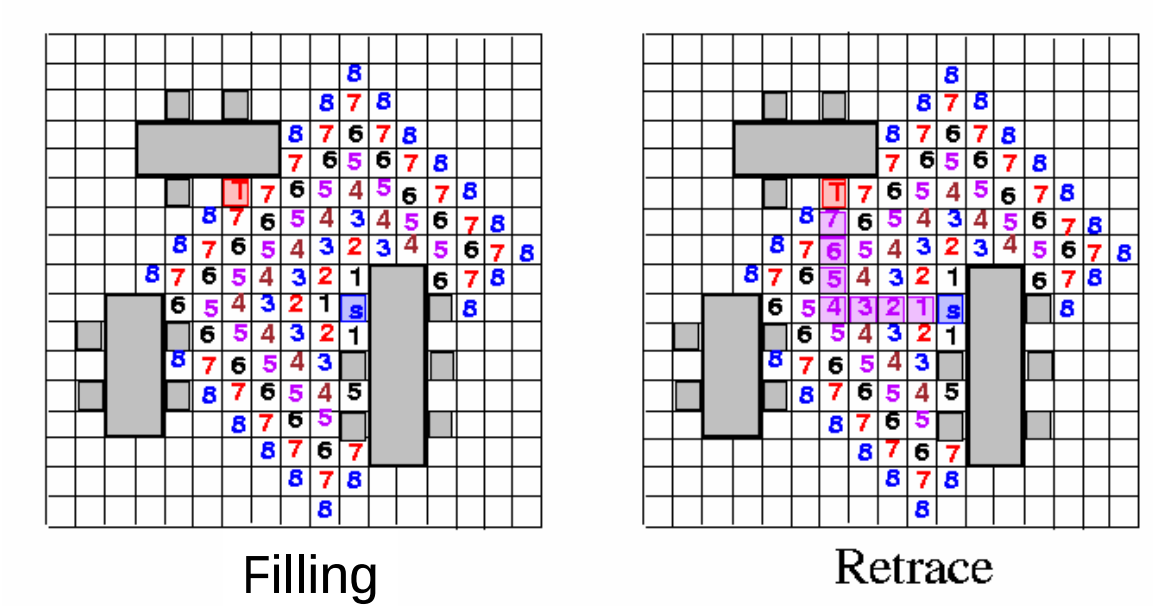
Unit 5D: Maze (Area) and Global Routing

- Course contents
  - Routing basics
  - Maze (area) routing
  - Global routing
- Readings
  - Chapters 9.1, 9.2, 9.5



Routing

placement

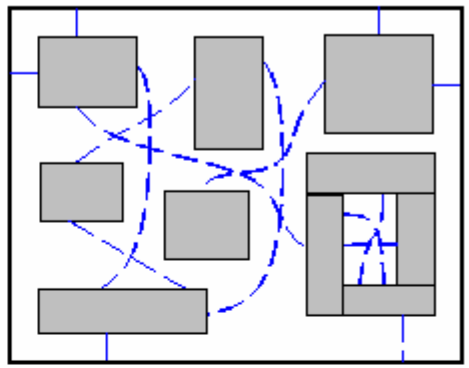
- Generates a "loose" route for each net.
- Assigns a list of routing regions to each net without specifying the actual layout of wires.

global routing

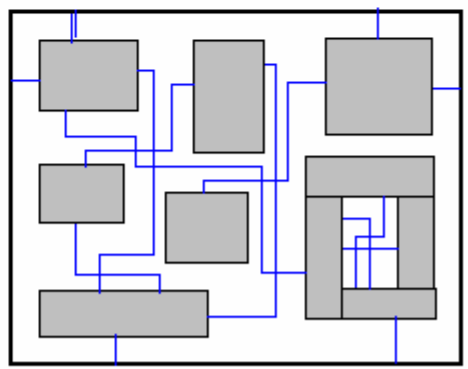
detailed routing

- Finds the actual geometric layout of each net within the assigned routing regions.

compaction



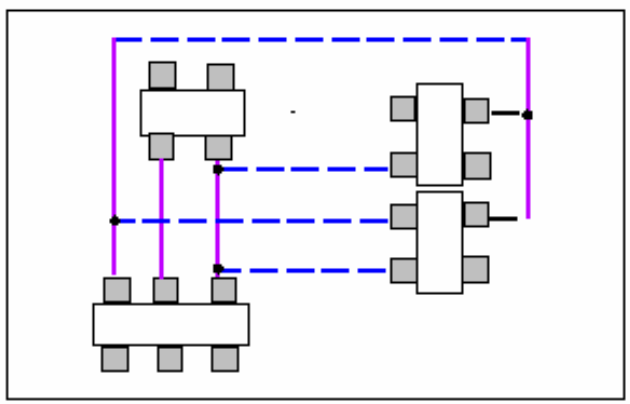
Global routing



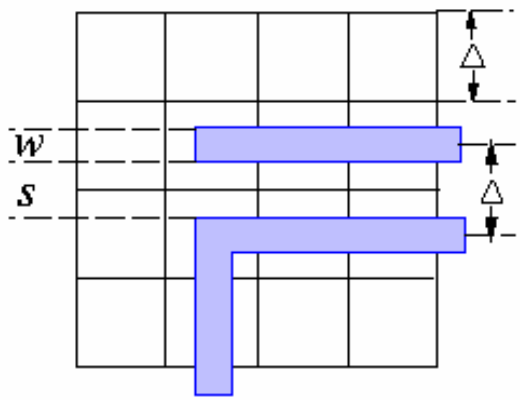
Detailed routing

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?
  - Process variations?

