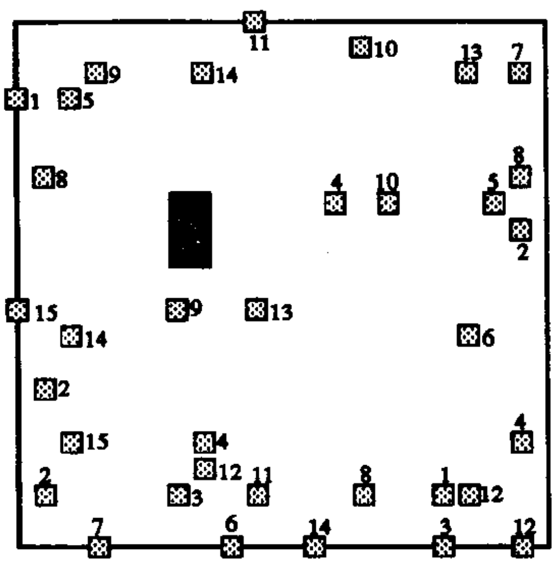
Multi-net Routing

ECE6133
Physical Design Automation of VLSI Systems

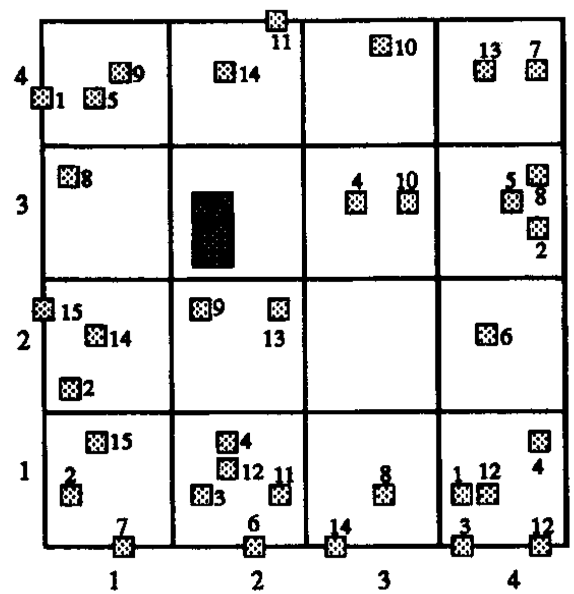
Prof. Sung Kyu Lim
School of Electrical and Computer Engineering
Georgia Institute of Technology

Global Routing

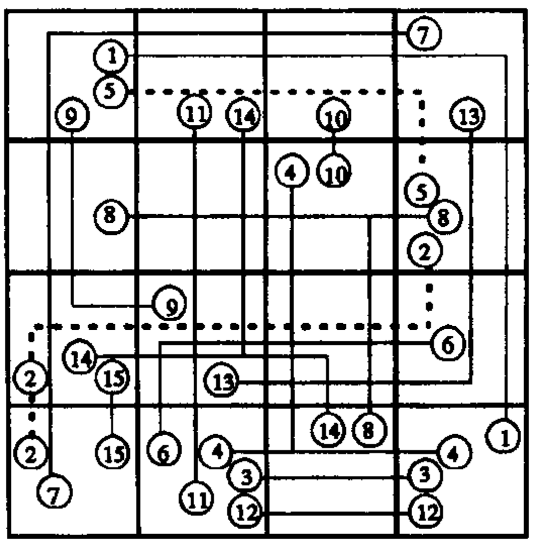
- **Global routing is planning**
  - Divide the routing into tiles
  - Build Steiner tree for each net
  - Routing is done in terms of tiles



