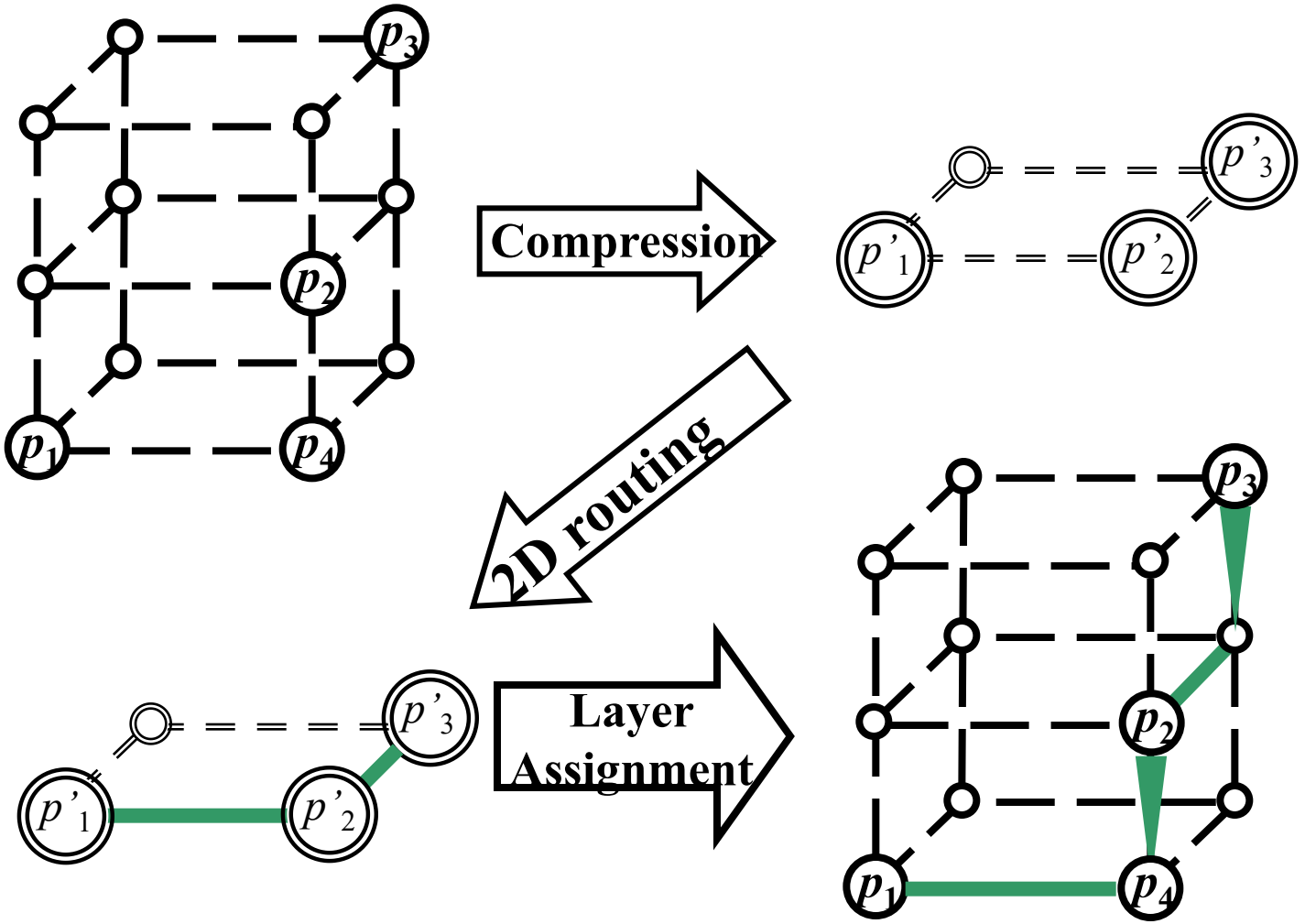
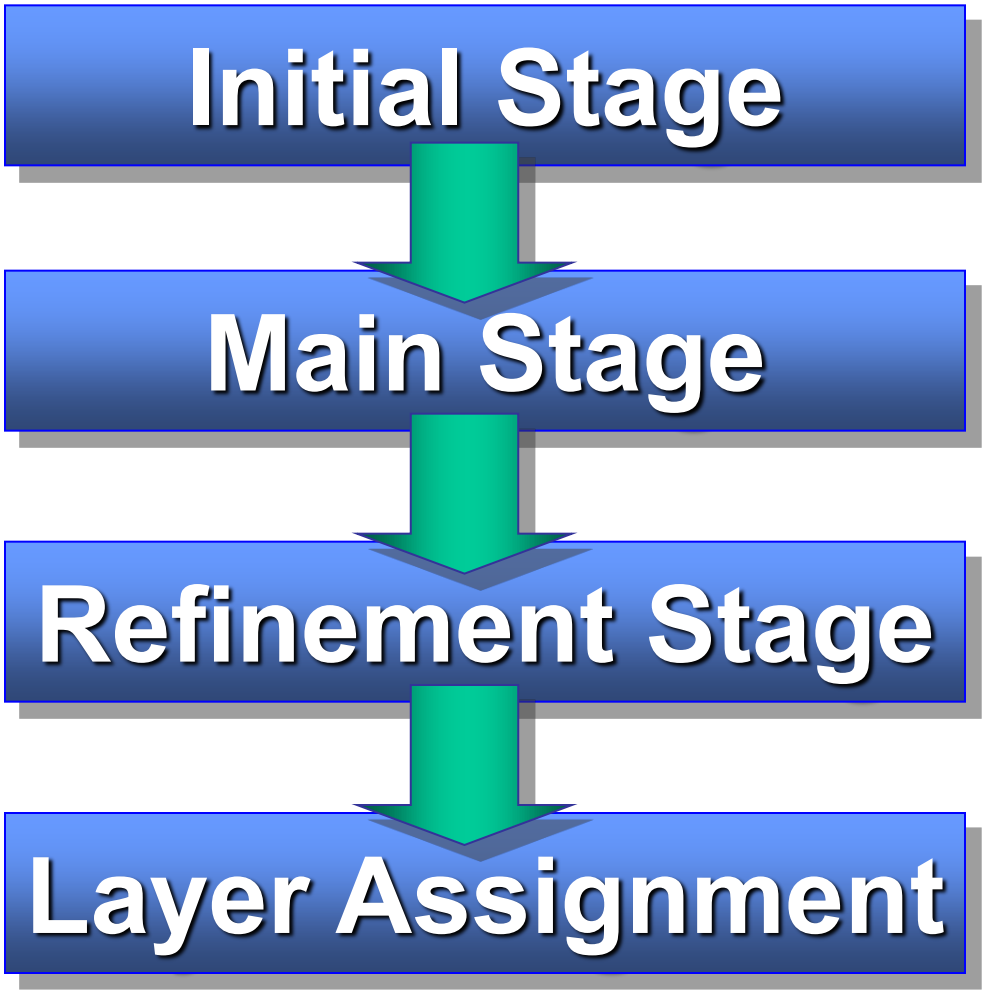
NTHU-Route 2.0

- Related publications:
  - Y.-J. Chang, Y.-T. Lee, J.-R. Gao, P.-C. Wu, and T.-C. Wang, "NTHU-Route 2.0: A Robust Global Router for Modern Designs," IEEE TCAD 2010.
  - Y.-J. Chang, Y.-T. Lee and T.-C. Wang, "NTHU-Route 2.0: A Fast and Stable Global Router," ICCAD 2008.
  - J.-R. Gao, P.-C. Wu and T.-C. Wang, "A New Global Router for Modern Designs," ASP-DAC 2008. (NTHU-Route)
  - T.-H. Lee and T.-C. Wang, "Congestion-Constrained Layer Assignment for Via Minimization in Global Routing," IEEE TCAD 2008.