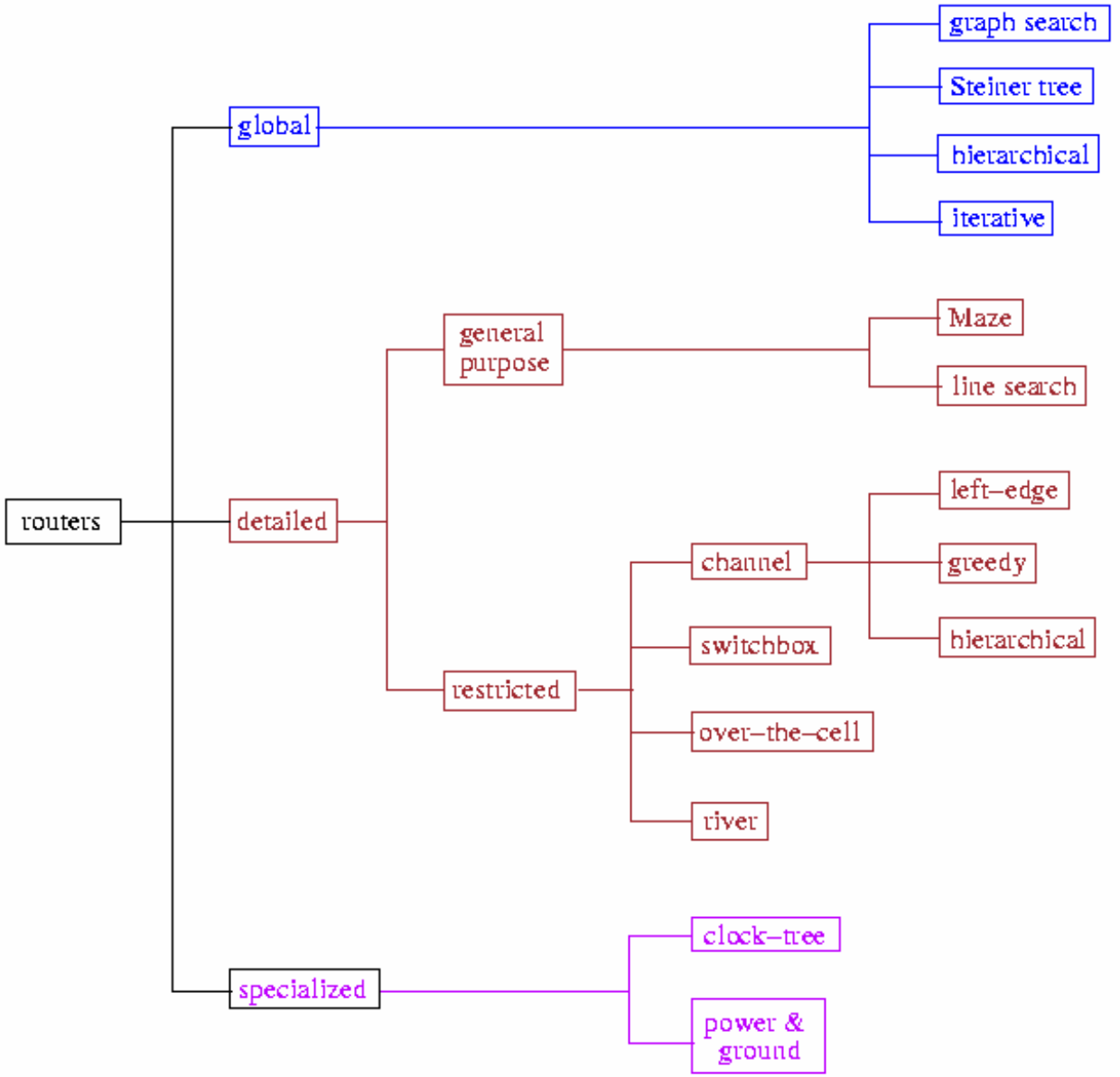


Two–layer routing



Geometrical constraint

Classification of Routing

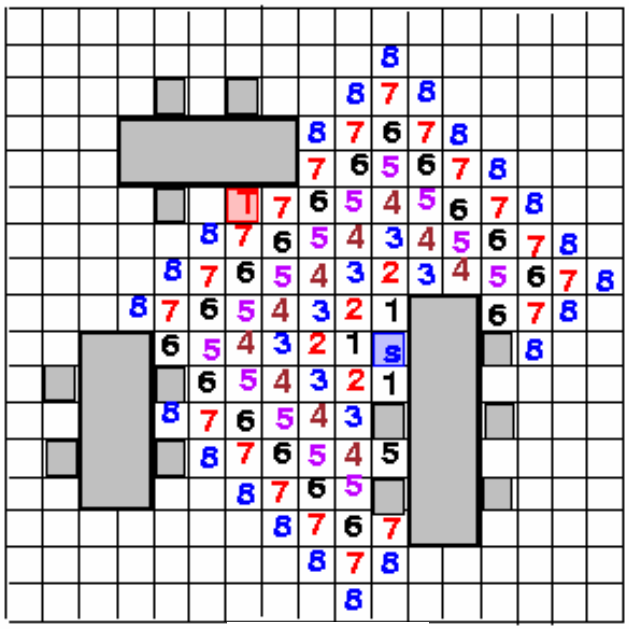


Maze Router: Lee Algorithm

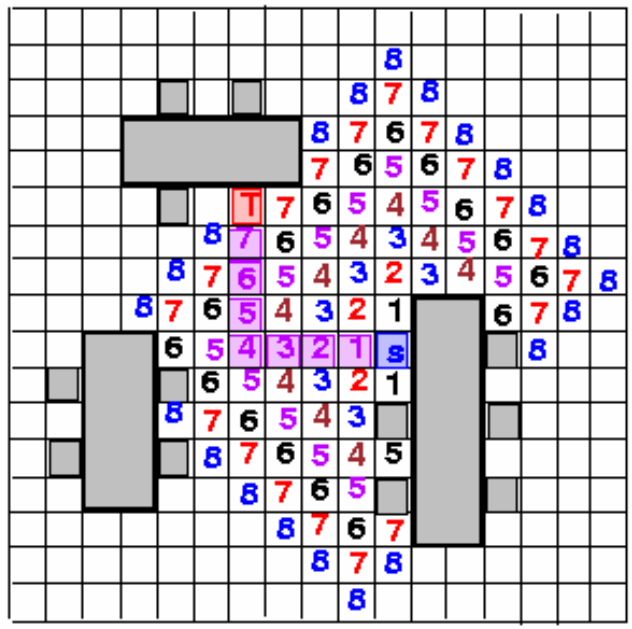
- Lee, “An algorithm for path connection and its application,” *IRE Trans. Electronic Computer*, EC-10, 1961.
- Discussion mainly on single-layer routing
- **Strengths**
  - Guarantee to find connection between 2 terminals if it exists.
  - Guarantee minimum path.
- **Weaknesses**
  - Requires large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

Lee Algorithm

- Find a path from *S* to *T* by “wave propagation”.



Filling

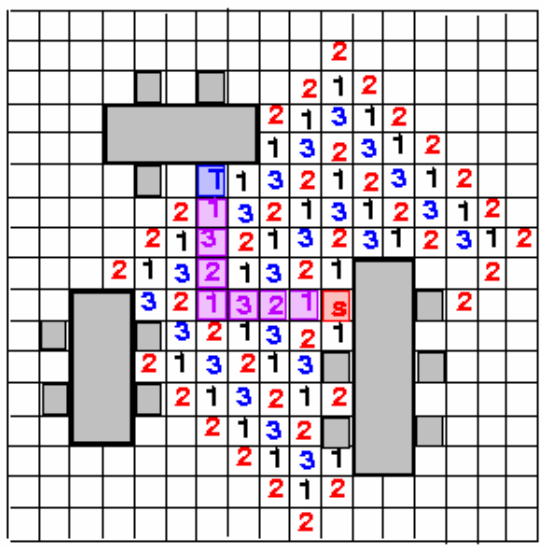


Retrace

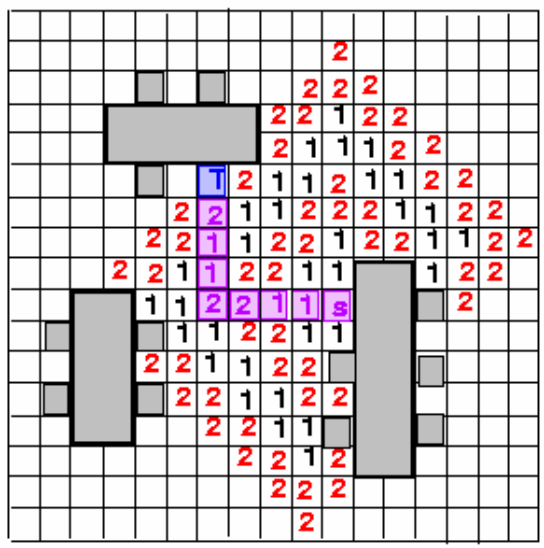
- Time & space complexity for an $M \times N$ grid: $O(MN)$ (**huge!**)

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, …; states: 1, 2, 3, *empty*, *blocked* (3 bits required)
- Way 2: coding sequence 1, 1, 2, 2, 1, 1, 2, 2, …; states: 1, 2, *empty*, *blocked* (need only 2 bits)



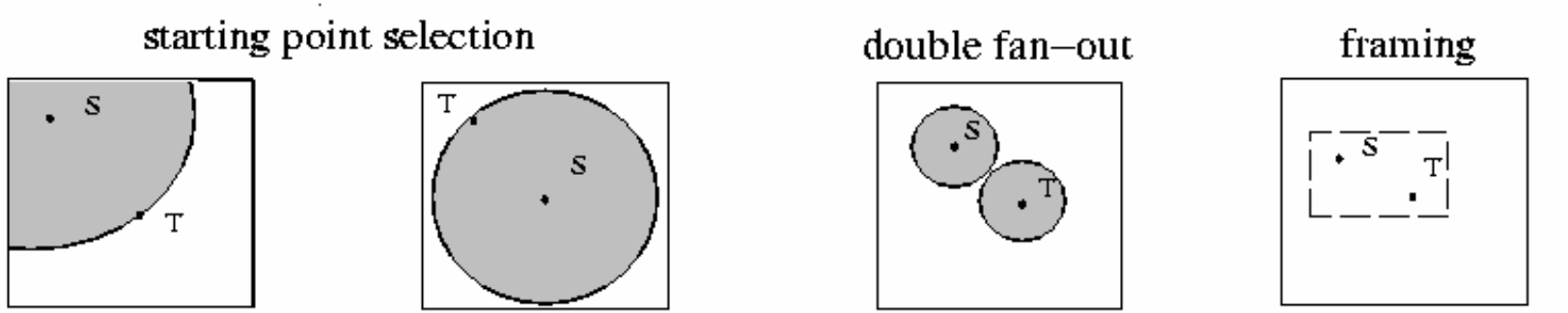
Sequence: 1, 2, 3, 1, 2, 3, ...



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

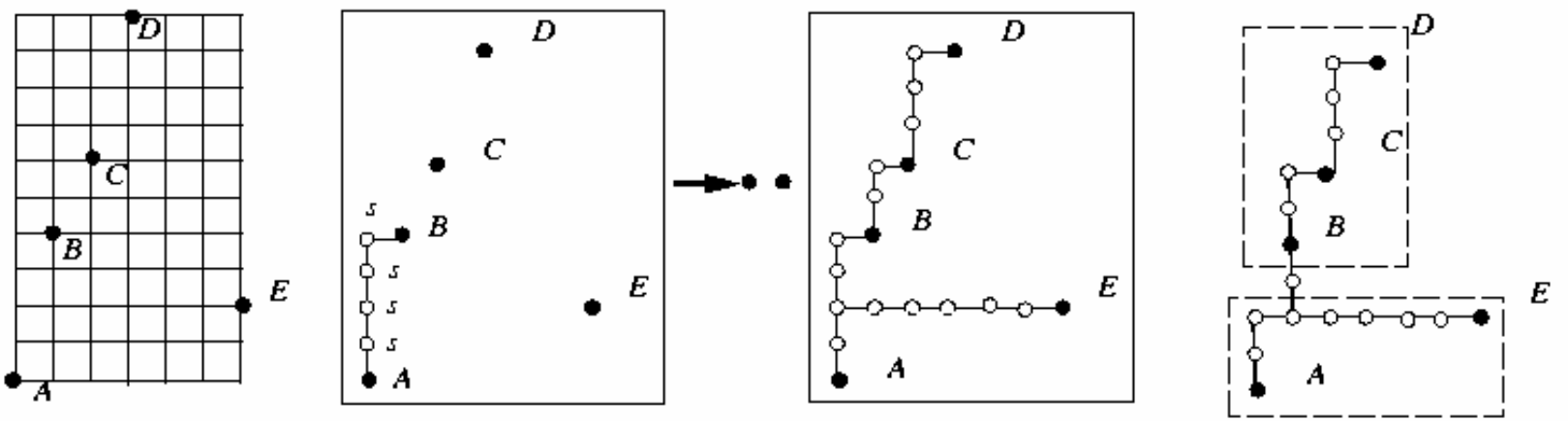
Reducing Running Time

- 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.



