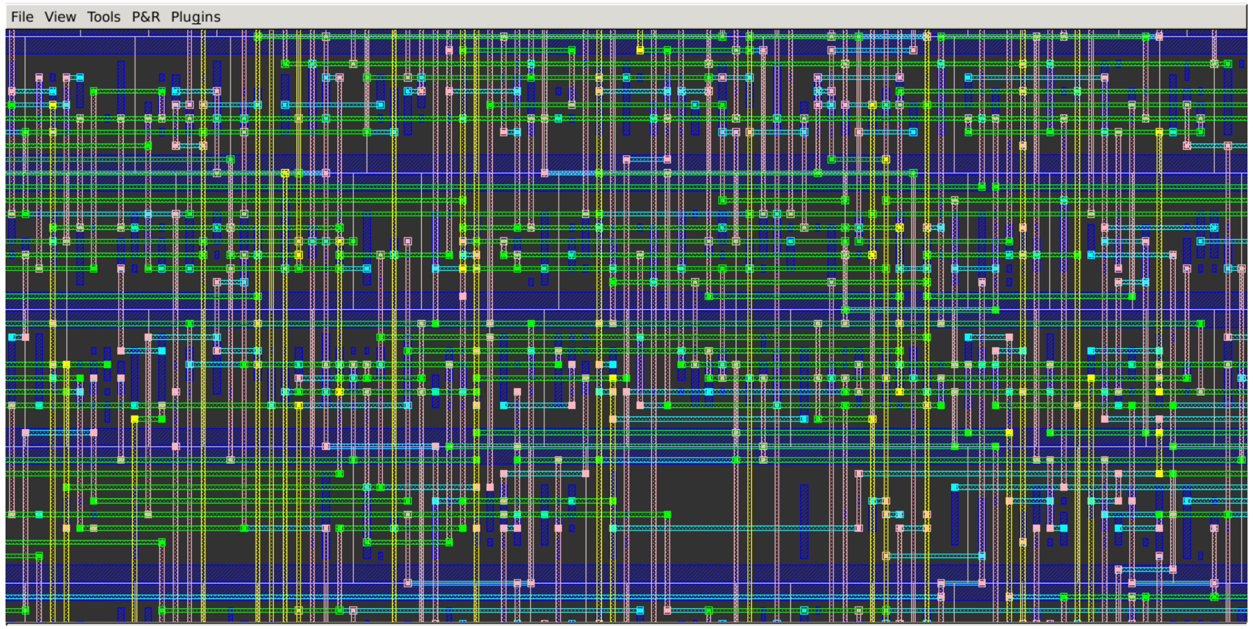
Routing

VLSI CAD

Compiled by Oleg Venger

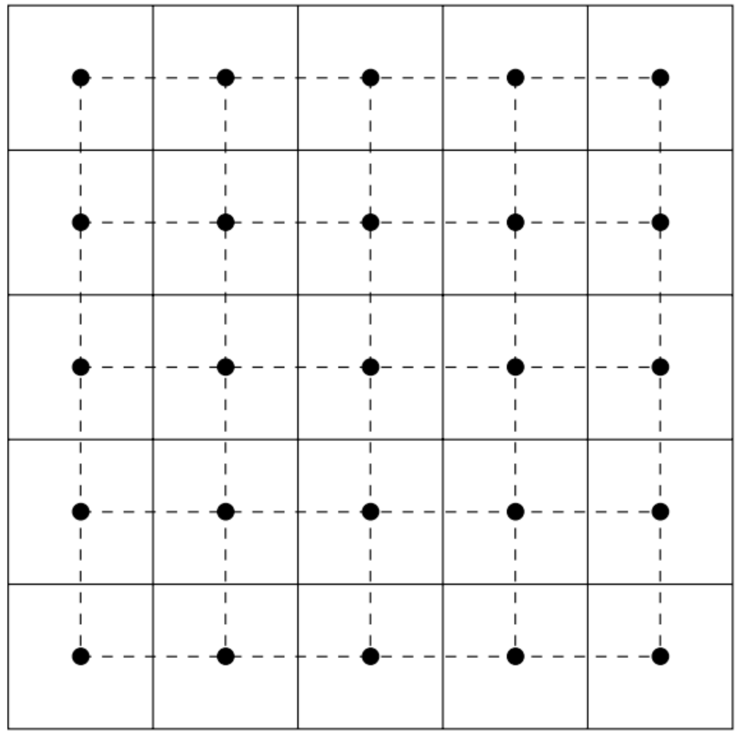
Routing problem

- The goal of routing is to produce complete wire layouts for all nets in a given placed netlist
- Because of the complexity it is solved in two steps:
  - Global routing defines approximate solution on a coarse grid
  - Detailed routing then finds the exact routes for all nets



