Interconnect Topology Optimization

Given: The driver and a set of sinks (receivers)
Question: Find the best interconnect topology for delay minimization.

Total wire length minimization
- Minimum spanning tree
- Steiner tree

Path length minimization
- Bounded radius bounded cost (BRBC) tree
- A-tree

Delay minimization
- WBA-tree
- P-tree
- Rats-tree
Prim Algorithm for Minimum Spanning Trees

- Grow a connected subtree from the source, one node at a time.
- At each step, choose the closest un-connected node and add it to the subtree.

Steiner Tree

- Steiner points and Steiner tree problem
- Heuristic algorithms for Steiner tree problem
  - Edge-merging based algorithm
  - 1-Steiner algorithm
  - Edge-insertion based algorithm
Merging-based Steiner Tree

1-Steiner Tree
Edge-Insertion based Steiner Tree

Interconnect Topology Optimization Under Linear Delay Model [Cong-Kahng-Robin, T-CAD’92]

- Conventional Routing Algorithms Are Not Good Enough
  - Minimum spanning tree may have very long source-sink path.
  - Shortest path tree may have very large routing cost.
Example: A Routing Problem

Definitions

Given Net $N$ with source $s$ and connected by tree $T$.

- Radius of net $N$: distance from the source to the furthest sink.
- Radius of a routing tree $r(T)$: length of the longest path from the root to a leaf.
- Cost of an edge: distance between two endpoints (other weights are ok).
- Cost of a routing tree $cost(T)$: sum of the edge costs in $T$.
- $minpath_G(u, v)$: shortest path from $u$ to $v$ in $G$
- $dist_G(u, v)$: cost of $minpath_G(u, v)$. 

radius of the net  
route tree
Problem Formulation

- Basic Idea: Restrict the tree radius while minimizing the routing cost
- Bounded radius minimum spanning tree problem (BRMST):
  - Given: A net \( N \) with radius \( R \)
  - Find: A minimum cost tree with radius \( r(T) \leq (1 + \varepsilon)R \)

Parameter \( \varepsilon \) controls the trade-off between radius and cost
- \( \varepsilon = \infty \) minimum spanning tree; \( \varepsilon = 0 \) shortest path tree

BPRIIM Algorithm Bounded-Radius Minimum Spanning Trees

- Given net \( N \) with source \( s \) and radius \( R \), and parameter \( \varepsilon \).
- Grow a connected subtree \( T \) from the source, one node at a time.
- At each step, choose the closest pair \( x \in T \) and \( y \in N - T \)
  - If \( \text{dist}_T(s, x) + \text{cost}(x, y) \leq (1 + \varepsilon)R \), add \((x, y)\)
  - Else backtrack along \( \text{minpath}_T(s, x) \) to find \( x' \) such that
    \( \text{dist}_T(s, x') + \text{cost}(x', y) \leq R \), then add \((x', y)\)

- Slack \( \varepsilon R \) is introduced at each backtrace so we do not have to backtrack too often.
An Improved Algorithm -- BRBC

[Cong-Kahng-Robin, T-CAD'92]

- Construct \( MST \) and \( SPT \), \( Q = MST \);
- Construct \( L \) -- a depth-first tour of \( MST \);
- Traverse \( L \) while keeping a running total \( S \) of traversed edge costs, when reaching \( L_i \):
  - If \( S \geq \epsilon \) \( \text{dist}_{SPT}(S, L_i) \) then add \( \text{minpath}_{SPT}(s, L_i) \) to \( Q \) and reset \( S = 0 \),
  - Else continue traverse \( L \);
- Construct the shortest path tree \( T \) of \( Q \).

BRBC Trees Have Bounded Radius

**Theorem 1:** \( r(T) \leq (1 + \epsilon) R \)

**Proof:** For any vertex \( x \), let \( y \) be the last vertex before \( x \) in \( L \) that we add \( \text{minpath}_{SPT}(s, L) \).

By the choice of \( y \), we have
\[
\text{dist}_{L}(y, x) \leq \epsilon \text{ dist}_{SPT}(s, x) \leq \epsilon R
\]

Therefore,
\[
\text{dist}_{Q}(s, x) \leq \text{dist}_{Q}(s, y) + \text{dist}_{L}(y, x) \\
\leq \text{dist}_{SPT}(s, y) + \text{dist}_{L}(y, x) \\
\leq R + \epsilon R = (1 + \epsilon) R
\]
BRBC Trees Have Bounded Cost

Theorem 2: \( \text{cost}(T) \leq (1+2/\varepsilon) \text{cost}(\text{MST}). \)

Proof: Let \( v_1, v_2, \ldots, v_k \) be the vertices that we add
\( \text{minpath}_{SP}(s,v_i) \)
Note that \( T \) is a subgraph of \( Q \)

\[
\text{cost}(Q) = \text{cost}(\text{MST}) + \sum_{i=1}^{k} \text{dist}_{SP}(s,v_i)
\]
\[
\leq \text{cost}(\text{MST}) + \sum_{i=1}^{k} \frac{1}{\varepsilon} \text{dist}_{L}(v_{i-1},v_i)
\]
\[
\leq \text{cost}(\text{MST}) + \frac{1}{\varepsilon} \text{cost}(L)
\]
\[
\leq \text{cost}(\text{MST}) + \frac{2}{\varepsilon} \text{cost}(\text{MST})
\]
\[
\leq (1+\frac{2}{\varepsilon})\text{cost}(\text{MST})
\]