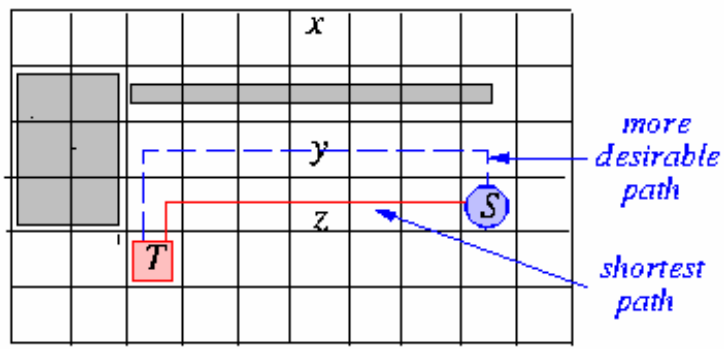
Connecting Multi-Terminal Nets

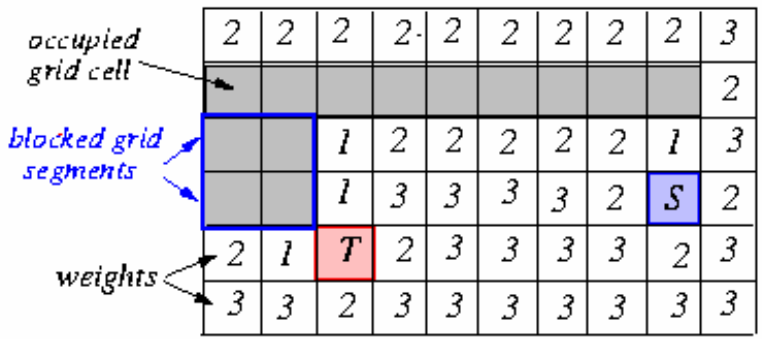
- Step 1: Propagate wave from the source *s* to the closet target.
- Step 2: Mark ALL cells on the path as *s*.
- Step 3: Propagate wave from ALL *s* cells to the other cells.
- Step 4: Continue until all cells are reached.
- Step 5: Apply heuristics to further reduce the tree cost.



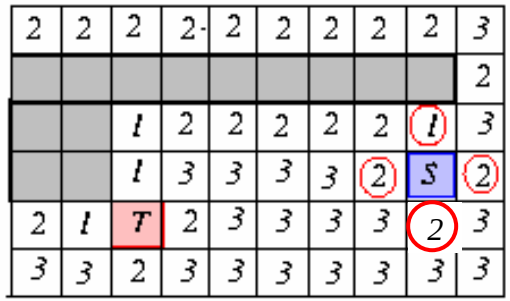
Routing on a Weighted Grid

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

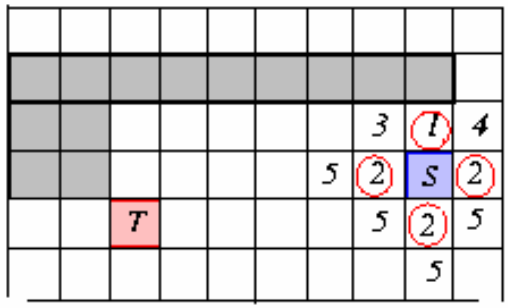




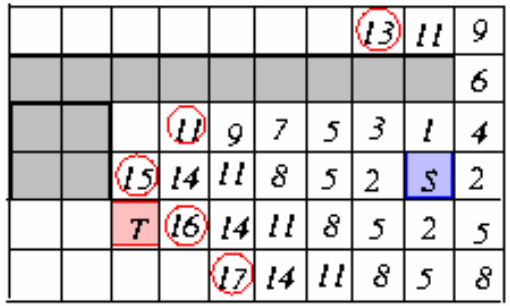
A Routing Example on a Weighted Grid

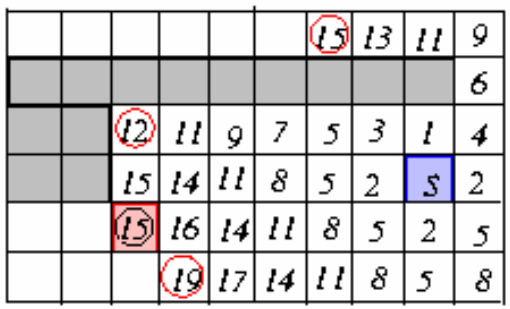


initialize cell weights

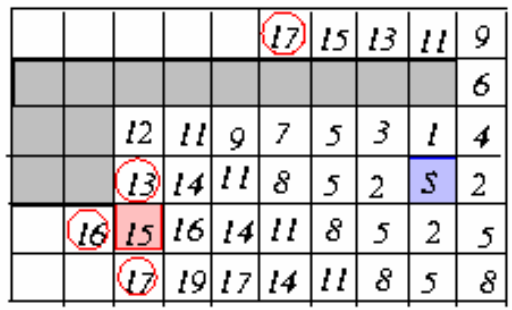


wave propagation

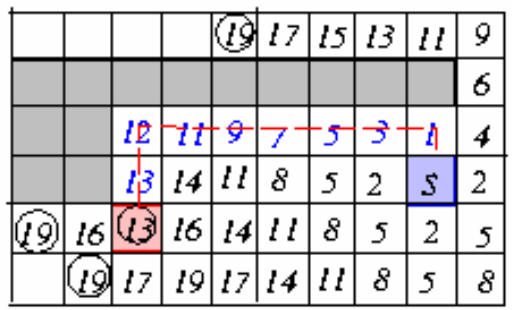




first wave reaches the target



finding other paths



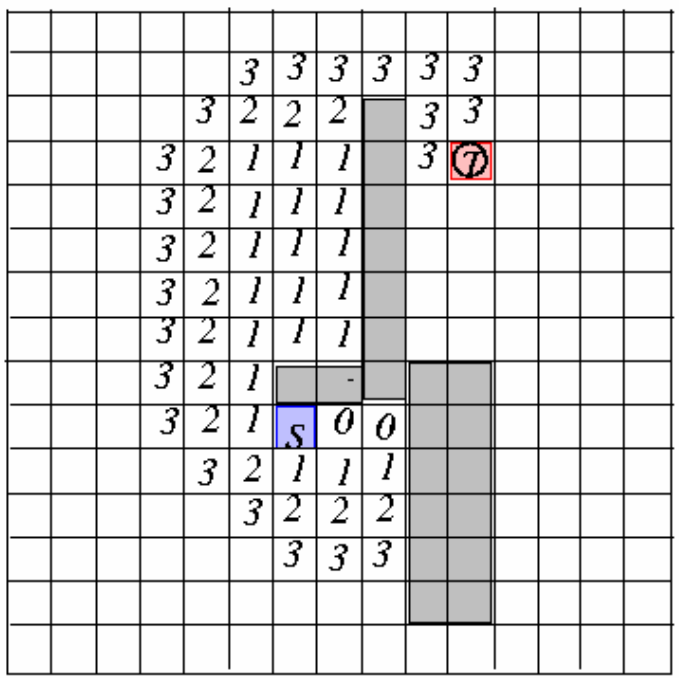
min−cost path found

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, $l(P)$: $l(P) = MD(S, T) + 2\,d(P)$.
  - $MD(S, T)$ 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.

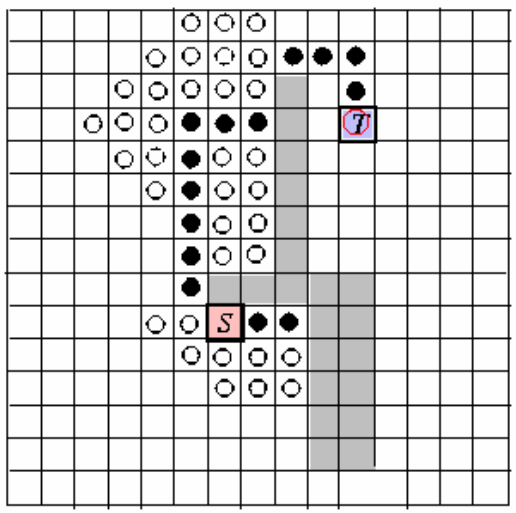
Hadlock's Algorithm (cont'd)

- $d(P)$: # 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.