Routing Problem



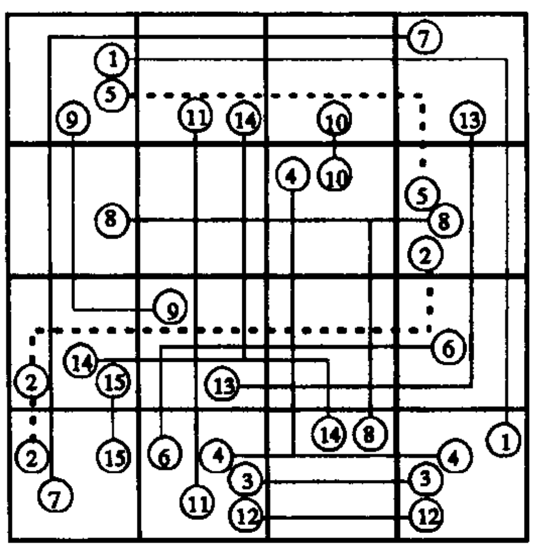
Global Routing Tiles



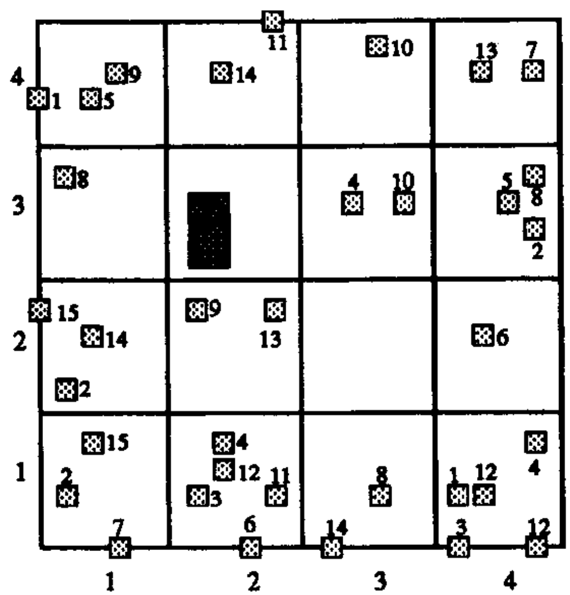
Global Routing Result

Cross Point Assignment

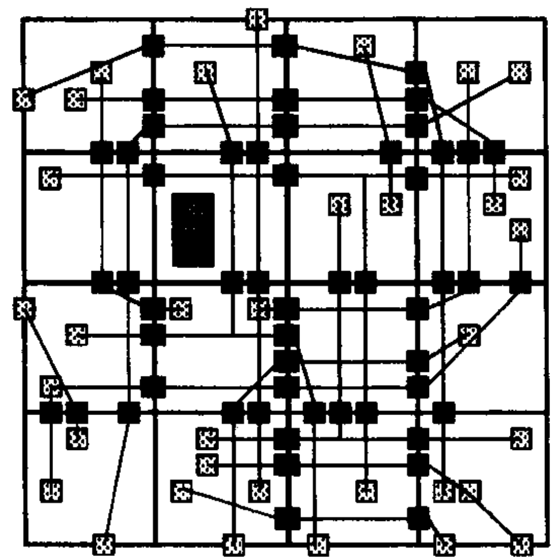
- **Key step before detailed routing**
  - CPA decides pin locations along tile boundaries
  - Key objective is routing completion, via usage, and wirelength



Global Routing Result (repeat)



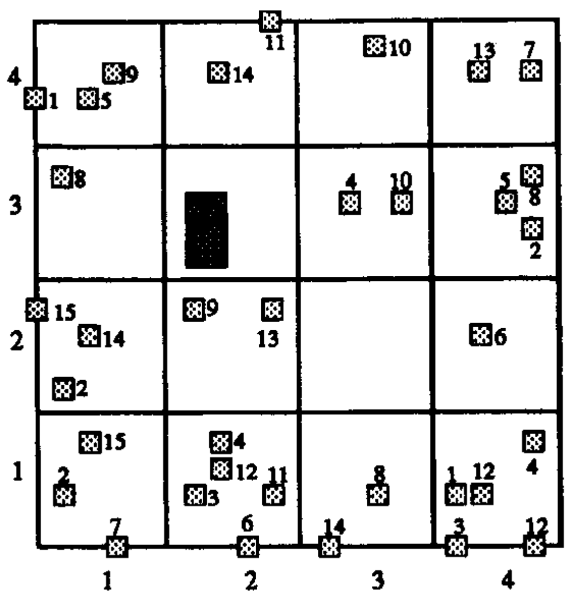
Routing Problem (repeat)



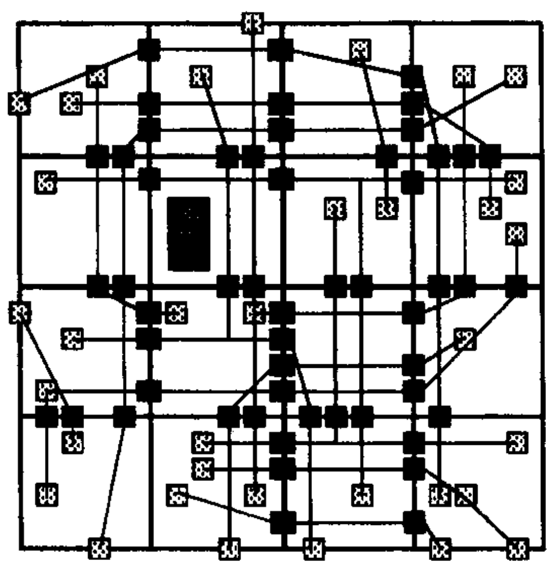
Cross Point Assignment

Detailed Routing

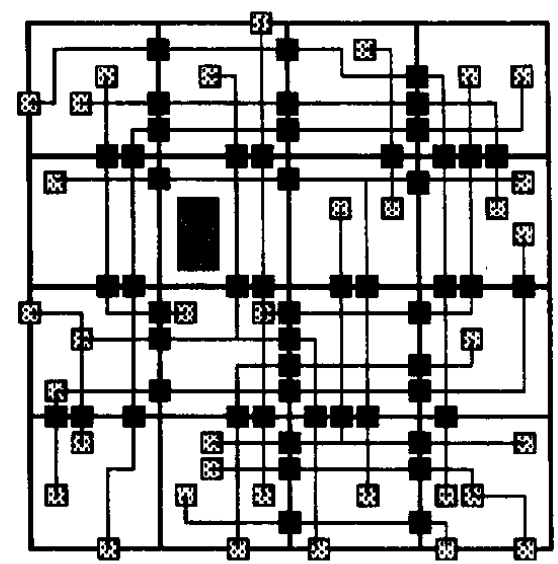
- Detailed routing decides exact topology
  - We use CPA results
  - We use the actual routing tracks and vias in each tile



Routing Problem (repeat)



