平面グラフでスタイナー林を求める並列アルゴリズム

鈴木均　山中智勢子　西関隆夫
東北大学工学部通信工学科

本論文では，平面グラフGと幾つかのネットが与えられたときにスタイナー林を求めるCREW PRAM上の並列アルゴリズムを与える．スタイナー林とはG上の点素な木で，各木が一つのネットの全ての端子を連結するものである．全ての端子がグラフGの外周上にある場合には，$O(n^3/\log n)$台のプロセッサを用いて$O(\log^2 n)$時間でスタイナー林を求めることができる．ここでnはグラフの点数である．またこのアルゴリズムから，各ネットの全ての端子が指定された定数個の面の一つの周上にある場合や，全ての端子が二つの面の周上にある場合(二つの周に端子をもつネットがあってもよい)についてのNCアルゴリズムが得られる．更に，平面グラフの指定された二点の間の最大本数の内素な道を求めるNCアルゴリズムも得られる．

Parallel Algorithms for Finding Steiner Forests in Planar Graphs

Hitoshi Suzuki, Chiseko Yamanaka and Takao Nishizeki

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

Abstract. 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. This paper presents several NC algorithms to solve the problems in parallel. An algorithm for the case all the terminals are located on the outer boundary of G runs in $O(\log^2 n)$ time and uses $O(n^3/\log n)$ processors on a CREW PRAM, where n is the number of vertices in G. An algorithm for the case all the terminals of each net lie on one of a fixed number of face boundaries runs in a poly-log time using a polynomial number of processors. On the other hand an algorithm for the case all terminals lie on two face boundaries runs in $O(\log^2 n)$ time using $O(n^6/\log n)$ processors. Furthermore we give an NC algorithm for finding a maximum number of internally disjoint paths between two specified vertices in planar graphs.

1. Introduction

A *Steiner forest* of an undirected graph G is a set of disjoint trees each of which interconnects all the terminals in each net. Although the well-known Steiner tree problem is to find a minimum tree interconnecting all specified terminals in a given weighted graph, we do not require that a Steiner forest has the minimum total weight of edges or a minimum number of edges. Therefore our problem is a generalization of the disjoint path problem. 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.

Robertson and Seymour [RS] and Suzuki, Akama and Nishizeki [SAN] obtained sequential algorithms to solve the problem for the case that G is planar and all the terminals are located on two face boundaries. Their algorithms run in $O(n^3)$ and $O(n \log n)$ time, respectively, where n is the number of vertices in G. Furthermore Schrijver showed that a Steiner forest of a planar graph can be found in $O(n^{h+2} \log^2 n)$ time if all the terminals lie on a fixed number h of face boundaries [Sch1,Sch2].

In this paper we present a parallel algorithm for finding a Steiner forest in a planar graph G in which all terminals are located on the outer boundary. Fig. 1 depicts a planar graph G and a Steiner forest, where all the terminals of ten nets lie on the outer boundary, and a Steiner forest of ten disjoint trees is drawn in thick lines. Our algorithm runs in $O(\log^2 n)$ time and uses $O(n^3 / \log n)$ processors on a CREW PRAM model. The algorithm can be extended for the case all the terminals of each net lie on one of h specified face boundaries. For that case we can find a Steiner forest in a poly-log time using a polynomial number of processors if h is constant. Furthermore, using similar algorithms, we can find a Steiner forest in a planar graph in which all terminals lie on two face boundaries in $O(\log^2 n)$ time using $O(n^6 / \log n)$ processors, and also find a maximum number of internally disjoint paths between two specified vertices in a planar graph in the same time using a polynomial number of processors.

