# Lecture 13: Routing

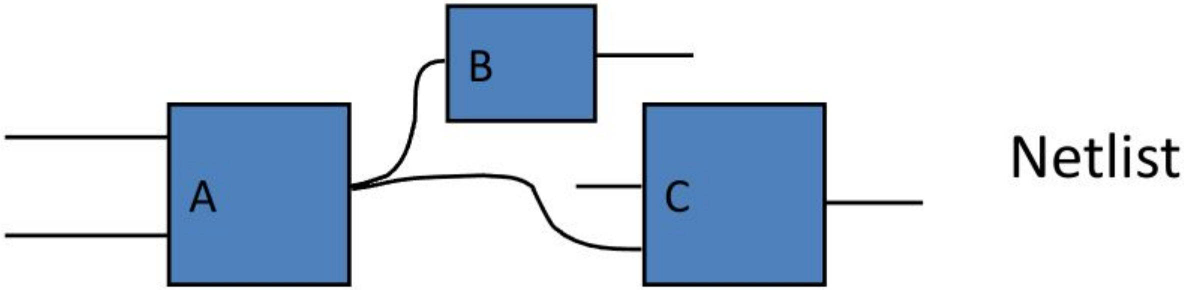
Matthew Guthaus
Professor
UCSC, Computer Science & Engineering
http://vlsida.soe.ucsc.edu
mrg@ucsc.edu

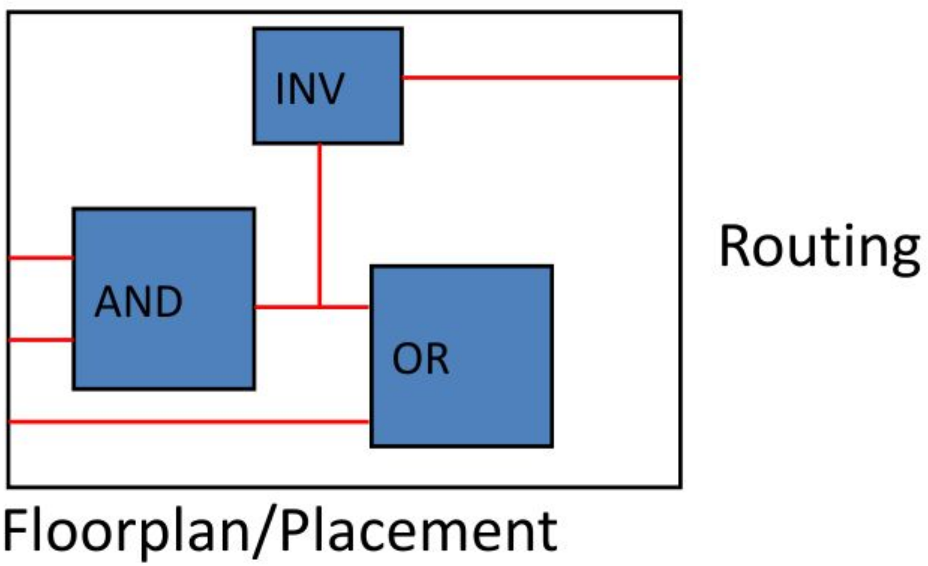
UC SANTA CRUZ VLSI DESIGN & AUTOMATION LAB

# Today's Lecture

- Global vs Detailed Routing
- Maze Routing
- Iterative Routing (Rip up and Reroute)
- OpenLane Routing

# Routing in design flow



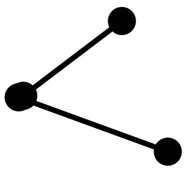



# The Routing Problem

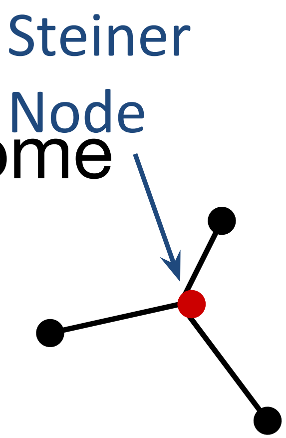
- Apply it after floorplanning/placement
- Input:
  - Netlist
  - Timing budget for, typically, critical nets
  - Locations of blocks and locations of pins
- Output:
  - Geometric layouts of all nets
- Objective:
  - Minimize the total wire length, the number of vias, or just completing all connections without increasing the chip area.
  - Each net meets its timing budget.

# Steiner Tree

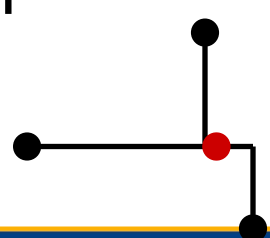
- For a multi-terminal net, we can construct a spanning tree to connect all the terminals together.
  - But the wire length will be large.



- Better use Steiner Tree:
  - A tree connecting all terminals and some additional nodes (Steiner nodes).