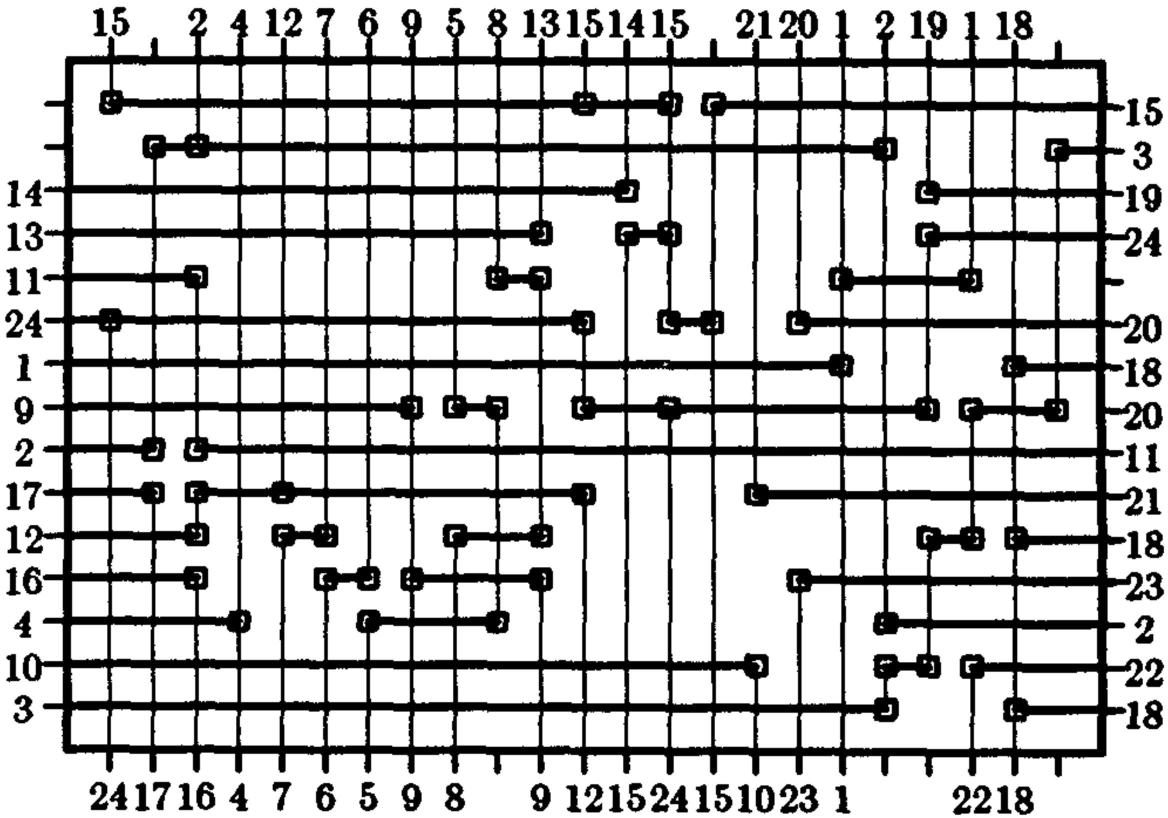
Cross Point Assignment



Detailed Routing

Type 1: Switchbox Routing

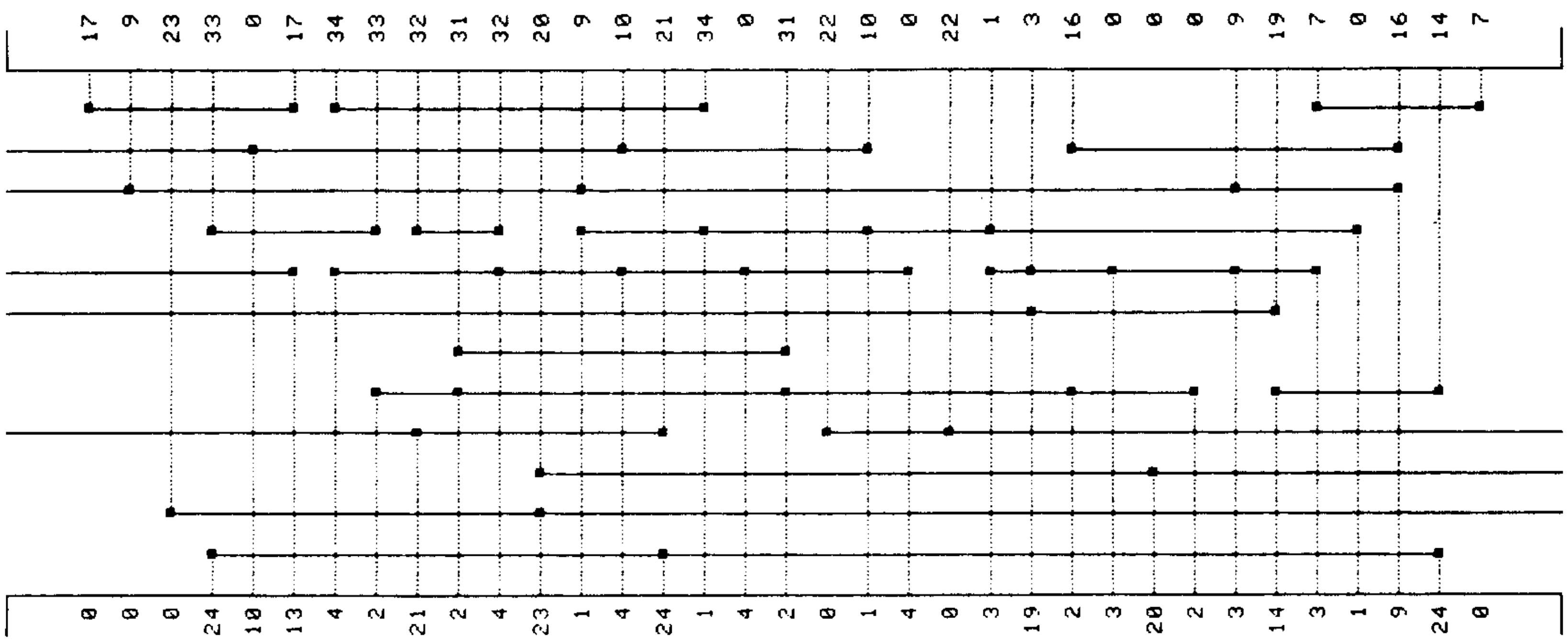
- Key problem for detailed routing
  - CPA gives pin locations on all 4 sides



we assume two metal layers (H and V) in this case

Type 2: Channel Routing

- **Key problem for detailed routing**
  - **CPA gives pin locations on 2 sides**