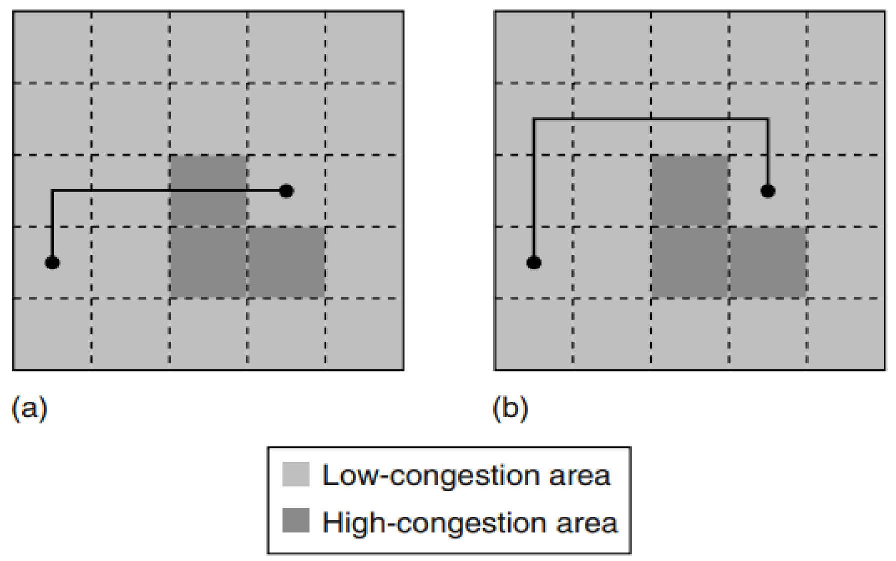
Tile-based graph model for global routing

- Routing is done on metal layers on top of cells
  - Each layer is used for either horizontal or vertical routing
- Layout is partitioned into rectangular regions and grid graph G is formed such that:
  - There is a vertex in each tile
  - Edges exist between neighbor tiles
- Each net terminal is assumed to lie at the center of tile that contains the terminal
- Each edge has associated capacity: maximum number of routing tracks that can pass along the edge



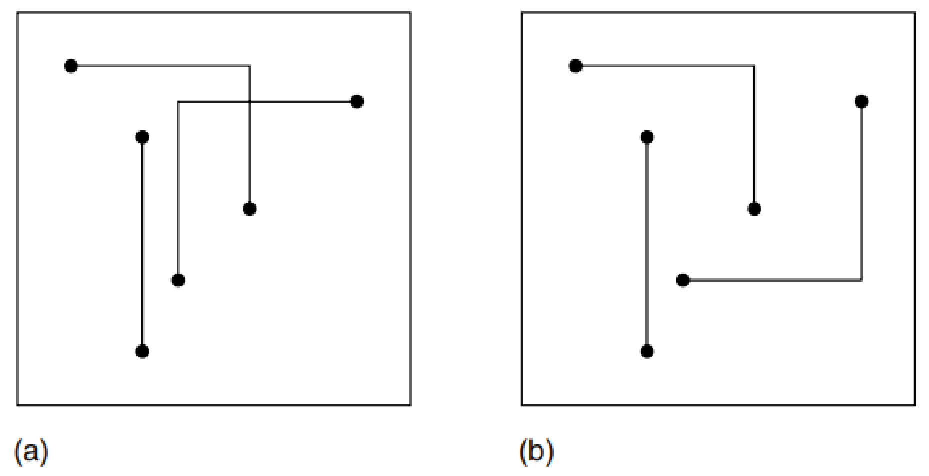
Routing optimization objectives

- Total wirelength
  - Reduced total wirelength is good for delay and power consumption
- Congestion
  - Edge is congested if total number of tracks assigned to it exceeds the edge capacity
  - Congested hotspots make detailed routing much harder
  - Even spread of congestion throughout routing region usually leads to better detailed routing
- Bend count
  - Each bend indicates the need for layer change and via
  - Vias have negative effects on delay, routing area and manufacturing yields



Routing optimization objectives

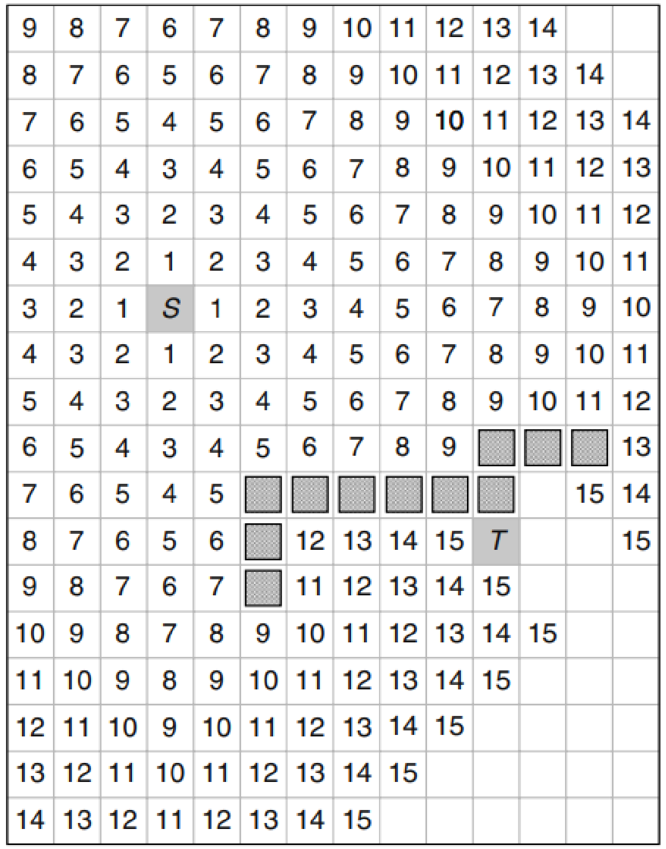
- Timing
  - Certain timing bounds are imposed on critical connections
- Coupling
  - Coupling capacitance between two wires is proportional to the amount of parallel overlap between them, and inversely proportional to the distance between them.
  - Coupling capacitance negatively affects timing

