ \le 3time(F_2', T_2') + O(m(F_2, T_2)) \\ = O(m(F_{2e}, T_2') + m(F_{2s}, T_2)). \end{aligned}$$

Note that $T_2 \in F_{2s}$ and $T_2' \in F_{2e}$ in this case.

Case of (9). The execution time of the loop for this case is bounded by the time for Case of (8) plus $time(F_1''', T_1''')$, i.e.,

$$O(m(F_{2e}, T_2') + m(F_{2s}, T_2) + m(F_{1e}, T_1''')).$$

The trees modified in the six cases above will not be modified again. Therefore one can know that the total execution time of the algorithm is

$$\begin{aligned} &\sum_{T \in F_{1s}} O(m(F_{1s}, T)) + \sum_{T \in F_{2s}} O(m(F_{2s}, T)) \\ &+ \sum_{T \in F_{1e}} O(m(F_{1e}, T)) + \sum_{T \in F_{2e}} O(m(F_{2e}, T)) + O(n) \\ &= O(n). \end{aligned}$$

Thus we have:

THEOREM 2. A Steiner forest of network $\mathcal{N} = (G, S)$ can be found in $O(n)$ time for the case $S_{12} = \phi$. ■

5. Case in which there is a net intersecting with $V(f_1)$ and $V(f_2)$

In this section, we present an algorithm FOREST3 for finding a Steiner forest of network $\mathcal{N} = (G, S)$ for the case $S_{12} \ne \phi$. FOREST3 is based on several results in [RS] and a new algorithm PATH for finding internally disjoint paths. We first outline FOREST3 consisting of three steps, using a concrete example.

STEP 1. This step reduces the Steiner forest problem to the disjoint path problem. Let $\mathcal{N}_1 = (G, S_1)$ and $\mathcal{N}_2 = (G, S_2)$. We first find two Steiner forests F_1 of $\mathcal{N}_1$ and F_2 of $\mathcal{N}_2$. In Fig. 7(a) F_1 and F_2 are drawn by thick lines. We then delete all vertices of $F_1 + F_2$ from G, and let $G' = G - V(F_1 + F_2)$. G' is drawn in Fig. 7(b). Let $S_1' = \{N \cap V(f_1) \mid N \in S_{12}\}$, and let $\mathcal{N}_1' = (G', S_1')$. Define S_2' and $\mathcal{N}_2'$ similarly. We then find two Steiner forests F_1' of $\mathcal{N}_1'$ and F_2' of $\mathcal{N}_2'$, and contract all edges of $F_1' + F_2'$ in G'. Let $\mathcal{N}_{12} = (G'', S_{12}'')$ be the resulting network. F_1' and F_2' are drawn by thick lines in Fig. 7(b), and $\mathcal{N}_{12}$ in Fig. 7(c). Each of the k_{12} nets in S_{12}'' consists of exactly two terminals; one on the outer boundary and the other on an inner boundary. Hence a Steiner forest F_{12} of $\mathcal{N}_{12}$ indeed consists of vertex-disjoint paths. (As shown later, if F_1, F_2, F_1', and F_2' are appropriately chosen, then $\mathcal{N}_{12}$ necessarily has a forest F_{12} and $F_1 + F_2 + F_1' + F_2' + F_{12}$ is a forest of $\mathcal{N}$. Baker and Pinter have solved the disjoint path problem of this type for a special class of grids [BP].)

STEP 2. The disjoint path problem for $\mathcal{N}_{12}$ is solved by this step and the succeeding Step 3. In this step we simply find k_{12} vertex-disjoint paths $\mathcal{P}$ in G'', each connecting a terminal on the outer boundary B_1 and a terminal on the inner boundary B_2 of network $\mathcal{N}_{12}$. $\mathcal{P}$ are drawn by thick lines in Fig. 7(c). Note that these paths do not necessarily connect terminals of the same net.

STEP 3. In this step we modify $\mathcal{P}$ so that each path connects two terminals of the same net. Let $\mathcal{P}'$ be the resulting disjoint paths, then $F = F_1 + F_2 + F_1' + F_2' + \mathcal{P}'$ is a Steiner forest of network $\mathcal{N}$. $\mathcal{P}'$ is drawn by thick lines in Fig. 7(d), and F in Fig. 1.