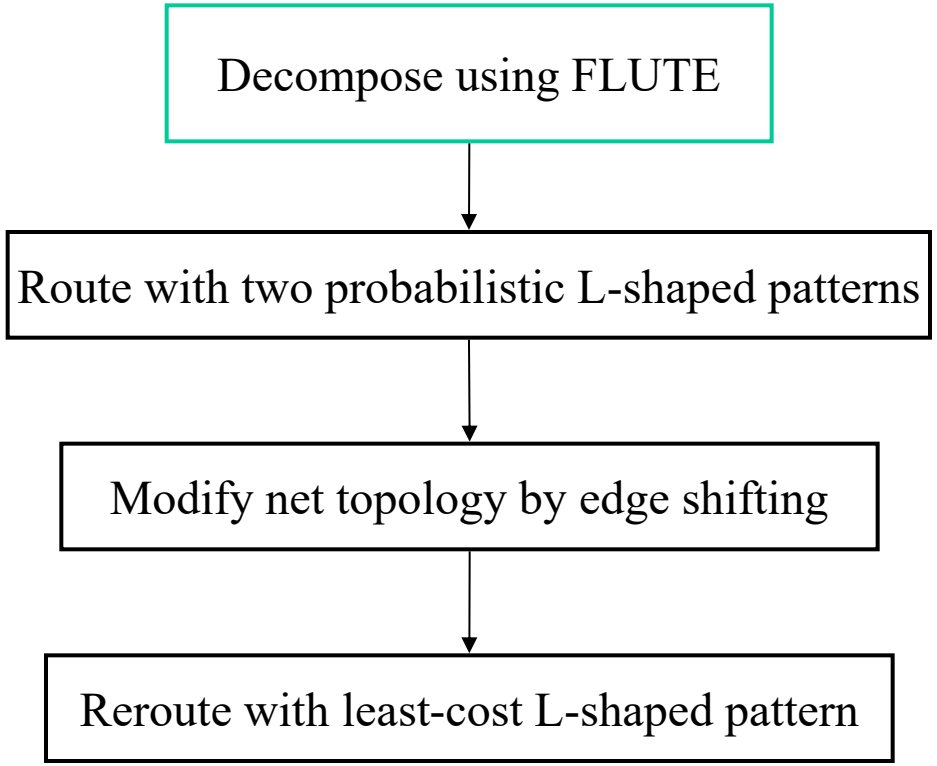
Multi-Layer Global Routing



Four Stages

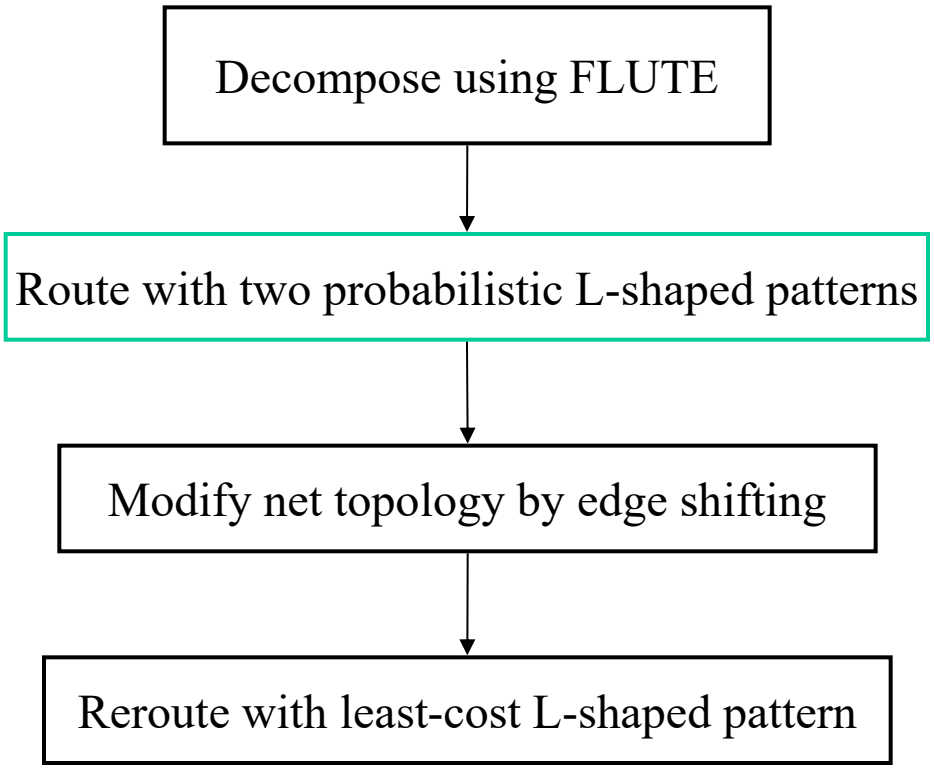

Initial Stage
Main Stage
Refinement Stage
Layer Assignment

Initial Stage

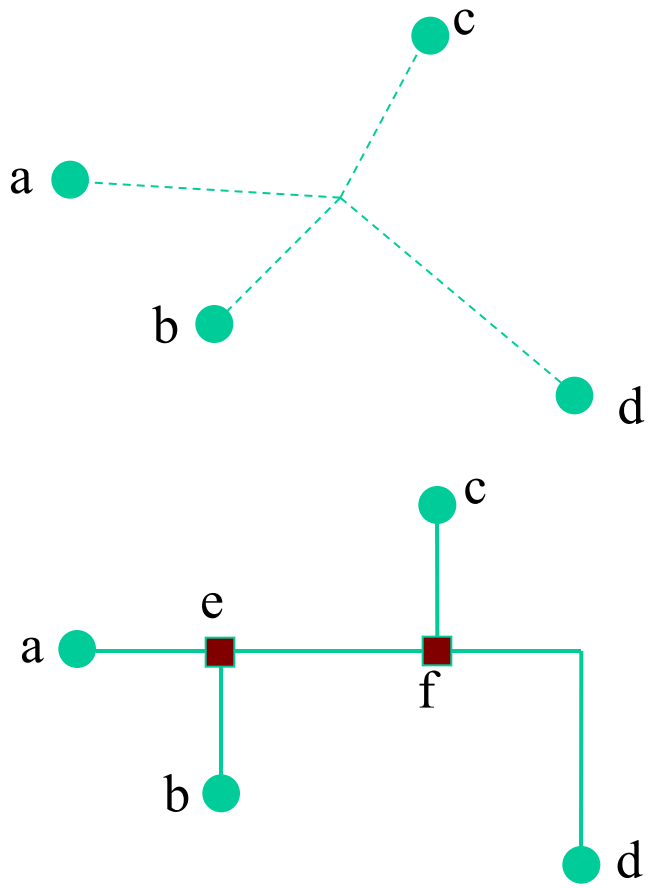


- First project the design onto a plane
- FLUTE [ISPD05]
  - Generation of a rectilinear Steiner minimal tree
  - A look-up table based method
  - Optimum for net with 9 or fewer pins