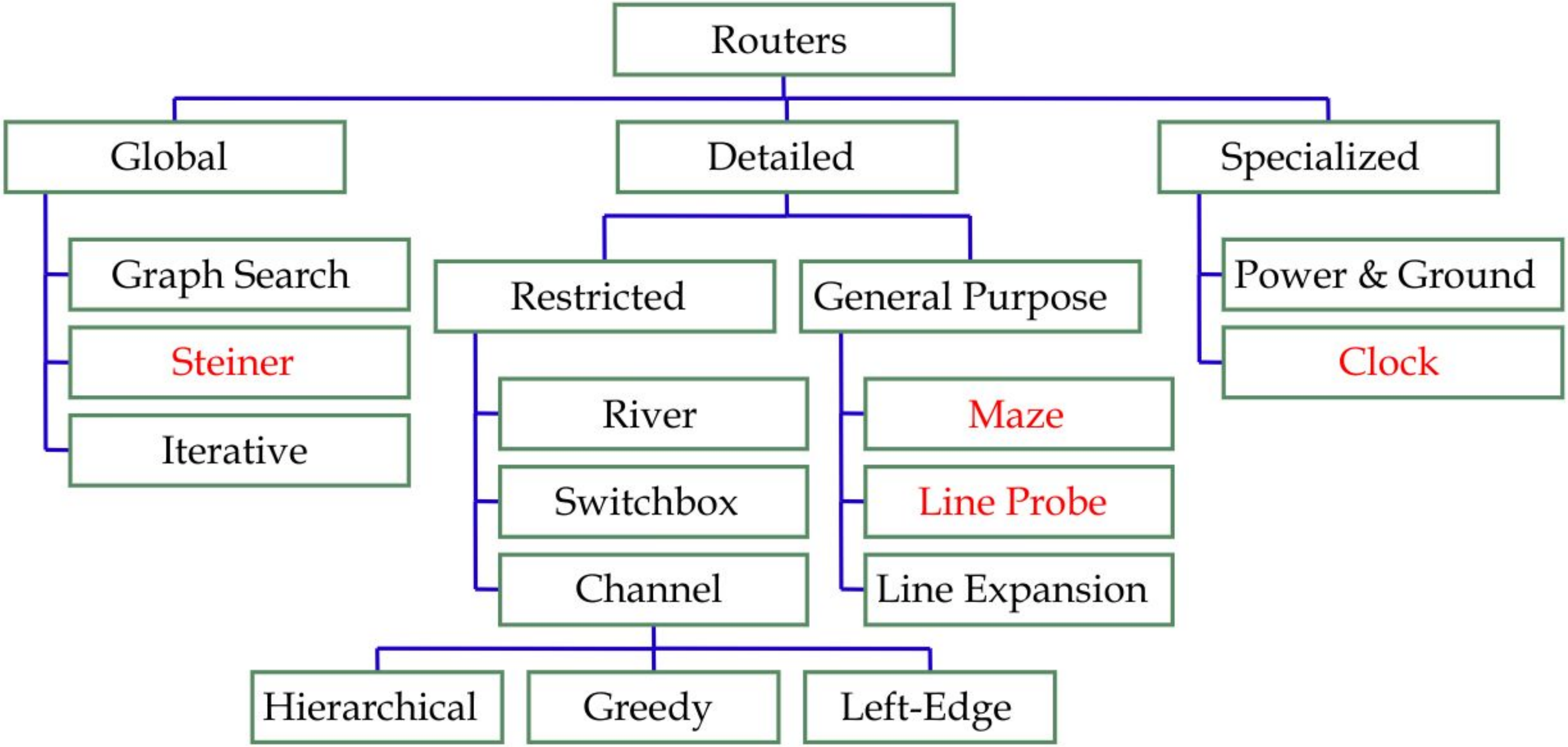

- Rectilinear Steiner Tree:
  - Steiner tree in which all the edges run horizontally and vertically.



# Routing Problem is Very Hard

- Minimum Steiner Tree Problem:
  - Given a net, find the Steiner tree with the minimum length.
  - This problem is NP-Complete!
- May need to route tens of thousands of nets simultaneously without overlapping.
- Obstacles may exist in the routing region.
- Minimum wirelength is not minimum delay.

