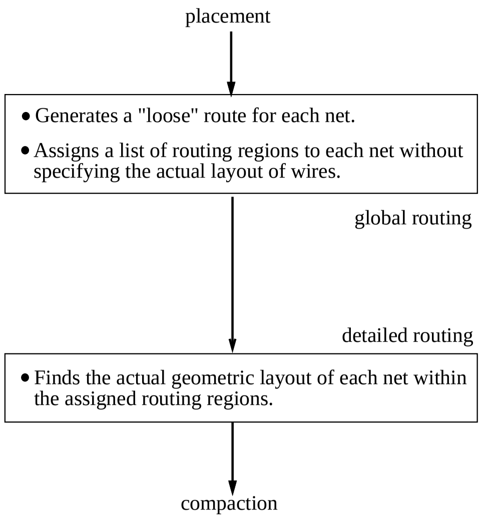
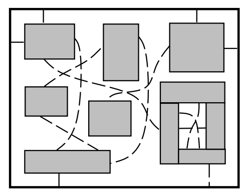
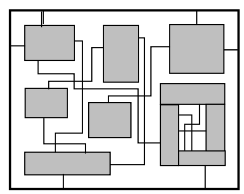
Routing





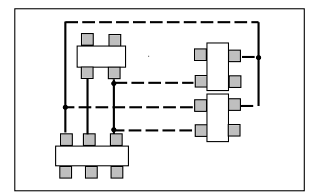
Global routing



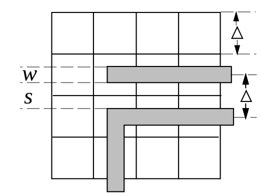
Detailed routing

Routing Constraints

- 100% routing completion + area minimization, under a set of constraints:
  - Placement constraint: usually based on fixed placement
  - Number of routing layers
  - Geometrical constraints: must satisfy design rules
  - Timing constraints (performance-driven routing): must satisfy delay constraints
  - Crosstalk?

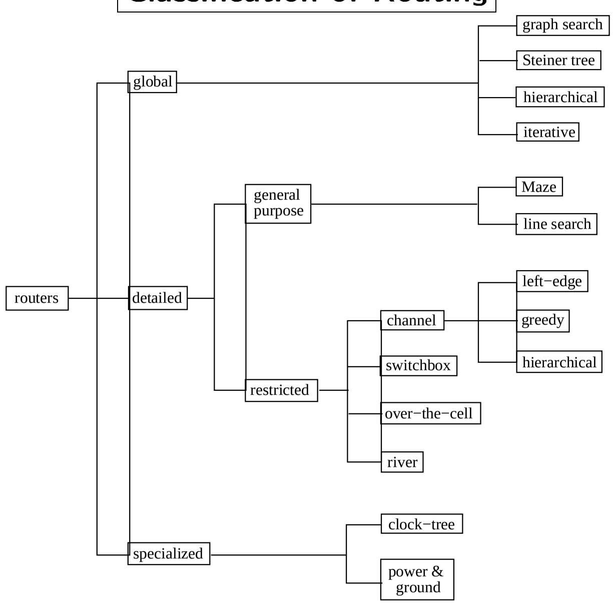


Two−layer routing



Geometrical constraint

Classification of Routing

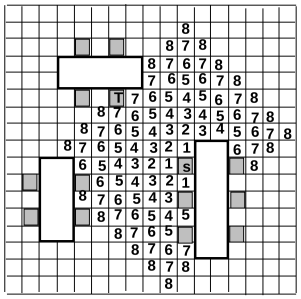


Maze Router: Lee Algorithm

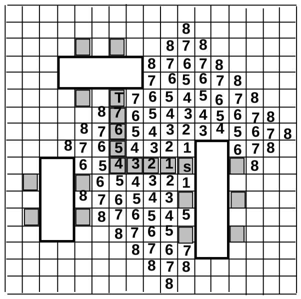
- Lee, "An algorithm for path connection and its application," *IRE Trans. Electronic Computer*, EC-10, 1961.
- Discussion mainly on single-layer routing
- **Strengths**
  - Guarantee to find connection between 2 terminals if it exists.
  - Guarantee minimum path.
- **Weaknesses**
  - Requires large memory for dense layout
  - Slow
- Applications: global routing, detailed routing

Lee Algorithm

- Find a path from S to T by "wave propagation".



Filing

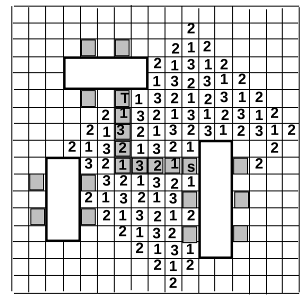


Retrace

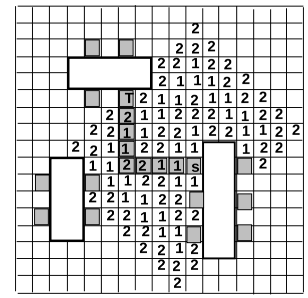
- Time & space complexity for an $M \times N$ grid: $O(MN)$ (**huge!**)

Reducing Memory Requirement

- Akers's Observations (1967)
  - Adjacent labels for k are either $k-1$ or $k+1$.
  - Want a labeling scheme such that each label has its preceding label different from its succeeding label.
- Way 1: coding sequence $1, 2, 3, 1, 2, 3, \ldots$; states: 1, 2, 3, *empty*, *blocked* (3 bits required)
- Way 2: coding sequence $1, 1, 2, 2, 1, 1, 2, 2, \ldots$; states: 1, 2, *empty*, *blocked* (need only 2 bits)



Sequence: 1, 2, 3, 1, 2, 3, ...

