VLSI DESIGN
1996, Vol. 4, No. 1, pp. 53-57
Reprints available directly from the publisher
Photocopying permitted by license only

Minimum-Cost Node-Disjoint Steiner Trees in Series-Parallel Networks

SUNIL CHOPRA
MEDS, J. L. Kellogg Graduate School of Management, Northwestern University, Evanston, IL 60208

KALYAN T. TALLURI
USAir, Operations Research Department, 2345 Crystal Drive, Arlington, VA 22227

(Received February 22, 1994, Revised February 3, 1995)

The routing problem in VLSI-layout can be modeled as a problem of packing node-disjoint Steiner trees in a graph. The problem is as follows: Given an undirected network $G = (V, E)$ and a net list $\Psi = \{N_i, i = 1, \ldots, r\}$, a family $\Gamma_G = \{T_{N_i} = (V_{N_i}, E_{N_i}), i = 1, \ldots, r\}$ is a node-disjoint family of Steiner trees spanning Ψ if T_{N_i} is a Steiner tree spanning N_i for $i = 1, \ldots, r$ and $V_{N_i} \cap V_{N_j} =$ for $i \neq j$. The edge-disjoint version of this problem is known to be NP-hard for series-parallel graphs (see Richey and Parker [5]). In this paper we give a $O(n^5)$ algorithm for finding a minimum-cost node-disjoint family of Steiner trees in series-parallel networks. Our algorithm can be extended to k-trees and is polynomial for fixed k.

Key Words: *Network Design, VLSI, Steiner Trees, Series-Parallel*

1 INTRODUCTION

The routing problem in VLSI-layout can be modeled as a problem of packing node-disjoint Steiner trees in a graph (Korte, et al. [4]). Given an undirected network $G = (V,E)$ and a *terminal set* (also referred to as a *net*) $N \subseteq V$, a tree $T_N = (V_N, E_N)$ is said to be a *Steiner tree* if it spans every node in N. Let $\Psi = \{N_i, i = 1, \ldots, r\}$, be a *family of nets* (or a *net list*) where $N_i \subseteq V$, $i = 1, \ldots, r$, and $|N_i \cap N_j| = 0$ for $i \neq j$. Assume edge costs $c_e \geq 0$ for $e \in E$ and node costs $n_v \geq 0$ for $v \in V$. Given a Steiner tree $T_N = (V_N, E_N)$, define its cost $c_{T_N} = \Sigma_{e \in E_N} c_e + \Sigma_{v \in V_N} n_v$. Given $G = (V, E)$ and a net list $\Psi = \{N_i, i = 1, \ldots, r\}$, a family $\Gamma_G = \{T_{N_i} = (V_{N_i}, E_{N_i}), i = 1, \ldots, r\}$, is a node-disjoint family of Steiner trees spanning Ψ, if T_{N_i} is a Steiner tree spanning N_i, $i = 1, \ldots, r$, and $|V_{N_i} \cap V_{N_j}| = 0$ for $i \neq j$. The cost of a family Γ_G is the sum of the costs of the trees in the family. In this paper, we consider the minimum-cost node-disjoint Steiner tree problem (MCNDST): given a network $G = (V, E)$ with non-negative edge and node costs and a net list Ψ, find a minimum-cost family of node-disjoint Steiner trees.

This problem is known to be NP-hard, since the Steiner tree problem is a special case where the net list is a single net. In general, the feasibility problem (NDST), i.e., the problem of checking the existence of a node-disjoint family of Steiner trees is also NP-hard.

A related problem that has been studied is one of finding a minimum-cost edge-disjoint family of Steiner trees (MCEDST). MCEDST and the feasibility version (EDST) are NP-hard. Richey and Parker [5] have shown MCEDST to be NP-hard even on series-parallel graphs. In this paper, in contrast, we give a polynomial-time algorithm to solve both NDST and MCNDST on series-parallel graphs with no restrictions on the net list N. *Series-parallel* graphs (see Duffin [2]) can be constructed from a single edge by a sequence of series and parallel operations. In a *series* operation we replace an existing edge by a path of length two. In a *parallel* operation, we add a new edge in parallel to an existing edge. For applications where series-parallel graphs arise, see Wald and Colbourn [6].