# Taxonomy of VLSI Routers

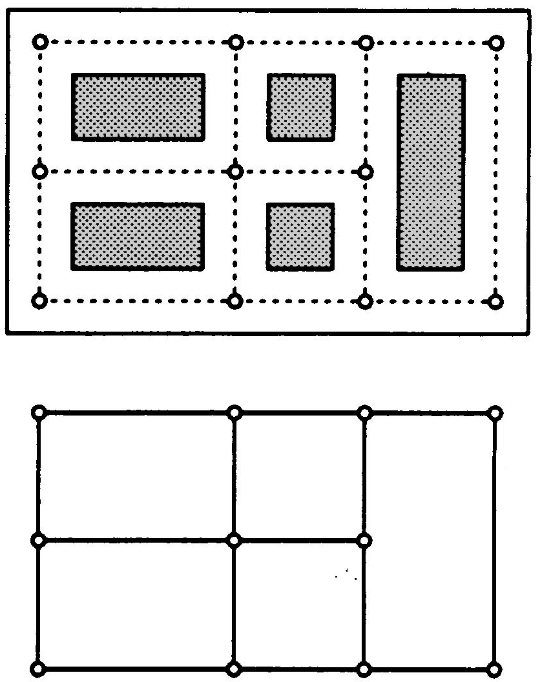


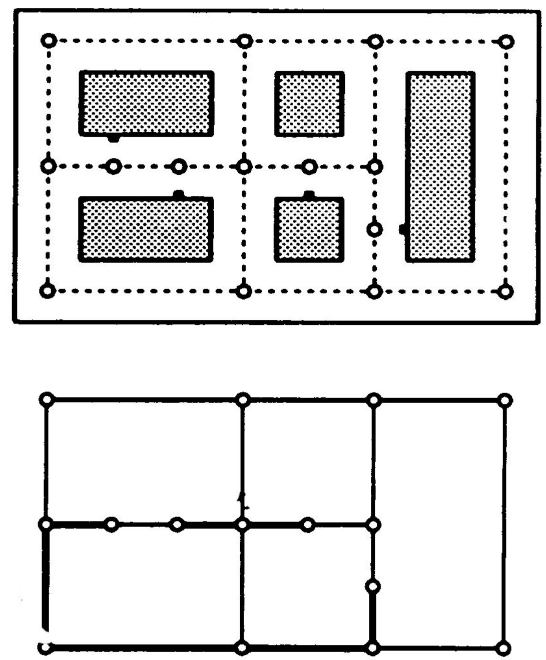
# Routing Algorithms

- Global routing
  - Guide the detailed router in large design
  - May perform quick initial detail routing
  - Commonly used in cell-based design, chip assembly, and datapath
  - Also used in floorplanning and placement
- Detail routing
  - Connect all pins in each net
  - Must understand most or all design rules
  - May use a compactor to optimize result
  - Necessary in all applications

# Channel Intersection Graph

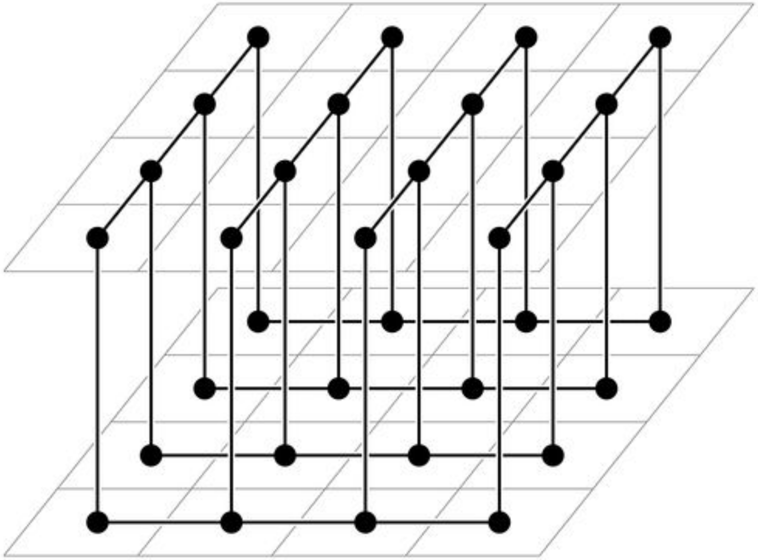
- Edges are channels, vertices are channel intersections (CI), v1 and v2 are adjacent if there exists a channel between ($CI_1$ and $CI_2$). Graph can be extended to include pins.