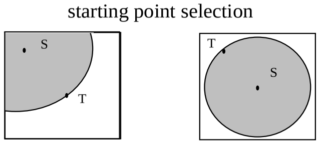


Sequence: 1, 1, 2, 2, 1, 1, 2, 2, ...

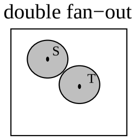
Reducing Running Time

- Starting point selection: Choose the point farthest from the center of the grid as the starting point.
- Double fan-out: Propagate waves from both the source and the target cells.
- Framing: Search inside a rectangle area 10–20% larger than the bounding box containing the source and target.
  - Need to enlarge the rectangle and redo if the search fails.

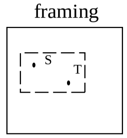
starting point selection



double fan−out

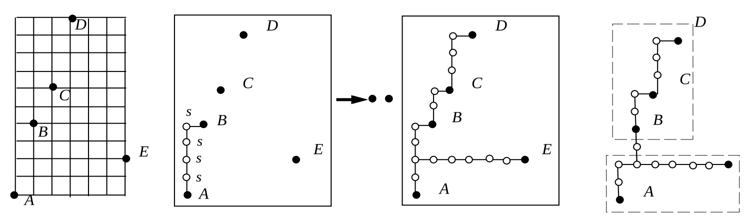


framing



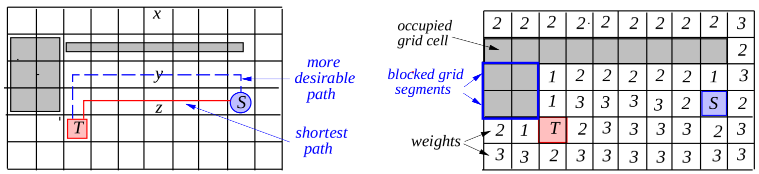
Connecting Multi-Terminal Nets

- Step 1: Propagate wave from the source s to the closet target.
- Step 2: Mark ALL cells on the path as s.
- Step 3: Propagate wave from ALL s cells to the other cells.
- Step 4: Continue until all cells are reached.
- Step 5: Apply heuristics to further reduce the tree cost.

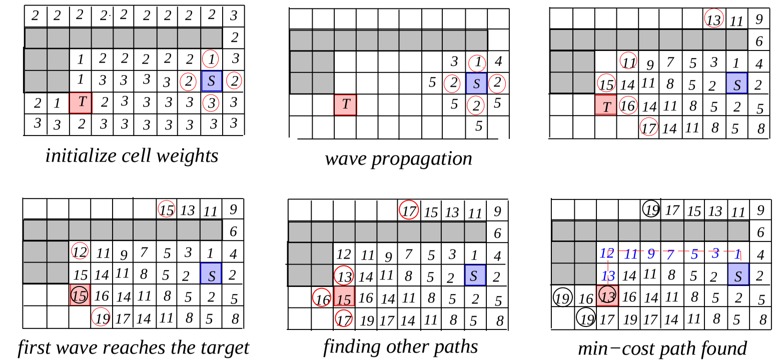


Routing on a Weighted Grid

- Motivation: finding more desirable paths
- $weight(grid\ cell) = \#\ of\ unblocked\ grid\ cell\ segments - 1$



A Routing Example on a Weighted Grid

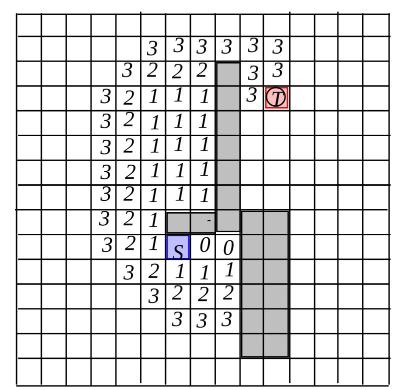


Hadlock's Algorithm

- Hadlock, "A shortest path algorithm for grid graphs," *Networks*, 1977.
- Uses detour number (instead of labeling wavefront in Lee's router)
  - Detour number, $d(P)$: # of grid cells directed **away from** its target on path P.
  - $MD(S,T)$: the Manhattan distance between S and T.
  - Path length of P, $l(P)$: $l(P) = MD(S,T) + 2d(P)$.
  - $MD(S,T)$ fixed! $\Rightarrow$ Minimize $d(P)$ to find the shortest path.
  - For any cell labeled i, label its adjacent unblocked cells **away from** T $i+1$; label i otherwise.
- Time and space complexities: $O(MN)$, but substantially reduces the # of searched cells.
- Finds the shortest path between S and T.