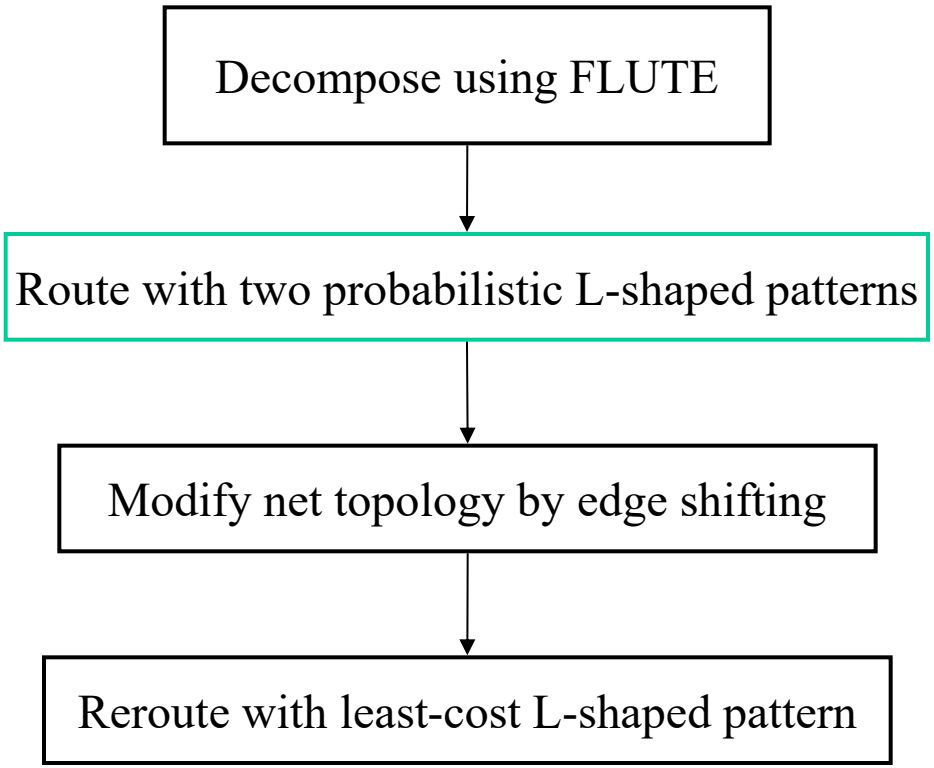
Initial Stage (cont'd)

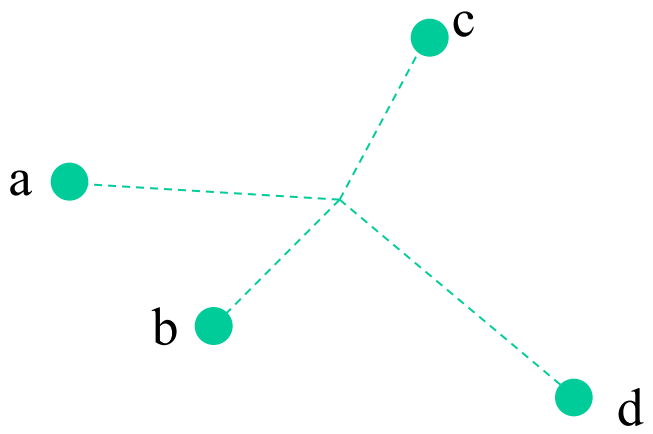


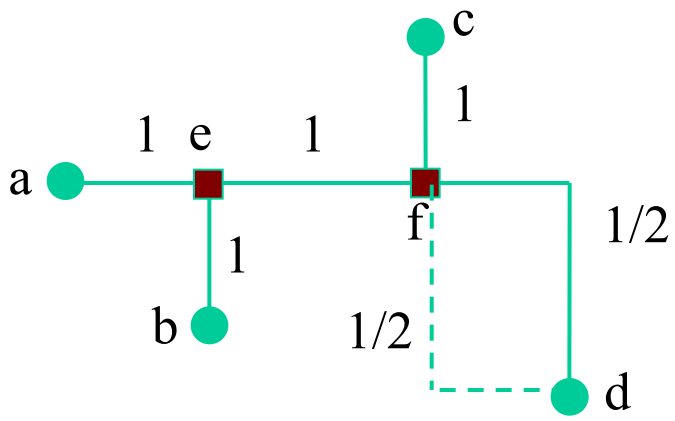
- Add ½ demand alone each edge of an L-shaped path



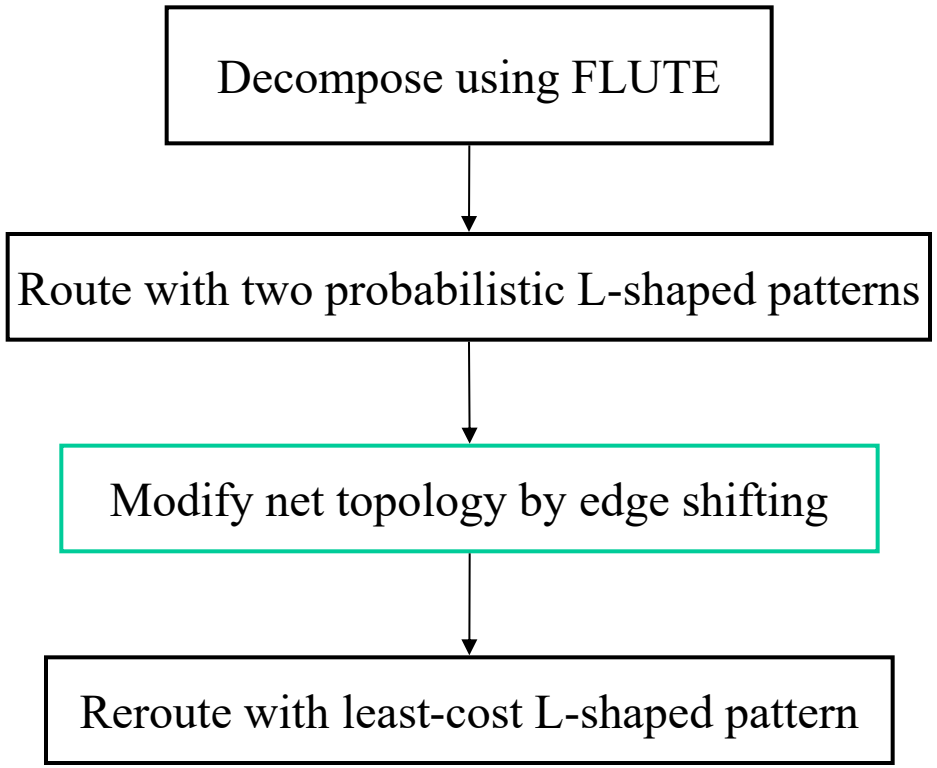
Initial Stage (cont'd)