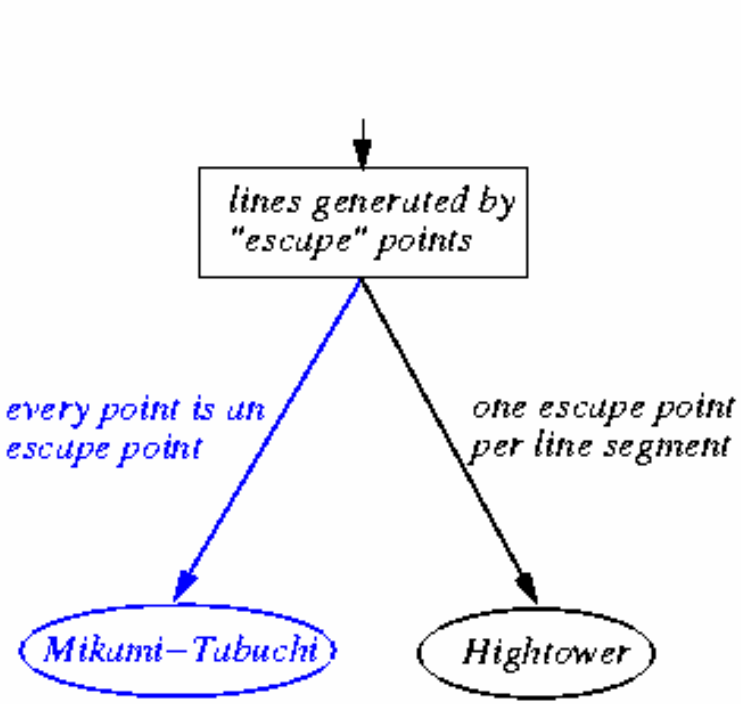


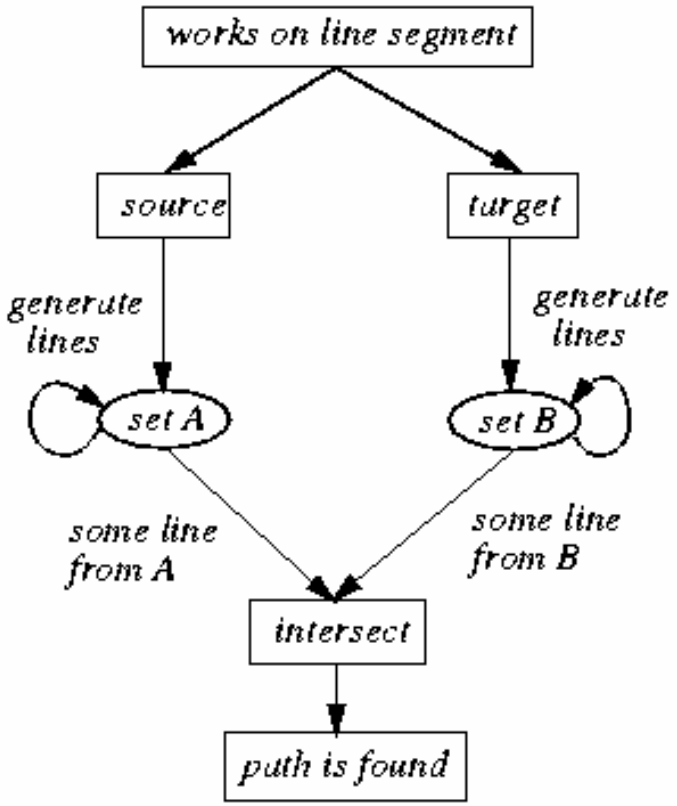
Soukup's Algorithm

- 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

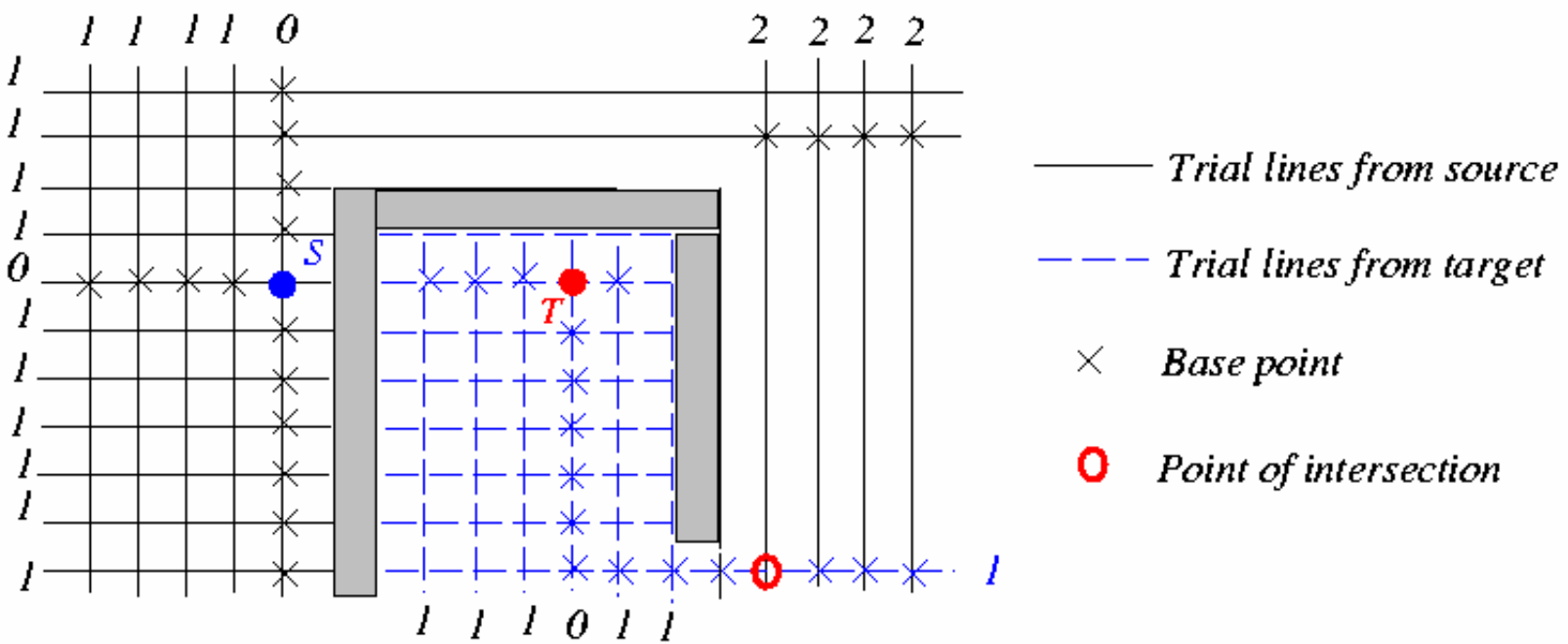




- Time and space complexities: $O(L)$, where L is the # of line segments generated.

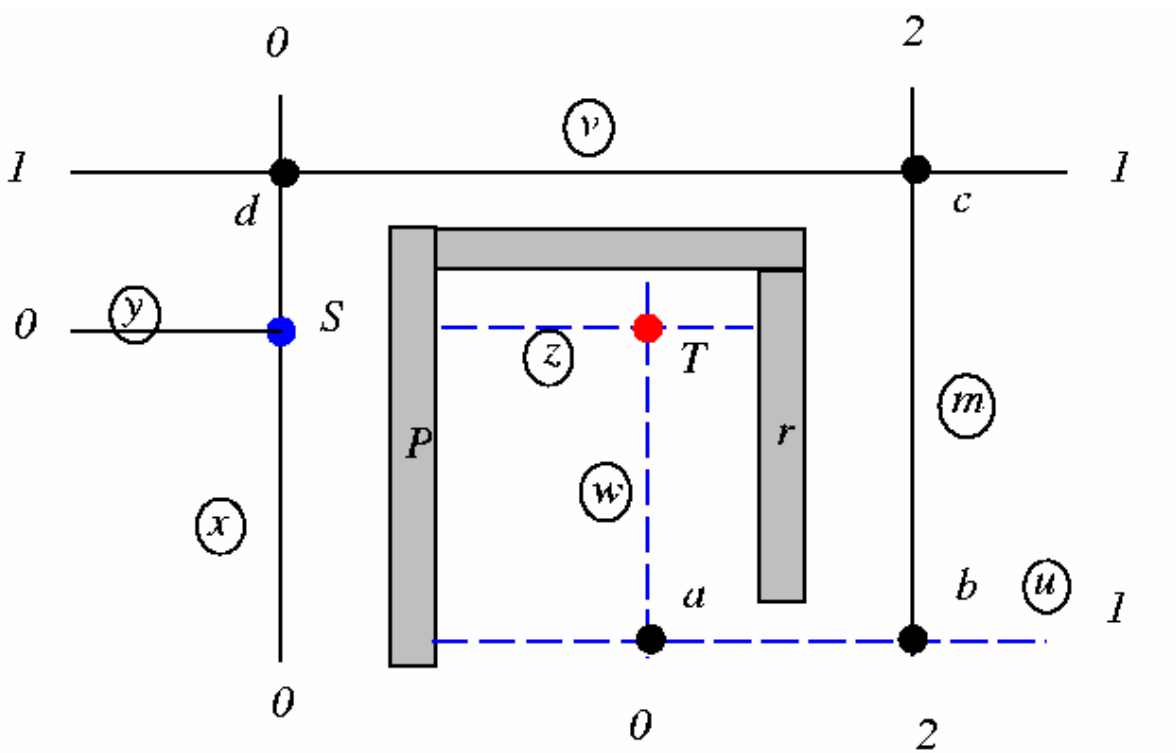
Mikami-Tabuchi's Algorithm

- 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.