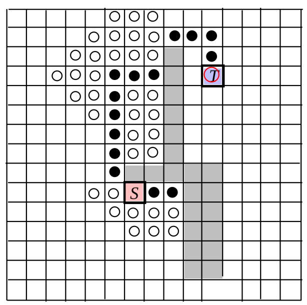
Hadlock's Algorithm (cont'd)

- $d(P)$: # of grid cells directed **away from** its target on path P.
- $MD(S,T)$: the Manhattan distance between S and T.
- Path length of P, $l(P)$: $l(P) = MD(S,T) + 2d(P)$.
- $MD(S,T)$ fixed! $\Rightarrow$ Minimize $d(P)$ to find the shortest path.
- For any cell labeled i, label its adjacent unblocked cells **away from** T $i+1$; label i otherwise.

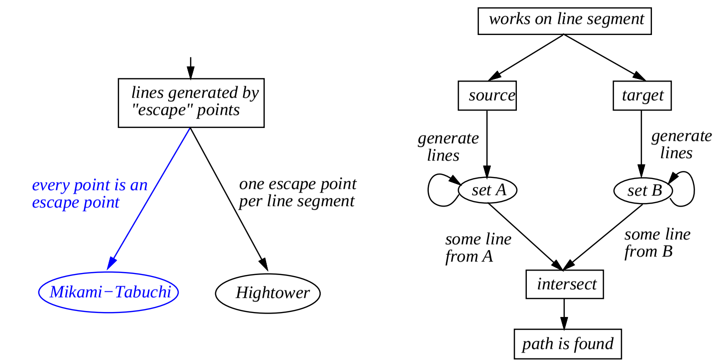


Soukup's Algorithm

- Soukup, "Fast maze router," DAC-78.
- Combined breadth-first and depth-first search.
  - Depth-first (**line**) search is first directed toward target T until an obstacle or T is reached.
  - Breadth-first (Lee-type) search is used to "bubble" around an obstacle if an obstacle is reached.
- Time and space complexities: $O(MN)$, but 10–50 times faster than Lee's algorithm.
- Find **a** path between S and T, but may not be the shortest!



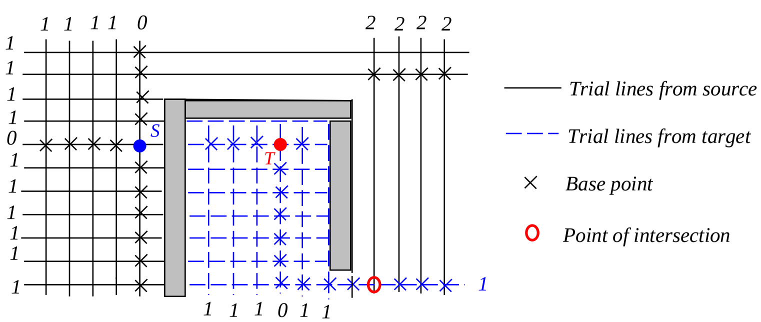
Features of Line-Search Algorithms



- Time and space complexities: $O(L)$, where L is the # of line segments generated.

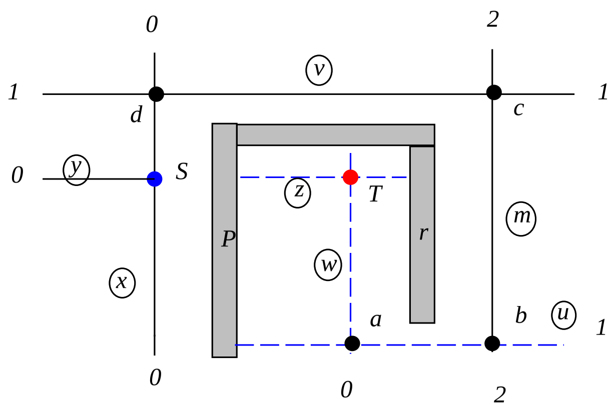
Mikami-Tabuchi's Algorithm

- Mikami & Tabuchi, "A computer program for optimal routing of printed circuit connectors," *IFIP*, H47, 1968.
- Every grid point is an escape point.



Hightower's Algorithm

- Hightower, "A solution to line-routing problem on the continuous plane," DAC-69.
- A single escape point on each line segment.
- If a line parallels to the blocked cells, the escape point is placed just past the endpoint of the segment.

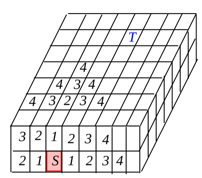


Comparison of Algorithms

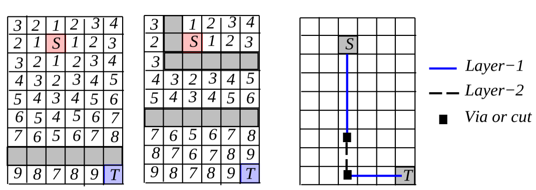
	Maze routing			Line search	
	Lee	Soukup	Hadlock	Mikami	Hightower
Time	$O(MN)$	$O(MN)$	$O(MN)$	$O(L)$	$O(L)$
Space	$O(MN)$	$O(MN)$	$O(MN)$	$O(L)$	$O(L)$
Finds path if one exists?	yes	yes	yes	yes	no
Is the path shortest?	yes	no	yes	no	no
Works on grids or lines?	grid	grid	grid	line	line

- Soukup, Mikami, and Hightower all adopt some sort of line-search operations $\Rightarrow$ cannot guarantee shortest paths.