We next present the details of Steps 1, 2, and 3 in subsections 5.1-5.3

5.1 STEP 1. We find four forests F_1, F_2, F_1', and F_2' for which there exist desired paths $\mathcal{P}'$. Since $k_{12} \ge 1$, the "homotopy" of F_l is uniquely determined. That is, if F_l can be a subgraph of a Steiner forest of $\mathcal{N}$, then $Z(F_l)$ is uniquely determined. (Observe this fact in Figs. 1 and 8(a) by considering all possible trees spanning net 1.) Among all such forests, we find F_l with the minimal $in(F_l)$. One can find such a forest F_l of $\mathcal{N}_l$ in $O(n)$ time by applying procedure FOREST1 in Section 3 to network $\mathcal{N}_l$, with choosing any terminal (on B_l) of a net in S_{12} as the starting vertex v_0. It may be assumed that a Steiner forest of $\mathcal{N}$ contains F_1 and F_2. Therefore we delete all vertices of $F_1 + F_2$ from G. Let $\mathcal{N}' = (G', S_{12})$ be the resulting network. $\mathcal{N}'$ is depicted in Fig. 7(b). If $k_{12} = 1$, then find in G' a tree T spanning the net in S_{12}, and output $F = F_1 + F_2 + T$ as a Steiner forest of $\mathcal{N}$. Thus we may assume that $k_{12} \ge 2$ and $S_{12} = \{N_1, N_2, ..., N_{k_{12}}\}$. Then the outer boundary of G' contains k_{12} vertex-disjoint walks $W_{1i}, 1 \le i \le k_{12}$ such that $N_i \subset V(W_{1i})$, $V(W_{1i}) \cap N_j = \phi$ for every $j \ne i, 1 \le j \le k_{12}$, and both of the ends of W_{1i} are terminals in $N_i \cap V(f_1)$. Similarly define k_{12} vertex-disjoint walks $W_{2i}, 1 \le i \le k_{12}$ in the inner boundary of G'. These $2k_{12}$ walks are drawn by thick lines in Fig. 7(b). Let F_1' be a forest of $\mathcal{N}_1'$ contained in $\sum W_{1i}$, and let F_2' be a forest of $\mathcal{N}_2'$ contained in $\sum W_{2i}$. One may assume that $F_1' + F_2'$ is contained in a Steiner forest of $\mathcal{N}$. So we contract all edges of these $2k_{12}$ walks in G', and let $\mathcal{N}_{12} = (G'', S_{12}'')$ be the resulting network. Thus we have reduced the Steiner forest problem for $\mathcal{N}$ to the disjoint path problem for $\mathcal{N}_{12}$.

Since FOREST1 runs in $O(n)$ time and the deletion and contraction of edges can be done in $O(n)$ time, STEP1 runs in $O(n)$ time.