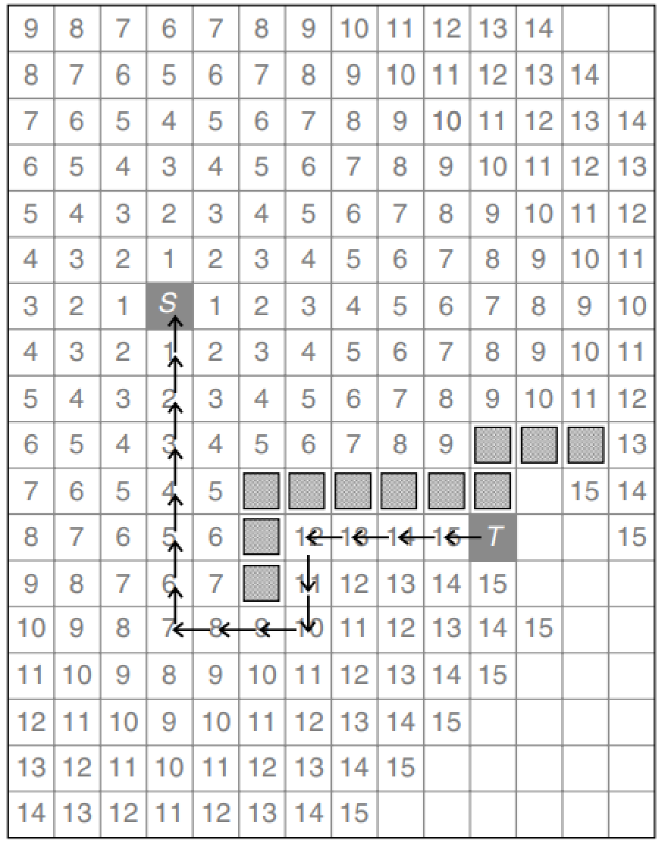


(a) Layout with coupling owing to long parallel wires and (b) layout with no coupling.

Maze routing

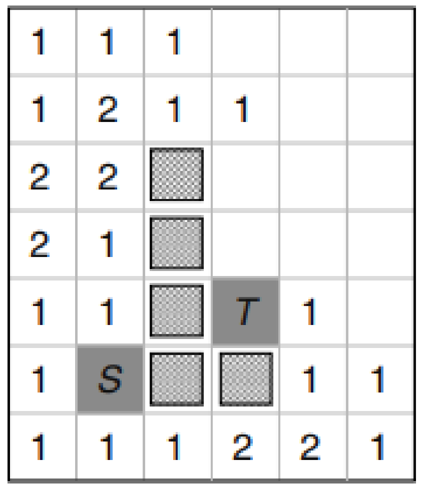
- Given a planar rectangular grid graph, two terminals S and T, obstacles modeled as blocked vertices it finds the shortest path between S and T
- Finds the shortest possible path if the path exists
- Two phases:
  - (a) wavefront expansion
  - (b) backtracking
- Modifications:
  - Start from pin that is located closer to the boundary
  - Expand from both terminals simultaneously
  - Define artificial bounding box on the search region


(a)

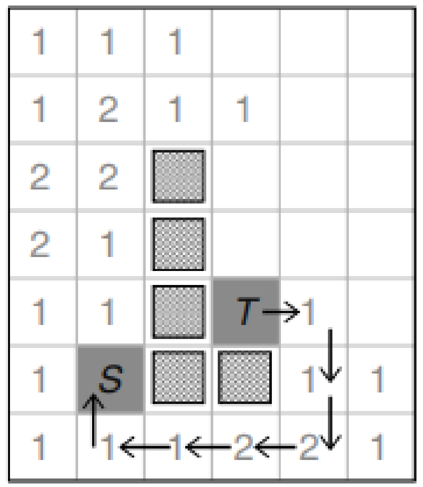

(b)

Aker’s labeling scheme

- The following sequence is used to label the cells during wavefront expansion phase: 1, 1, 2, 2, 1, 1, 2, 2, . ..
- In this case the predecessor of each cell *C* is labeled different from the successor of cell *C*.
- During backtracking, the same sequence is used to construct the path from target to source
- This reduces the memory requirements of the algorithm