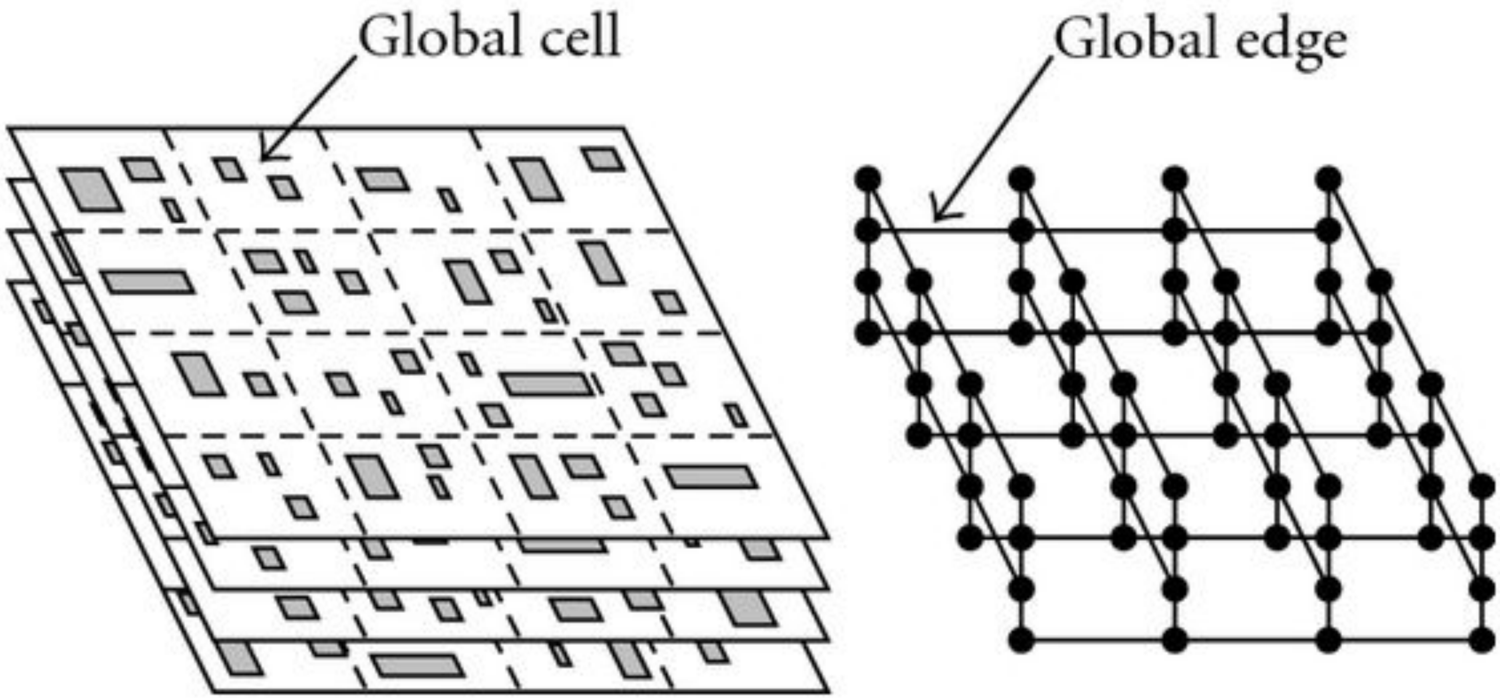
# Multilayer Routing

- Instead of 2D grid graph, use a 3D graph
- Vias should have cost greater than a wire
- Preferred direction routing
  - Calculate the “momentum” of a layer
  - Off-grid should cost more (like vias) but should allow short non-preferred layer connections



# Global Routing "Grid" Graph

- 4 layer grid graph example
- Non-optional preferred directions
- No blockages shown



# Graph Edge Capacity

- Number of wires that can pass through a region depends on
  - Wire routing pitch (width + spacing)
  - Width (or height) of channel
  - Number of layers
  - Blockages
  - Reserved tracks (for clock or power)

# Approaches for Routing

- Sequential Approach:
  - Route nets one at a time.
  - Order depends on factors like criticality, estimated wire length, and number of terminals.
  - When further routing of nets is not possible because some nets are blocked by nets routed earlier, apply ‘Rip-up and Reroute’ technique (or ‘Shove-aside’ technique).
- Concurrent Approach:
  - Consider all nets simultaneously, i.e., no ordering.
  - Can be formulated as integer programming.

# Sequential Approaches

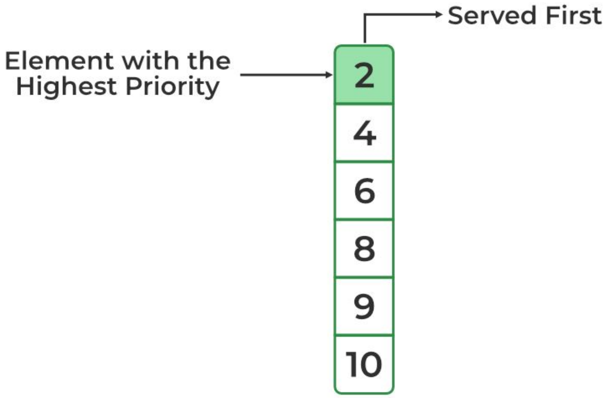
- Solve a single net routing problem
- Differ depending on whether net is two- or multi-terminal
- Two-terminal algorithms
  - Maze routing algorithms
  - Line probing
  - Shortest-path based algorithms
- Multi-terminal algorithms
  - Steiner tree algorithms

# Two-Terminal Routing: Maze Routing

- Maze routing finds a path between source (s) and target (t) in a graph
- Grid graph model is used
- Available routing areas are unblocked vertices, obstacles are blocked vertices
- Basic idea = wave propagation (Lee, 1961)
  - Breadth-first search + back-tracing after finding shortest path
  - Guarantees to find the shortest path
- Time and space complexity O(*h* x *w*)