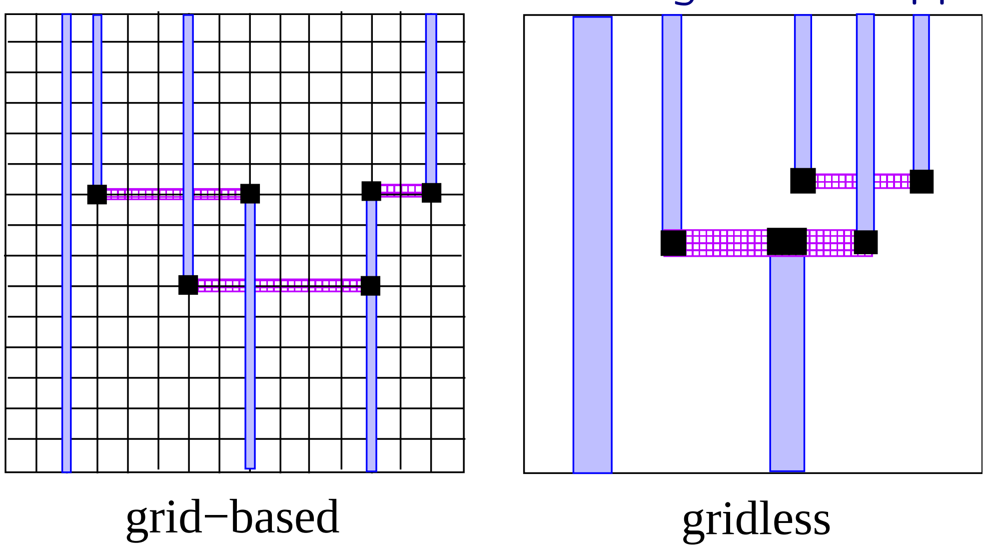


two metal layers (H and V) again

number of tracks = 12
maximum density = 12

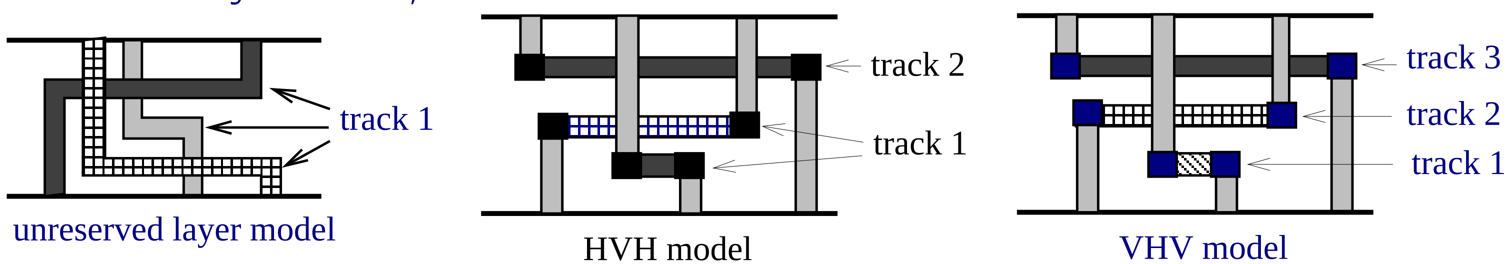
Routing Models

- **Grid-based model:**
  - A grid is super-imposed on the routing region.
  - Wires follow paths along the grid lines.
- **Gridless model:**
  - Any model that does not follow this "gridded" approach.



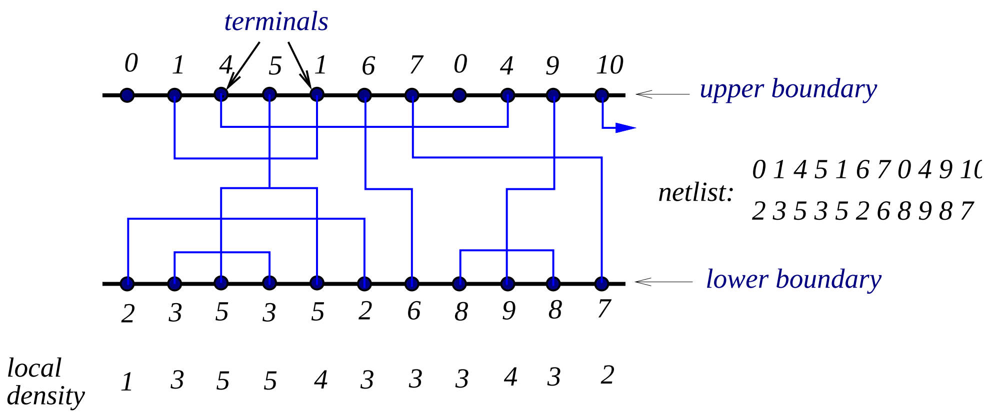
Models for Multi-Layer Routing

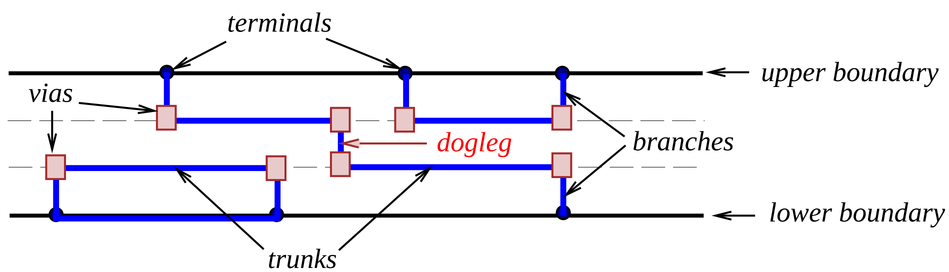
- **Unreserved layer model:** Any net segment is allowed to be placed in any layer.
- **Reserved layer model:** Certain type of segments are restricted to particular layer(s).
  - Two-layer: HV (horizontal-Vertical), VH
  - Three-layer: HVH, VHV



3 types of 3−layer models

Terminology for Channel Routing Problems





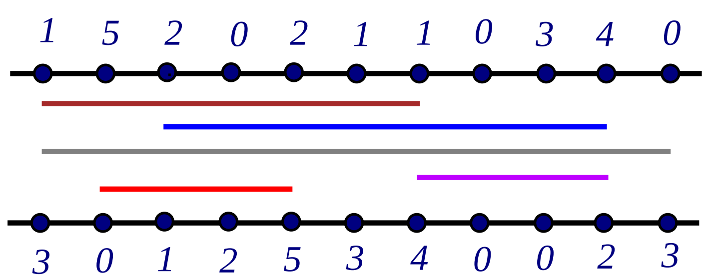
- Local density at column i: total # of nets that crosses column i.
- Channel density: maximum local density; # of horizontal tracks required $\geq$ channel density.

