論文 / 著書情報
Article / Book Information

Title	Cost-Radius Balanced Spanning/Steiner Trees
Authors	Hideki Mitsubayashi, Atsushi Takahashi, Yoji Kajitani
Citation	IEICE Trans. Fundamentals, Vol. E80-A, No. 4, pp. 689-694
Pub. date	1997, 4
URL	http://search.ieice.org/
Copyright	(c) 1997 Institute of Electronics, Information and Communication Engineers

PAPER Special Section on Discrete Mathematics and Its Applications

Cost-Radius Balanced Spanning/Steiner Trees

Hideki MITSUBAYASHI†, *Nonmember*, Atsushi TAKAHASHI†, *and* Yoji KAJITANI†, *Members*

SUMMARY The most crucial factor that degrades a high-speed VLSI is the signal propagation delay in a routing tree. It is estimated by the sum of the delay caused by the source-to-sink path length and by the total length. To design a routing tree in which these two are both small and balanced, we propose an algorithm to construct such a spanning tree, based on the idea of constructing a tree combining the minimum-spanning-tree and shortest-path-tree algorithms. This idea is extended to finding a rectilinear Steiner tree. Experiments are presented to illustrate how the source-to-sink path length and total length can be balanced and small.

key words: *delay, spanning tree, steiner tree, VLSI layout*

1. Introduction

In recent VLSIs, the signal propagation delay is one of the most critical factors which influence the performance of a circuit. To minimize this delay from the source terminal to more than one sink terminal of a net, many routing algorithms have been proposed [1]–[4], [6], [8]–[10].

In the Elmore delay model [11], the objective is to reduce both the total length of the routing tree and the path length from the source to each sink on the tree. The minimum-spanning-tree (MST) algorithm realizes the minimum total length but may generate very long source-to-sink paths. The shortest-path-tree (SPT) algorithm realizes the minimum source-to-sink path length but may generate excessive total length. The difficulty is in constructing a routing tree that has balanced small total length (*cost*) and source-to-sink path length (*radius*).

The rectilinear Steiner tree is a more practical routing tree used in conventional designs. Thus, the problem is extended to construct a rectilinear Steiner tree with balanced small cost and radius.

The MST or SPT can be obtained using polynomial time algorithms. Therefore, to find a cost-radius-balanced tree, there are alternative ways of constructing a *cost-bounded minimum-radius tree* or a *radius-bounded minimum-cost tree.* However, it is difficult even to find a feasible solution for the former, while it is easy for the latter. Hence, reasonable heuristics will be Prim based [3], Kruskal based [10], Dijkstra based or their combination [1], and the idea of the limit of radius will be introduced. Note that the problem of finding a minimum spanning tree or Steiner tree with a bounded path length is NP-hard [5], [7].

Alpert et al. [1] showed a trade-off between cost and radius. Pyo et al. [10] showed a Kruskal radius-bounded based algorithm (called BKRUS) which constructs lower-cost trees than previous algorithms under the radius-bounded condition.

Some heuristics proposed so far take the radius into consideration after the radius exceeds a certain value, and others follow the path-length-oriented strategy in all stages. As a consequence, the former tends to increase in cost in the later stage, and the latter may miss a lower-cost connection for a point merely because its path length is longer than the costlier connection, even if the former connection is not critical with respect to the radius. Therefore, a good algorithm should continuously take the radius into greater consideration according to the distance from the source to the chosen point.

By introducing a distance-dependent coefficient, we realize the above idea in the form of new heuristic algorithms which can be used to design a routing spanning/Steiner tree with balanced small cost and radius. These trees perform better than the existing methods with respect to cost and radius balance. An important feature is that the radius-limit can be achieved by setting a parameter to an appropriate value. This parameter also causes the algorithms to construct a variety of trees from which we can choose an appropriate one for minimizing the delay.

2. Definitions

A *signal net* $V = \{v_s, v_1, \ldots, v_n\}$ is a set of terminals to be connected on the *Manhattan (L1 metric) plane*, with v_s is for the *source* and $v_1, \ldots, v_n$ are for *sinks*. If $n \leqq 2$, every problem concerned is trivial. A *routing tree* $T = (V_T, E_T)$ is a spanning tree or a rectilinear Steiner tree connecting V. V_T consists of all the terminals and Steiner points (if the tree is a Steiner tree), and the edges in E_T connect those points in V_T.