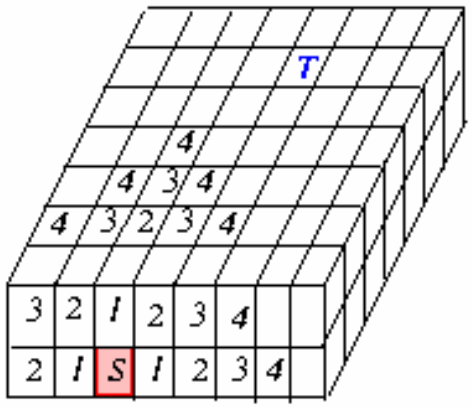
Comparison of Algorithms

	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

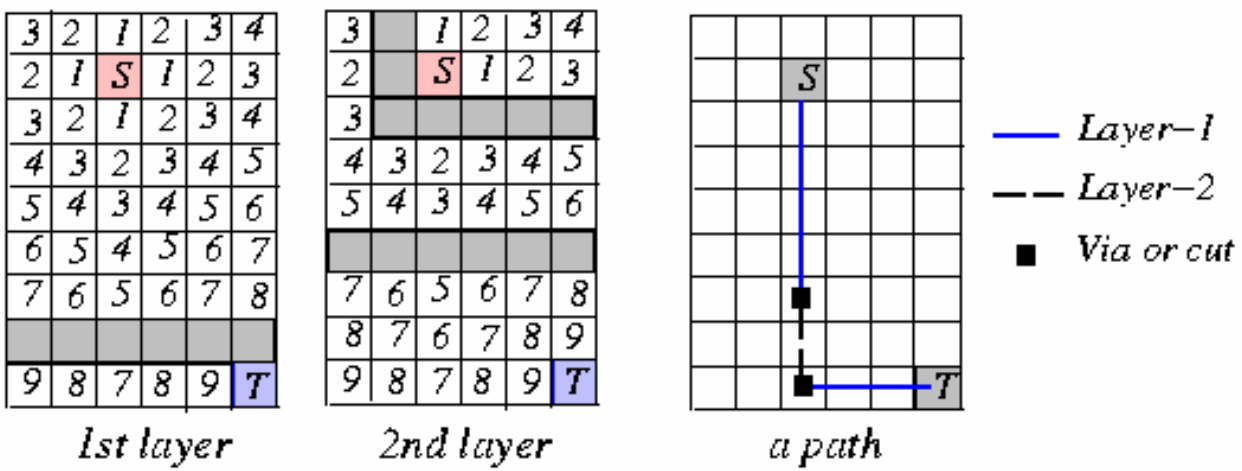
- Soukup, Mikami, and Hightower all adopt some sort of line-search operations $\Rightarrow$ cannot guarantee shortest paths.

Multi-layer Routing

- 3-D grid:

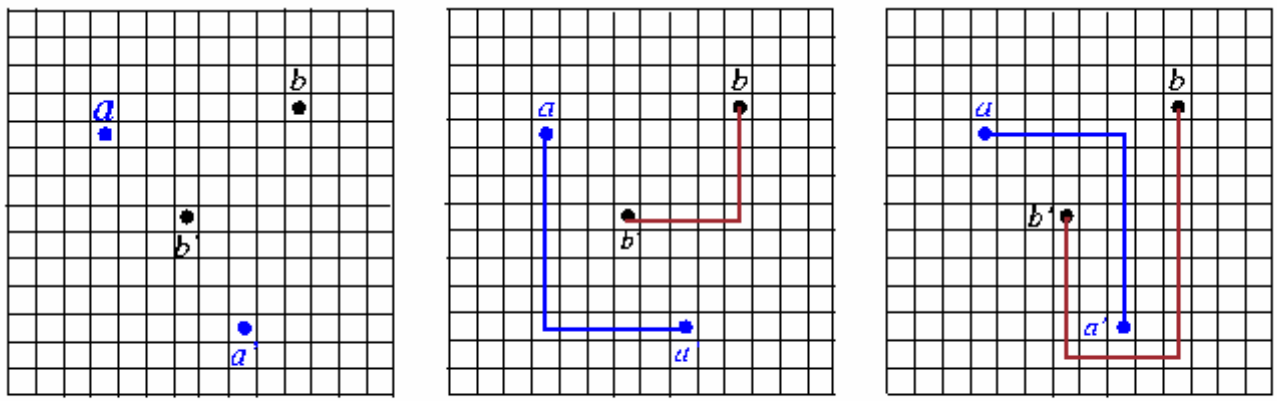


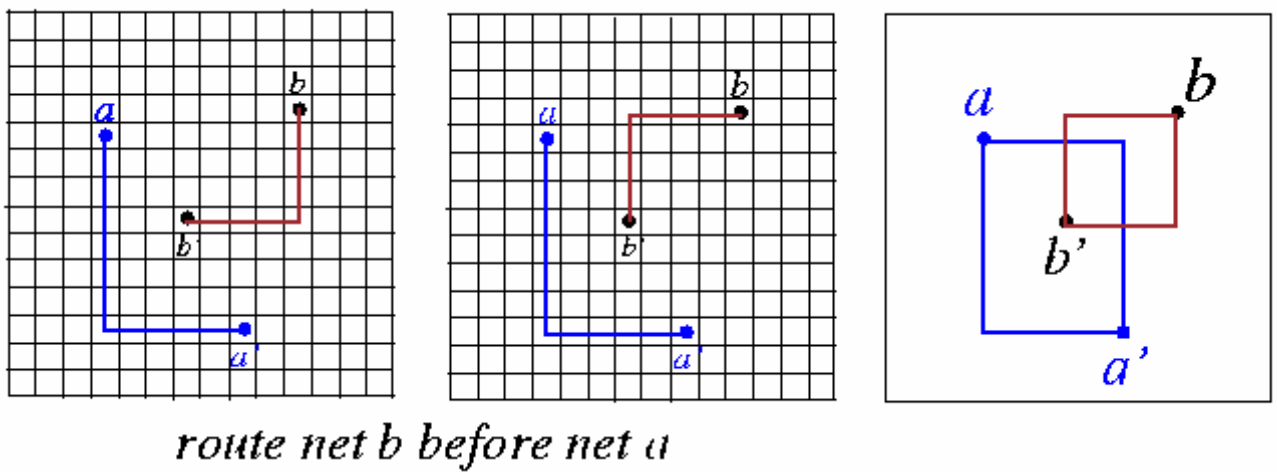
- Two planar arrays:
  - Neglect the weight for inter-layer connection through via.
  - Pins are accessible from both layers.



Net Ordering

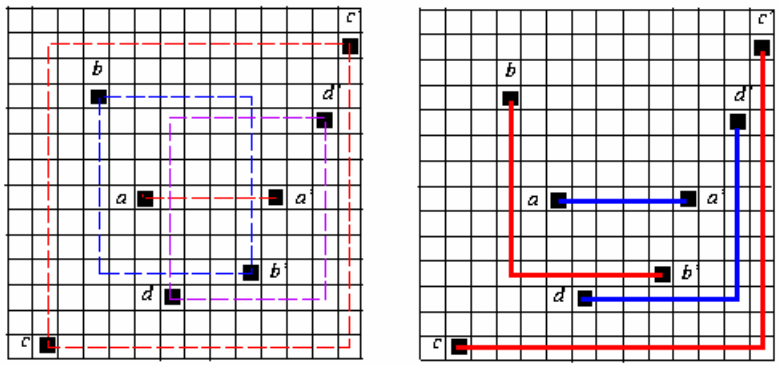
- Net ordering greatly affects routing solutions.
- In the example, we should route net *b* before net *a*.



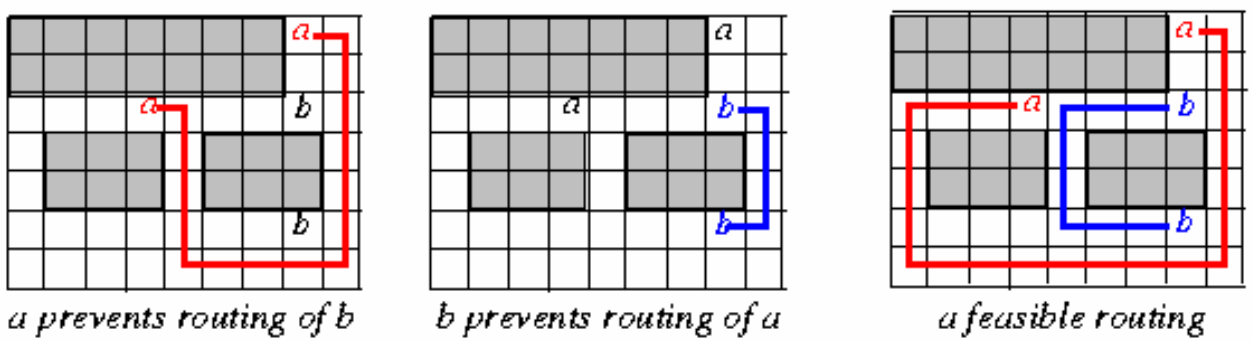


Net Ordering (cont'd)

- 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.