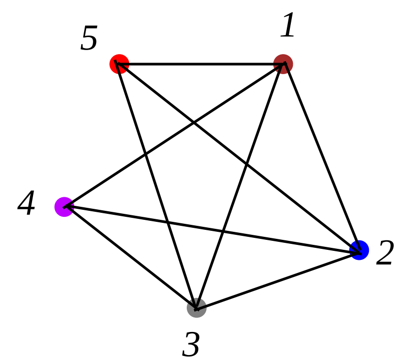
Channel Routing Problem

- **Assignments of horizontal segments of nets to tracks.**
- **Assignments of vertical segments to connect.**
  - horizontal segments of the same net in different tracks, and
  - the terminals of the net to horizontal segments of the net.
- **Horizontal and vertical constraints must not be violated.**
  - Horizontal constraints between two nets: The horizontal span of two nets overlaps each other.
  - Vertical constraints between two nets: There exists a column such that the terminal on top of the column belongs to one net and the terminal on bottom of the column belongs to the other net.
- **Objective: Channel height is minimized** (i.e., channel area is minimized).

Horizontal Constraint Graph (HCG)

- HCG $G = (V, E)$ is **undirected** graph where
  - $V = \{v_i | v_i$ represents a net $n_i\}$
  - $E = \{(v_i, v_j) |$ a horizontal constraint exists between n_i and $n_j\}$.
- For graph G: vertices $\Leftrightarrow$ nets; edge $(i, j) \Leftrightarrow$ net i overlaps net j.

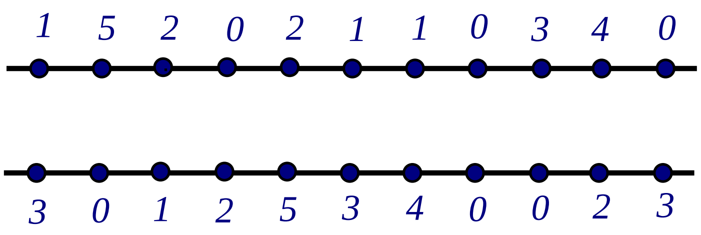




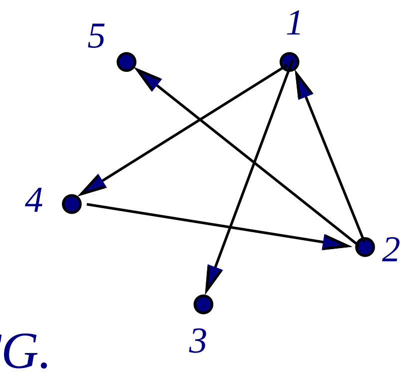
A routing problem and its HCG.

Vertical Constraint Graph (VCG)

- VCG $G = (V, E)$ is **directed** graph where
  - $V = \{v_i | v_i$ represents a net $n_i\}$
  - $E = \{(v_i, v_j)|$ a vertical constraint exists between n_i and $n_j\}$.
- For graph G: vertices $\Leftrightarrow$ nets; edge $i \rightarrow j \Leftrightarrow$ net i must be above net j.



A routing problem and its VCG.



2-L Channel Routing: Basic Left-Edge Algorithm

- Hashimoto & Stevens, "Wire routing by optimizing channel assignment within large apertures," DAC-71.
- **No vertical constraint.**
- HV-layer model is used.
- **Doglegs are not allowed.**
- Treat each net as an interval.
- Intervals are sorted according to their left-end x-coordinates.
- Intervals (nets) are routed one-by-one according to the order.
- For a net, tracks are scanned from top to bottom, and the first track that can accommodate the net is assigned to the net.
- Optimality: produces a routing solution with the minimum # of tracks (if no vertical constraint).

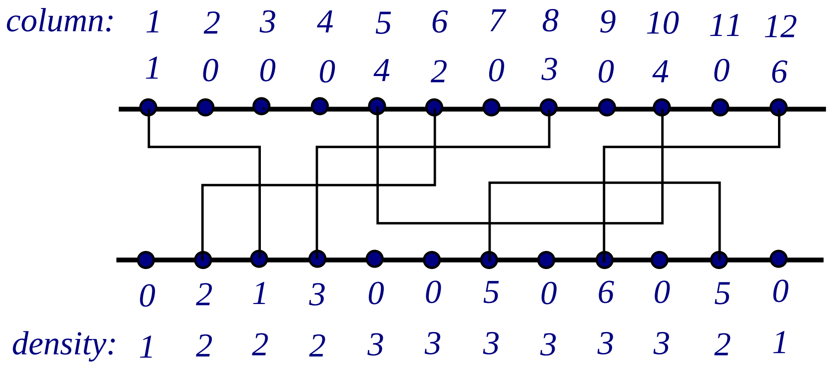
Basic Left-Edge Algorithm

Algorithm: Basic_Left-Edge($U, track[j]$)
U: set of unassigned intervals (nets) $I_1, \ldots, I_n$;
$I_j = [s_j, e_j]$: interval j with left-end x-coordinate s_j and right-end e_j;
$track[j]$: track to which net j is assigned.