$D(p_i, p_j)$ is the distance between two points p_i and p_j (terminals or Steiner points) on the plane. The length of an edge is defined as the distance between two end

Manuscript received September 11, 1996.
Manuscript revised November 15, 1996.

†The authors are with the Faculty of Engineering, Tokyo Institute of Technology, Tokyo, 152 Japan.

points. For the two points p_i and p_j on T, the length of the path connecting them on T is denoted by $D_T(p_i, p_j)$. The *cost* $W(T)$ is the sum of lengths of the edges of T, and the *radius* of T is $R(T) = max_{v \in V}(D_T(v_s, v))$.

Let $R_{max} = max_{v \in V}(D(v_s, v))$, the distance from the source to the farthest sink. If the radius limit is concerned, the limit should be set not less than R_{max}.

3. Spanning Tree Algorithm

The proposed algorithm called the cost-radius-balanced tree (CRBT) yields a spanning tree. CRBT begins with the tree consisting only of source v_s. The algorithm iteratively adds sink $v_k \notin V_T$ to V_T connecting $v_i \in V_T$ by an edge e_{ik}. The pair v_i and v_k is chosen to minimize the cost function:

$$H(v_i, v_k) = C(v_s, v_k, P) \cdot D_T(v_s, v_i) + D(v_i, v_k),$$

where $C(v_s, v_k, P) = D(v_s, v_k)/P$, with P ($\geqq R_{max}$) being an algorithm parameter.

As P is smaller than or the vertex v_k under consideration is more distant from v_s, C monotonically increases with increasing stress on the first term. In the extreme case when C is very small ($P = \infty$ or v_k is close to v_s), $H(v_i, v_k) \cong D(v_i, v_k)$. This increase of C is the increase of total length when v_k is added to T. Then, the CRBT continues like Prim's algorithm toward an MST. In contrast, if C approaches 1 ($D(v_s, v_k) \cong P$), $H(v_i, v_k)$ approximates the path length from v_s to v_k from the vertices on T. Then, the CRBT performs like Dijkstra's algorithm toward an SPT.

Algorithm CRBT

1. $V_T = \{v_s\}$, $E_T = \phi$.
2. Let $v_i \in V_T$ and $v_k \notin V_T$ be the pair which minimizes the cost function H.
3. $V_T = V_T \cup \{v_k\}$, $E_T = E_T \cup \{e_{ik}\}$.
4. If $V \neq V_T$, return to step 2.

Theorem 1: The CRBT algorithm yields a spanning tree T with $R(T) \leqq P$.

Proof: Clearly, the initial tree satisfies $R(T) \leqq P$. Assume that a tree T' under construction satisfies $R(T') \leqq P$, but tree T'', which is made by adding edge (v_i, v_k) to T' in the algorithm, does not.

Since $D_{T''}(v_s, v_k) > P$,

$$\begin{aligned} H(v_i, v_k) &= \frac{D(v_s, v_k)}{P} \cdot D_{T'}(v_s, v_i) + D(v_i, v_k) \\ &> \frac{D(v_s, v_k)}{D_{T''}(v_s, v_k)} \cdot D_{T'}(v_s, v_i) + D(v_i, v_k) \\ &= \frac{D(v_s, v_k)}{D_{T''}(v_s, v_k)} \\ &\quad \times \left(D_{T'}(v_s, v_i) + \frac{D_{T''}(v_s, v_k)}{D(v_s, v_k)} \cdot D(v_i, v_k) \right). \end{aligned}$$

Since $D_{T''}(v_s, v_k) > P \geqq D(v_s, v_k)$,

$$D_{T''}(v_s, v_k)/D(v_s, v_k) > 1,$$

and since $D_{T'}(v_s, v_i) + D(v_i, v_k) = D_{T''}(v_s, v_k)$, we have

$$\begin{aligned} &D_{T'}(v_s, v_k) + \frac{D_{T''}(v_s, v_k)}{D(v_s, v_k)} \cdot D(v_i, v_k) \\ &\quad > D_{T''}(v_s, v_k). \end{aligned}$$

Therefore, $H(v_i, v_k) > D(v_s, v_k)$. However $H(v_s, v_k) = D(v_s, v_k)$ by definition. This contradicts the assumptions that $H(v_i, v_k)$ is minimal and that the pair v_i and v_k is selected in the algorithm. □

The time complexity of the CRBT is $O(n^2)$, since the algorithm performs like the Prim- and Dijkstra-based algorithms.