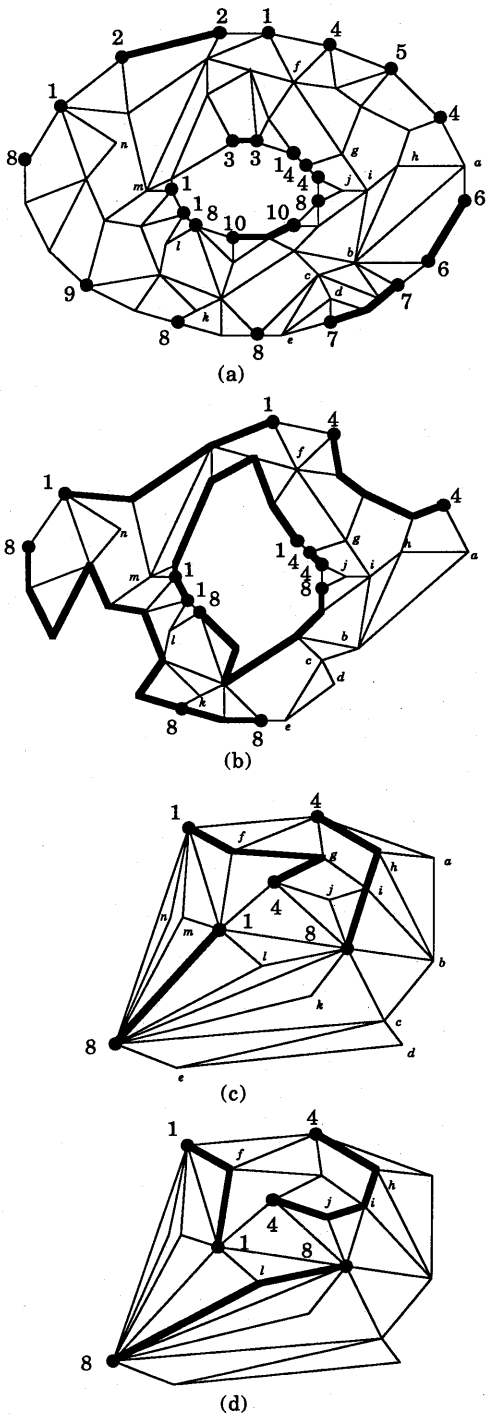

Fig. 7(a) F_1 and F_2 for network $\mathcal{N} = (G, S)$ in Fig. 1;
(b) $\mathcal{N}' = (G', S_{12})$ and $2k_{12}$ walks;
(c) $\mathcal{N}_{12}$ and vertex-disjoint paths $\mathcal{P}$; and
(d) $\mathcal{P}'$.

5.2 **Step 2.** In this step we find k_{12} vertex-disjoint paths in graph G''. Let G^+ be the graph obtained from G'' as follows:

(1) add two new vertices s and t to G'';
(2) join s with each of the k_{12} terminals on the inner face boundary; and
(3) join t with each of the k_{12} terminals on the outer face boundary.

Find k_{12} internally disjoint s-t paths in G^+. Then k_{12} vertex-disjoint paths of G'' can be immediately obtained from them. Applying network flow algorithms, one can find k_{12} internally disjoint paths in $O(\mathrm{MIN}\{k_{12}n, n\sqrt{n}\})$ [ET]. Furthermore we can find them in $O(\mathrm{MIN}\{k_{12}n, n\log n\})$ time using none of flow algorithms. Therefore Step 2 can be done in $O(\mathrm{MIN}\{k_{12}n, n\log n\})$ time.

5.3 **Step 3.** The paths found in Step 2 do not necessarily connect terminals of the same net. In Step 3 we find vertex-disjoint paths of $\mathcal{N}_{12} = (G'', S''_{12})$, each connecting the terminals of the same net. Let $S''_{12} = \{(s_i, t_i) | 1 \le i \le k_{12}\}$, where $s_i \neq t_i$, and s_i lies on the inner boundary and t_i on the outer boundary. One may assume that G'' is embedded in the ring $\Sigma = \{(x, y) | 1 \le x^2 + y^2 \le 2^2\}$ bounded by two circles C_1, $x^2 + y^2 = 1$, and C_2, $x^2 + y^2 = 2^2$, as follows:

$Image(s_i) = (\cos \frac{2\pi i}{k_{12}}, \sin \frac{2\pi i}{k_{12}})$,
$Image(t_i) = (2\cos \frac{2\pi i}{k_{12}}, 2\sin \frac{2\pi i}{k_{12}})$, and
$Image(G'') \cap (C_1 \cup C_2)$
$= \{Image(s_i), Image(t_i) | 1 \le i \le k_{12}\}$.

Let O be the origin of the x-y plane. Let P be a path in Σ which starts with s_i and ends with t_j. Let θ be the total angle turned through (measured counterclockwise) by the line OX, where point X moves on P from s_i to t_j. Possibly $|\theta| > 2\pi$. We define the *angle* $\theta(P)$ *of path* P by $\theta(P) = k_{12}\theta/2\pi$. Thus $\theta(P)$ is an integer.

Let $\mathcal{P} = \{P_1, P_2, ..., P_{k_{12}}\}$ be the k_{12} vertex-disjoint paths in G'' found by STEP2, where P_i starts with s_i but does not necessarily end with t_i. Clearly $\theta(P_i)$, $1 \le i \le k_{12}$, are all equal. We denote this value by $\theta(\mathcal{P})$ and call it *the angle of vertex-disjoint paths* $\mathcal{P}$. If $\theta(\mathcal{P})$ is a multiple of k_{12}, then each path in $\mathcal{P}$ would have connected terminals of the same net. Thus we may assume that $\theta(\mathcal{P})$ is not a multiple of k_{12}. In Step 3, we modify $\mathcal{P}$ so that $\theta(\mathcal{P})$ becomes a multiple of k_{12}. Our algorithm uses the following result (Theorem (5.9) in [RS, p.134]).