2. Tight Steiner Forest

Let $G = (V, E)$ be an undirected planar graph with vertex set V and edge set E. We sometimes write $V = V(G)$. Let n be the number of vertices in G, that is, $n = |V|$. We assume that G is connected and embedded in the plane. We denote by B the outer boundary of G. For two graphs G and G', $G + G'$ means a graph $(V(G) \cup V(G'), E(G) \cup E(G'))$. A set of vertices on the outer boundary B of G are designated as *terminals*. A *net* is a set of terminals that are all to be interconnected. 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 $F = T_1 + T_2 + \cdots + T_k$ in G such that $N_i \subseteq V(T_i)$ for each tree T_i in F. For simplicity we often call F a *forest* of $\mathcal{N}$. Hereafter we assume that there exists a Steiner forest in a given network $\mathcal{N}$. In this section and the next section, we assume that the outer boundary of G is a simple cycle, and that every vertex on the outer boundary is designated as a terminal(see Fig. 2). We will show later in Section 4 that this assumption does not lose any generality.

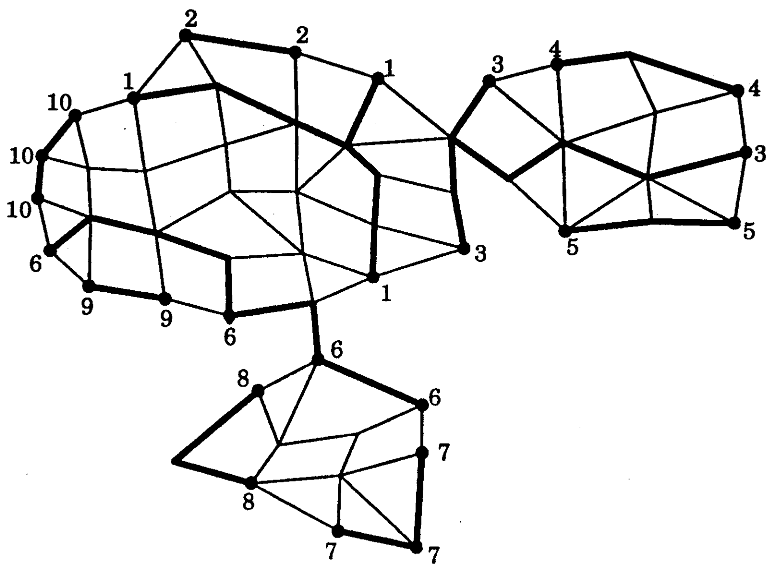

Fig. 1 A network and a Steiner forest.

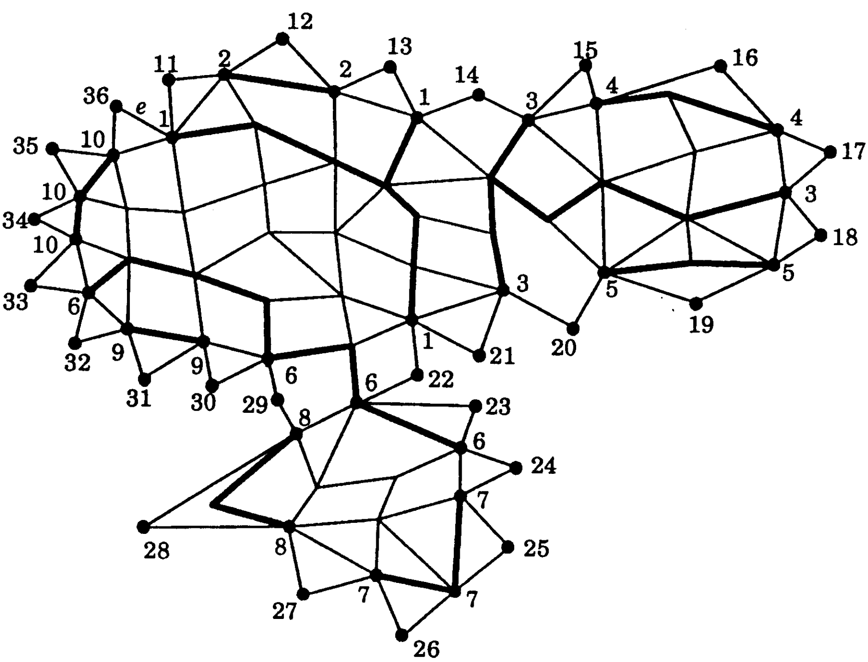

Fig. 2 A network after applying step (1) in Section 4 to the network in Fig. 1.