Multi-layer Routing



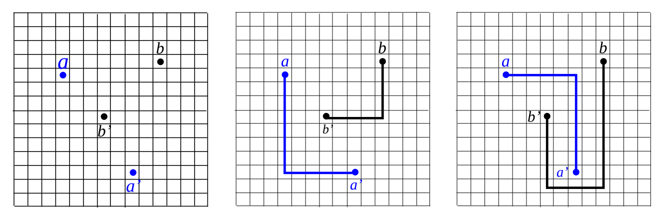
- 3-D grid:
- Two planner arrays:
  - Neglect the weight for inter-layer connection through via.
  - Pins are accessible from both layers.



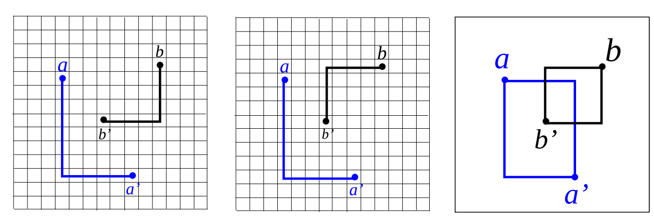
1st layer — 2nd layer — a path

Net Ordering

- Net ordering greatly affects routing solutions.
- In the example, we should route net b before net a.



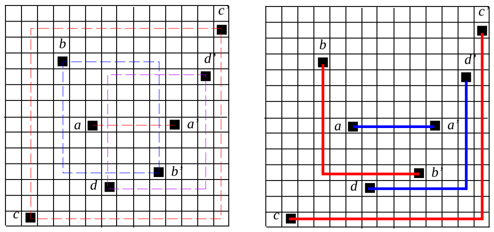
route net a before net b



route net b before net a

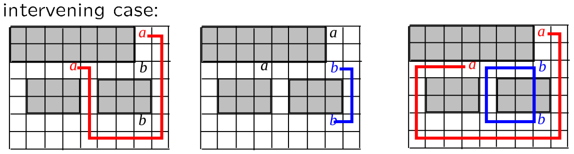
Net Ordering (cont'd)

- Order the nets in the ascending order of the # of pins within their bounding boxes.
- Order the nets in the ascending (or descending??) order of their lengths.
- Order the nets based on their timing criticality.



routing ordering: a (0) −> b (1) −> d (2) −> c (6)

- A mutually intervening case:



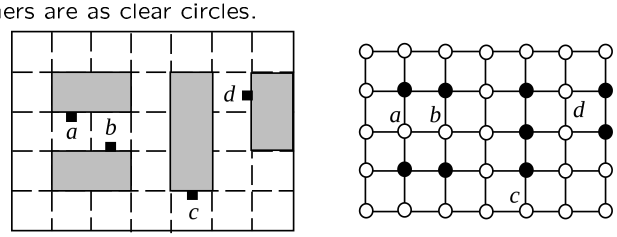
a prevents routing of b *b prevents routing of a* *a feasible routing*

Rip-Up and Re-routing

- Rip-up and re-routing is required if a global or detailed router fails in routing all nets.
- Approaches: the manual approach? the automatic procedure?
- Two steps in rip-up and re-routing
  1. Identify bottleneck regions, rip off some already routed nets.
  2. Route the blocked connections, and re-route the ripped-up connections.
- Repeat the above steps until all connections are routed or a time limit is exceeded.

Graph Models for Global Routing: Grid Graph

- Each cell is represented by a vertex.
- Two vertices are joined by an edge if the corresponding cells are adjacent to each other.
- The occupied cells are represented as filled circles, whereas the others are as clear circles.

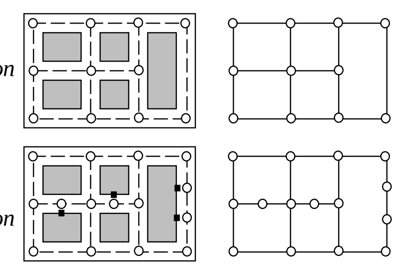


Graph Model: Channel Intersection Graph

- Channels are represented as edges.
- Channel intersections are represented as vertices.
- Edge weight represents channel capacity.
- Extended channel intersection graph: terminals are also represented as vertices.

channel intersection graph

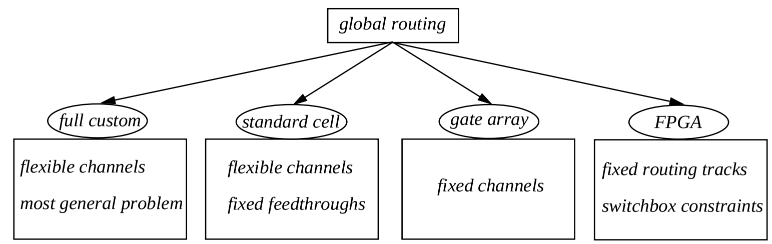
extended channel intersection graph



Global-Routing Problem

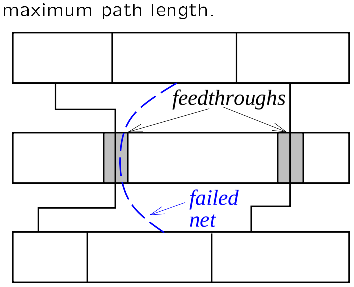
- Given a netlist N=$\{N_1, N_2, \ldots, N_n\}$, a routing graph $G = (V, E)$, find a Steiner tree T_i for each net N_i, $1 \leq i \leq n$, such that $U(e_j) \leq c(e_j)$, $\forall e_j \in E$ and $\sum_{i=1}^{n} L(T_i)$ is minimized, where
  - $c(e_j)$: capacity of edge e_j;
  - $x_{ij} = 1$ if e_j is in T_i; $x_{ij} = 0$ otherwise;
  - $U(e_j) = \sum_{i=1}^{n} x_{ij}$: # of wires that pass through the channel corresponding to edge e_j;
  - $L(T_i)$: total wirelength of Steiner tree T_i.
- For high-performance, the maximum wirelength ($\max_{i=1}^{n} L(T_i)$) is minimized (or the longest path between two points in T_i is minimized).