Theorem 3. If G'' has vertex-disjoint paths $\mathcal{P}_1$ and $\mathcal{P}_2$ with $\theta(\mathcal{P}_1) \le \theta(\mathcal{P}_2)$, then G'' has vertex-disjoint paths $\mathcal{P}$ with $\theta(\mathcal{P}) = \alpha$ for any integer α, $\theta(\mathcal{P}_1) \le \alpha \le \theta(\mathcal{P}_2)$. ■

One may assume that $0 < \theta(\mathcal{P}) < k_{12}$; otherwise, re-embed G'' in Σ with fixing C_1 and rotating C_2 appropriately. Then Theorem 3 implies that G'' has vertex-disjoint paths $\mathcal{P}'$ such that either $\theta(\mathcal{P}') = 0$ or $\theta(\mathcal{P}') = k_{12}$. We now present an algorithm which finds vertex-disjoint paths $\mathcal{P}'$ with $\theta(\mathcal{P}') = 0$ whenever there exists such $\mathcal{P}'$. Similarly one can find vertex-disjoint paths $\mathcal{P}'$ with $\theta(\mathcal{P}') = k_{12}$. From now on, indices are counted modulo k_{12}, so $P_{k_{12}} = P_0$.

Let $\alpha = \theta(\mathcal{P})$, and let $P_i \in \mathcal{P}, 1 \le i \le k_{12}$, be a path in G'' connecting s_i and $t_{i+\alpha}$. Let R_i, $1 \le i \le k_{12}$, be the walk on the inner face boundary counterclockwise going from s_i to s_{i+1}. Although R_i is updated during the execution of the algorithm, we always denote it by R_i. If there exist vertex-disjoint paths $\mathcal{P}'$ with $\theta(\mathcal{P}') = 0$ in G'', then there exist disjoint paths $\mathcal{P}'$ with $\theta(\mathcal{P}') = 0$ in a graph obtained from G'' by the following operation (a), (b), or (c):

(a) Delete the edge e on R_i incident with s_i if $e \notin E(P_i)$.
(b) Contract the edge e incident with s_i if s_i has degree one. (The vertex adjacent with s_i is not on C_2; otherwise, the vertex would be terminal $t_{i+\alpha}$, and consequently vertex-disjoint paths $\mathcal{P}'$ with $\theta(\mathcal{P}') = 0$ would not exist.)
(c) If path P_i intersects with R_{i-1} except at s_i, then let v be the vertex on R_{i-1} which appears last on P_i, and replace the subpath of P_i from s_i to v with the path on R_{i-1} clockwise going from s_i to v.

One can easily observe that operations (b) and (c) preserve disjoint paths $\mathcal{P}'$ with $\theta(\mathcal{P}') = 0$. On the other hand it is intuitively clear but nontrivial that operation (a) preserves paths $\mathcal{P}'$ with $\theta(\mathcal{P}') = 0$. This fact can be immediately derived from Theorems (5.5) and (5.10) in [RS]. Apply (a)-(c) above to $\mathcal{N}_{12}$ and $\mathcal{P}$ repeatedly, and let $\mathcal{N}_{12}$ be the new network for which none of (a)-(c) can be applied. Then each path $P_i - s_i$ intersects with R_i but does not intersect with R_{i-1}. In this case the following operation (d) can "rotate" such paths by angle -1.

(d) Let $u_i, 1 \leq i \leq k_{12}$, be the vertex on R_{i-1} which appears last on P_{i-1}. Note that $u_i \neq s_{i-1}$. Let Q_i be the walk on R_{i-1} clockwise going from s_i to u_i, and let U_i be the subpath of P_{i-1} going from u_i to $t_{i-1+\alpha}$. Replace path P_i connecting s_i and $t_{i+\alpha}$ with path $Q_i + U_i$ connecting s_i and $t_{i-1+\alpha}$ through u_i.