In Section 2 we state the algorithm on series-parallel graphs and analyze its complexity. Section 3 generalizes the results of Section 2 to obtain an algorithm to solve

MCNDST on partial k-trees that is polynomial for a fixed value of k. The paper is structured in this manner since the results are easy to follow on series-parallel graphs and generalize naturally to k-trees.

2 MCNDST ON SERIES-PARALLEL GRAPHS

In this section we give a polynomial-time algorithm ($O(n^5)$) to solve MCNDST (and thus NDST) for series-parallel graphs. We impose no restrictions on the net list N.

2.1 Decomposition of Series-Parallel Graphs

Given a graph $G = (V, E)$, $S \subseteq V$ is said to be a $|S|$-*node cutset* if the graph $G \setminus S$ is not connected. A graph is said to be *k-connected* if there is no m-node cutset for $m < k$. From results of Duffin [2] it follows that no series-parallel graph can be 3-connected.

Let $G = (V, E)$ be any 2-connected graph with a 2-node cutset $C = \{u, v\}$. $G \setminus C$ has at least two connected components. We can partition the nodes of $G \setminus C$ into two sets $\tilde{V}_1$ and $\tilde{V}_2$ where there is no edge in $G \setminus C$ between $\tilde{V}_1$ and $\tilde{V}_2$. $\tilde{V}_1$ and $\tilde{V}_2$ may not be uniquely defined if $G \setminus C$ has more than two connected components. Define $V_i = \tilde{V}_i \cup C$ and let $G_i = (V_i, E_i)$ be the subgraph of G induced by nodes in V_i, $i = 1, 2$. Theorem 2.1 below follows from the results in Gilbert, Rose and Edenbrandt [3]. It is stated here without proof since the proof is similar to that given in Gilbert et al. [3].

Theorem 2.1 *Let $G = (V, E)$ be a 2-connected series-parallel graph with $|V| \geq 4$. There exists a 2-node cutset $\{u, v\}$ such that $|V_i| \geq |V|/3 + 1$, $i = 1, 2$, where $G_i = (V_i, E_i)$, $i = 1, 2$, are as defined above.*

Theorem 2.1 is tight in the sense that one can find series-parallel graphs for which either $|V_1|$ or $|V_2|$ is equal to $|V|/3 + 1$ for all 2-node cutsets.

A *2-tree* can be defined recursively as follows. A triangle is a 2-tree. Given a 2-tree and an edge (u, v) of the 2-tree, we can add a new vertex z adjacent to both u and v to obtain a new 2-tree. Wald and Colbourn [6] show that series-parallel graphs are subgraphs of 2-trees. They also give a $O(|V|)$ algorithm to complete a series-parallel graph to a 2-tree. Using this algorithm and a modified version of the algorithm in Gilbert et al. [3], one can find an $O(|V|)$ algorithm to identify a 2-node cutset that satisfies Theorem 2.1.

2.2 Polynomial-Time Algorithm

In this section we present a polynomial-time algorithm for finding minimum-cost node-disjoint Steiner trees in a given series-parallel network. The idea is to find a two node cutset as in Theorem 2.1 such that each of the networks has at least $|V|/3 + 1$ nodes. The problem is then solved on each of the smaller networks. The solutions are pieced back together to obtain a feasible solution for the original network. First, we describe the piecing together operation.

Let $C = \{v_j, j = 1, \ldots, k\} \subseteq V$, $k \leq 2$, be a cutset that separates the series-parallel graph G into two components $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ (cf. Theorem 2.1). For $i = 1, 2$, let c^i and n^i represent edge and node costs for G_i respectively. Let

$$\nu_C = |\{N_i : |N_i \cap V_1| \geq 1, |N_i \cap V_2| \geq 1\}|$$

represent the number of nets separated by C. If $\nu_C > |C|$, there exists no feasible packing of node-disjoint Steiner trees.