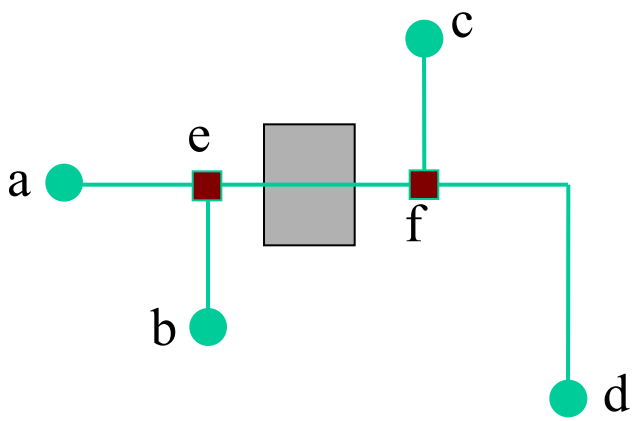




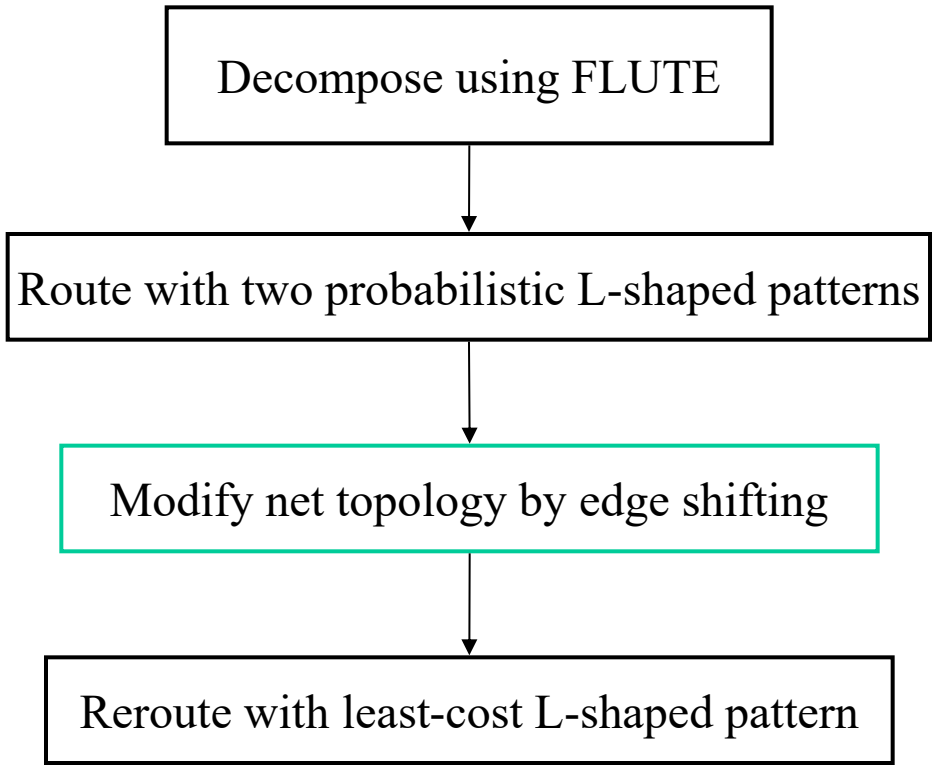
Initial Stage (cont'd)



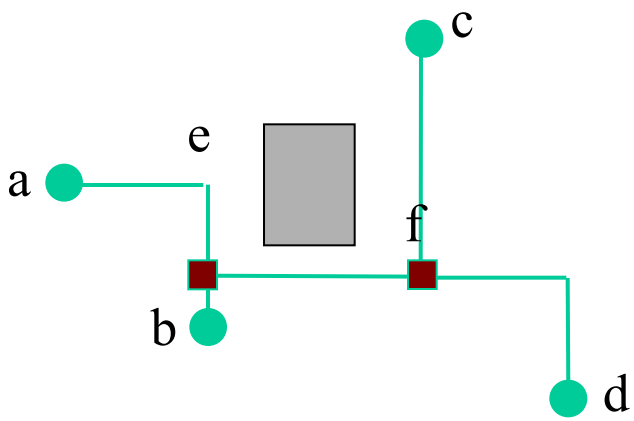
- Edge shifting [ICCAD06]
  - Move edges to less congested regions
  - Does not increase wirelength



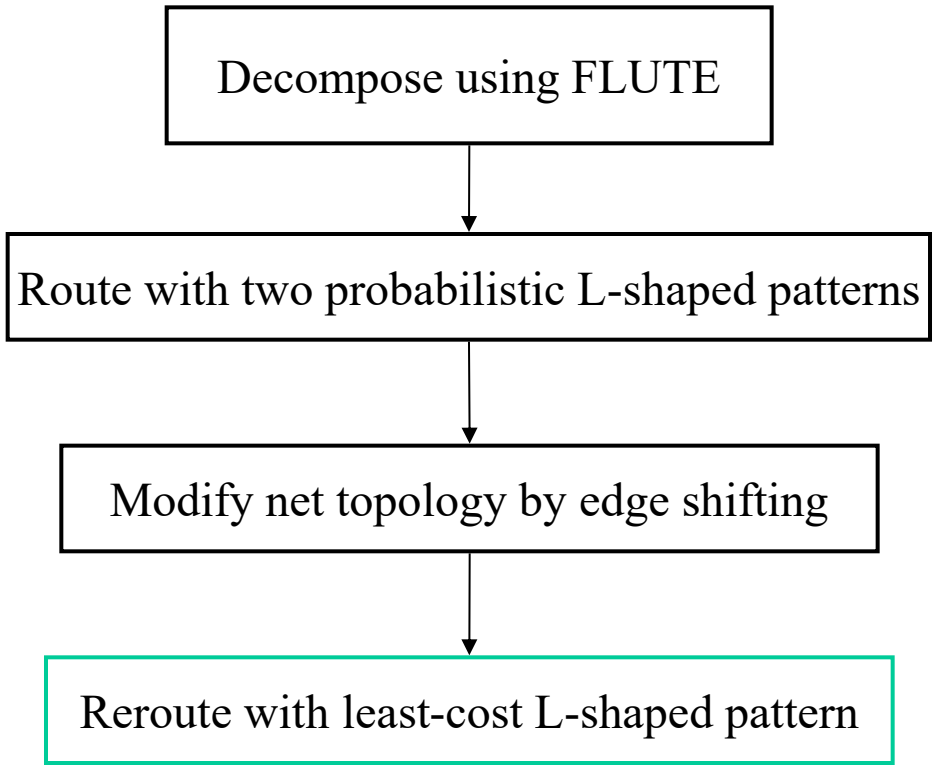
Initial Stage (cont'd)



- Edge Shifting [ICCAD06]
  - Move edges to less congested regions
  - Does not increase wirelength



Initial Stage (cont'd)