Let $e = (v_b, v_0)$ be an arbitrary edge on the outer boundary of G, and let the vertices $v_0, v_1, v_2, \cdots, v_b$ appear on the outer boundary clockwise in this order. For each vertex v on the outer boundary, $\mathrm{index}(v)$ denotes the index of v, that is, $\mathrm{index}(v) = i$ if $v = v_i$. Informally a Steiner forest F of $\mathcal{N}$ is *tight* if F is compacted as far away from e as possible. We now formally define a tight forest below.

Let $N_i, N_j \in S$. The *starting terminal* $s(N_i)$ of a net N_i is the terminal of N_i appearing first on the outer boundary clockwise going from v_0, while the *end terminal* $t(N_i)$ of N_i is the terminal appearing last. If $\mathrm{index}(s(N_j)) < \mathrm{index}(s(N_i)) \leq \mathrm{index}(t(N_i)) < \mathrm{index}(t(N_j))$, then we write $N_i \prec N_j$, and N_j is called an *ancestor* of N_i, and N_i is called a *descendant* of N_j. Note that either $\mathrm{index}(s(N_j)) < \mathrm{index}(s(N_i)) \leq \mathrm{index}(t(N_i)) < \mathrm{index}(t(N_j))$ or $\mathrm{index}(s(N_j)) < \mathrm{index}(t(N_j)) < \mathrm{index}(s(N_i)) < \mathrm{index}(t(N_i))$ holds if $\mathrm{index}(s(N_j)) < \mathrm{index}(s(N_i))$ and $\mathcal{N}$ has a Steriner forest. The relation $\prec$ is a partial order. We also use symbols $\preceq$ and $\succ$. If N_i has one or more ancestors, then there exists its minimum (youngest) ancestor, which is called the *parent* of N_i. N_j is a *child* of N_i if N_i is the parent of N_j.

Let F be a Steiner forest of $\mathcal{N}$. If, for every net N_i and every vertex $v \in (V(G) - V(B)) \cap V(T_i)$, there is a face whose boundary contains both vertex v and a vertex on the tree of a child of N_i, then F is called *tight for edge e* or simply *tight*. One can easily prove the following lemma.

LEMMA 1. If a network $\mathcal{N} = (G, S)$ has a Steiner forest, then also has a tight Steiner forest for any edge e on the outer boundary of G. ■

Suzuki, Akama and Nishizeki obtained a linear-time sequential algorithm for finding a Steiner forest of network $\mathcal{N} = (G, S)$ [SAN]. The algorithm finds a Steiner forest by repeating the following steps (1) and (2):

(1) for a net N_i which has no child, find the walk on the outer boundary of G going clockwise from $s(N_i)$ to $t(N_i)$, and let T_i be a spanning tree in the walk;

(2) remove the walk from G.

One can easily implement the algorithm above to run in linear time. However, it seems that the algorithm above cannot be easily transformed into a poly-log algorithm on a PRAM. In the algorithm shown below, we find all trees T_i in parallel using a shortest path algorithm.

3. Lemmas

For two vertices u and u' in G, a *vf-path* between u and v is a sequence of vertices $w_0 w_1 \cdots w_m$ such that $w_0 = u$, $w_m = u'$ and w_i, and w_{i+1} lie on the same finite face boundary for every i, $0 \leq i < m$. We define the *length* a vf-path $w_0 w_1 \cdots w_m$ to be m, and define the *distance* $\mathrm{d}(u, u')$ between two vertices u and u' to be the minimum length of vf-paths between u and u'.