If $|C| = 1$ and $\nu_C = 1$, we can solve the problem on G_1 and G_2, fixing the cut-node as a terminal belonging to the net that it separates. We divide the remaining possibilities into four cases.

(a) $|C| = 1$ and $\nu_C = 0$;
(b) $|C| = 2$ and $\nu_C = 2$;
(c) $|C| = 2$ and $\nu_C = 1$;
(d) $|C| = 2$ and $\nu_C = 0$.

If $|C| = 1$, let $C = \{v\}$ and if $|C| = 2$, let $C = \{u, v\}$. Each of the above cases is resolved separately. Let M be a very large number ($M = 1 + \Sigma_{e \in E} C_e + \Sigma_{v \in V} n_v$ is large enough). In the following proofs we assign a node weight of M to a node instead of deleting it. This is done primarily for the sake of consistency and to simplify the notation in the proofs.

Case (a): $|C| = 1$ and $\nu_C = 0$.

Let $|V_1| \leq |V_2|$. Let Γ'_{G_1} be the solution to MCNDST(G_1) with all the node and edge costs same as that of G and let $\Gamma^0_{G_1}$ be the solution to MCNDST(G_1) with all costs the same, except $n^1_v = M$. In G_2, let $n^2_v = n_v + c(\Gamma^0_{G_1}) - c(\Gamma'_{G_1})$ and all other costs same as in G. Let Γ_{G_2} be the solution for MCNDST(G_2). Combine the solutions as follows: if $v \in \Gamma_{G_2}$, then $\Gamma_G = \Gamma_{G_2} + \Gamma^0_{G_1}$ and if $v \notin \Gamma_{G_2}$, then $\Gamma_G = \Gamma_{G_2} + \Gamma'_{G_1}$.

Case (b): $|C| = 2$ and $\nu_C = 2$.

Let N_1, N_2 be the two sets separated by C. Solve MCNDST(G_i), $i = 1, 2$, for the following two instances.

(i) Set u in N_1, v in N_2 in both G_1 and G_2. $\Gamma^1_{G_1}$ and $\Gamma^1_{G_2}$ are the respective solutions.

(ii) Set u in N_2, v in N_1 in both G_1 and G_2. $\Gamma^2_{G_1}$ and $\Gamma^2_{G_2}$ are the respective solutions.

For $i = 1, 2$, let $\Gamma^i_G = \Gamma^i_{G_1} + \Gamma^i_{G_2}$ corresponding to the two instances above. If an assignment is infeasible (for example, (i) is infeasible if $u \in N_2$ in net list) the corresponding case is dropped.

Case (c): $|C| = 2$ and $v_c = 1$.

Let N_1 be the net separated by C. Solve MCNDST(G_i), $i = 1, 2$, for each of the following six instances.

(i) In G_1 set u in N_1, $n^1_v = M$, and in G_2, set u in N_1 $n^2_v = n_v$.
(ii) In G_1 set u in N_1, $n^1_v = n_v$, and in G_2, set u in N_1 $n^2_v = M$.
(iii) In G_1 set v in N_1, $n^1_u = M$, and in G_2, set v in N_1, $n^2_u = n_u$.
(iv) In G_1 set u in N_1, $n^1_u = n_u$, and in G_2, set v in N_1, $n^2_u = M$.
(v) In G_1 set u and v in N_1. In G_2, identify u and v into a single node w with $w \in N_1$.
(vi) Same as in case (v) with roles of G_1 and G_2 reversed.

All other costs for G_1 and G_2 are the same as in G. $\Gamma^i_{G_1}$ and $\Gamma^i_{G_2}$, $i = 1, \ldots, 6$, are the solutions obtained in the six cases above. For $i = 1, \ldots, 6$, define $\Gamma^i_G = \Gamma^i_{G_1} + \Gamma^i_{G_2}$.