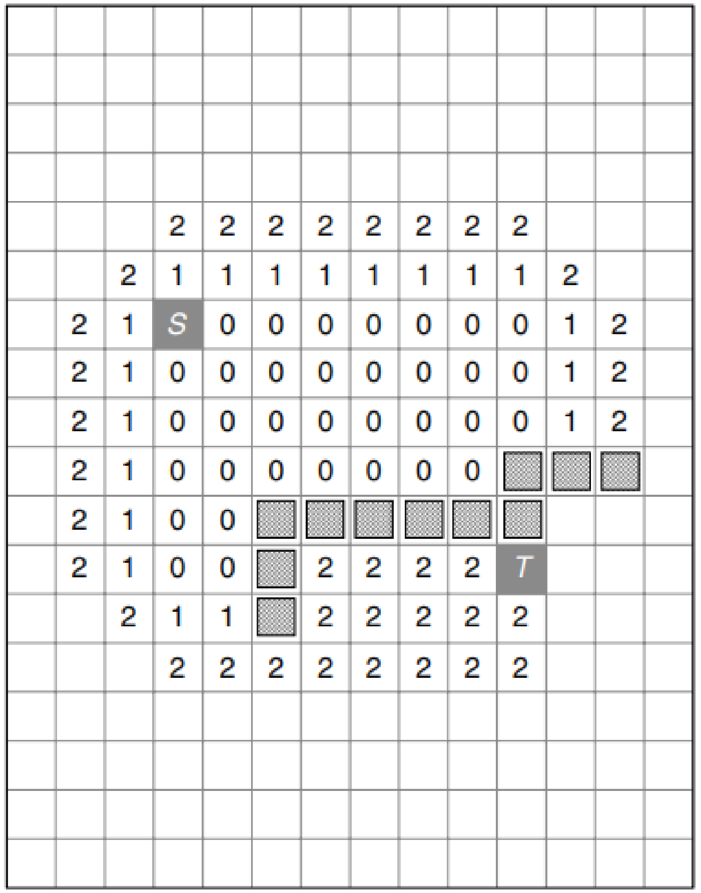


(a)



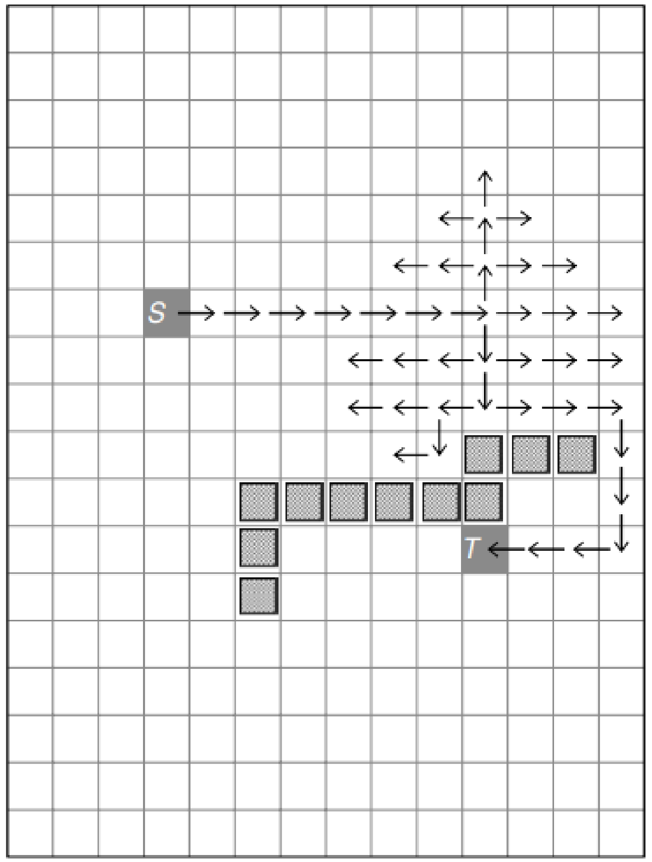
(b)

Hadlock’s minimum detour scheme



- Uses the detour numbers as the cell labels
- The cells with smaller detour numbers are expanded before the cells with higher detour numbers.
- Finds the shortest path if one exists

Soukup's fast maze

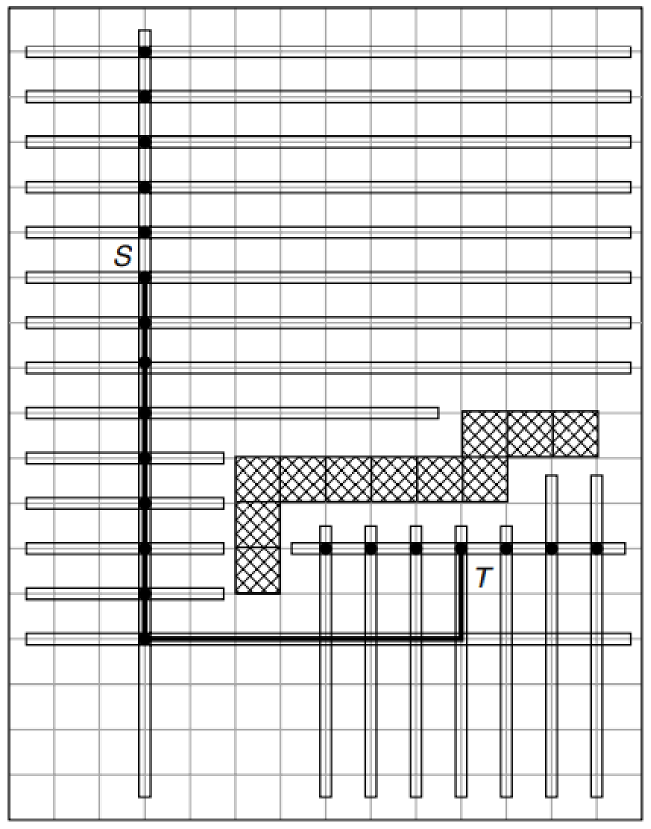


- Phase1: wavefront expansion is done toward the target without changing direction until an obstacle is reached
- phase2: the same wavefront expansion methodology as the original maze-routing algorithm is used to search around the obstacle
- Once a cell in the direction of the target is found, the first phase begins again for a directed search toward the target
- The path found is not guaranteed to be the shortest