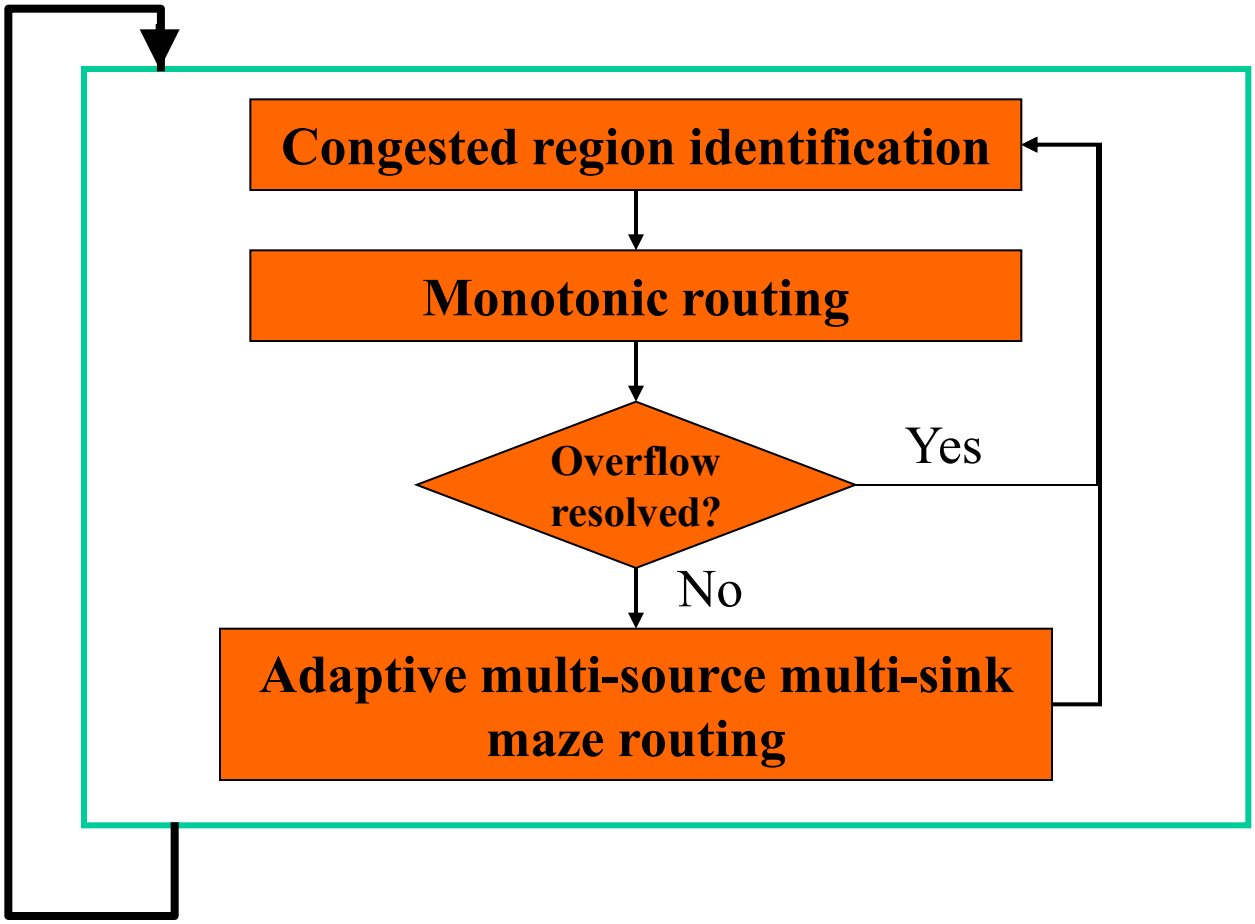
- Reroute all 2-pin nets in increasing order of the size of bounding box
- Add 1 demand to each edge along the chosen L-shaped path

$$cost_e = 1 + \frac{a}{1 + e^{-b*(d(e)-s(e))}}$$

[ICCAD06]

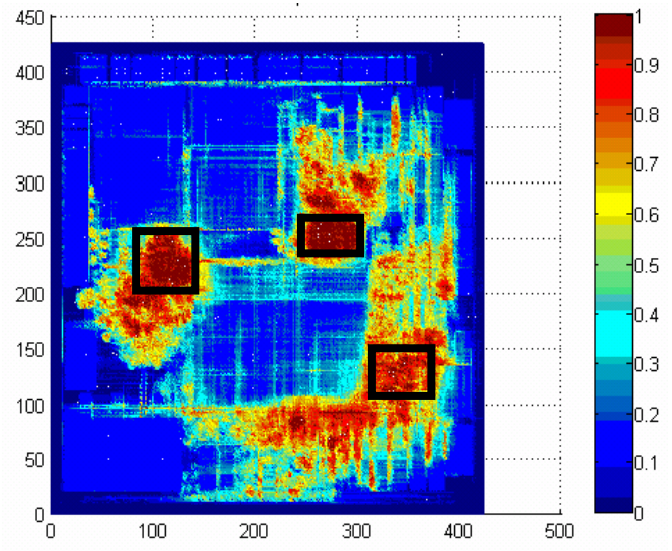
Main Stage

For a number of iterations



Rip-up and Reroute Based on Congested Region Identification

- The order of nets to be ripped up & rerouted affects the routing quality very much
- An algorithm is proposed to identify congested regions and rip-up & reroute two-pin nets with similar congestion at a time



(ISPD07 contest)

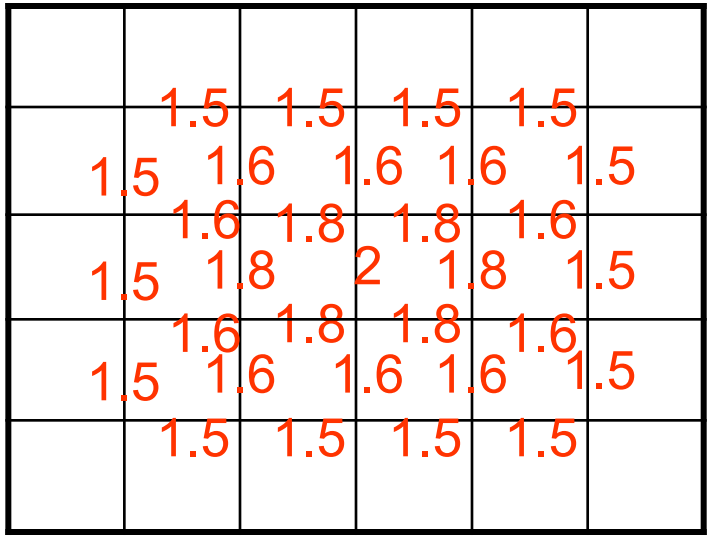
Congested Region Identification

- Partition the interval between the maximum congestion value and 1 into 10 sub-intervals
  - Each congested edge belongs to a sub-interval
- For each sub-interval I_k
  - Expand a rectangular region r_e from each edge e until $avg\_cong(r_e)$ is smaller than the lower bound of I_k

$$avg\_cong(r_e) = \frac{\sum d(e)}{\sum s(e)} \qquad e_i \text{ is an edge inside } r_e$$

  - Find two-pin nets within each region
  - For each 2-pin net in non-increasing order of bounding box size
    - Rip-up & reroute using monotonic routing
    - Adaptive multi-source multi-sink maze routing, if necessary

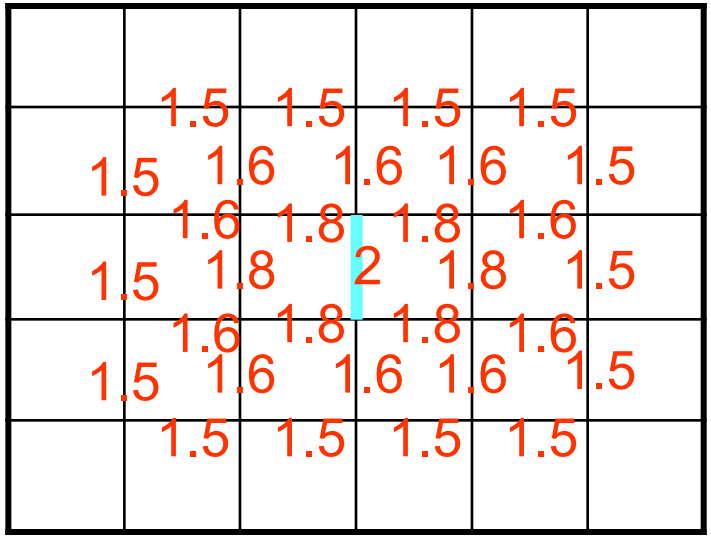
An Example of Region Identification



- Partition [2, 1]