For every terminal v in a net N_j, we denote by $\mathrm{anc}(v, i)$ the ith ancestor of the net N_j, that is, if N_j has l ancestors,

$$\mathrm{anc}(v,i) = \begin{cases} N_j, & i = 0; \\ \text{the parent of } \mathrm{anc}(v, i-1), & 1 \le i < l; \\ \infty, & l \le i. \end{cases}$$

For every vertex $v \in V(G)$, define

$$\mathrm{F}(v) = \mathrm{MIN}\{\mathrm{anc}(u, \mathrm{d}(u, v)) \mid u \in V(B)\},$$

where $N \prec \infty$ for every net $N \in S$ and $\mathrm{MIN}\{N, N'\} = N$ if $N \preceq N'$.

The following two lemmas guarantee that $\mathrm{F}(v)$ is well-defined and induces a Steiner forest of $\mathcal{N}$.

LEMMA 2. If $\mathcal{N} = (G, S)$ has a Steiner forest, then the set $C(v) = \{\mathrm{anc}(u, d(u, v)) \mid u \in V(B)\}$ for every vertex $v \in V$ is a totally ordered set on the relation $\preceq$, that is, either $\mathrm{anc}(u, \mathrm{d}(u, v)) \preceq \mathrm{anc}(u', \mathrm{d}(u', v))$ or $\mathrm{anc}(u', \mathrm{d}(u', v)) \preceq \mathrm{anc}(u, \mathrm{d}(u, v))$ holds for every two vertices u and u' on the outer boundary of G.

PROOF. Suppose for a contradiction that, for a pair of distinct nets $N_i, N_j \in C(v)$, both $N_i \not\preceq N_j$ and $N_j \not\preceq N_i$ hold. Assume that $N_i = \mathrm{anc}(u, \mathrm{d}(u, v))$ and $N_j = \mathrm{anc}(u', \mathrm{d}(u', v))$. Let R be a vf-path between u and v with length $\mathrm{d}(u, v)$, and R' be a vf-path between u and v with length $\mathrm{d}(u', v)$. Let R^* be a path connecting u and u' in $R + R'$. Then the length of R^* is at most $\mathrm{d}(u, v) + \mathrm{d}(u', v)$, that is, $|V(R^*)| \le d(u, v) + d(u', v) + 1$.

On the other hand, each net $N_l \in \{\mathrm{anc}(u, i) \mid 0 \le i \le \mathrm{d}(u, v)\} \cup \{\mathrm{anc}(u', i) \mid 0 \le i \le \mathrm{d}(u', v)\}$ is separated by the path R^*, that is, any tree connecting N_l must occupy a vertex on R^*. Since $\{\mathrm{anc}(u, i) \mid 0 \le i \le \mathrm{d}(u, v)\} \cap \{\mathrm{anc}(u', i) \mid 0 \le i \le \mathrm{d}(u', v)\} = \phi$, the vertices on R^* must be occupied by $\mathrm{d}(u, v) + \mathrm{d}(u', v) + 2$ different trees, a contradiction. ■

LEMMA 3. For every net N_i, a component of the induced subgraph by $V_i = \{v \in V \mid \mathrm{F}(v) = N_i\}$ contains all the terminals in N_i.

PROOF. Let F_t be a tight Steiner forest of $\mathcal{N}$ for edge e. Let T_i be the tree of F_t connecting N_i, and let v be a vertex on T_i. Then we may prove that $\mathrm{F}(v) = N_i$. Assume that, for every descendant N_j of N_i, $\mathrm{F}(w) = N_j$ for every vertex w on the tree T_j in F_t.