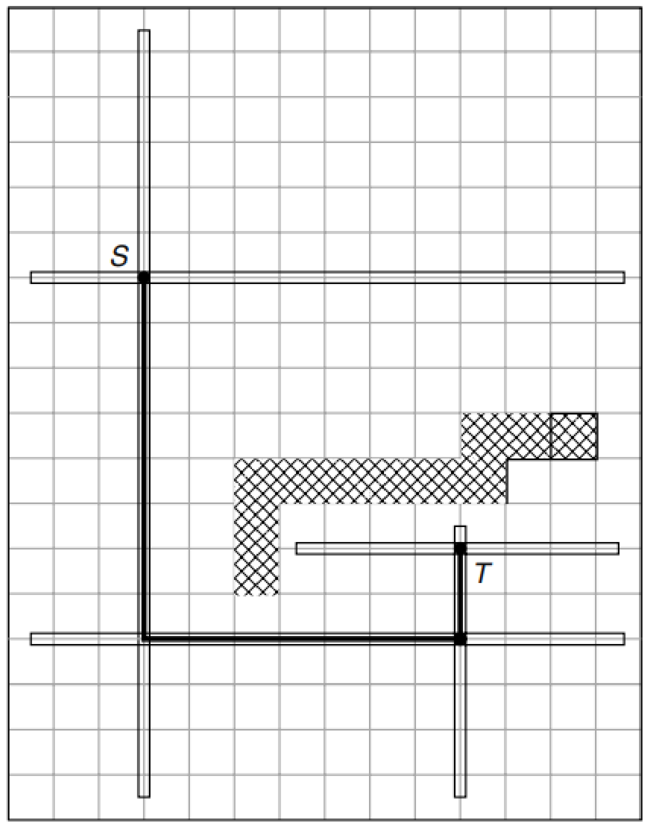
Line-search algorithms

- Mikami-Tabuchi algorithm (a)
  - Start with expanding horizontal and vertical line segments for S and T
  - Repeat for each point at recently added segments until one of segments originated from S intersects with one originated from T

- Hightower algorithm (b)
  - Escape lines on the most recently created line segments identified based on blockages



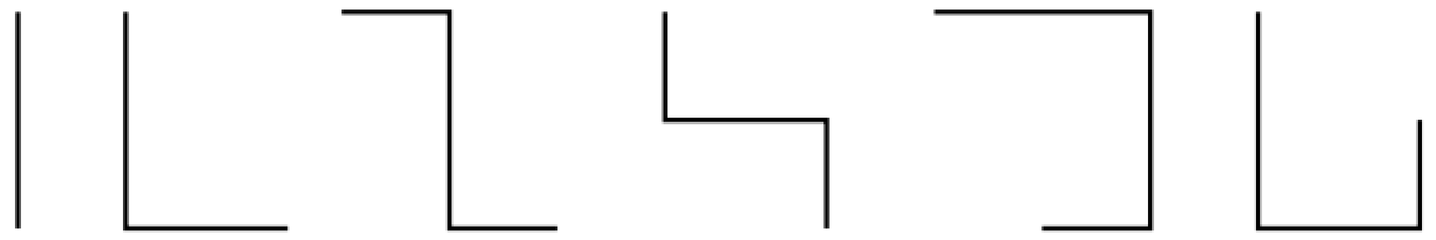
(a)



(b)

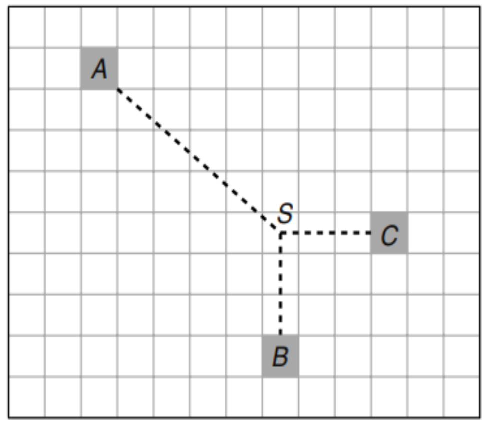
Pattern routing

- Solution space is restricted to routes with predefined patterns, such as I-, L-, Z-, and U-shaped patterns
- Can be effectively used in global routing to reduce the runtime requirements

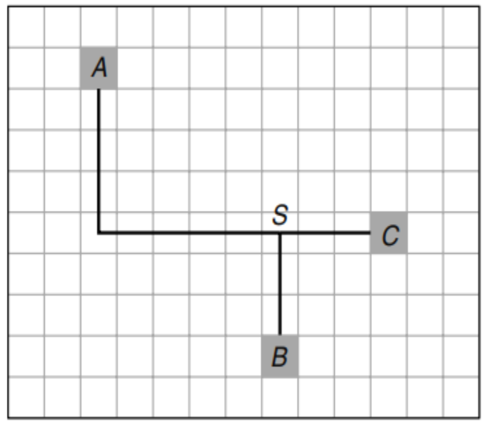


Routing for multiple terminal net

- Finding the optimal route for a multiple terminal net is NP-complete problem
- Fortunately there are good heuristic algorithms that
  a. Generate Steiner topology T and
  b. Perform point-to-point routing between points of that topology T