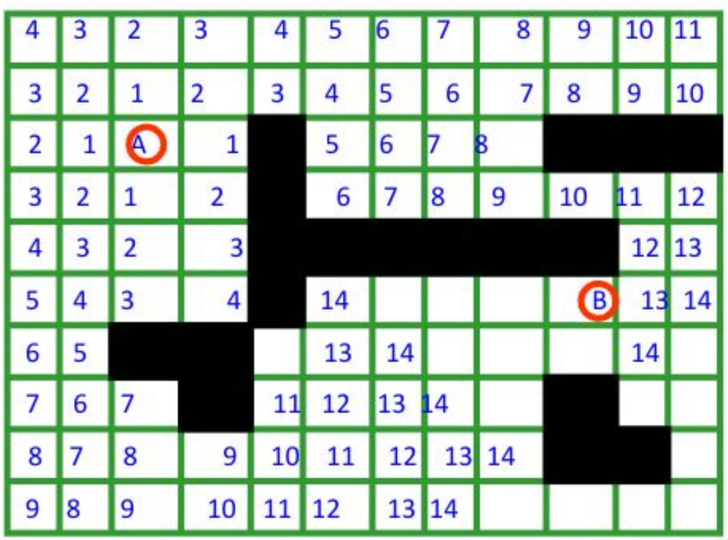
# Priority Queue

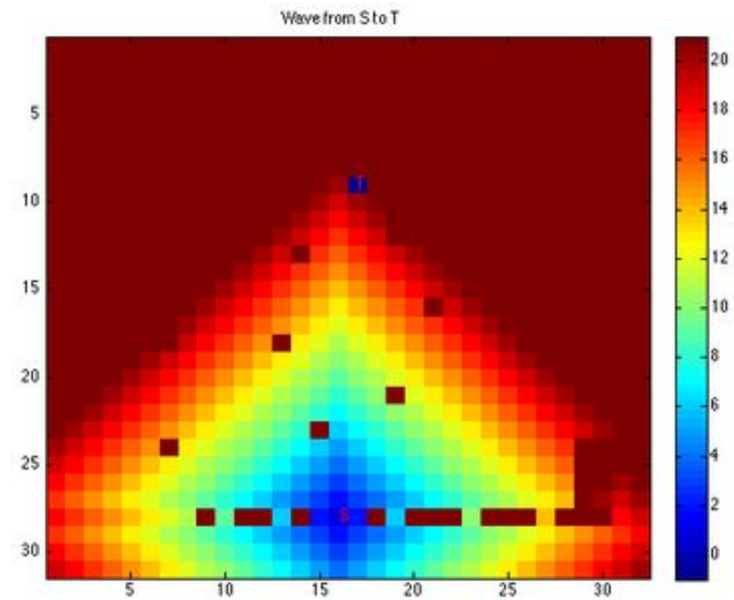
- Must prioritize partial routes based on a cost
  - Keep them in a sorted priority queue
  - Pull cheapest element
  - Insert element with a cost
- General idea: Keep expanding the cheapest option so far.



# Maze Routing

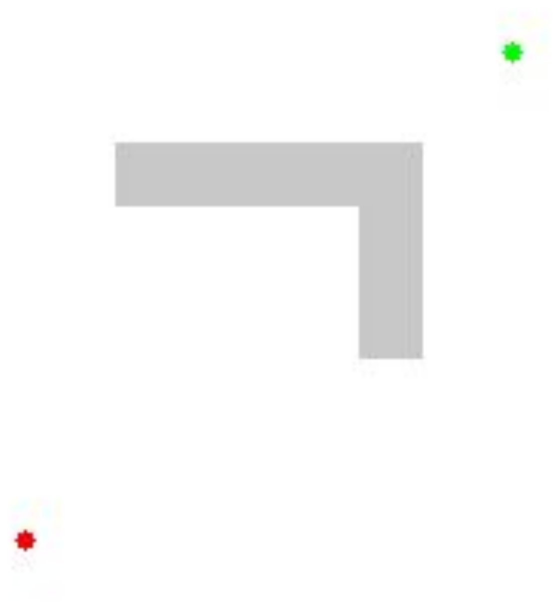
- Each cell is adjacent to another
- Minimum cost keeps track of a “frontier” in the priority queue




https://en.wikipedia.org/wiki/Lee_algorithm

# Dijkstra’s Routing Example



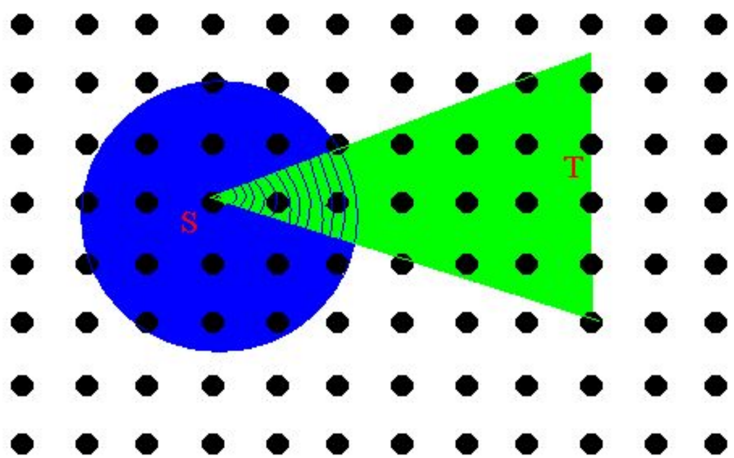
https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

# Dijkstra vs Lee Maze Routing

- Dijkstra's
  - Uses depth first (min-cost) priority queue
  - Used on general graphs, not just grid graphs
  - May terminate once target is found to optimize run time
- Lee maze router
  - Uses breadth first search
  - Guarantees minimal route
  - More time consuming!