  → {[1.1,1), [1.2, 1.1),..., [1.9, 1.8), [2, 1.9)}

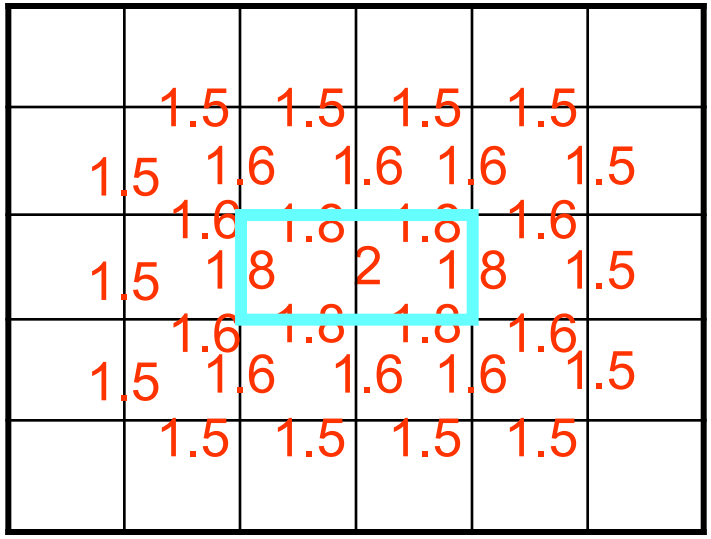
An Example of Region Identification



- Partition [2, 1]

  → {[1.1,1), [1.2, 1.1),..., [1.9, 1.8), [2, 1.9)}

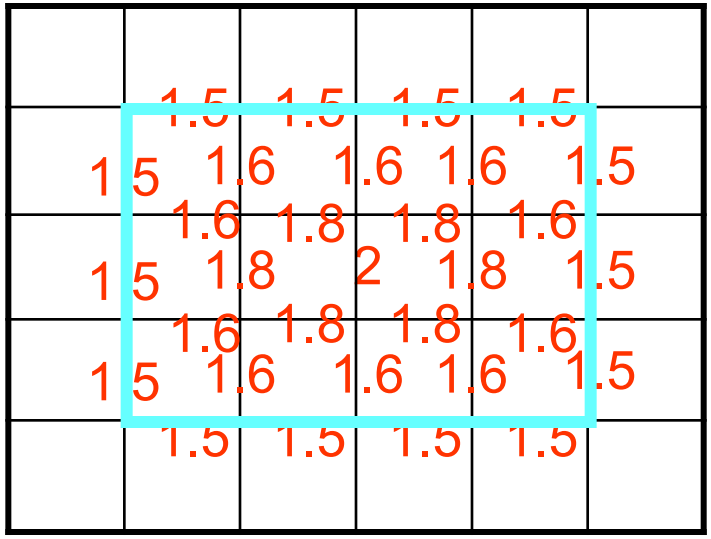
An Example of Region Identification



- Partition [2, 1]

  → {[1.1,1), [1.2, 1.1),..., [1.9, 1.8), [2, 1.9)}

An Example of Region Identification

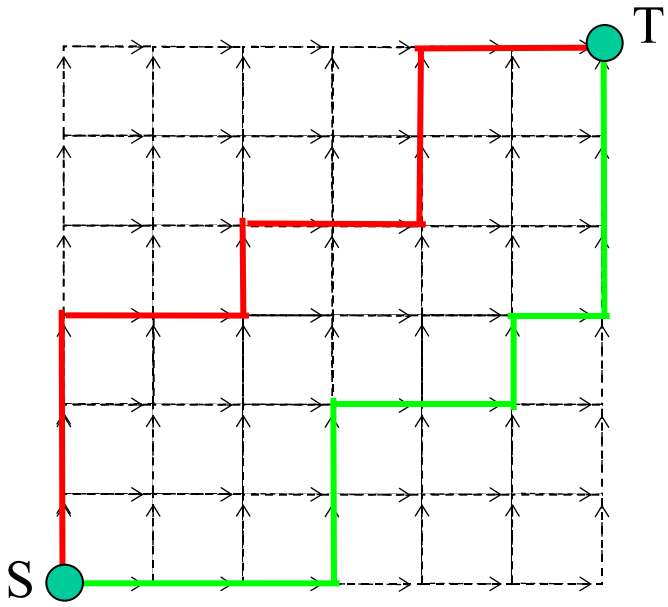


- Partition [2, 1]

  → {[1.1,1), [1.2, 1.1),..., [1.9, 1.8), [2, 1.9)}

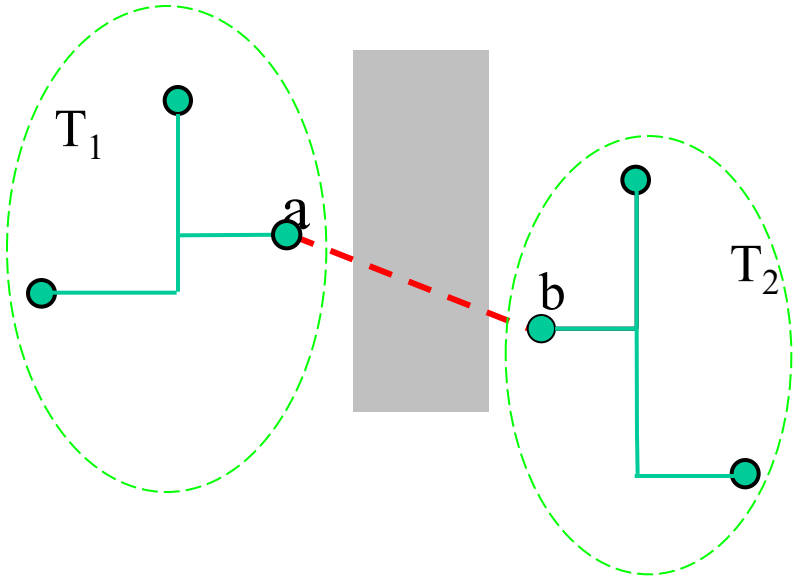
Monotonic Routing

- Proposed in [ASPDAC07]
- Finds the routing path in the direction toward *T* within the bounding box formed by *S* and *T*
- Can be done by dynamic programming with the same complexity as Z-shaped pattern routing