4. Steiner Tree Algorithm

The proposed algorithm called the cost-radius-balanced Steiner tree (CRBST) yields a rectilinear Steiner tree T with $R(T) \leqq P$.

Let V_{RST} be the set of vertices which are terminals or Steiner points contained in the current rectilinear Steiner tree (RST) during construction and E_{RST} the set of edges. $e_{ij} \in E_{RST}$ denotes a directed edge from v_i to v_j. The edge direction is introduced to show the direction from the source. Let v_t be a vertex which is closest to the source v_s.

The CRBST begins with $V_{RST} = \{v_s, v_t\}$, $E_{RST} = \{e_{st}\}$. The algorithm chooses an edge $e_{ij} \in E_{RST}$ and a sink $v_k \in V - V_{RST}$ which minimize the cost function $I(e_{ij}, v_k)$ defined below.

For e_{ij} and v_k, let v_m be the *middle point* of v_i, v_j, v_k, which is a point whose x-coordinate and y-coordinate are both the second largest of the x-coordinates and y-coordinates of v_i, v_j, v_k, respectively. Then (see Fig. 1),

$$\begin{aligned} I(e_{ij}, v_k) = C(v_s, v_k, P) \cdot (D_T(v_s, v_i) + D(v_i, v_m)) \\ + D(v_m, v_k), \end{aligned}$$

where $C(v_s, v_k, P) = D(v_s, v_k)/P$.

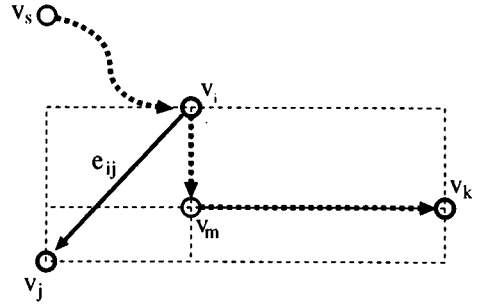


Fig. 1 Determination of v_m and $I(e_{ij}, v_k)$.

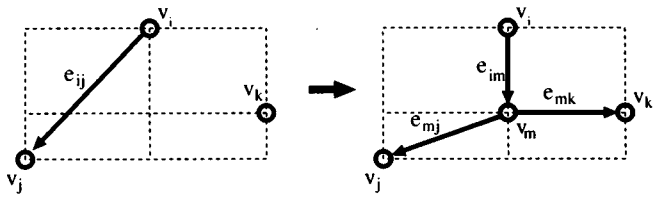


Fig. 2 Reconfiguration of T.

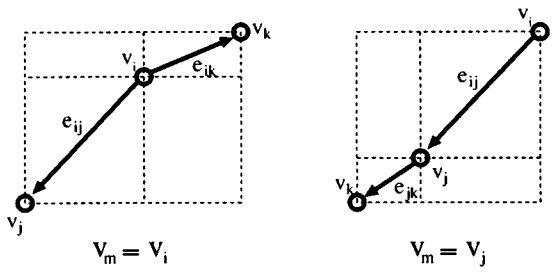


Fig. 3 Degenerated cases of Fig. 2.

Algorithm CRBST

1. $V_{RST} = \{v_s, v_t\}$, $E_{RST} = \{e_{st}\}$.
2. Let $e_{ij} \in E_{RST}$ and $v_k \notin V_{RST}$ be the pair which minimizes the cost function I.
3. $V_{RST} = V_{RST} \cup \{v_k, v_m\}$.
   $E_{RST} = (E_{RST} - \{e_{ij}\}) \cup \{e_{im}, e_{mj}, e_{mk}\}$.
   (v_m is the middle point of v_i, v_j, v_k).
4. If $V \subseteq V_{RST}$, then goto step 5, else return to step 2.
5. Change the remaining slant edges into an L-shaped layout.