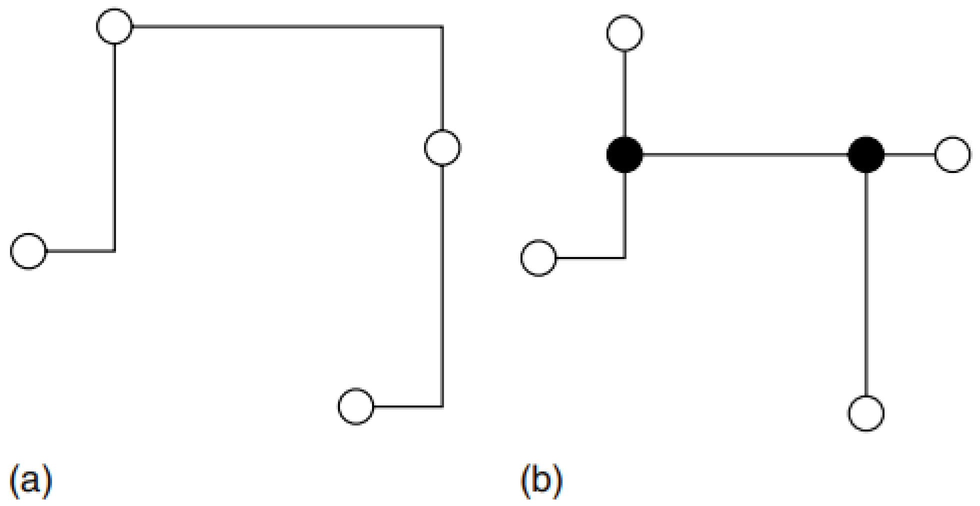
(a)



(b)

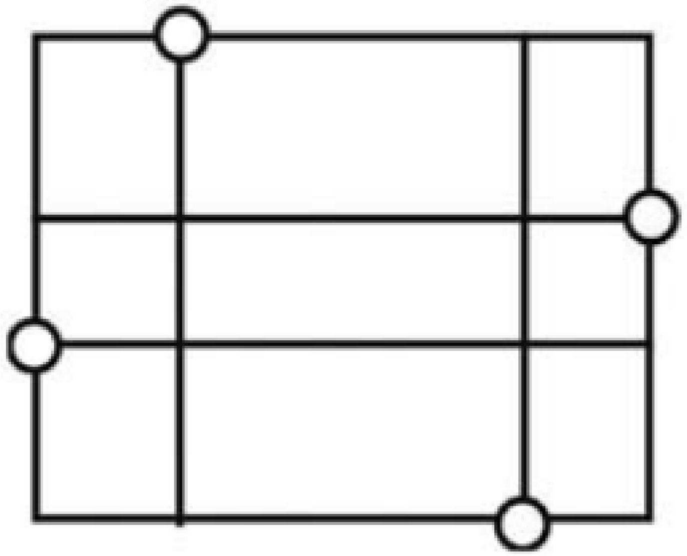
Steiner minimal tree (SMT) problem

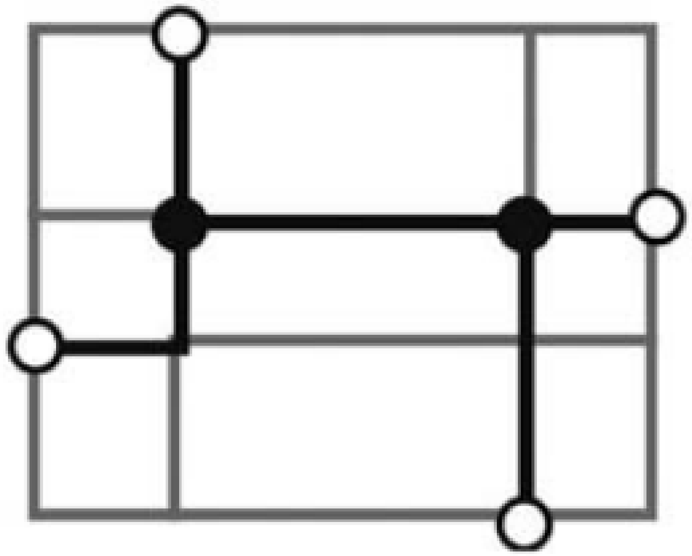
- Given a set P of n points determine a set S of Steiner points such that the MST cost over $P \cup S$ is minimized.
- The cost between pair of pins is modeled by Manhattan distance:
  - $|x_1 - x_2| + |y_1 - y_2|$
- It was shown (Hwang, 1976) that the MST over P is a good approximation to the SMT with cost ratio $\leq \frac{3}{2}$



Hanan’s theorem

- There exists an SMT with Steiner points chosen from the Hanan grid, i.e., intersection points of all horizontal and vertical lines drawn through the points.