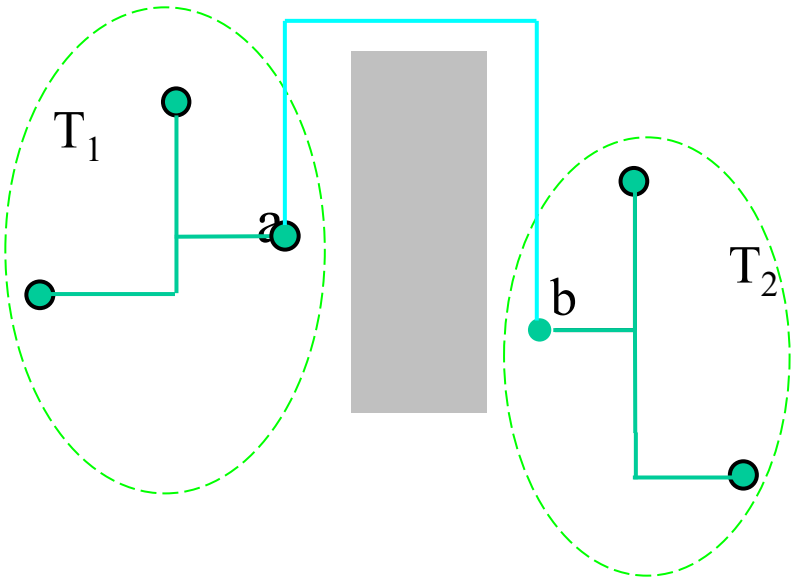


Adaptive Multi-Source Multi-Sink Maze Routing

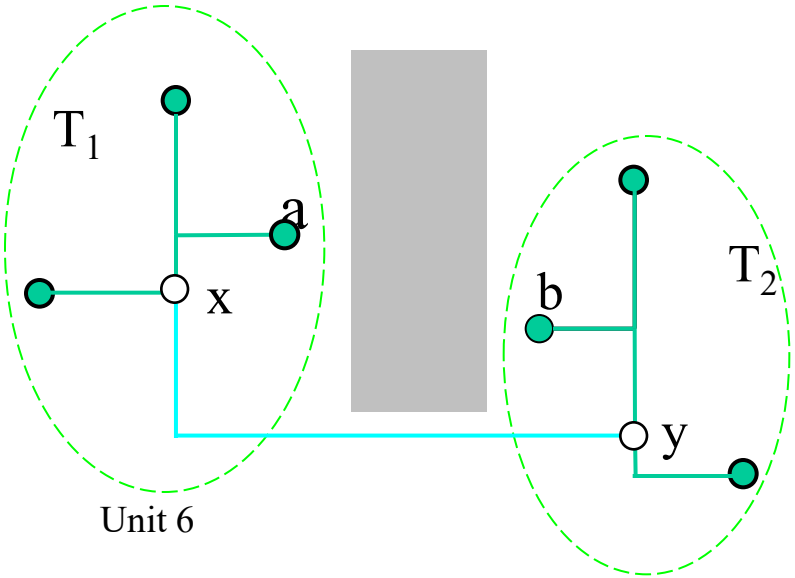
- General maze routing
  - One-source one-sink
- Multi-source multi-mink maze routing
  - Treat all grid points on one subtree as sources
  - Sinks are grid points on another subtree
- Adaptive multi-source multi-mink maze routing
  - Treat only pins or Steiner points as sources and sinks
  - More efficient



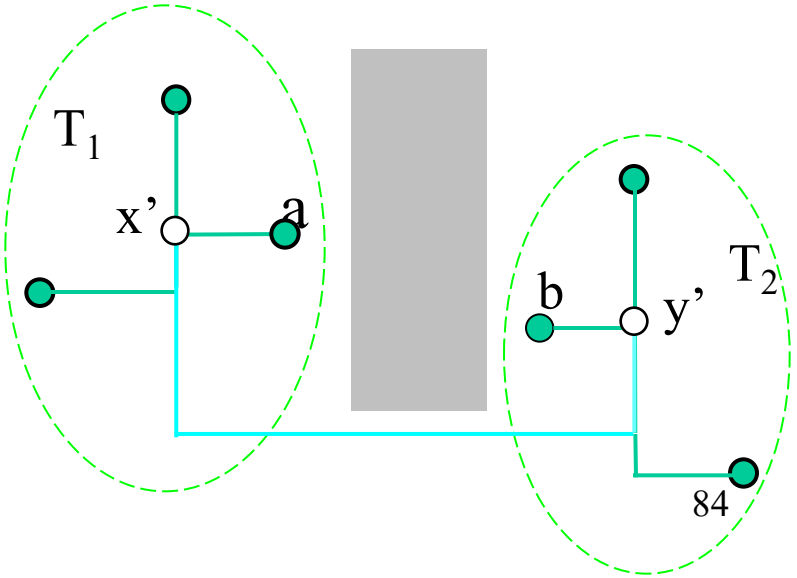
One-source one-sink



Multi-source multi-sink



Adaptive multi-source multi-sink



History Based Cost Function

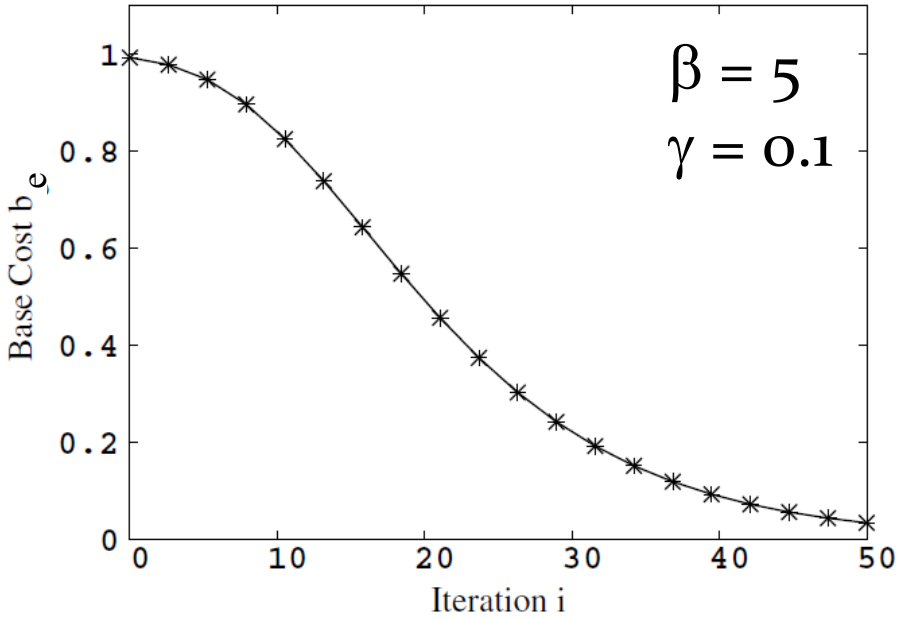
$$cost_e = b_e + h_e \times p_e + vc_e$$

- b_e: adaptive base cost function
- $h_e \times p_e$: congestion cost function with overflow prediction
- vc_e: via cost function for multi-layer design
- Sharing edges does not need additional cost

Adaptive Base Cost Function

- Provides a base cost between 1 to 0 adaptively

$$b_e = 1 - e^{-\beta e^{-\gamma i}}$$



Congestion Cost Function with Overflow Prediction

- $h_e \times p_e$: congestion cost
  - h_e: historical term

$$h_e^{i+1} = \begin{cases} h_e^i + 1 & \text{if } e \text{ has overflow} \\ h_e^i & \text{otherwise} \end{cases}$$