We first prove that $\mathrm{F}(v) \preceq N_i$. From the definition of anc, we have $\mathrm{F}(v) \preceq \mathrm{anc}(v, \mathrm{d}(v, v)) = \mathrm{anc}(v, 0) = N_i$, if $v \in V(B)$. On the other hand, if $v \notin V(B)$, then there is a face whose boundary contains both v and a vertex w on the tree of a child N_j of N_i. Since there is a vertex u on the outer boundary such that $\mathrm{anc}(u, \mathrm{d}(u, w)) = N_j$ and since $\mathrm{d}(u, v) \le \mathrm{d}(u, w) + 1$, $\mathrm{F}(v) \preceq \mathrm{anc}(u, \mathrm{d}(u, v)) \preceq N_i$. Therefore $F(v) \preceq N_i$.

We may assume that $\mathrm{F}(v) \prec N_i$. Let $N_j = F(v)$, let u' be a vertex on the outer boundary such that $\mathrm{anc}(u', \mathrm{d}(u', v)) = N_j$, and let Q be a vf-path between u' and v with length $\mathrm{d}(u', v)$. Since u' is a terminal of a descendant of N_j, $Q - \{v\}$ intersects the tree T_j of N_j at a vertex w. From the assumption of the lemma, we have $\mathrm{F}(w) = N_j$. Therefore $\mathrm{F}(v) \succ N_j$, a contradiction. ■

4. Algorithm and Complexity

The input to our problem is an embedding list of the given graph G together with a net set S. We find a Steiner forest of a network $\mathcal{N} = (G, S)$ as follows:

(1) modify the given graph G so that the outer boundary is a simple cycle and every vertices on the outer boundary is designated as a terminal, and compute $\mathrm{index}(v)$ for every vertex v on the outer boundary;

(2) compute $\mathrm{anc}(v, i)$ for every vertex v on G and every integer i, $0 \leq i \leq n$;

(3) compute $\mathrm{d}(u, v)$ for every vertex pair u and v;

(4) compute $\mathrm{F}(v)$ for every vertex v on G; and

(5) construct the subgraph $G_s = (V, \{(u, v) \in E | \mathrm{F}(u) = \mathrm{F}(v)\})$, find a spanning forest F, and remove from F the trees having no terminals.

We now show the detail of the above steps below.

(1) We first rank the edges of the outer boundary in clockwise order. For each terminal v, assign to v the minimum rank of the edges each of which joins t and a clockwise next vertex on the outer boundary. Assume that there are m terminals in $\mathcal{N}$ and let $r_1, r_2, \ldots, r_m$ be the ranks of them listed in increasing order. For each two terminals v and v' with ranks r_i and r_{i+1}, $1 \leq i \leq m$, connect them by two series edges, designate the common vertex of the two edges as a terminal, and add to net set S a new net consisting of only the terminal(see Fig. 2). Where $r_{m+1} = r_1$. Finally compute $\mathrm{index}(v)$ for every vertex v on the outer boundary. This step can be done in $O(\log n)$ time using $O(n)$ processors.

(2) For each net N_i, compute $s(N_i)$ and $t(N_i)$, and compute the parent $p(N_i)$ of N_i as follows. Initially assign to $p(N_i)$ the net N_j which contains the terminal v with $\mathrm{index}(v) = \mathrm{index}(t(N_i)) + 1$ if $\mathrm{index}(t(N_i)) < b$ and set $p(N_i) = \infty$ if $\mathrm{index}(t(N_i)) = b$, where b is the number of vertices on the outer boundary B (or the number of terminals in S). Then repeat the following procedure $O(\log n)$ times:

for each net N_i **in parallel do**
  if $p(N_i) \not\succ N_i$ **then** $p(N_i) := p(p(N_i))$.

Then one can easily compute $\mathrm{anc}(v, i)$, for every vertex v on the outer boundary and every integer i, $0 \leq i < n$. This step can be done in $O(\log n)$ time using $O(n^2)$ processors.

(3) For every face of G, find all the vertices on the boundary of the face. Then one can compute $\mathrm{d}(u, v)$ for all pairs of vertices u and v in G by using an algorithm for finding shortest paths between all pairs of vertices. The algorithm runs in $O(\log^2 n)$ time using $O(n^3 / \log n)$ processors [GR].