Related work by [Awarbuch - Baratz - Peleg, PODC-90]

Interconnect Topology Design Formulation Under Distributed RC Delay Model

\[
t(T) = \sum_{s \in \text{sources}} R_s C_s = \sum_{s \in \text{sources}} (R_s + R_s \cdot \text{pl}_s(T)) (C_s + C_s)
\]
\[
t_1(T) = R_s \cdot C_s \cdot \text{length}(T) \quad \text{MST}
\]
\[
t_2(T) = \sum_{s \in \text{sources}} C_s \cdot \text{pl}_s(T) \quad \text{SPT}
\]
\[
t_3(T) = R_s \cdot \sum_{s \in \text{sources}} \text{pl}_s(T) \quad \text{QMST}
\]
\[
t_4(T) = C_s \quad \text{Constant}
\]

Objective: Minimize
\[
\alpha \cdot \text{length}(T) + \beta \cdot \sum_{s \in \text{sources}} \text{pl}_s(T) + \gamma \cdot \sum_{s \in \text{sources}} \text{pl}_s(T)
\]
Impact of Resistance Ratio

Definition: \( \frac{R_d}{R_0} \)
- Driver resistance versus unit wire resistance

Determined by the Technology:
- Reduce device dimension \( \Rightarrow \) \( R_0 \uparrow \) \( R_d \downarrow \)
- \( \frac{R_d}{R_0} \downarrow \)

• Impact on Interconnect Optimization:

\[
\begin{align*}
    t_1(T) &= R_d \cdot C_s \cdot \text{length}(T) \quad \text{OST} \quad \text{SPT} \\
    t_2(T) &= R_s \cdot \sum_{i=\text{max}} C_s \cdot p_i(T) \quad \text{QMST} \\
    t_3(T) &= R_s \cdot C_s \cdot \sum_{i=\text{min}} p_i(T) \quad \text{QMST}
\end{align*}
\]

Comparison of Three Types of Trees

<table>
<thead>
<tr>
<th></th>
<th>OST cost</th>
<th>SPT cost</th>
<th>QMST cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>OST</td>
<td>9 (optimal)</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>SPT</td>
<td>37</td>
<td>29 (optimal)</td>
<td>31</td>
</tr>
<tr>
<td>QMST</td>
<td>45</td>
<td>36</td>
<td>34 (optimal)</td>
</tr>
</tbody>
</table>
Results on Interconnect Topology Optimization

A-TREE: Generalized Rectilinear Minimum Steiner Arborescence

- Advantage of A-tree:
  - Always a SPT ($t_2(T)$ is minimum)
  - Minimizing $t_3(T)$ for most A-trees
- Objective: Minimize the total wirelength of A-tree:
  - Simultaneous minimization of $t_1(T), t_2(T)$ and $t_3(T)$.

Rectilinear Steiner Arborescence Algorithm

[Rao-Sadayappan-Huang ’92]

- RSA: Rectilinear Steiner Arborescence (shortest path Steiner tree), A-tree in short
- RSA algorithm
  - Start with a forest of n single-node A-trees, repeatedly combining two A-trees into a new one
Performance of RSA Algorithm

- Time Complexity $O(n \log n)$.
- Wirelength of the tree by RSA algorithm
  $\leq 2 \text{ length(RSMA)}$
  - RSMA: Rectilinear Steiner Minimum Arborescence

A-tree Algorithm

[Cong-Leung-Zhou, DAC’93]

- Start with a forest of n single-node A-trees
- Apply a sequence of moves
  - Grow an existing A-tree, or
  - Combine two A-trees into a new one
- Terminate when only one A-tree is left
A-tree Algorithm (Cont’d)

- p dominates q if \( p_x \geq q_x \) and \( p_y \geq q_y \).
- \( \text{DOM}(p, F_k) \): the set of nodes in \( F_k \) dominated by \( p \).

Given a node \( p \), define
- \( \text{NW}(p) \): \( \{(x, y) | x < p_x, y > p_y\} \).
- \( \text{SE}(p), \text{SW}(p), \text{NE}(p), \text{N}(p), \text{S}(p), \text{W}(p), \text{E}(p) \) similarly.

Given node \( p \), define:
- \( \text{MF}(p, F_k) \) -- nodes in \( \text{DOM}(p, F_k) \) with minimum rectilinear distance from \( p \),
- \( d_f(p, F_k) \) -- the distance from \( p \) to any node in \( \text{MF}(p, F_k) \).
- \( \text{mf}_{\text{west}}(p, F_k) \) -- the one with smallest x-coordinate in \( \text{MF}(p, F_k) \).
- \( \text{mf}_{\text{south}}(p, F_k) \) defined similarly.