# Maze Routing Cost Function and Directed Search

- Points can be popped from queue according to a multivariable cost function
- Cost = function(overflow, coupling, wire length, … )




S denotes the source point
T is the sink point

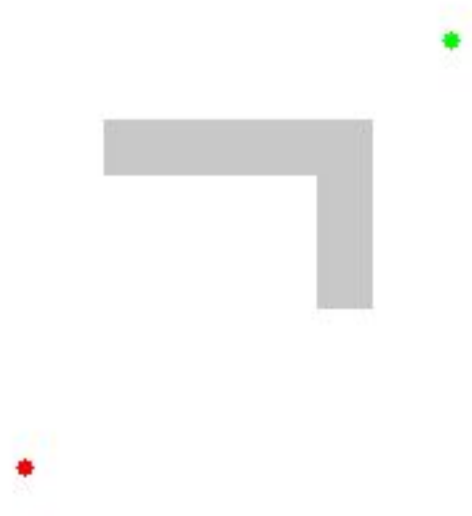
Directed search limits the search space when all other cost variables are equal

Non directed search expands in a circular fashion from S

Directed search expands in a conical fashion from S to T

# A* Search

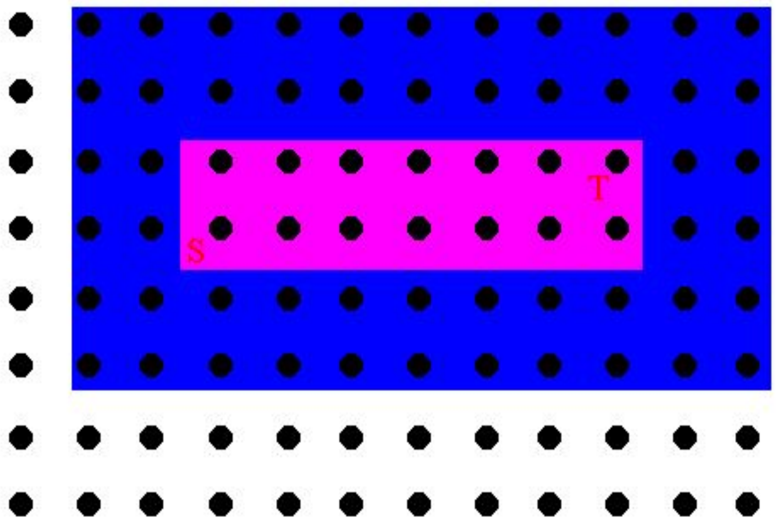
- Add <distance to sink> to cost function  directed search
  - Allows maze router to explore points around the direct path from source to sink first



https://en.wikipedia.org/wiki/A*_search_algorithm

# Limiting the Search Region

- Since majority of nets are routed within the bounding box defined by S and T, can limit points searched by maze router to those within bounding box
  - Allows maze router to finish sooner with little or no negative impact on final routing cost
  - Router will not consider points that are unlikely to be on the route path




S denotes the source point
T is the sink point

Bounding box of S and T

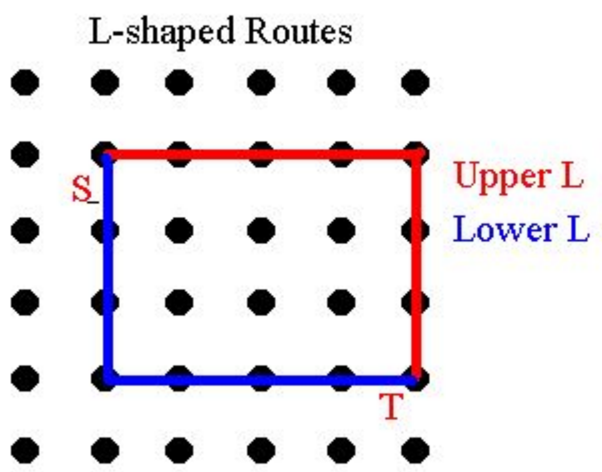
Normally, the search region is restricted to the bounding box + X