  - p_e: current congestion penalty

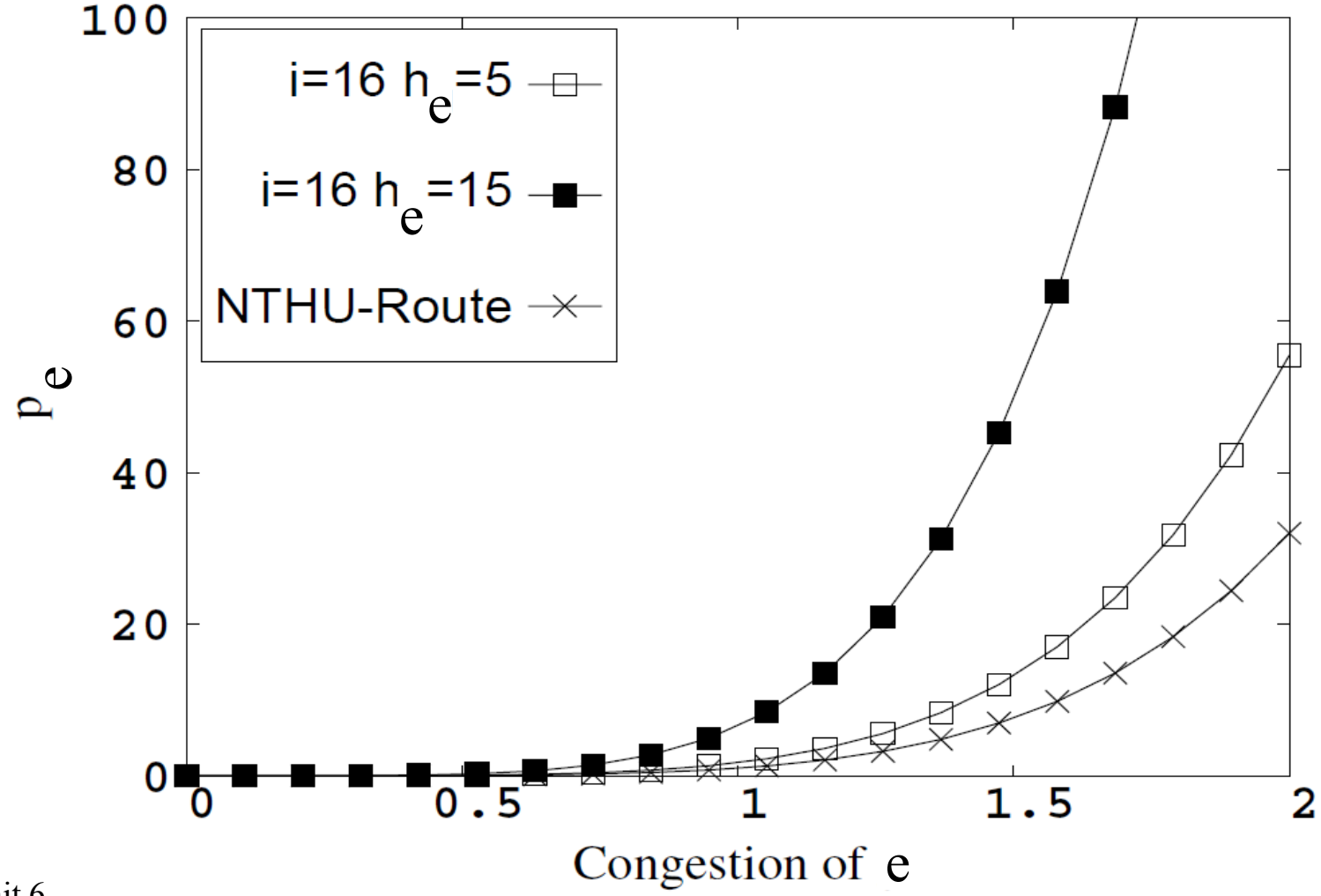
$$p_e = \left(\frac{d_e + 1}{c_e} \times f(h_e, i) \right)^{k1}$$

$$f(h_e, i) = \left(\frac{i \times (k_2 + adj(i))}{i \times (k_2 + adj(i)) - (h_e - 1)} \right)$$

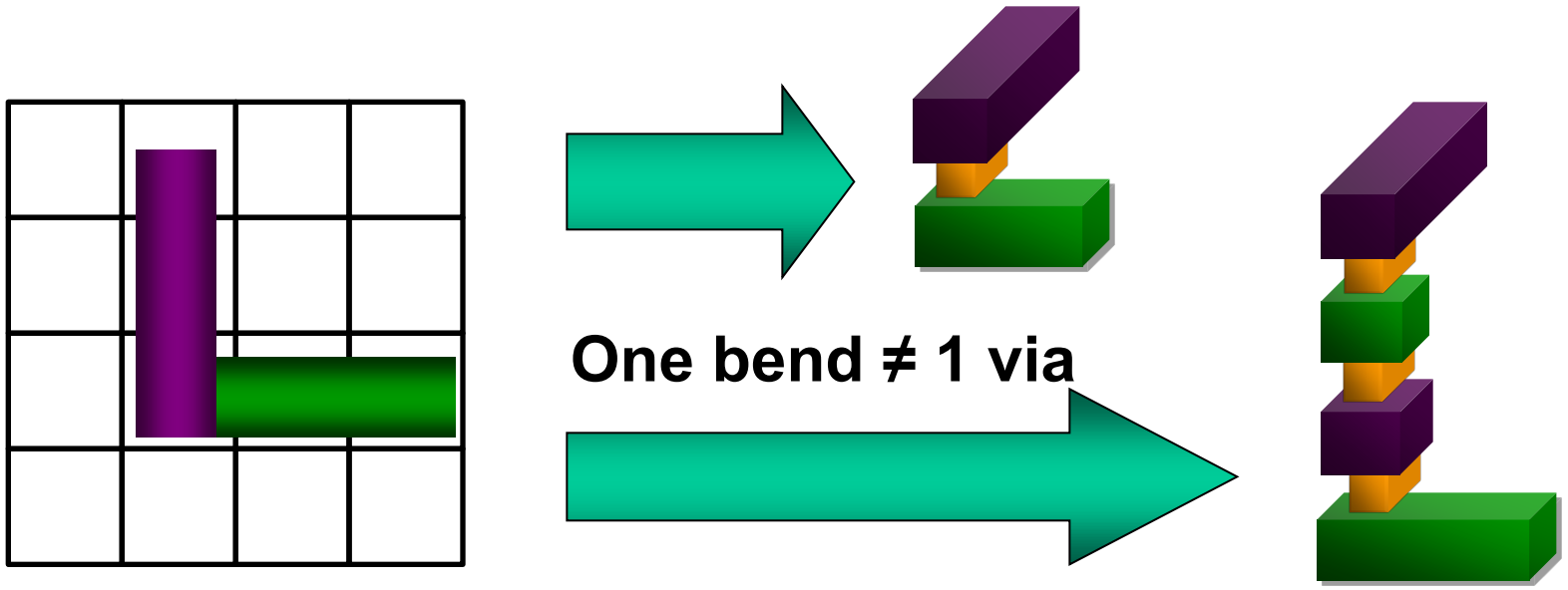
$$adj(i) = k_3 \times (1 - e^{-\beta e^{-\gamma i}})$$

    - $f(h_e, i)$: amplify congestion value if e has high probability to have overflow

Curves of Penalty Terms



Via Cost Function for Multi-layer Design



$$vc_e = \begin{cases} v_e \times c_e \times b_e & \text{if passing e makes a bend,} \\ 0 & \text{otherwise} \end{cases}$$

- v_e: the expected amount of vias for a bend
- c_e: the wirelength of a via
- b_e: adaptive base cost function