Global Routing in different Design Styles



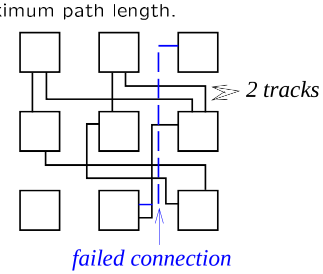
Global Routing in Standard Cell

- Objective
  - Minimize total channel height.
  - Assignment of **feedthrough**: Placement? Global routing?
- For high performance,
  - Minimize the maximum wire length.
  - Minimize the maximum path length.



Global Routing in Gate Array

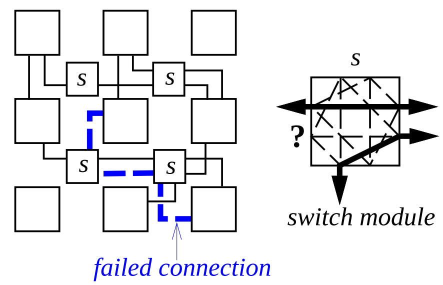
- Objective
  - **Guarantee 100% routability.**
- For high performance,
  - Minimize the maximum wire length.
  - Minimize the maximum path length.



Each channel has a capacity of 2 tracks.

Global Routing in FPGA

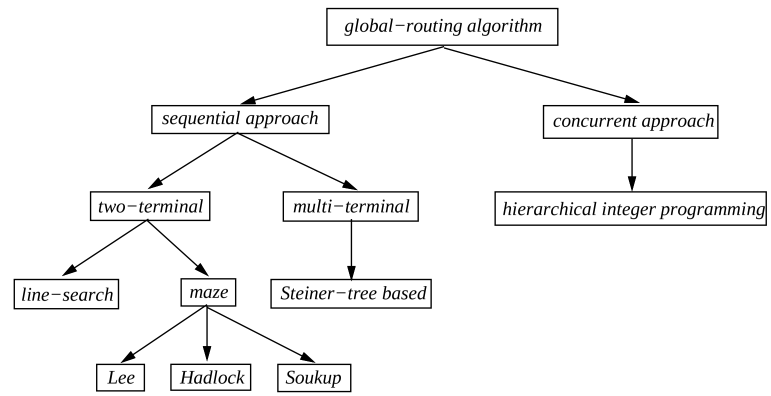
- Objective
  - Guarantee 100% routability.
  - Consider **switch-module architectural constraints.**
- For performance-driven routing,
  - **Minimize # of switches used.**
  - Minimize the maximum wire length.
  - Minimize the maximum path length.



Each channel has a capacity of 2 tracks.

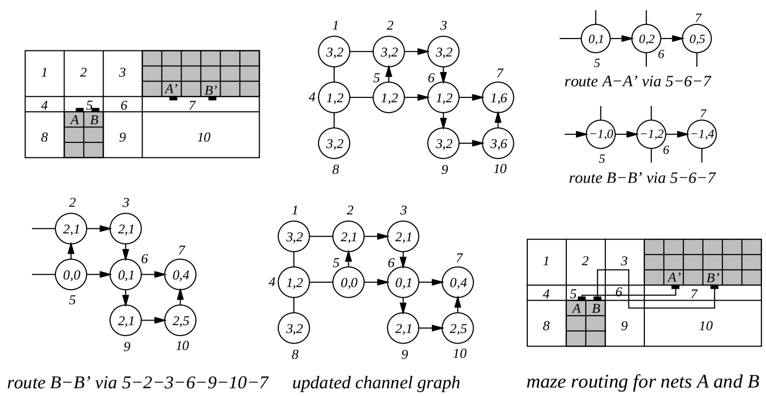
Classification of Global-Routing Algorithm

- **Sequential approach:** Assigns priority to nets; routes one net at a time based on its priority (net ordering?).
- **Concurrent approach:** All nets are considered at the same time (complexity?)