(4) Using the result of (2) and (3) we can compute $\mathrm{F}(v)$ for every vertex $v \in V$ in $O(\log n)$ time using $O(n^2)$ processors.

(5) One can construct G_s in $O(\log n)$ time using $O(n)$ processors. Furthermore one can find a spanning forest of G_s in $O(\log^2 n)$ time using $O(n^2 / \log^2 n)$ processors [CLC].

Thus we have the following theorem.

THEOREM 1. A Steiner forest in a planar graph can be found in $O(\log^2 n)$ time using $O(n^3/\log n)$ processors on a CREW PRAM, if all the terminals lie on the outer face boundary. ■

5. Some Extensions

1. Suppose that all the terminals of every net are located on one of the h face boundaries $B_1, B_2, \ldots, B_h$. Then we can find a Steiner forest in such a network $\mathcal{N} = (G, S)$ by using the algorithm above as follows:

let $S_i \subset S$ be the net set on B_i for every face boundary B_i;
for each permutation $B_{i_1}, B_{i_2}, \ldots, B_{i_h}$ of the h face boundaries and each $e_{i_1} \in B_{i_1}, e_{i_2} \in B_{i_2}, \ldots, e_{i_{h-1}} \in B_{i_{h-1}}$ **in parallel do**
  begin
    for each boundary B_{i_j}, $1 \leq j \leq h-1$ **do**
      begin
        find a tight Steiner forest F_{i_j} for edge e_{i_j} in network $\mathcal{N}_{i_j} = (G, S_{i_j})$;
        remove the forest F_{i_j} from the graph G, that is, $G := G - F_{i_j}$
      end;
    find a tight Steiner forest F_{i_h} for an arbitrary edge $e_{i_h} \in B_{i_h}$ in network $\mathcal{N}_{i_h} = (G, S_{i_h})$;
    check whether $F = F_1 + F_2 + \cdots + F_h$ is a Steiner forest of $\mathcal{N}$ or not
  end.

Therefore, a Steiner forest of $\mathcal{N}$ can be found in $O(h \log^2 n)$ time using $O(n^{h+2}/\log n)$ processors, and in poly-log time if h is constant.

2. Suppose that two face boundaries B_1 and B_2 are specified and every net consists of two terminals, one on B_1 and the other on B_2. Then we can find a Steiner forest (disjoint paths) in such a network in $O(\log^2 n)$ time and $O(n^6/\log n)$ processors by using an algorithm which is similar to the algorithm presented in Section 4. Using the algorithm above for finding disjoint paths, we can find a Steiner forest in a planar network $\mathcal{N}$ in which all terminals lie on two face boundaries in $O(\log^2 n)$ time using $O(n^6/\log n)$ processors. Note that $\mathcal{N}$ may have a net intersecting with both boundaries B_1 and B_2. Furthermore using the disjoint path algorithm we can also find a maximum number of internally disjoint paths between two specified vertices in a planar graph in $O(\log^2 n)$ time using a polynomial number of processors.

Acknowledgment

We would like to thank Professors N. Saito for fruitful discussions.

References

[CLC] F. Y. Chin, J. Lam and I. Chen, Efficient parallel algorithms for some graph problems, Communications of the ACM 25, 9(1982).

[GR] A. Gibbons and W. Rytter, Efficient Parallel Algorithms, Cambridge University Press, Cambridge(1988).

[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 SIGDA Newsletter 5:3, pp. 31-65 (1975).

[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).

[Sch1] A. Schrijver, Disjoint homotopic trees in a planar graph, manuscript(1988).

[Scha] A. Schrijver, personal communication, 1988.

[SAN] H. Suzuki, T. Akama and T. Nishizeki, Finding Steiner forests in planar graphs, Proc. 1st ACM-SIAM Symp. on Discrete Algorithms, pp. 444-453(1990).