Repeating the operations above α times, one can construct vertex-disjoint paths $\mathcal{P}'$ with $\theta(\mathcal{P}') = 0$ as below.

procedure STEP3($\mathcal{N}_{12}, \mathcal{P}$);
begin
let $\theta(\mathcal{P}) = \alpha$;
for $i := 1$ **to** k_{12} **do** T_i :=an empty graph;
{ initialization of the vertex-disjoint paths $\mathcal{P}' = \{T_1, T_2, ..., T_{k_{12}}\}$}
(1) **for** $m := \alpha$ **downto** 1 **do**
begin
(2) **for** $i := 1$ **to** k_{12} **do**
if $V(P_i - s_i) \cap V(R_{i-1}) \neq \phi$
then SHORTCUT(P_i, R_{i-1}); {combined operation of (a)-(c)}
$i := 1$;
(3) **while** $V(P_i - s_i) \cap V(R_{i-1}) \neq \phi$ **do**
begin
SHORTCUT(P_i, R_{i-1});
$i := i + 1$
end;
{none of operations (a), (b), and (c) is applicable to $\mathcal{N}_{12}$. Each path $P_i - s_i$ intersects with R_i and does not intersect with R_{i-1}. Operation (d) is next executed}
for $i := 1$ **to** k_{12} **do**
begin
let u_i be the vertex on R_{i-1} that appears last on P_{i-1};
let Q_i be the walk on R_{i-1} clockwise going from s_i to u_i;
let U_i be the subpath of P_{i-1} from u_i to t_{i-1+m}
end;
for $i := 1$ **to** k_{12} **do** $P_i := Q_i + U_i$
end; {Statement (1) ends}
for $i := 1$ **to** k_{12} **do** $T_i := T_i + P_i$;
$\mathcal{P}' := \{T_1, T_2, ..., T_{k_{12}}\}$
end;

procedure SHORTCUT(P_i, R_{i-1});
begin
let v be the vertex on R_{i-1} which appears last on P_i; $\{v \neq s_i\}$
let R_v be the subwalk of R_{i-1} clockwise going from s_i to v;
$T_i := T_i + R_v$; {operations (a), (b) and (c). R_v is determined to be included in P_i}
$G := G - (V(R_v) - v)$;
$s_i := v$;
update P_i, R_i and R_{i-1};
assume that the edges $e_1, e_2, ..., e_d$ incident with vertex s_i are embedded in this order clockwise around s_i, where e_1 is the first edge of R_i;
let e_j be the first edge of P_i;
delete edges $e_1, e_2, ..., e_{j-1}$, and update R_i {operation (a)}
end;

We next show that procedure STEP3 runs correctly. If none of (a)-(c) is applicable to $\mathcal{N}_{12}$ when the **while** loop (3) terminates, then Operation (d) is applicable and the algorithm finds vertex-disjoint paths $\mathcal{P}'$ with $\theta(\mathcal{P}') = \theta(\mathcal{P}) - 1$. Therefore it suffices to show that none of Operations (a)-(c) is applicable when the **while** loop terminates. Just after SHORTCUT(P_i, R_{i-1}) is executed in the **while** loop, $P_i - s_i$ does not intersect with R_{i-1}, and none of (a)-(c) is applicable for i. Thereafter $P_i - s_i$ and R_{i-1} remain vertex-disjoint as long as R_{i-1} is not altered. R_{i-1} is altered only when either SHORTCUT(P_i, R_{i-1}) or SHORTCUT(P_{i-1}, R_{i-2}) is executed. Therefore none of (a)-(c) is applicable for any i, $1 \leq i \leq k_{12}$, when the **while** loop terminates.

We next analyze the execution time of STEP3. The total execution time of STEP3, excluding the time needed to find intersections of P_i and R_{i-1} in the **for** loop (2) and in the **while** loop (3), is proportional to the number of edges deleted from G, and hence is $O(n)$. Thus we shall show that the time needed to find intersections of P_i and R_{i-1} is $O(n)$ in total. When executing the **for** statement (2), one can find all the intersections of P_i and R_{i-1} by traversing the inner face boundary. On the other hand, traversing only the edges newly appearing on the inner face boundary, one can find new intersections of P_i and R_{i-1} caused by the modification of the graph. That is, just after executing SHORTCUT(P_{i-1}, R_{i-2}), traverse counterclockwise the new portion of inner face boundary from the end($\neq v$) of e_j, and, if we meet vertices on P_i, then choose the first one as new u_i. Thus when executing the **for** loop (1) for $m = \alpha$, we traverse at most once each of the edges appearing on the inner face boundary to find the intersections of P_i and R_{i-1}. Furthermore, all the edges on the inner face boundary which are traversed during the execution of the **for** loop (1) for m and are not deleted from the graph are necessarily deleted during the execution of the **for** loop for $m - 1$. Therefore each edge in G is traversed at most twice, and hence the time needed to find the intersections of P_i and R_{i-1} is $O(n)$ in total. Thus Step 3 runs in $O(n)$ time.