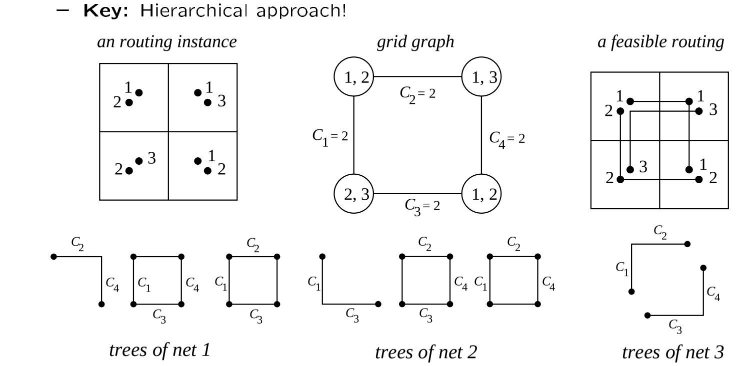
Global-Routing: Maze Routing

- Routing channels may be modelled by a weighted undirected graph called **channel connectivity graph**.
- Node $\leftrightarrow$ channel; edge $\leftrightarrow$ two adjacent channels; capacity: $(width, length)$



Global Routing by Integer Programming

- Suppose that for each net i, there are n_i possible trees $t_1^i, t_2^i, \ldots, t_{n_i}^i$ to route the net.
- Constraint I: For each net i, only one tree t_j^i will be selected.
- Constraint II: The capacity of each cell boundary c_i is not exceeded.
- Minimize the total tree cost.
- **Question:** Feasible for practical problem sizes?
  - **Key:** Hierarchical approach!



An Integer-Programming Example

Boundary	t_1^1	t_2^1	t_3^1	t_1^2	t_2^2	t_3^2	t_1^3	t_2^3
B1	0	1	1	1	0	1	1	0
B2	1	0	1	0	1	1	1	0
B3	0	1	1	1	1	0	0	1
B4	1	1	0	0	1	1	0	1

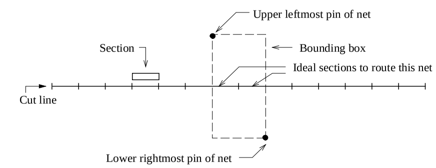
- $g_{i,j}$: cost of tree $t_j^i \Rightarrow g_{1,1} = 2, g_{1,2} = 3, g_{1,3} = 3, g_{2,1} = 2, g_{2,2} = 3, g_{2,3} = 3, g_{3,1} = 2, g_{3,2} = 2$.

Minimize $2x_{1,1} + 3x_{1,2} + 3x_{1,3} + 2x_{2,1} + 3x_{2,2} + 3x_{2,3} + 2x_{3,1} + 2x_{3,2}$
subject to

$$
\begin{array}{rcll}
x_{1,1} + x_{1,2} + x_{1,3} & = & 1 & (Constraint\ I : t^1) \\
x_{2,1} + x_{2,2} + x_{2,3} & = & 1 & (Constraint\ I : t^2) \\
x_{3,1} + x_{3,2} & = & 1 & (Constraint\ I : t^3) \\
x_{1,2} + x_{1,3} + x_{2,1} + x_{2,3} + x_{3,1} & \leq & 2 & (Constraint\ II : B1) \\
x_{1,1} + x_{1,3} + x_{2,2} + x_{2,3} + x_{3,1} & \leq & 2 & (Constraint\ II : B2) \\
x_{1,2} + x_{1,3} + x_{2,1} + x_{2,2} + x_{3,2} & \leq & 2 & (Constraint\ II : B3) \\
x_{1,1} + x_{1,2} + x_{2,2} + x_{2,3} + x_{3,2} & \leq & 2 & (Constraint\ II : B4) \\
x_{i,j} & = & 0, 1, 1 \leq i, j \leq 3 &
\end{array}
$$

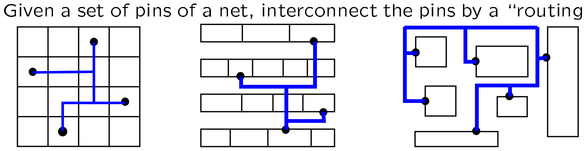
Hierarchical Global Routing

- Marek-Sadowska, "Router planner for custom chip design," ICCAD, 1986.
- At each level of the hierarchy, an attempt is made to minimize the cost of nets crossing cut lines.
- At the lowest level of the hierarchy, the layout surface is divided into $R \times R$ grid regions with boundary capacity equal to C tracks.
- Let R_l be the # of grid regions of a given cut line l; a cut line can be divided into $M = \frac{R_l}{C}$ sections.
- Global routing can be formulated as a linear assignment problem:
  - $x_{i,j} = 1$ if net i is assigned to section j; $x_{i,j} = 0$ otherwise.
  - Each net crosses the cut line exactly once: $\sum_{j=1}^{M} x_{ij} = 1, 1 \leq i \leq N$.
  - Capacity constraint of each section: $\sum_{i=1}^{N} x_{ij} \leq C, 1 \leq j \leq M$.
  - w_{ij}: cost of assigning net i to section j. Minimize $\sum_{i=1}^{N} \sum_{j=1}^{M} w_{ij} x_{ij}$.



The Routing-Tree Problem

- **Problem:** Given a set of pins of a net, interconnect the pins by a "routing tree."