Case (d): $|C| = 2$ and $v_C = 0$.

Again, except when specified, all costs for G_1 and G_2 will be the same as that of G.

(i) In G_1, identify u and v into a single node w with $n_w = n_u + n_v$. In G_2, the nodes u and v form a new net.
(ii) Same as in case (i) with roles of G_1 and G_2 reversed.
(iii) In G_1, let $n^1_u = M$, $n^1_v = n_v$ and in G_2, let $n^2_u = n_u$, $n^2_v = M$.
(iv) In G_1, let $n^1_u = n_u$, $n^1_v = M$ and in G_2, let $n^2_u = M$, $n^2_v = n_v$.
(v) In G_1, let $n^1_u = M = n^1_v$ and in G_2, let $n^2_u = n_u$, $n^2_v = n_v$.
(vi) In G_1, let $n^1_u = n_u$, $n^1_v = n_v$ and in G_2, let $n^2_u = M = n^2_v$.

$\Gamma^i_{G_1}$ and $\Gamma^i_{G_2}$, $i = 1, \ldots, 6$, are the solutions obtained in the six cases above.

For $i = 1, \ldots, 6$, define $\Gamma^i_G = \Gamma^i_{G_1} + \Gamma^i_{G_2}$.

We will show below that the optimal solution Γ_G will be the minimum of Γ^i_G, for $i = 1,2$ for case (b) and for $i = 1, \ldots, 6$, for cases (c) and (d).

Theorem 2.2 *For a cutset C, $|C| \leq 2$, one of the following holds*:

(a) If $|C| = 1$, $v_C = 0$ then Γ_G of case (a) is an optimal solution to G.

If $|C| = 2$ and

(b) $v_C = 2$, $\Gamma_G = \arg\min_{i=1,2} \{c(\Gamma^i_G)\}$ of case (b), is an optimal solution to G,

(c) $v_C = 1$, $\Gamma_G = \arg\min_{i=1,6} \{c(\Gamma^i_G)\}$ of case (c) is an optimal solution to G,

(d) $v_C = 0$, $\Gamma_G = \arg\min_{i=1,6} \{c(\Gamma^i_G)\}$ of case (d) is an optimal solution to G.

Proof: We prove cases (a) and (d) since (b) and (c) are similar to (d).

Proof of (a): Let Γ^*_G be an optimal solution to G with $\Gamma^*_{G_1}$ and $\Gamma^*_{G_2}$ its restriction to G_1 and G_2, respectively. There are two possible cases: (A) $v \notin \Gamma^*_G$, and (B) $v \in \Gamma^*_G$, which we further split into subcases (B_1) through (B_4).

(A) If $v \notin \Gamma^*_G$, $c(\Gamma^*_{G_1}) = c(\Gamma^0_{G_1})$ and since $d = c(\Gamma^0_{G_1}) \geq 0$, $\Gamma^*_{G_2}$ is also an optimal solution to G_2 with $n^2_v = n_v + d$. Thus $c(\Gamma^*_{G_2}) = c(\Gamma_{G_2})$. Therefore $c(\Gamma_G) = c(\Gamma^*_G)$ and Γ_G is an optimal solution.

(B_1) Now consider the case $v \in \Gamma^*_G$, $v \in \Gamma^*_{G_2}$ and $v \in \Gamma_{G_2}$. In this case $c(\Gamma_{G_2}) = c(\Gamma^*_{G_2})$ and $c(\Gamma^0_{G_1}) = c(\Gamma^*_{G_1})$ since $v \notin \Gamma^0_{G_1}$ and $v \notin \Gamma^*_{G_1}$. Therefore $c(\Gamma_G) = c(\Gamma^0_{G_1} + \Gamma_{G_2}) = c(\Gamma^0_{G_1}) + c(\Gamma_{G_2}) = c(\Gamma^*_{G_1}) + c(\Gamma^*_{G_2}) = c(\Gamma^*_G)$. Thus, Γ_G is also an optimal solution.