A slant edge shall be transformed into a rectilinear layout to get a rectilinear Steiner tree. Of an infinite number of possibilities, cost function I used in choosing a new vertex is based on the expectation that the connection will be completed with the shortest possible length. Step 3 is to realize this shortest length by inserting a Steiner point. In the general case, an edge is replaced with three edges, as shown in Fig. 2. If the edge is slant, its partial layout is determined to achieve the shortest connection. The possibilities that are not concerned with the shortest connection are left as replaced by a new slant edge e_{mj}. In the degenerated case when the middle point coincides with v_i or v_j, an edge is added to T as shown in Fig. 3. Step 5 is to make the slant edges left after Step 3 rectilinear. An example of the entire procedure for $n = 4$ is shown in Fig. 4.

Theorem 2: The CRBST algorithm yields a rectilinear Steiner tree T with $R(T) \leqq P$.

A proof analogous to the proof of Theorem 1 can be done, and is omitted here.

The time complexity of the CRBST is $O(n^2 \log n)$. This is attained using the following popular ideas of the data structure. For each terminal $v_k \notin V_{RST}$, keep the evaluation I with respect to every edge $e \in E_{RST}$ in a heap. In Step 2, find the minimum I that can be achieved in $O(n)$. In Step 3, the addition and reconfiguration can be done in a constant time, and renewal of each heap after reconfiguration can be achieved in $O(\log n)$. In total, the time complexity of Step 3 takes $O(n \log n)$. Note that the Steiner points need not be counted in the vertices since any Steiner point belongs to T from the beginning. Other operations are smaller than this in order. Since the repetition is n, the total time is $O(n^2 \log n)$.

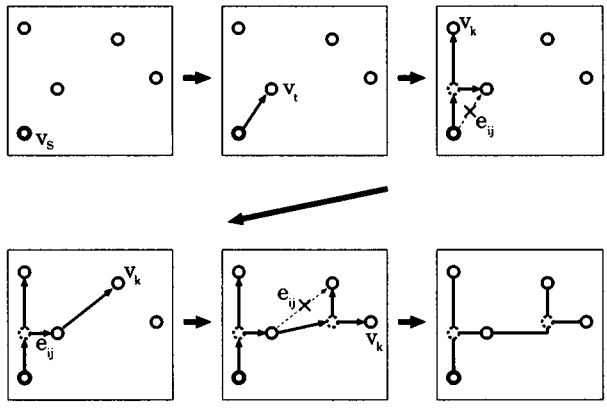


Fig. 4 Flow of algorithm CRBST.

5. Experimental Results

5.1 Environments

5.1.1 Spanning Tree Problem

We tested the CRBT, Pyo's BKRUS, and Alpert's algorithms.

P, the only parameter, is set to $P = R_{max}/c$ where c is changed between $0.00 \sim 1.00$.

The first term to be compared is the *Radius Ratio* which is the radius of resultant trees of each algorithm over that of the *SPT* algorithm, the latter being the smallest. Therefore, the optimum radius ratio is close to, but not less than, 1.

The second term to be compared is the *Cost Ratio* which is the cost for each algorithm over that for the *MST* algorithm, the latter being the minimum. Therefore, the cost ratio is close to, but not less than, 1.

The third term to be compared is the CPU time required on an Intel DX4 PC.

Three hundred randomly generated nets were tested and the resultant value averages are shown in Table 1.

5.1.2 Steiner Tree Problem

Steiner tree versions of the above three algorithms, the CRBST, Pyo's BKST, and the Steiner version of Alpert's

Table 1 Comparison of spanning tree algorithms (*size* is the number of terminals of the net).