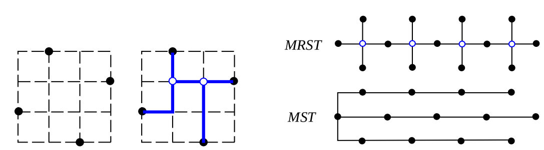
gate array *standard cell* *building block*

- **Minimum Rectilinear Steiner Tree (MRST) Problem:** Given n points in the plane, find a minimum-length tree of rectilinear edges which connects the points.
- $MRST(P) = MST(P \cup S)$, where P and S are the sets of original points and Steiner points, respectively.

minimum spanning tree MST *MRST*

Theoretic Results for the MRST Problem

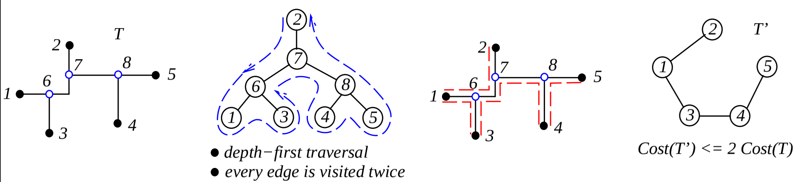
- **Hanan's Thm:** There exists an MRST with all Steiner points (set S) chosen from the intersection points of horizontal and vertical lines drawn points of P.
  - Hanan, "On Steiner's problem with rectilinear distance," *SIAM J. Applied Math.*, 1966.
- **Hwang's Theorem:** For any point set P, $\frac{Cost(MST(P))}{Cost(MRST(P))} \leq \frac{3}{2}$.
  - Hwang, "On Steiner minimal tree with rectilinear distance," *SIAM J. Applied Math.*, 1976.
- Best existing approximation algorithm: Performance bound $\frac{61}{48}$ by Foessmeier *et al.*
  - Foessmeier *et al*, "Fast approximation algorithm for the rectilinear Steiner problem," Wilhelm Schickard-Institut für Informatik, TR WSI-93-14, 93.
  - Zelikovsky, "An $\frac{11}{6}$ approximation algorithm for the network Steiner problem," *Algorithmica.*, 1993.



Hanan grid *Cost(MST)/Cost(MRST) −> 3/2*

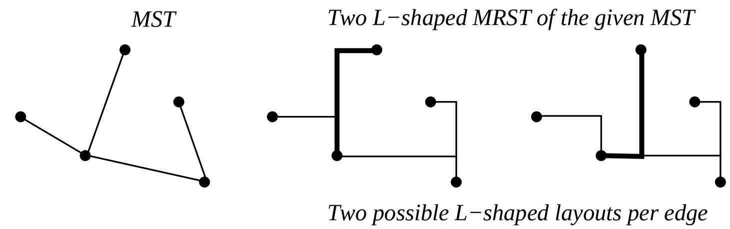
A Simple Performance Bound

- Easy to show that $\frac{Cost(MST(P))}{Cost(MRST(P))} \leq 2$.
- Given any MRST T on point set P with Steiner point set S, construct a spanning tree T' on P as follows:
  1. Select any point in T as a root.
  2. Perform a depth-first traversal on the rooted tree T.
  3. Construct T' based on the traversal.



Coping with the MRST Problem

- Ho, Vijayan, Wong, "New algorithms for the rectilinear Steiner problem," *IEEE TCAD*, 1990.
  1. Construct an MRST from an MST.
  2. Each edge is straight or L-shaped.
  3. Maximize overlaps by dynamic programming.
- About 8% smaller than $Cost(MST)$.

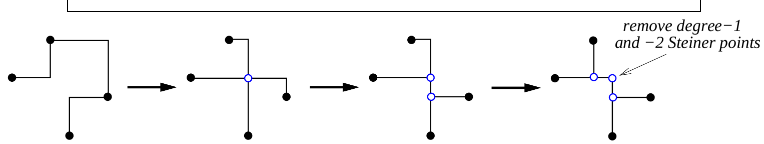


The Iterated 1-Steiner Heuristic for the MRST Problem

- Smith & Liebman, "Steiner trees, Steiner circuits and the interference problem in building design," *Engineering Optimization 4*, 1979.
- Extended by Kahng & Robins, " A new class of Steiner tree heuristics with good performance: the iterated 1-Steiner approach," *ICCAD*, 1990.

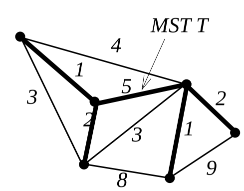
Algorithm: Iterated_1-Steiner(P)
P: set P of n points.

begin
$S \leftarrow \emptyset$;
/* $H(P \cup S)$: set of Hanan points */
/* $\Delta MST(A, B) = Cost(MST(A)) - Cost(MST(A \cup B))$ */
while $(Cand \leftarrow \{x \in H(P \cup S) | \Delta MST(P \cup S, \{x\}) > 0\} \neq \emptyset)$ **do**
Find $x \in Cand$ which maximizes $\Delta MST(P \cup S), \{x\})$;
$S \leftarrow S \cup \{x\}$;
Remove points in S which have degree ≤ 2 in $MST(P \cup S)$;
Output $MST(P \cup S)$;
end



Bounded-Radius (-Diameter) Minimum Spanning Tree

- **Problem:** Given a parameter $\epsilon \geq 0$ and a signal net with radius R (diameter D), find a minimum-cost spanning tree T with radius $r(T) \leq (1+\epsilon)R$ $(d(T) \leq (1+\epsilon)D)$.
  - Awerbuch, *et al.*, "Cost-sensitive analysis of communication protocols," *ACM Symp. Principles of Distributed Computing*, 1990.
  - Cong, Kahng, Robins, Sarrafzadeh, Wong, "Performance-driven global routing for cell based IC's," ICCD-91 (& TCAD, June 1992).
- MST (minimum spanning tree) $\longleftrightarrow$ minimum cost; SPT (shortest path tree) $\longleftrightarrow$ minimum radius.
- Question: How to find a spanning tree with a good trade-off between cost and radius?