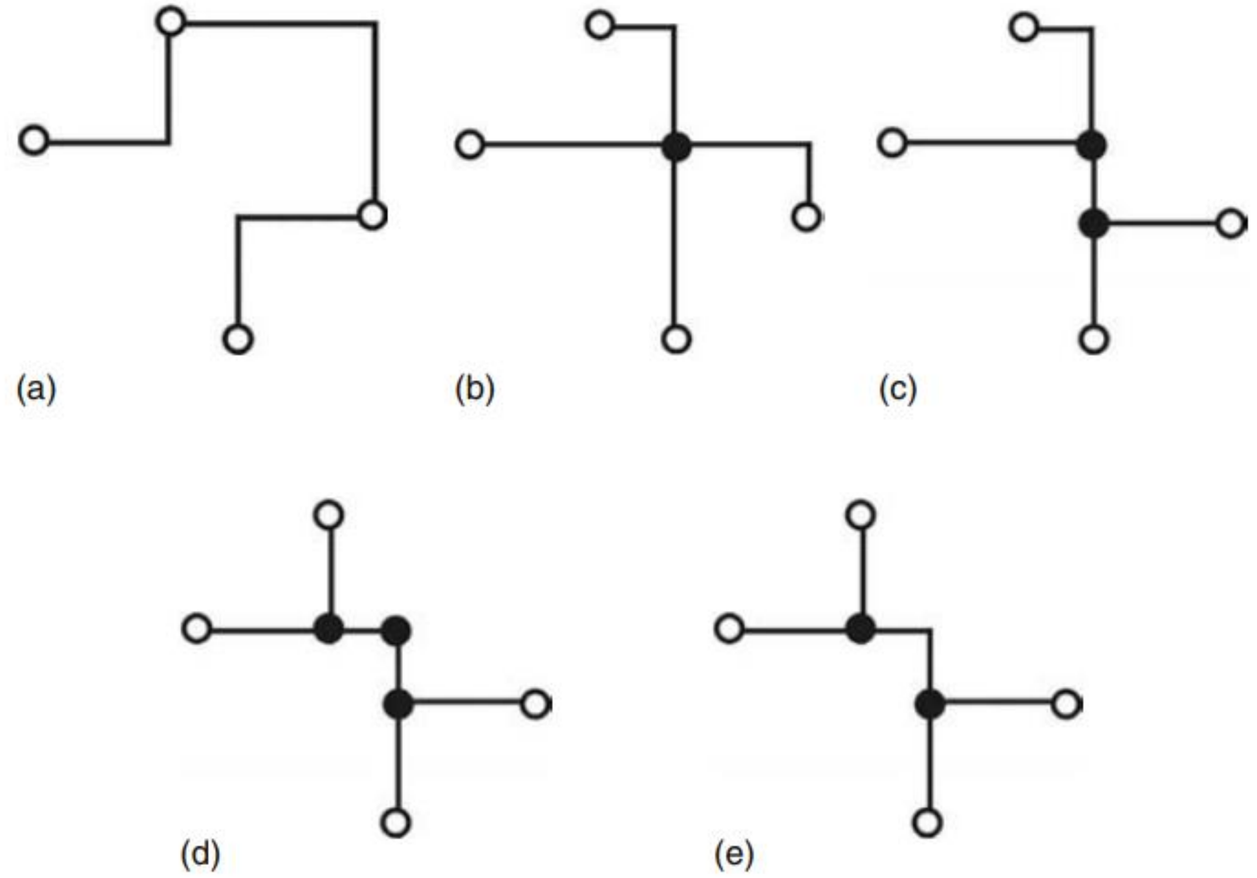




Iterated 1-Steiner heuristic

```
**Iterated 1-Steiner (I1S) heuristic**
**Input**: Set $P$ of $n$ points
**Output**: Rectilinear Steiner tree spanning $P$
$S = \emptyset$
**While** Candidate_Set $= \{x \in H(P \cup S) \mid \Delta MST(P \cup S, \{x\}) > 0\} \neq \emptyset$ **Do**
    **Find** $x \in$ Candidate_Set which maximizes $\Delta MST(P \cup S, \{x\})$
    $S = S \cup \{x\}$
    **Remove** points in $S$ which have degree $\leq 2$ in $MST(P \cup S)$
**Output** MST$(P \cup S)$
```