begin
$U \leftarrow \{I_1, I_2, \ldots, I_n\}$;
$t \leftarrow 0$;
while ($U \neq \emptyset$) **do**
  $t \leftarrow t + 1$;
  $watermark \leftarrow 0$;
  while (there is an $I_j \in U$ s.t. $s_j > watermark$) **do**
    Pick the interval $I_j \in U$ with $s_j > watermark$,
    nearest $watermark$;
    $track[j] \leftarrow t$;
    $watermark \leftarrow e_j$;
    $U \leftarrow U - \{I_j\}$;
end

Basic Left-Edge Example

- $U = \{I_1, I_2, \ldots, I_6\}$; $I_1 = [1,3]$, $I_2 = [2,6]$, $I_3 = [4,8]$, $I_4 = [5,10]$, $I_5 = [7,11]$, $I_6 = [9,12]$.
- $t = 1$:
  - Route I_1: $watermark = 3$;
  - Route I_3: $watermark = 8$;
  - Route I_6: $watermark = 12$;
- $t = 2$:
  - Route I_2: $watermark = 6$;
  - Route I_5: $watermark = 11$;
- $t = 3$: Route I_4



Constrained Left-Edge Algorithm

Algorithm: Constrained_Left-Edge($U, track[j]$)

U: set of unassigned intervals (nets) $I_1, \ldots, I_n$;

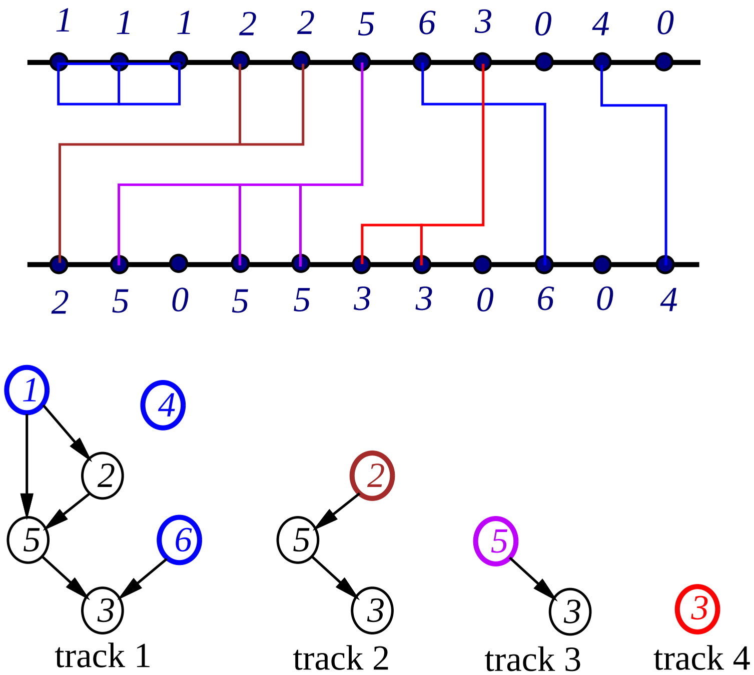
$I_j = [s_j, e_j]$: interval j with left-end x-coordinate s_j and right-end e_j;

$track[j]$: track to which net j is assigned.

begin
$U \leftarrow \{I_1, I_2, \ldots, I_n\}$;
$t \leftarrow 0$;
while $(U \neq \emptyset)$ **do**
  $t \leftarrow t + 1$;
  $watermark \leftarrow 0$;
  while (there is an **unconstrained** $I_j \in U$ s.t. $s_j > watermark$) **do**
      Pick the interval $I_j \in U$ that is unconstrained,
      with $s_j > watermark$, nearest $watermark$;
      $track[j] \leftarrow t$;
      $watermark \leftarrow e_j$;
      $U \leftarrow U - \{I_j\}$;
  end

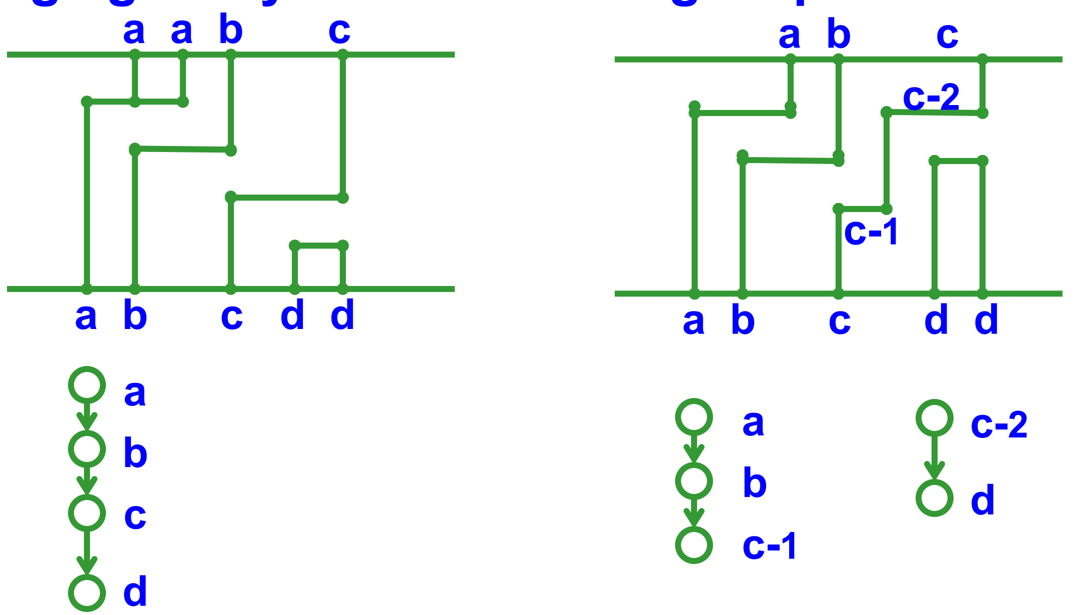
Constrained Left-Edge Example

- $I_1 = [1,3]$, $I_2 = [1,5]$, $I_3 = [6,8]$, $I_4 = [10,11]$, $I_5 = [2,6]$, $I_6 = [7,9]$.
- Track 1: Route I_1 (cannot route I_3); Route I_6; Route I_4.
- Track 2: Route I_2; cannot route I_3.
- Track 3: Route I_5.
- Track 4: Route I_3.