size	c	CRBT(Ours)			BKRUS(Pyo's)			Alpert's		
		Radius Ratio	Cost Ratio	CPU (s)	Radius Ratio	Cost Ratio	CPU (s)	Radius Ratio	Cost Ratio	CPU (s)
6	0.00	1.310	1.000	0.013	1.357	1.000	0.013	1.310	1.000	0.013
	0.50	1.088	1.037	0.013	1.348	1.001	0.013	1.112	1.057	0.013
	0.75	1.027	1.115	0.013	1.146	1.039	0.013	1.026	1.143	0.013
	1.00	1.000	1.275	0.013	1.000	1.207	0.013	1.000	1.318	0.013
12	0.00	1.476	1.000	0.020	1.476	1.000	0.043	1.476	1.000	0.020
	0.50	1.134	1.058	0.020	1.430	1.004	0.043	1.149	1.081	0.020
	0.75	1.043	1.128	0.020	1.215	1.042	0.043	1.033	1.221	0.020
	1.00	1.000	1.321	0.020	1.000	1.213	0.043	1.000	1.430	0.020
18	0.00	1.608	1.000	0.033	1.627	1.000	0.166	1.608	1.000	0.033
	0.50	1.130	1.063	0.033	1.473	1.005	0.166	1.147	1.090	0.033
	0.75	1.053	1.142	0.033	1.236	1.041	0.166	1.034	1.243	0.033
	1.00	1.000	1.308	0.033	1.000	1.217	0.166	1.000	1.499	0.033
24	0.00	1.653	1.000	0.060	1.632	1.000	0.503	1.653	1.000	0.060
	0.50	1.129	1.063	0.060	1.555	1.005	0.503	1.143	1.097	0.060
	0.75	1.046	1.127	0.060	1.254	1.039	0.503	1.039	1.240	0.060
	1.00	1.000	1.334	0.060	1.000	1.195	0.503	1.000	1.530	0.060
30	0.00	1.705	1.000	0.103	1.705	1.000	1.213	1.705	1.000	0.103
	0.50	1.142	1.061	0.103	1.606	1.002	1.213	1.152	1.093	0.103
	0.75	1.053	1.135	0.103	1.277	1.046	1.213	1.031	1.265	0.103
	1.00	1.000	1.347	0.103	1.000	1.179	1.213	1.000	1.533	0.103

Table 2 Comparison of Steiner tree algorithms.

size	c	CRBST			BKST			Alpert's		
		Radius Ratio	Cost Ratio	CPU (s)	Radius Ratio	Cost Ratio	CPU (s)	Radius Ratio	Cost Ratio	CPU (s)
6	0.00	1.133	0.910	0.020	1.179	0.928	0.025	1.157	0.916	0.015
	0.50	1.028	0.927	0.020	1.135	0.928	0.025	1.042	0.923	0.015
	0.75	1.009	0.939	0.020	1.082	0.940	0.025	1.015	0.938	0.015
	1.00	1.000	0.978	0.020	1.000	1.041	0.020	1.000	0.956	0.015
12	0.00	1.243	0.906	0.028	1.338	0.924	0.070	1.340	0.914	0.020
	0.50	1.055	0.946	0.028	1.315	0.927	0.065	1.065	0.949	0.020
	0.75	1.013	0.970	0.028	1.150	0.938	0.070	1.011	0.994	0.020
	1.00	1.000	1.029	0.028	1.000	1.038	0.075	1.000	1.024	0.020
18	0.00	1.356	0.905	0.044	1.411	0.924	0.170	1.395	0.909	0.035
	0.50	1.061	0.947	0.044	1.396	0.925	0.175	1.072	0.960	0.040
	0.75	1.020	0.978	0.044	1.180	0.944	0.180	1.014	1.022	0.040
	1.00	1.000	1.045	0.044	1.000	1.046	0.205	1.000	1.066	0.040
24	0.00	1.394	0.905	0.068	1.485	0.924	0.385	1.452	0.909	0.065
	0.50	1.058	0.943	0.068	1.433	0.923	0.385	1.074	0.963	0.065
	0.75	1.019	0.971	0.068	1.193	0.942	0.395	1.013	1.035	0.065
	1.00	1.000	1.043	0.068	1.000	1.040	0.440	1.000	1.093	0.070
30	0.00	1.438	0.904	0.104	1.510	0.923	0.710	1.502	0.906	0.115
	0.50	1.064	0.947	0.104	1.466	0.927	0.720	1.078	0.969	0.115
	0.75	1.017	0.977	0.104	1.223	0.945	0.735	1.018	1.042	0.115
	1.00	1.000	1.048	0.104	1.000	1.043	0.830	1.000	1.111	0.115

algorithm, are compared in the same environments. The results are shown in Table 2. An alternate way of extending the CRBT for a Steiner tree is simply to transform the output of the CRBT to a rectilinear Steiner tree in the same way as in Alpert's algorithm. We experimentally compared this method with the CRBST and the Steiner version of Alpert's algorithms. In most cases, extended CRBT and CRBST are superior to Alpert's, and CRBST is superior to extended CRBT in terms of the cost-radius balance.