(B_2) $v \in \Gamma^*_G$, $v \in \Gamma^*_{G_2}$ but $v \notin \Gamma_{G_2}$. Let $d = c(\Gamma^0_{G_1}) - c(\Gamma'_{G_1})$. We have,

$$c(\Gamma_{G_2}) \leq c(\Gamma^*_{G_2}) + d. \tag{1}$$

Note that $\Gamma_G = \Gamma_{G_2} + \Gamma'_{G_1}$ and $c(\Gamma^*_{G_1}) = c(\Gamma^0_{G_1})$. The inequality (1) thus implies $c(\Gamma_{G_2}) \leq c(\Gamma^*_{G_2}) + d = c(\Gamma^*_{G_2}) + c(\Gamma^0_{G_1}) - c(\Gamma'_{G_1}) = c(\Gamma^*_{G_2}) + c(\Gamma^*_{G_1}) - c(\Gamma'_{G_1})$.

Thus, $c(\Gamma_G) \leq c(\Gamma^*_G)$ and Γ_G is an optimal solution to G.

(B_3) If $v \in \Gamma^*_{G_1}$ and $v \in \Gamma_{G_2}$, then $c(\Gamma_{G_2}) + d \leq c(\Gamma^*_{G_2})$ and since $c(\Gamma^*_{G_1}) = c(\Gamma'_{G_1})$, we have

$$c(\Gamma_{G_2}) + c(\Gamma^0_{G_1}) - c(\Gamma^*_{G_1}) \leq c(\Gamma^*_{G_2}).$$

Thus $c(\Gamma_G) \leq c(\Gamma^*_{G_1})$, proving that Γ_G is an optimal solution to G.

(B_4) If $v \in \Gamma_{G\dagger}$ and $v \notin \Gamma_{G_2}$, we have $c(\Gamma'_{G_1}) \leq c(\Gamma^*_{G_1})$ and $c(\Gamma_{G_2}) \leq c(\Gamma^*_{G_2})$. This proves that $c(\Gamma_G) \leq c(\Gamma^*_{G_1})$; i.e., Γ_G is an optimal solution.

Proof of (d): Let Γ^*_G be an optimal solution to G and $\Gamma^*_{G_1}$ and $\Gamma^*_{G_2}$ its restriction to G_1 and G_2, respectively. Depending on the presence of u and v in $\Gamma^*_{G_1}$, we will show that one of Γ^i_G for $i = 1, \ldots, 6$, should have the same cost as $\Gamma^*_{G_2}$. There are five possible cases, (A), (B), (C), (D) and (E), with case (E) further subdivided into eight cases, (E_1) through (E_8).

(A) $u, v \in \Gamma^*_{G_1}$. Then $c(\Gamma^6_{G_2}) \leq c(\Gamma^*_{G_2})$ and $c(\Gamma^6_{G_1}) \leq c(\Gamma^*_{G_1})$. Thus, $c(\Gamma^6_G) \leq c(\Gamma^*_G)$.

(B) $u, v \in \Gamma^*_{G_2}$. Then $c(\Gamma^5_{G_2}) \leq c(\Gamma^*_{G_2})$ and $c(\Gamma^5_{G_1}) \leq c(\Gamma^*_{G_1})$, implying Then $c(\Gamma^5_G) \leq c(\Gamma^*_G)$.

(C) Exactly one of u, v (say u) belongs to Γ^*_G. Assume that $u \in \Gamma^*_{G_1}$ and $v \notin \Gamma^*_G$. Then $c(\Gamma^6_{G_1}) \leq c(\Gamma^*_{G_1})$ and therefore $c(\Gamma^6_G) \leq c(\Gamma^*_G)$ and Γ^6_G is an optimal solution.