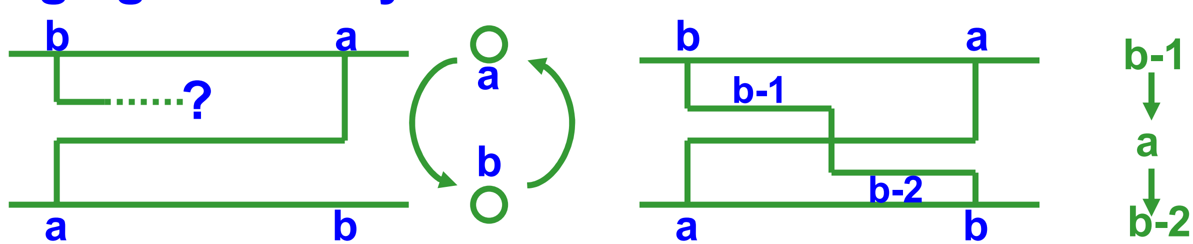


Doglegs in Channel Routing

- Doglegs may reduce the longest path in VCG

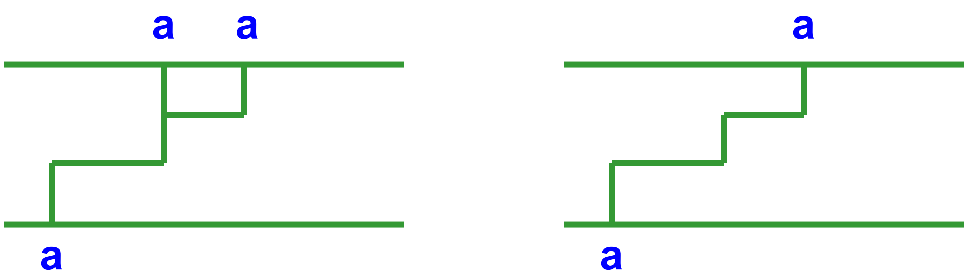


- Doglegs break cycles in VCG



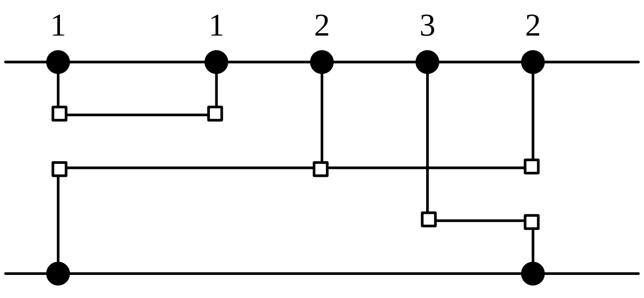
Doglegs in Channel Routing(Cont'd)

- Restricted Dogleg vs unrestricted dogleg

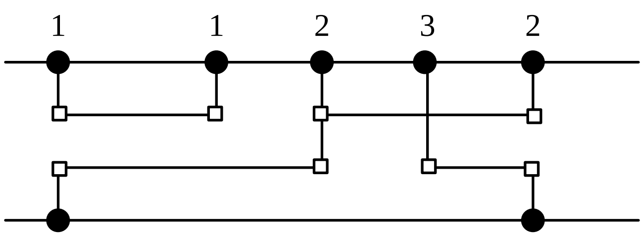


Dogleg Router

- Drawback of LEA: the entire net is on a single track.
- Doglegs are used to place parts of a net on different tracks, thereby minimizing channel height.



(a)

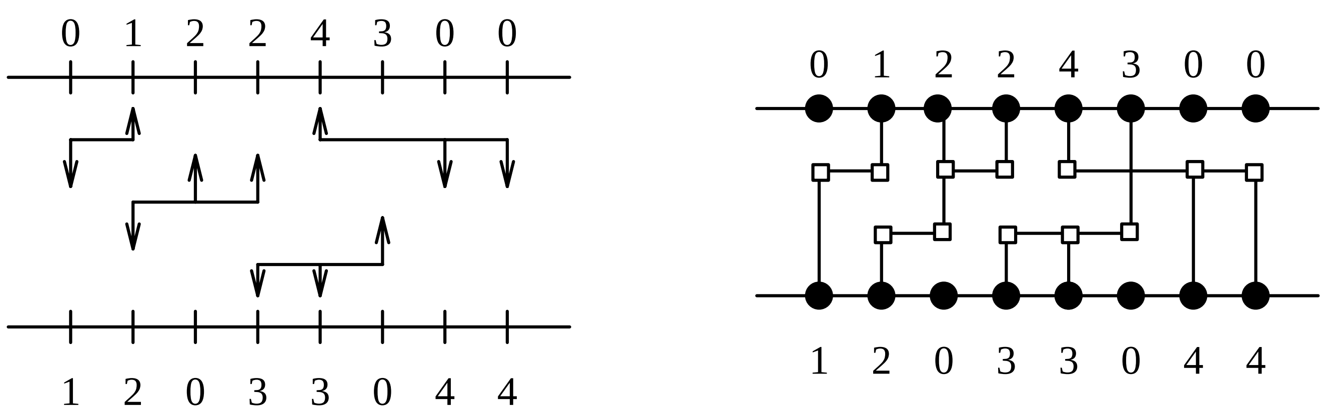


(b)

Using a dogleg to reduce channel height

Dogleg Router

- Each Multi-terminal net is broken into a set of two-terminal nets.
- Two parameters are used to control routing:
  1. range: Determine the number of consecutive two-terminal subnets of the same net that can be placed on the same track.
  2. routing sequence: Specifies the starting position and the direction of routing along the channel.
- Modified LEA is applied to each subnet.