- A mutually intervening case:

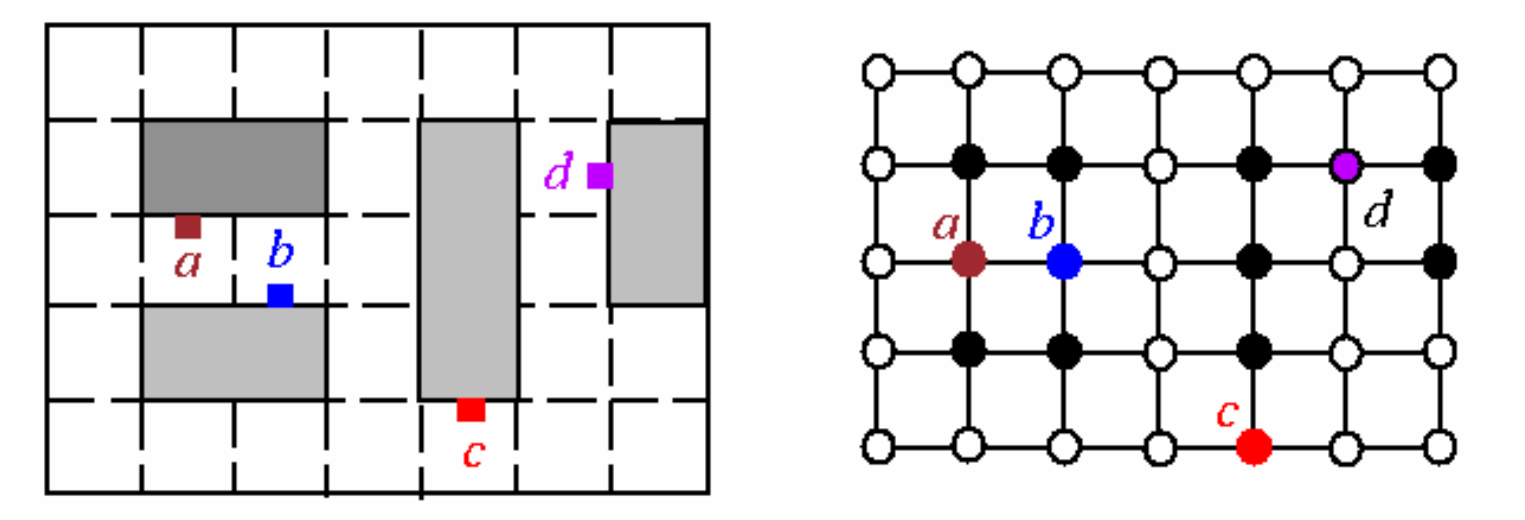


Rip-Up and Re-routing

- 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.

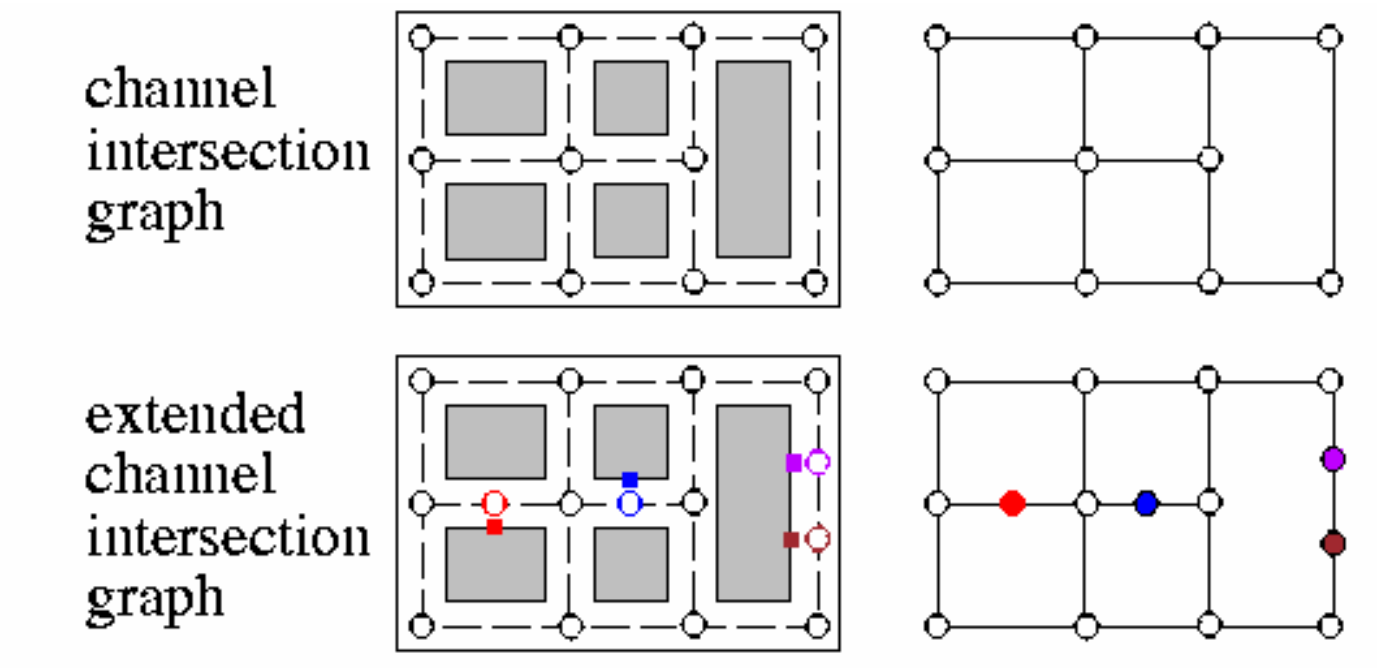
Graph Models for Global Routing: Grid Graph

- 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

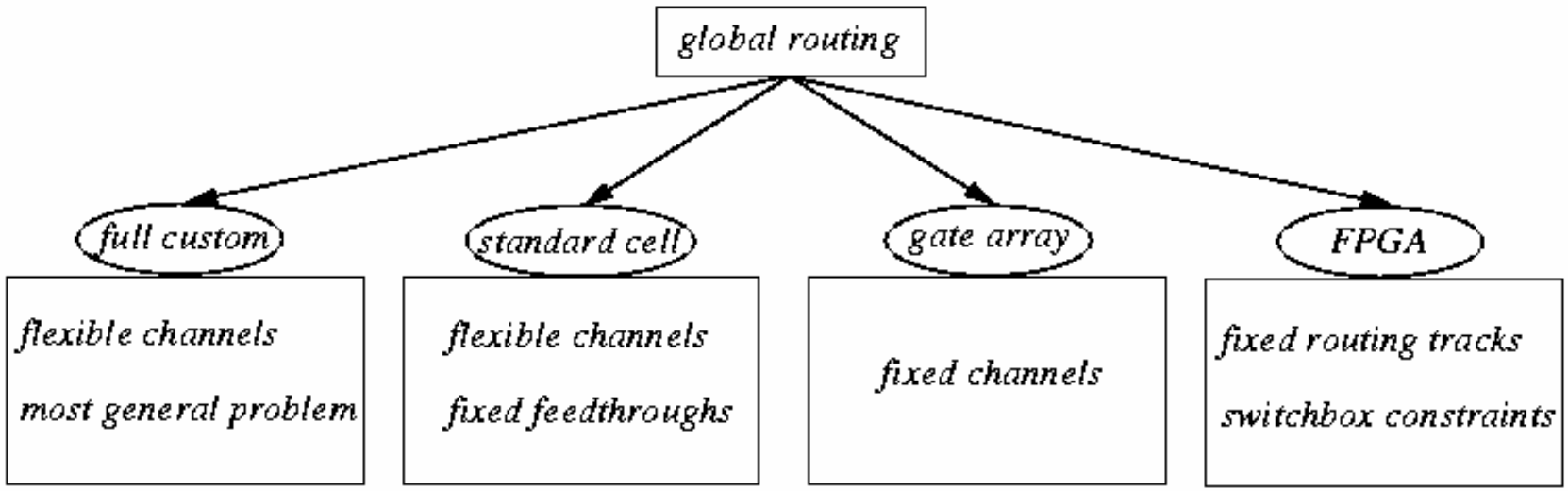
- Channels are represented as edges.
- Channel intersections are represented as vertices.
- Edge weight represents channel capacity.
- Extended channel intersection graph: terminals are also represented as vertices.