5.2 Analysis

The CRBT is designed to take the path length into consideration from a very early stage. Therefore, the CRBT tends to construct a tree whose radius is far smaller than P which is a provable radius bound, as in Alpert's algorithm. On the other hand, since BKRUS tries to minimize the cost as long as the radius limit is not violated, its radius is close to the limit. Therefore, under the same parameter c $(= R_{max}/P)$, it is expected that the CRBT and Alpert's algorithms will both generate small-radius large-cost trees while the BKRUS method will generate a large-radius small-cost trees. The experiments verified

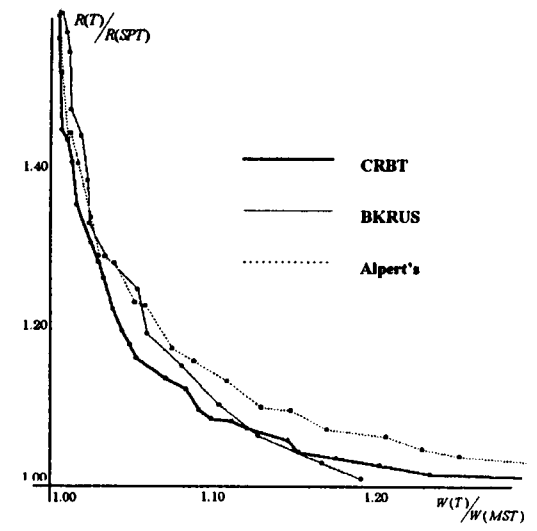


Fig. 5 Cost-Radius balance of spanning tree algorithms.

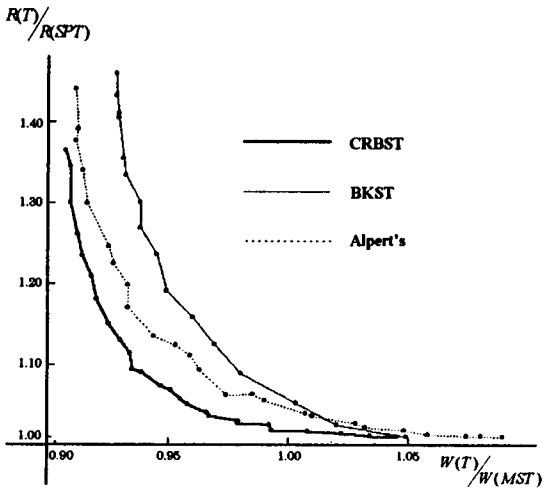


Fig. 6 Cost-Radius balance of Steiner tree algorithms.

this observation, as is clearly seen in Table 1. For the Steiner trees, a similar tendency was found.

A *cost-radius balance* graph showing the radius ratio versus cost ratio is given in Fig. 5. The curve for each algorithm is shown with c ranging from 0 to 1 in increments of 0.04. The samples represent 200 randomly generated nets of 20 terminals for each P $(= R_{max}/c)$. The results illustrate that the CRBT constructs slightly better cost-radius balanced trees. This feature is more apparent for Steiner trees, as shown in Fig. 6, under the same environment.

As the net becomes large, the difference in CPU time becomes significant because the time complexity of BKRUS is $O(n^3)$, but those of the CRBT and Alpert's algorithm are only $O(n^2)$.

6. Conclusions

In looking for an efficient algorithm to construct a practical and useful routing tree with respect to delay reduction, we have presented two heuristic algorithms, CRBT and CRBST, which yield a spanning tree and a Steiner tree with a fairly good balance between cost and radius, and demonstrated the performance of our algorithms.

A future task is to generalize these algorithms to allow the radius limit to be defined to each sink. However, the basic problem, which is beyond the scope of our work, is to realize a best balanced routing tree which may depend on the device process used.

Acknowledgments

This work is supported in part by the Research Body of CAD21 at the Tokyo Institute of Technology.