Example



Analysis

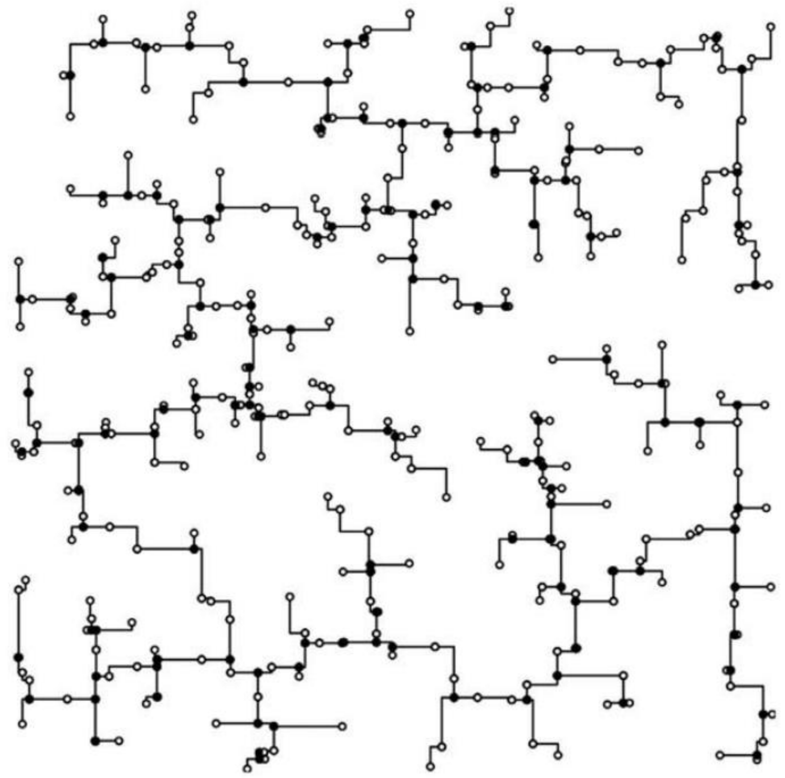
- Need to construct MST for each of $O(n^2)$ members of the Steiner candidate set (i.e., Hanan grid points)
- Each MST computation can be performed in $O(\mathrm{n}\log(n))$
- $O(n^3\log(n))$ time method to find a single 1-Steiner point.
- In practice, the number of iterations performed by I1S averages less than *n*/2 for uniformly distributed random pointsets.
- I1S heuristic is provably optimal for four or less points

Directions to improve SMT runtime and quality

- Pre-filtering of candidate Steiner points
- Incremental/dynamic MST updating schemes
- Local Steiner tree restructuring along the way

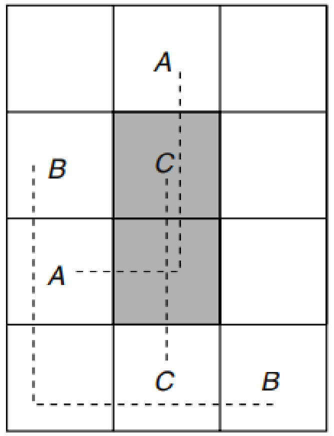
Performance of iterated 1-Steiner

- On average, solution cost is within 0.5% of optimal.
- Produce optimal solutions on 90 percent of all random eight-point instances (and on more than half of all random 15-point instances)
- For n = 30 points, on average only about 0.3 percent away from optimal, and yields optimal solutions in about one quarter of the cases

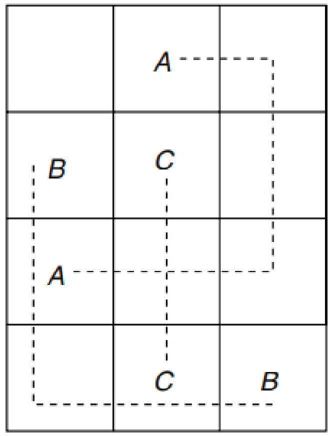

Sample solution on random set of 300 points

Routing multiple nets

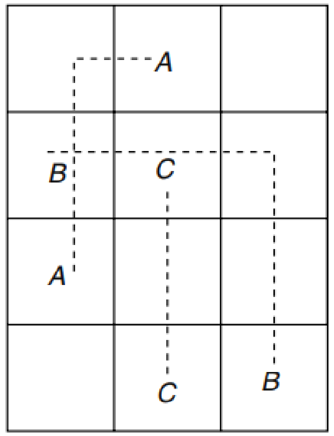
- Routing for one net may impact the routing of other nets because common resources are being used. Order of routing matters.



(a)



(b)



(c)

Assume that horizontal and vertical edge capacity = 1 and not more than 2 bends is allowed

a) A–B–C leads to a solution with two overcongested tiles

b) C–B–A leads to a solution where A is detoured to avoid congestion

c) C–A–B leads to the congestion-free solution with optimal routing for each net

Routing multiple nets

- Sequential routing route nets in a specific order
  - More critical nets are routed first
  - Route nets with less routing alternatives before other nets. In practice this means short nets first
- Nets are first routed allowing congestion, and then the nets in the overcongested regions are ripped up and rerouted in the later iterations
- Rip-up-and-reroute strategy is typically a combination of progressive and iterative schemas

Progressive rip-up-and-reroute

```
1. For pass = 1 to $k$
2. For each net $n$
3. For each connection $r$ in $n$
    a. Remove $r$
    b. Reroute $r$
```

- Pros:
  - Simplicity of design
- Issues:
  - Detouring. The case when the length of the routed connection is much longer than the length of a connection if congestion is not considered.
  - Net ordering
  - Divergence. Commonly happens when a design is infeasible

Iterative improvement scheme

```
1. Route nets
2. Identify areas where the design constraints are violated
3. Identify nets/connections $r$ to
   a. Remove $r$
   b. Reroute $r$
```

- Select one violated design constraint and then attempt to resolve it by rip-up and reroute without introducing new violations.
- Issues:
  - Insufficient rip-up
  - Net ordering
  - Oscillations

References

- Handbook of Algorithms for Physical Design Automation, Chapter 23, Muhammet Mustafa Ozdal and Martin D.F. Wong, Global Routing Formulation and Maze Routing
- Handbook of Algorithms for Physical Design Automation, Chapter 24, Minimum Steiner Tree Construction, Gabriel Robins and Alexander Zelikovsky
- Handbook of Algorithms for Physical Design Automation, Chapter 31, Jeffrey S. Salowe, Rip-up and reroute