Global-Routing Problem

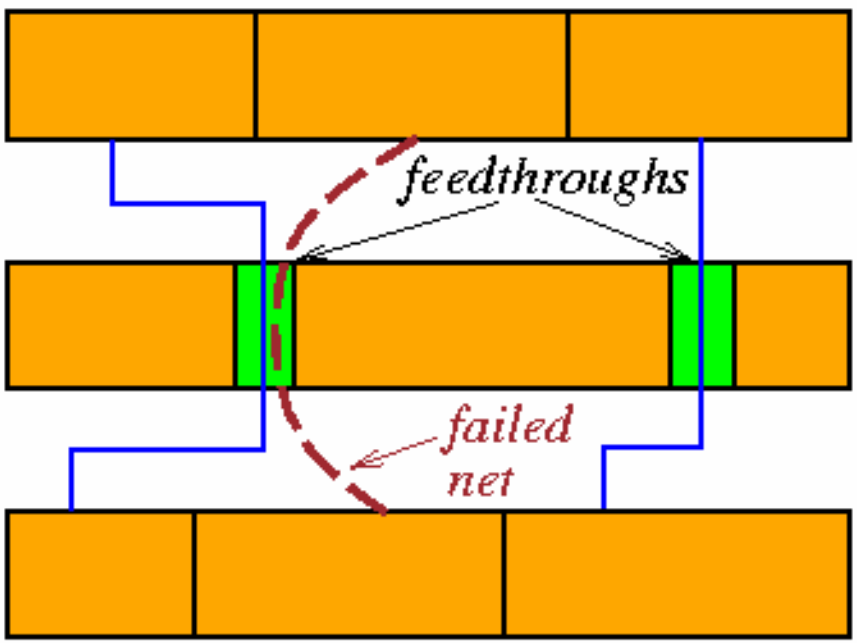
- Given a netlist N={N_1, N_2, …, N_n }, a routing graph G = (V, E), find a Steiner tree T_i for each net N_i, $1 \le i \le n$, such that $U(e_j) \le c(e_j)$, $\forall\ e_j \in E$ and $\sum_{i=1}^{n} L(T_i)$ is minimized, where
  - $c(e_j)$: capacity of edge e_j;
  - x_{ij}=1 if e_j is in T_i; x_{ij} = 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



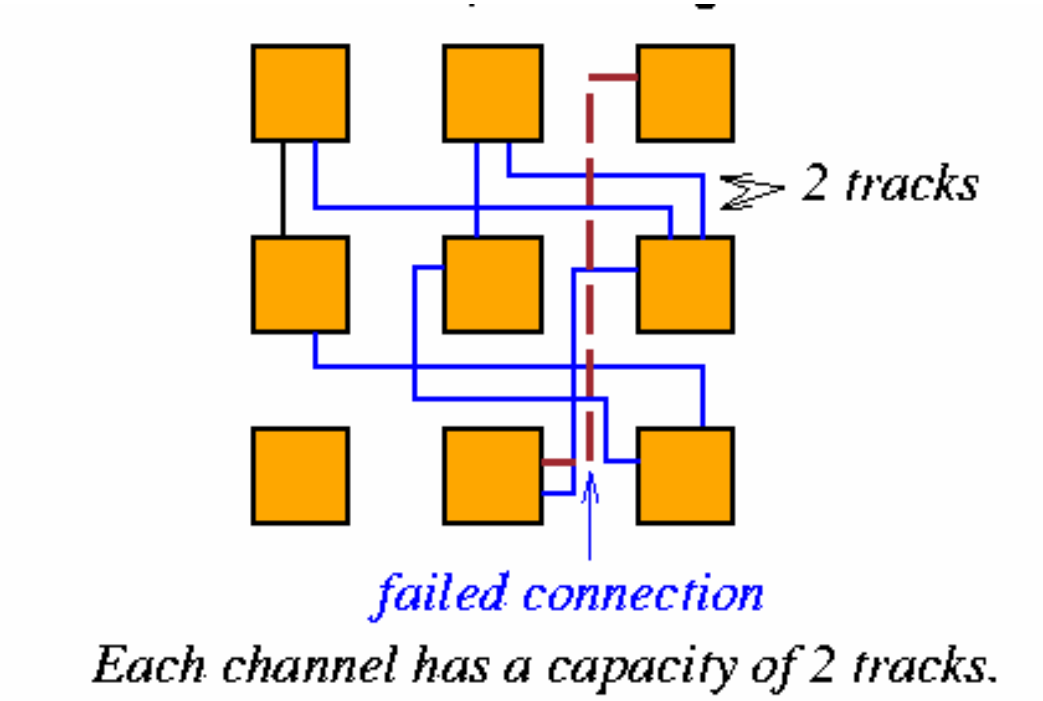
Global Routing in Standard Cell

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



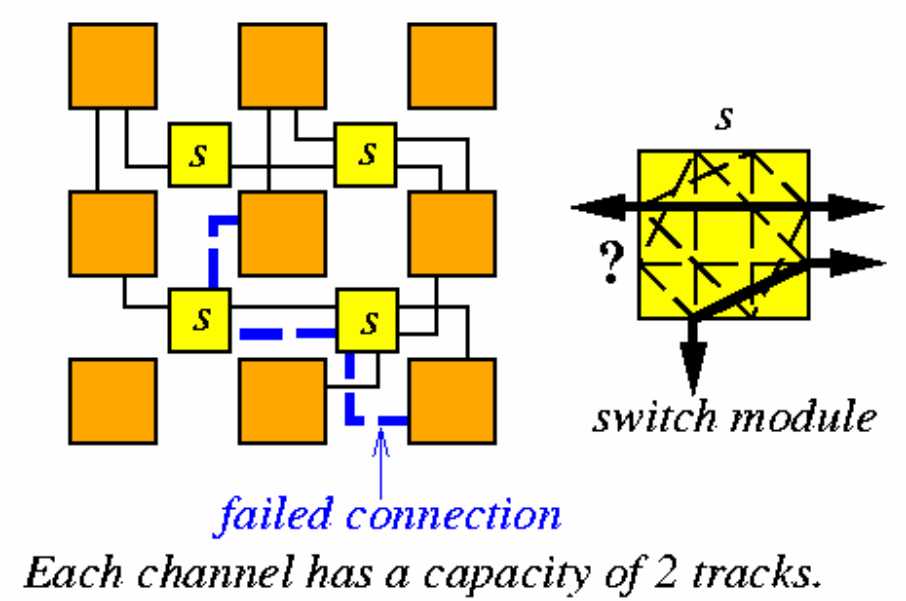
Global Routing in Gate Array

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



Global Routing in FPGA

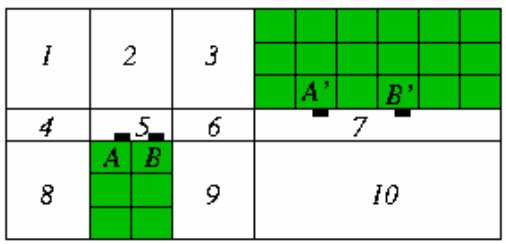
- Objective
  - Guarantee 100% routability.
  - Consider **switch-module architectural constraints.**
- For performance-driven routing,
  - **Minimize # of switches used.**
  - Minimize the maximum wire length.
  - Minimize the maximum path length.

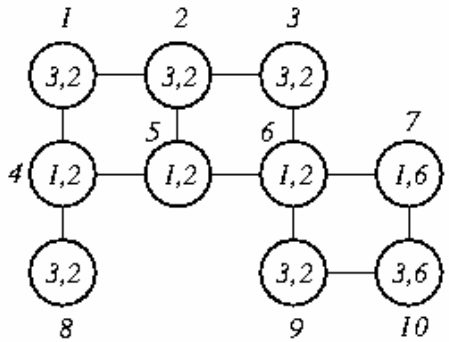


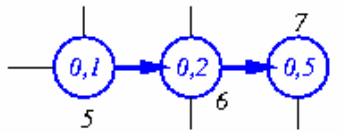
Each channel has a capacity of 2 tracks.

Global-Routing: Maze Routing

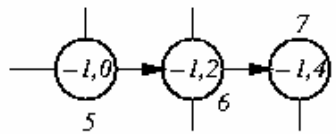
- Routing channels may be modelled by a weighted undirected graph called **channel connectivity graph**.
- Node ↔ channel; edge ↔ two adjacent channels; capacity: (*width*, *length*)