(D) If u is spanned by a net belonging to G_2, consider Γ^5_G instead of Γ^6_G and proceed as in case (C).

(E) Both u and v belong to Γ_G

(E_1) If both u and v are spanned by a net belonging to G_1 and the path between u and v in the tree spanning that net is in G_2, then consider $\Gamma^1_{G_1}$. $\Gamma^*_{G_1} + (u, v)$, $\Gamma^*_{G_2}$ are feasible solutions to G_1 and G_2, respectively, and $c(\Gamma^1_{G_1}) \leq c(\Gamma^*_{G_1})$; $c(\Gamma^1_{G_2}) \leq c(\Gamma^*_{G_2})$. Thus $c(\Gamma^1_G) \leq c(\Gamma^*_G)$ and Γ^1_G is an optimal solution.

(E_2) If the path between u and v in Γ^*_G is in G_1 itself, consider Γ^6_G. $\Gamma^*_{G_1}$ and $\Gamma^*_{G_2}$ are feasible in G_1 and G_2. Hence, $c(\Gamma^6_G) \leq c(\Gamma^*_G)$ and Γ^6_G is an optimal solution.

($E_{3,4}$) For the symmetric case when u and v are spanned by a net of G_2, consider Γ^2_G and Γ^5_G. One of them is an optimal solution.

($E_{5,6}$) If u and v are spanned by two different nets in Γ^*_G but both belong to $G_1(G_2)$, consider Γ^6_G (Γ^5_G).

(E_7) If u is spanned by a net of G_1 and v by a net of G_2 in Γ^*_G, consider Γ^4_G. $\Gamma^*_{G_1}$ is feasible in G_1 and $\Gamma^*_{G_2}$ in G_2 for case (4). Hence, $c(\Gamma^4_G) \leq c(\Gamma^*_G)$ and Γ^4_G is an optimal solution.

(E_8) Consider Γ^3_G instead of Γ^4_G for the symmetric case when v is spanned by a net of G_1 and u by a net of G_2. □

Finally, we show that the above procedure leads to an $O(n^5)$ algorithm where $n = |V|$.

Proposition 2.3 *The algorithm described above runs in $O(n^5)$ steps.*

Proof: Let $R(n)$ represent the running time to solve the problem on a network with n nodes. We prove the result by induction on n. So assume $R(l) \leq l^5$ for all $l < n$.

Suppose G has a cut node which divides the graph into G_1 and G_2 with $n_1 = |V_1|$ and $n_2 = |V_2|$. Assume $3 \leq n_1 \leq n_2$. We have $n_1 + n_2 = n + 1$. The algorithm solves the problem twice on G_1 and once on G_2. Therefore,

$$\begin{aligned} R(n) &= 2R(n_1) + R(n_2) \\ &\leq 2n_1^5 + n_2^5 \\ &\leq (n_1 + n_2 - 1)^5 \text{ (since } 3 \leq n_1 \leq n^2) \\ &= n^5. \end{aligned}$$

Next, let C be a two-node cutset with $|V_1|, |V_2| \geq \lceil n/3 \rceil + 1$. The work involved in finding such a cutset is $\mathcal{O}(|V|)$ and is dominated by the rest of the algorithm, so we ignore it. The maximum amount of work involved is when case (d) occurs, so we restrict our analysis to this case. Here, $n_1, n_2 \geq n/3 + 1$ and

$$R(n) \leq 6R(n_1) + 6R(n_2).$$

Using the fact that $R(1) = R(2) = 1$, $R(3) = 2$, $R(4) = 4$, $R(5) = 5$, and $R(6) = 6$, one can verify that $R(n) \leq n^5$ for all $n \leq 40$. This can be done by writing a short program for calculating $R(n)$ using the recursion $R(n) = 2R(n_1) + R(n_2)$ with the initial values as given above. We omit giving the actual numbers that we obtained, but just state that the inequality $R(n) \leq n^5$ is true by enumeration for $n \leq 40$. Note that $n_1 = a + 1$ and $n_2 = b + 1$ where $a + b = n$. Thus, consider the problem

$$\begin{aligned} &\max \ (1 + a)^5 + (1 + b)^5 \\ &\text{s.t} \\ &\qquad a + b = n, \\ &\qquad a, b \geq n/3. \end{aligned}$$