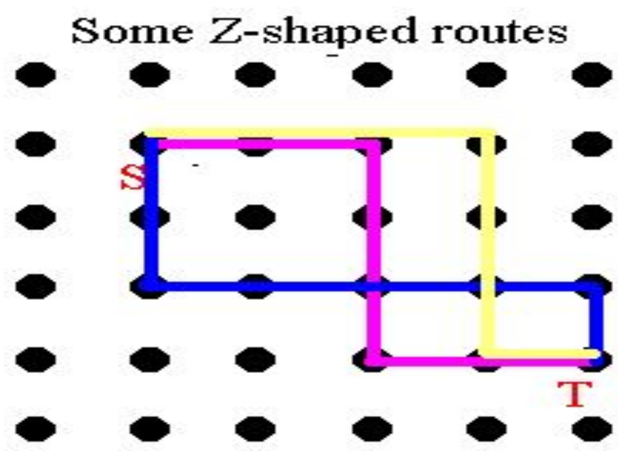
In this example, X = 2

The points outside of blue area are not considered by the maze router

# Pattern-Based Routing

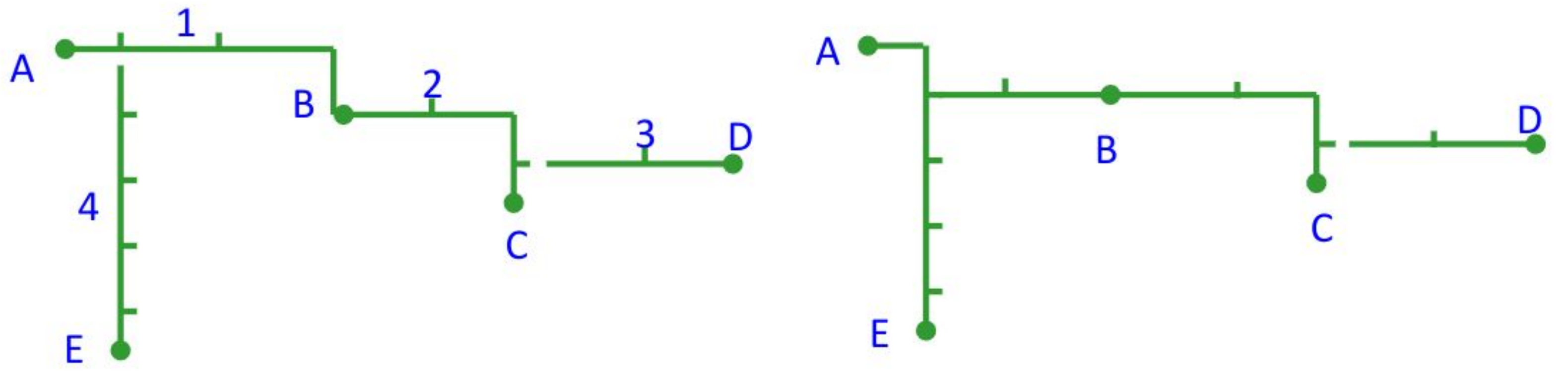
- Restrict routing of net to certain basic templates
- Basic templates are L-shaped (1 bend) or Z-shaped (2 bends) routes between a source and sink
- Templates allow fast routing of nets since only certain edges and points are considered



# Multi-Terminal Nets

- In general, maze and line-probe routing are not well-suited to multi-terminal nets
- Several attempts made to extend to multi-terminal nets
  - Connect one terminal at a time
  - Use the entire connected subtrees as sources or targets during expansion
  - Ripup/Reroute to improve solution quality (remove a segment and re-connect)
- Results are sub-optimal
- Inherit time and memory cost of maze and line-probe algorithms



# Problems with Sequential Routing Algorithms

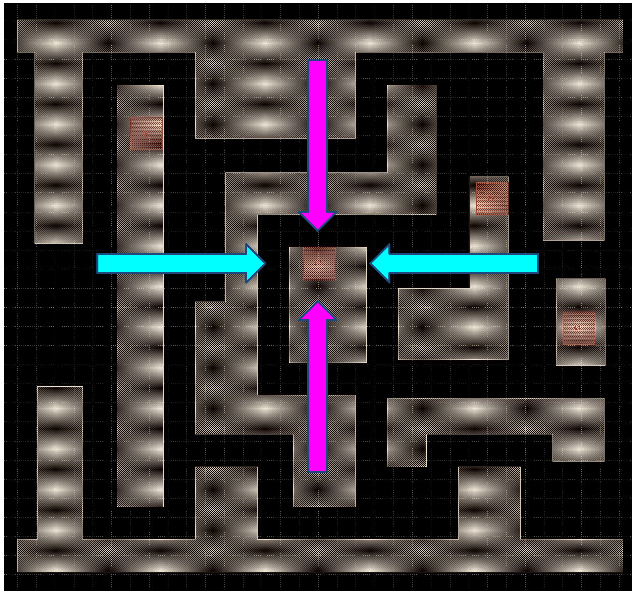
- Net ordering
- Must route net by net, but difficult to determine best net ordering!
- Difficult to predict/avoid congestion
  - What can be done
- Use other routers
  - Channel/switchbox routers
  - Hierarchical routers
  - Rip-up and reroute