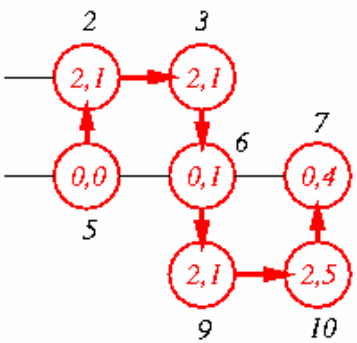




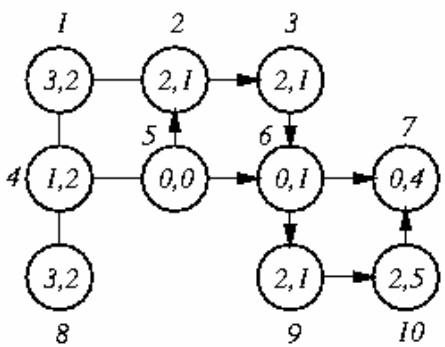
route A–A' via 5–6–7



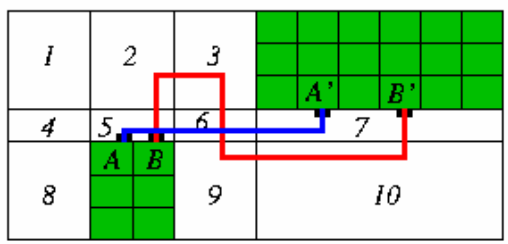
route B–B' via 5–6–7



route B–B' via 5–2–3–6–9–10–7



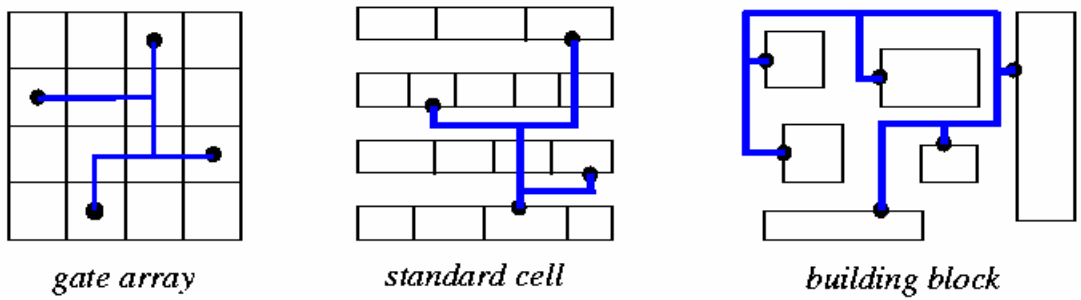
updated channel graph



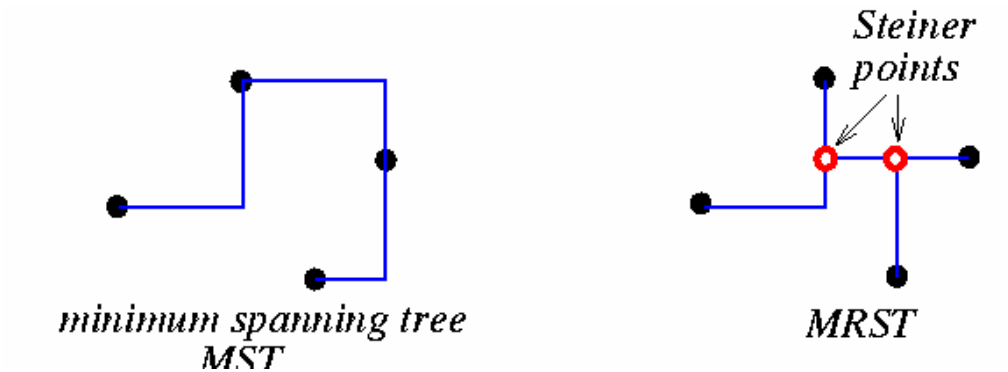
maze routing for nets A and B

The Routing-Tree Problem

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

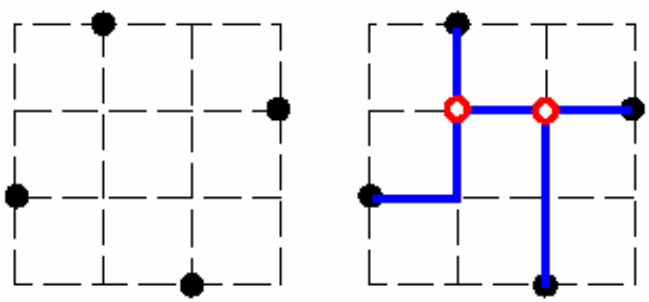


- **Minimum Rectilinear Steiner Tree (MRST) Problem:** Given *n* points in the plane, find a minimum-length tree of rectilinear edges which connects the points.
- *MRST*(*P*) = *MST*(*P* ∪ *S*), where *P* and *S* are the sets of original points and Steiner points, respectively.

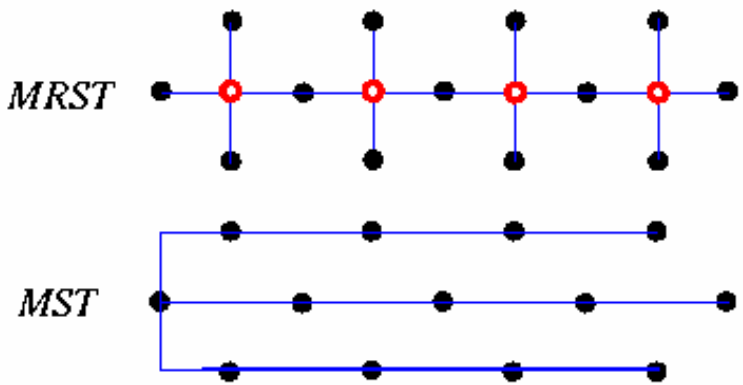


Theoretic Results for the MRST Problem

- **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 61/48 by Foessmeier *et al*.



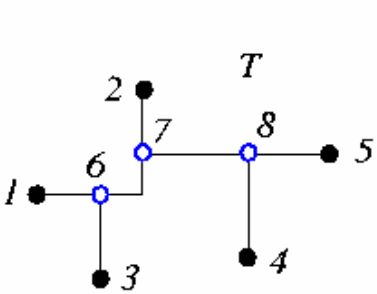
Hanan grid

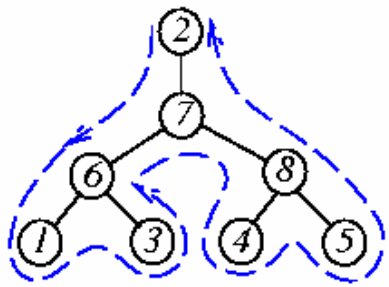


Cost(MST)/Cost(MRST) –> 3/2

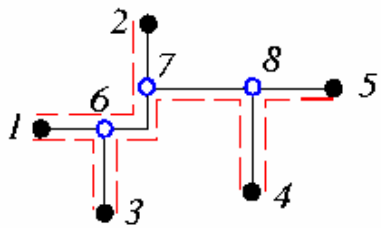
A Simple Performance Bound

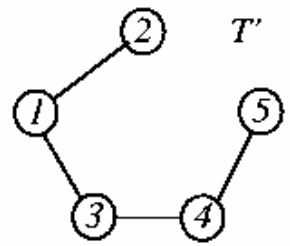
- Easy to show that $\frac{Cost(MST(P))}{Cost(MRST(P))} \leq 2$
- Given any MRST *T* on point set *P* with Steiner point set *S*, construct a spanning tree *T'* on *P* as follows:
  1. Select any point in *T* as a root.
  2. Perform a depth-first traversal on the rooted tree *T*.
  3. Construct *T'*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





- *depth–first traversal*
- *every edge is visited twice*

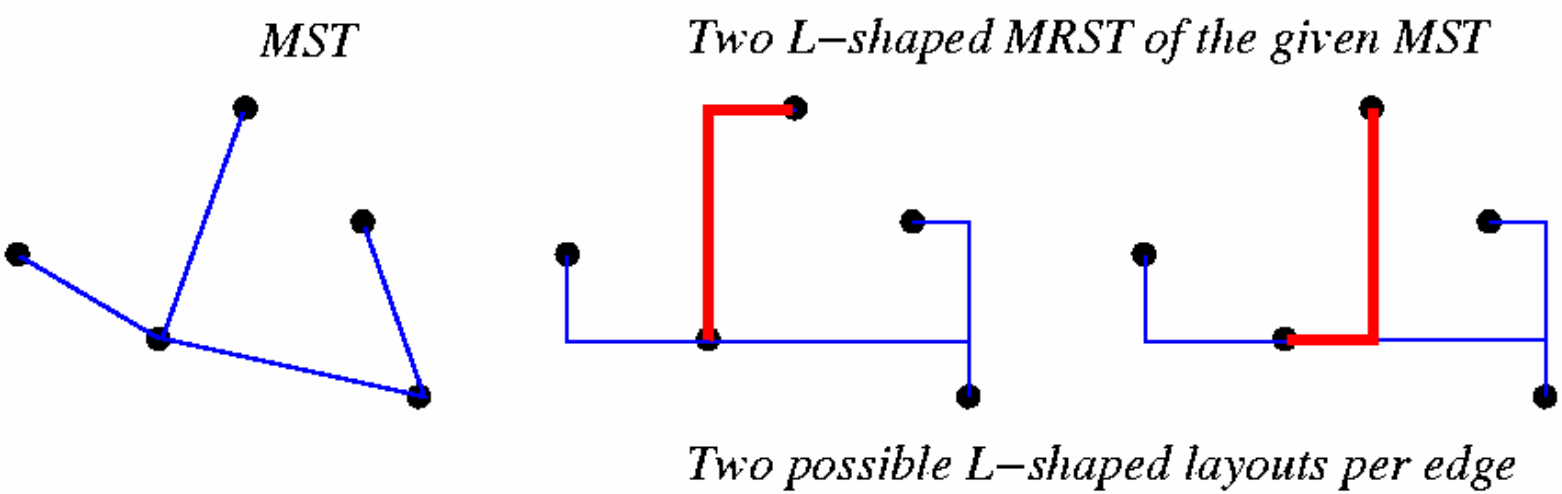




Cost(T') <= 2 Cost(T)

Coping with the MRST Problem

- Ho, Vijayan, Wong, “New algorithms for the rectilinear Steiner problem,”
  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*).



Iterated 1-Steiner Heuristic for MRST

- Kahng & Robins, "A new class of Steiner tree heuristics with good performance: the iterated 1-Steiner approach," *ICCAD-90.*.

Algorithm: Iterated_1-Steiner(*P*)
***P*: set *P* of *n* points.**
1 begin
2 $S \leftarrow \emptyset$;
/* $H(P \cup S)$: set of Hanan points */
/* $\Delta MST(A, B) = Cost(MST(A)) - Cost(MST(A \cup B))$ */
3 while ($Cand \leftarrow \{x \in H(P \cup S) | \Delta MST(P \cup S, \{x\}) > 0\} \neq \emptyset$) do
4 Find $x \in C$ and which maximizes $\Delta MST(P \cup S), \{x\})$;
5 $S \leftarrow S \cup \{x\}$;
6 Remove points in *S* which have degree ≤ 2 in $MST(P \cup S)$;
7 Output $MST(P \cup S)$;
8 end