We have proved that FOREST3 runs in $O(\text{MIN}\{k_{12} n, n\log n\})$, and can conclude this section by the main theorem of this paper.

Theorem 4. A Steiner forest of network $\mathcal{N} = (G, S)$ can be found in $O(\text{MIN}\{(k_{12}+1)n, n\log n\})$ time if all the terminals lie on at most two face boundaries of a planar graph G. ■

Remark. Theorem 4 can be generalized as follows: a Steiner forest of $\mathcal{N} = (G, S)$ can be found in $O(n^{h-2}\text{MIN}\{(k_{12}+1)n, n\log n\})$ time if all the terminals lie on a constant number of face boundaries $B_1, B_2, ..., B_h$, $h \geq 2$, of a planar graph G and each net $N \in S$ satisfies either $N \subset V(B_1) \cup V(B_2)$ or $N \subset V(B_i)$ for some $i, 3 \leq i \leq h$, where k_{12} is the number of nets N with $N \cap V(B_1) \neq \phi$ and $N \cap V(B_2) \neq \phi$.

Let $h \geq 3$ and let $F = F_1 + F_2 + ... + F_h + F_{12}$ be a forest of $\mathcal{N}$, where $F_i, 1 \leq i \leq h$, is a forest of trees spanning nets N with $N \subset V(B_i)$, and F_{12} is a forest of trees spanning nets N with $N \cap V(B_1) \neq \phi$ and $N \cap V(B_2) \neq \phi$. Then the relation $in(F_i) \subset in(F_j)$ among $F_3, F_4, ..., F_h$ defines a genealogy forest of $h - 2$ nodes. Since h is a constant, there are a constant number (at most $(h-1)^{h-2}$) of genealogy forests. We consider all these genealogy forests. For each of them we construct $F_3, F_4, ..., F_h$ in the postorder. Thus, if F_3 is a leaf in the genealogy forest, then we first construct F_3. By Lemma 1(c) and Corollary 1(b) it suffices to consider at most n tight forests F_3 with distinct $Z(F_3)$. Thus for a fixed genealogy forest there are at most n^{h-2} patterns of $\{Z(F_3), ..., Z(F_h)\}$. Noting these facts and using FOREST3, one can immediately give an algorithm of the claimed complexity to find a Steiner forest of $\mathcal{N}$.

Acknowledgment. We would like to thank Professor N. Saito for many stimulating discussions. This research is partly supported by Grant in Aid for Scientific Research of the Ministry of Education, Science, and Culture, under grant number: General Research (C) 62550253 (1987).

References

[BP] B. S. Baker and R. Y. Pinter, An algorithm for optimal placement and routing of a circuit within a ring of pads, 24th Ann. Symp. FOCS, pp. 360-370 (1983).

[ET] S. Even and R. E. Tarjan, Network flow and testing graph connectivity, SIAM J. Compt. 4, 4, pp. 507-518 (1975).

[FF] L. R. Ford and D. R. Fulkerson, Maximal flow through a network, Canad. J. Math., 8, pp. 399-404 (1956).

[GT] H. N. Gabow and R. E. Tarjan, A linear-time algorithm for a special case of disjoint set union, Journal of Computer and System Sciences, 30, pp. 209-221 (1985).

[HJ] R. Hassin and D. B. Johnson, An $O(n\log^2 n)$ algorithm for maximum flow in undirected planar networks, SIAM J. Compt., 14, pp. 612-624 (1985).

[KL] M. R. Kramer and J. van Leeuwen, Wire-routing is NP-complete, Report No. RUU-CS-82-4, Department of Computer Science, University of Utrecht, Utrecht, the Netherlands (1982).

[Lyn] J. F. Lynch, The equivalence of theorem proving and the interconnection problem, ACM Sigma Newsletter 5:3, pp. 31-65 (1975).

[Men] K. Menger, Zur allgemeinen Kurventheorie, Fund. Math.10, pp. 95-115 (1927).

[Rei] J. H. Reif, Minimum s-t cut of a planar undirected network in $O(n\log^2(n))$ time, SIAM J. on Compt., 12, pp. 71-81 (1981).

[Ric] D. Richards, Complexity of single-layer routing, unpublished manuscript (1981).

[RS] N. Robertson and P. D. Seymour, Graph minors. VI. Disjoint paths across a disc, Journal of Combinatorial Theory, Series B, 41, pp. 115-138 (1986).