A-tree Algorithm (Cont’d)

- Given \( p \) as a root in \( F_k \):
  - \( \text{mx}(p, F_k) \) -- the node in \( \text{NW}(p) \cap \text{ROOT}(F_k) \), not blocked from \( p \) and have the minimum horizontal distance from \( p \).
  - \( \text{dx}(p, F_k) \) -- the horizontal distance between \( p \) and \( \text{mx}(p, F_k) \) (or \( \propto \) if \( \text{mx}(p, F_k) \) doesn’t exist)
  - \( \text{my}(p, F_k), \text{dy}(F_k) \) similarly defined
**Type-1 Safe Move**

“Combine” two A-trees into a new one

\[
\begin{align*}
&dx(p, F_k) \geq df(p, F_k) \\
&dy(p, F_k) \geq df(p, F_k)
\end{align*}
\]

Move:

\[p \text{ to } mf_{\text{west}}(p, F_k)\]

**Type-2 Safe Move**

\[
\begin{align*}
&dx(p, F_k) \geq df(p, F_k) \\
&dy(p, F_k) < df(p, F_k)
\end{align*}
\]

Move:

southward, \(p\) to \(p'\) with length

\[\min\{\text{dist}_{y}(mf_{\text{south}}(p, F_k), p), dy(p, F_k)\}\]
**Type-3 Safe Move**

Move:
westward, \(p\) to \(p'\) with length
\[\min\{\text{dist}(\text{mf}_{\text{west}}(p, F_k), p), dx(p, F_k)\}\]

\[dx(p, F_k) < df(p, F_k)\]
\[dy(p, F_k) >= df(p, F_k)\]

---

**Heuristic Moves**

- **Type-1 and Type-2 Heuristic Moves:**
  - "Combines" two A-trees into a new one

**H1:** select node \(p\) such that \(p' = \text{mf}_{\text{west}}(p, F_k)\)
is farthest away from the source and introduce a path from \(p\) to \(p'\)

**H2:** select two nodes \(p_1, p_2\) such that
\(p' = \min\{\min((p_1)_x, (p_2)_x), \min((p_1)_y, (p_2)_y)\}\)
is farthest away from the source and connect \(p_1\) and \(p_2\) to \(p'\)
Optimality of a Move

- Definitions:
  - \( F_k \) is the forest constructed after the \( k \)-th move by the A-tree algorithm
  - \( T(F_k) \) is the minimum-cost rectilinear Steiner A-tree containing \( F_k \) as a subgraph

- \( T(F_0) \) is the optimal A-tree

- \( T(F_m) \) is the A-tree constructed by our algorithm

- Optimality of a Move:
  - If \( cost(T(F_k)) = cost(T(F_{k+1})) \) then the \( k+1 \)-th is called an optimal move

- Theorem All three types of safe moves are optimal moves

---

Lower Bound Computation of Optimal A-Tree

- **Lower Bound Computation:**
  - After the \( k \)-th move, we can compute an upper bound \( ERROR_k \) of \( cost(T(F_{k+1})) - cost(T(F_k)) \)
  - Note that
    - (i) \( ERROR_k = 0 \) if the \( k \)-th move is a safe move
    - (ii) \( ERROR_k > 0 \) if the \( k \)-th move is a heuristic move
  - \( cost(T) \leq cost(T^*) + \sum_k ERROR_k \)
  - \( T \): constructed by the A-tree algorithm
  - \( T^* \): optimal A-tree

- **Results obtained from the Lower Bound Computation:**
  - 94% of the moves are safe moves
  - 45% of the A-trees constructed by the algorithm are optimal
  - the A-trees constructed by the algorithm are at most 4% from optimal
Steiner Elmore Routing Tree (SERT) Heuristic
[Boese-Kahng-Robins, DAC’93]

- Use Elmore Delay Model directly in construction of routing tree $T$.
- Add nodes to $T$ one-by-one like Prim’s MST algorithm.
- At each step, choose $v \not\in T$ and $u \in T$ s.t.
  1. The linear weighted Elmore-delay to the critical sinks has minimum increase (SERT-C algorithm) or
  2. The maximum Elmore-delay to any sink has minimum increase (SERT algorithm).

Examples of SERT-C Construction

a) Node 2 (or 4) critical
b) Node 3 (or 7) critical (also 1-Steiner tree)
c) Node 5 critical
d) Node 6 critical
e) Node 8 critical (also Steiner ERT)
f) Node 9 critical
Some Theoretical Results on Optimal Routing Tree Under Elmore Delay Model  
[Boese-Kahng-McCoy-Robins, DAC’94]

- When minimizing linear weighted Elmore delay to the critical sinks, there exists optimal tree on the Hanan-grid.

- When minimizing maximum Elmore delay in routing tree, optimal tree may not lie on the Hanan-grid.

Performance Driven Multisource Routing Problem  
[Cong-Madden, ISCAS ‘95]

- Previous work:
  Assume a single source driving one or more sink nodes

- Contribution:
  First work to address nets where each pin may act as a source, sink, or both.

- Signal busses are examples of multisource nets.
An Example Multisource Routing Problem

Minimize $\sum W(i,j) \times delay(p_i, p_j)$ weighted delay

$\sum L(T)$ total tree length