# Rip Up and Reroute

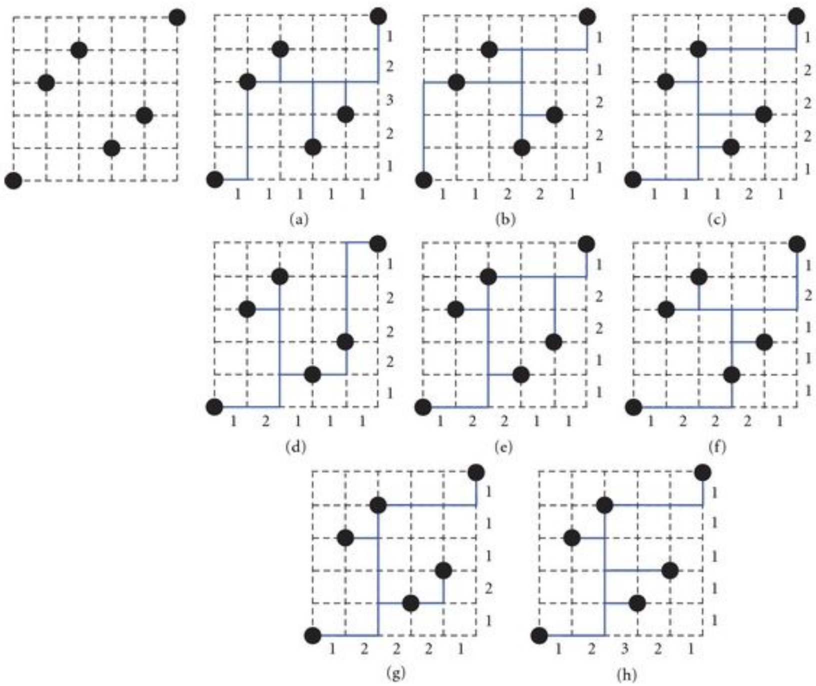
- Maze route each net
  - If unable to route all, rip and reroute nets, i.e., select a number of nets based on a cost function (e.g., congestion of regions through which net travels), then remove the net and reroute it
- Main objective: reduce overflow
  - Edge overflow = 0 if num_nets less than or equal to the capacity
  - Edge overflow = num_nets – capacity if num_nets is greater than capacity
  - Overflow = Σ (edge overflows) over all edges

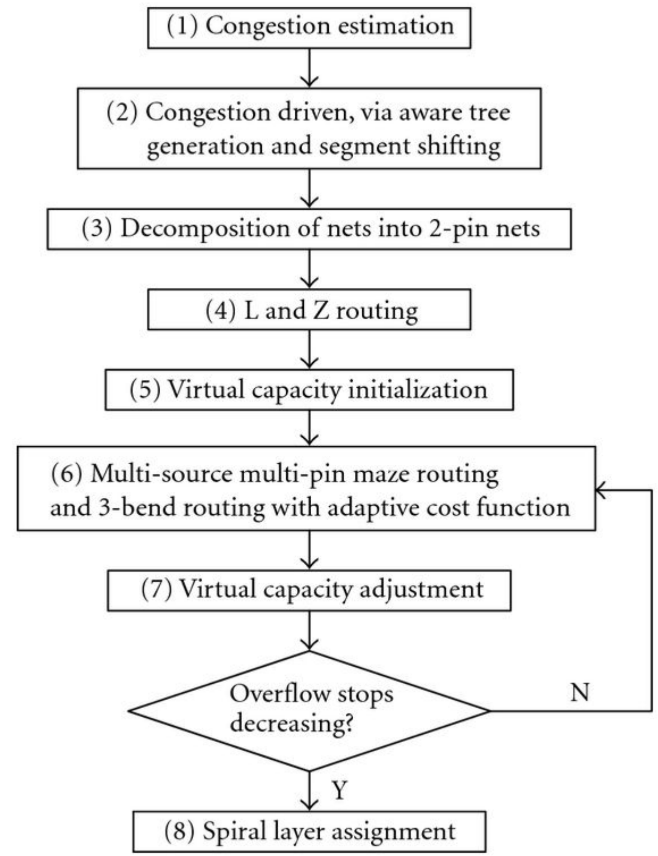
# Detailed Routing

- Modern design rules are quite complex
  - Variable spacing based on length
  - Line extension
  - Different pitches on different layers
- Gridded routing
  - On-grid pins
  - Center-line convention



# Global: FastRoute, Detailed: TritonRoute




FastRoute:
https://www.hindawi.com/journals/vlsi/2012/608362/
TritonRoute:
https://vlsicad.ucsd.edu/Publications/Journals/j133.pdf

# OpenLane Options

- GLB_RT_ADJUSTMENT: Adjusts percentage of available global route
- GLB_RT_MACRO_EXTENSION: Reserves space around macros
- **GLB_RT_OVERFLOW_ITERS: Global iterations**
- GLB_RESIZER_*
- **DRT_OPT_ITERS: Detailed iterations**
- DRT_MIN_LAYER/DRT_MAX_LAYER: Detail routing can use more layers than global.
- ROUTING_CORES: Multi-threading! (default 2?)

# Parameters Not Exposed in OpenLane

https://openroad.readthedocs.io/en/latest/main/src/grt/README.html

- `critical_nets_percentage` : Set the percentage of nets with the worst slack value that are considered timing critical, having preference over other nets during congestion iterations (e.g. `-critical_nets_percentage 30` ). The default percentage is 0%.

# Next Lecture

- Clock Tree Synthesis