Since the objective function is convex over a bounded polyhedral region, the maximum is attained at an extreme point, say, $a = n/3$, $b = 2n/3$. Thus

$$R(n) \leq 6(n/3 + 1)^5 + 6(2n/3 + 1)^5 \leq n^5$$

for $n \geq 40$. This proves the result. □

k-trees generalize 2-trees and can be inductively characterized as follows. (1) A complete graph on k nodes (referred to as a *k-clique*) is a k-tree. (2) If $G = (V, E)$ is a k-tree and $V' \subseteq V$ is a set of k nodes that induces a complete subgraph in G, then the graph obtained by adding a new vertex v to V together with an edge from v to every vertex in V' is also a k-tree. Note that the

vertices $V' \cup \{v\}$ induce a $(k+1)$-clique. $G = (V, E)$ is a *partial k-tree* if it is the subgraph of some k-tree $\overline{G} = (V, \overline{E})$. Arnborg and Proskurowski [1] have shown that, for any fixed k, one can in polynomial time determine whether an arbitrary graph G is a partial k-tree and construct a corresponding k-tree, although the complexity of the algorithm grows exponentially with k and the general problem of determining the minimal k such that G is a partial k-tree is NP-hard.

We would like to state here that our algorithm for MCNDST stated in this section extends naturally to k-trees and will be polynomial for a fixed k. We omit the details.

3 CONCLUSIONS

In this paper we have shown that the minimum-cost node-disjoint Steiner tree problem is polynomially solvable on series-parallel graphs. Our algorithm also generalizes to k-trees. This contrasts with the fact that the minimum-cost edge-disjoint Steiner tree problem is NP-hard on series-parallel graphs.

Acknowledgements

We thank S. Raghavan of MIT for bringing the paper of Richey and Parker [5] to our attention.

References

[1] Arnborg, S. and A. Proskurowski, "Linear Time Algorithm for NP-hard Problems for Graphs Embedded in k-trees". TRITA-NA-8404, Royal Institute of Technology, S-100 44 Stockholm, Sweden (1984).

[2] Duffin, R., "Topology of Series-Parallel Networks". *Journal of Mathematical Analysis and Applications*, **10**, 303-318 (1965).

[3] Gilbert, J. R., D. J. Rose and A. Edenbrandt, "A Separator Theorem for Chordal Graphs". *SIAM Journal of Algebraic and Discrete Methods*, **5(3)**, 306–313 (1984).

[4] Korte, B., H. J. Prömel and A. Steger, "Steiner Trees in VLSI-layout", in *Paths, Flows, and VLSI-Layout*, Korte, B., L. Lovász, H. J. Prömel and A. Schrijver (Eds.), Springer-Verlag (1990).

[5] Richey, M. B. and R. G. Parker, "On Multiple Steiner Subgraph Problems". *Networks*, **16**, 423–438 (1986).

[6] Wald, J. A. and C. J. Colbourn, "Steiner Trees, Partial 2-Trees, and Minimum IFI *Networks*, **13**, 159–167 (1983).

Biographies

SUNIL CHOPRA is Associate Professor at the Kellogg Graduate School of Management of Northwestern University in the MEDS Department. Prior to working at Kellogg, he has taught at New York University. He obtained his Ph.D from SUNY at Stony Brook. His interests are Network Design, Logistics and Operations Management.

KALYAN TALLURI works in the USAir Operations Research Department in Arlington, VA, as Senior Operations Research Analyst. He graduated from MIT in 1991. His research interests are Network Design and Transportation and Logistics problems.