References

[1] C.J. Alpert, T.C. Hu, J.H. Huang, A.B. Kahng, and D. Karger, "Prim-Dijkstra tradeoffs for improved performance driven routing tree design," IEEE Trans. Computer-Aided Design, vol.14, no.7, pp.890–896, July 1995.

[2] K.D. Boese, J. Cong, A.B. Kahng, K.S. Leung, and D. Zhou, "On highspeed VLSI interconnects: analysis and design," Proc. Asia-Pacific Conf. Circuits Syst., pp.35–40, Dec. 1992.

[3] J. Cong, A.B. Kahng, G. Robins, M. Sarrafzadeh, and C.K. Wong, "Provably good performance-driven global routing," IEEE Trans. CAD, vol.11, no.6, pp.739–752, June 1992.

[4] J. Cong, K. Leung, and D. Zhou, "Performance-driven interconnect design based on distributed RC delay model," Proc. ACM/IEEE Design Automat. Conf., pp.606–611, June 1993.

[5] M. Garey and D.S. Johnson, "The rectilinear Steiner problem is NP-complete," SIAM J. Appl. Math., vol.32, no.4, pp.826–834, April 1977.

[6] M. Hanan, "On Steiner's problem with rectilinear distance," SIAM J. APPL. MATH., pp.255–265, March 1966.

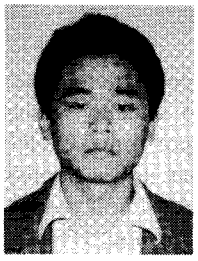
[7] J. Ho, D.T. Lee, C.H. Chang, and C.K. Wong, "Bounded diameter spanning trees and related problems," Proc. ACM Symp. Computational Geometry, pp.276–282, 1989.

[8] J-M. Ho, G. Vijayan, and C.K. Wong, "New algorithms for rectilinear Steiner trees," IEEE Trans. Computer-Aided Design, vol.9, no.2, pp.185–193, Feb. 1990.

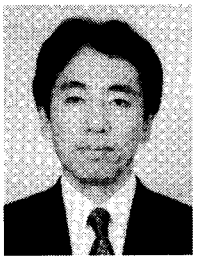
[9] S. Khuller, B. Raghavachari, and N. Young, "Balancing minimum spanning and shortest path trees," Proc. ACM/SIAM Symp. Discrete Algorithms, pp.243–250, Jan. 1993.

[10] I. Pyo, J. Oh, and M. Pedram, "Constructing minimal spanning/Steiner trees with bounded path length," Proc. European Design & Test Conf., pp.244–249, March 1996.

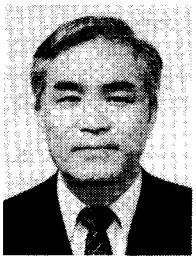
[11] J. Rubinstein, P. Penfield, and M.A. Horowitz, "Signal delay in RC tree networks," IEEE Trans. Computer-Aided Design, vol.2, no.3, pp.202–211, July 1983.



Hideki Mitsubayashi received his B.E. degree in electrical and electronic engineering from the Tokyo Institute of Technology, Tokyo, Japan, in 1996. He is currently a master student of electronic engineering at the Tokyo Institute of Technology. His research interests are in VLSI layout design and combinatorial algorithms.



Atsushi Takahashi received his B.E., M.E., and D.E. degrees in electrical and electronic engineering from the Tokyo Institute of Technology, Tokyo, Japan, in 1989, 1991, and 1996, respectively. Since 1991, he has been with the Tokyo Institute of Technology, where he is now a research associate in the Department of Electrical and Electronic Engineering. His research interests are in VLSI layout design and combinatorial algorithms. He is a member of the Information Processing Society of Japan.



Yoji Kajitani received his B.S., M.S. and D.E. degrees from the Tokyo Institute of Technology, Tokyo, all in electrical engineering, in 1964, 1966 and 1969, respectively. He has been a professor in the Department of Electrical and Electronic Engineering at the Tokyo Institute of Technology since 1985, and has been a professor at the Japan Advanced Institute of Science and Technology from 1992 to 1995. His current research interest is in combinatorial algorithms applied to VLSI layout design. He was awarded an IEEE Fellowship in 1992.