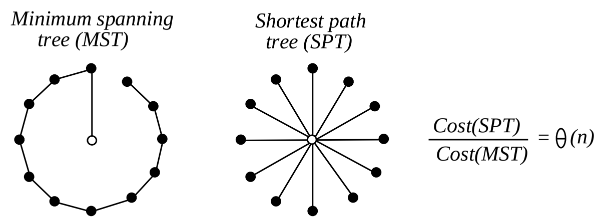


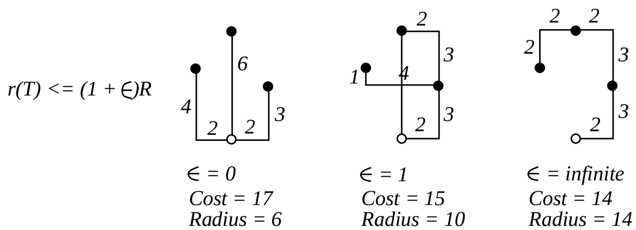
Cost(T)=1+2+5+1+2=11

Radius(T)=max{1+2, 1+5+1, 1+5+2}=8

MST vs SPT Trade-off

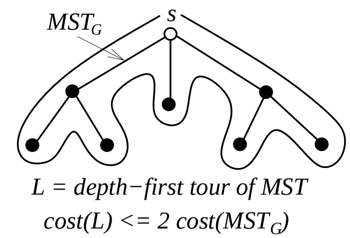
- $Cost(SPT)$ may be $\Omega(|n|)$ times larger than $Cost(MST)$.





Algorithm: Bounded-Radius_Bounded-Cost_Spanning_Tree(G)

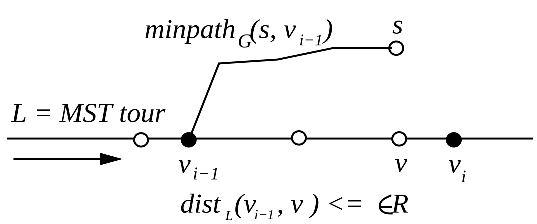
```
begin
Compute MST_G and SPT_G;
Q ← MST_G;
L ← depth-first tour of MST_G;
S ← 0;
for i ← 1 to |L| − 1
    S ← S + cost(L_i, L_{i+1});
    if S ≥ ε · dist_G(s, L_{i+1}) then
        Q ← Q ∪ minpath_G(s, L_{i+1});
        S ← 0;
T = shortest path tree of Q;
end
```



Bounded-Radius Bounded-Cost Spanning Tree

- For any weighted graph G and parameter ϵ, the routing tree T constructed by the algorithm has radius $r(T) \leq (1+\epsilon)R$.
- v_{i-1}: the last node before v on L for which we added $minpath_G(s, v_{i-1})$ to Q.

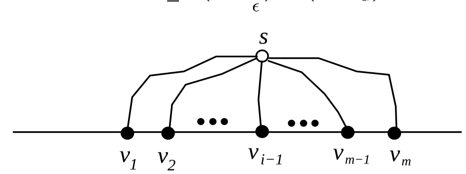
$$\begin{aligned} dist_T(s,v) &\leq dist_T(s,v_{i-1}) + dist_L(v_{i-1},v) \\ &\leq dist_G(s,v_{i-1}) + \epsilon R \\ &\leq R + \epsilon R \\ &= (1+\epsilon)R \end{aligned}$$



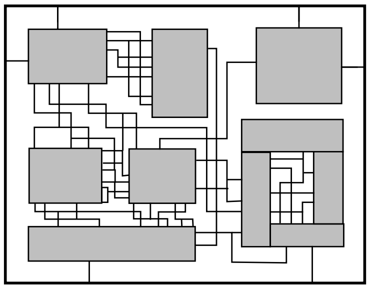
Bounded-Radius Bounded-Cost Spanning Tree

- For any weighted graph G and parameter ϵ, the routing tree T constructed by the algorithm has cost $cost(T) \leq (1+\frac{2}{\epsilon})cost(MST_G)$.
- Let $v_1, v_2, \ldots, v_m$ be the set of nodes to which the algorithm added shortest paths from source s.

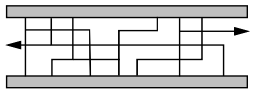
$$\begin{aligned} cost(T) &\leq cost(MST_G) + \sum_{i=1}^{m} dist_G(s, v_i) \\ dist_L(v_{i-1}, v_i) &\geq \epsilon \cdot dist_G(s, v_i) \\ cost(T) &\leq cost(MST_G) + \sum_{i=1}^{m} \frac{dist_G(v_{i-1}, v_i)}{\epsilon} \\ &\leq cost(MST_G) + \frac{cost(L)}{\epsilon} \\ &\leq cost(MST_G) + \frac{2 \cdot cost(MST_G)}{\epsilon} \\ &\leq (1+\frac{2}{\epsilon})cost(MST_G) \end{aligned}$$



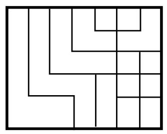
Channel & Switchbox Routing



Detailed routing



channel routing



switchbox routing