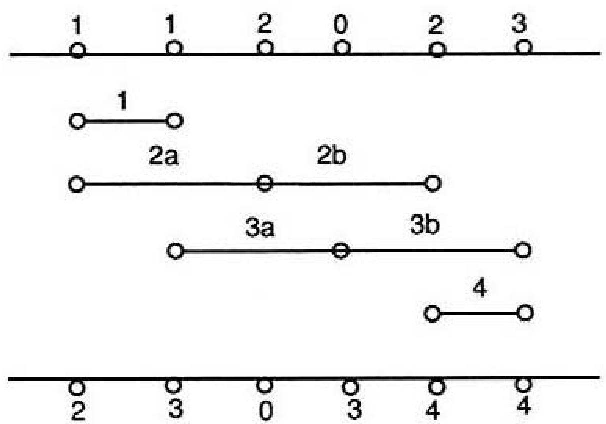
(a) (b)

Example of Dogleg Router

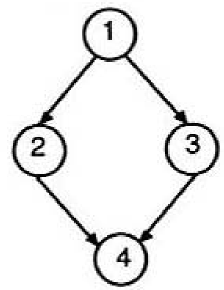
Deutsch, *ICCAD* 1985

Dogleg Router: Example

- **Decompose multi-terminal nets into two-terminal nets**

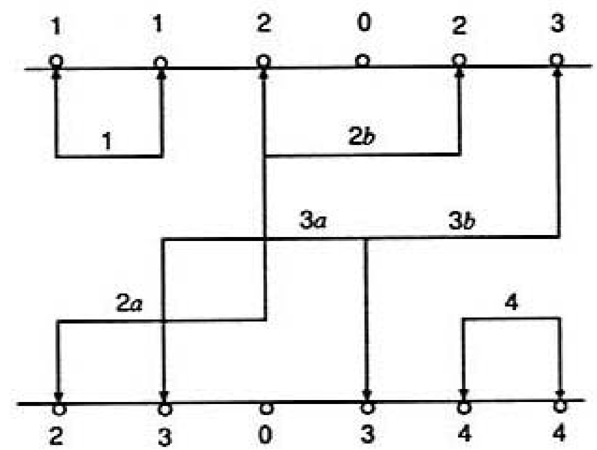


(a)



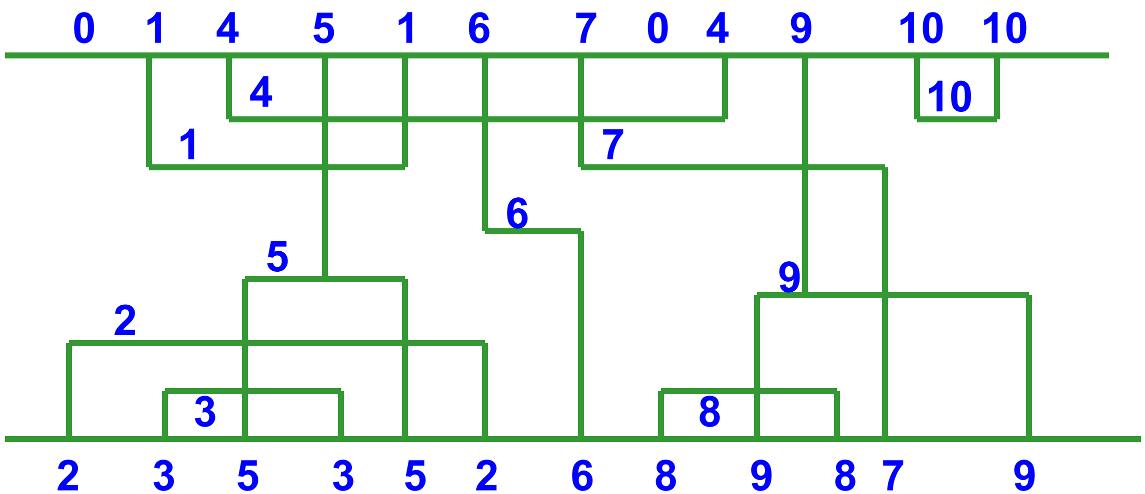
(b)

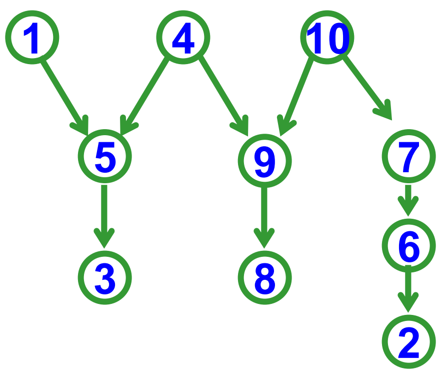
(c)



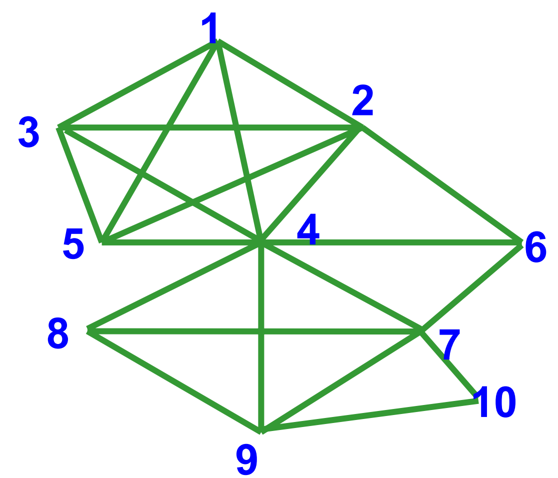
Final solution

Characterizing Channel Routing Problem





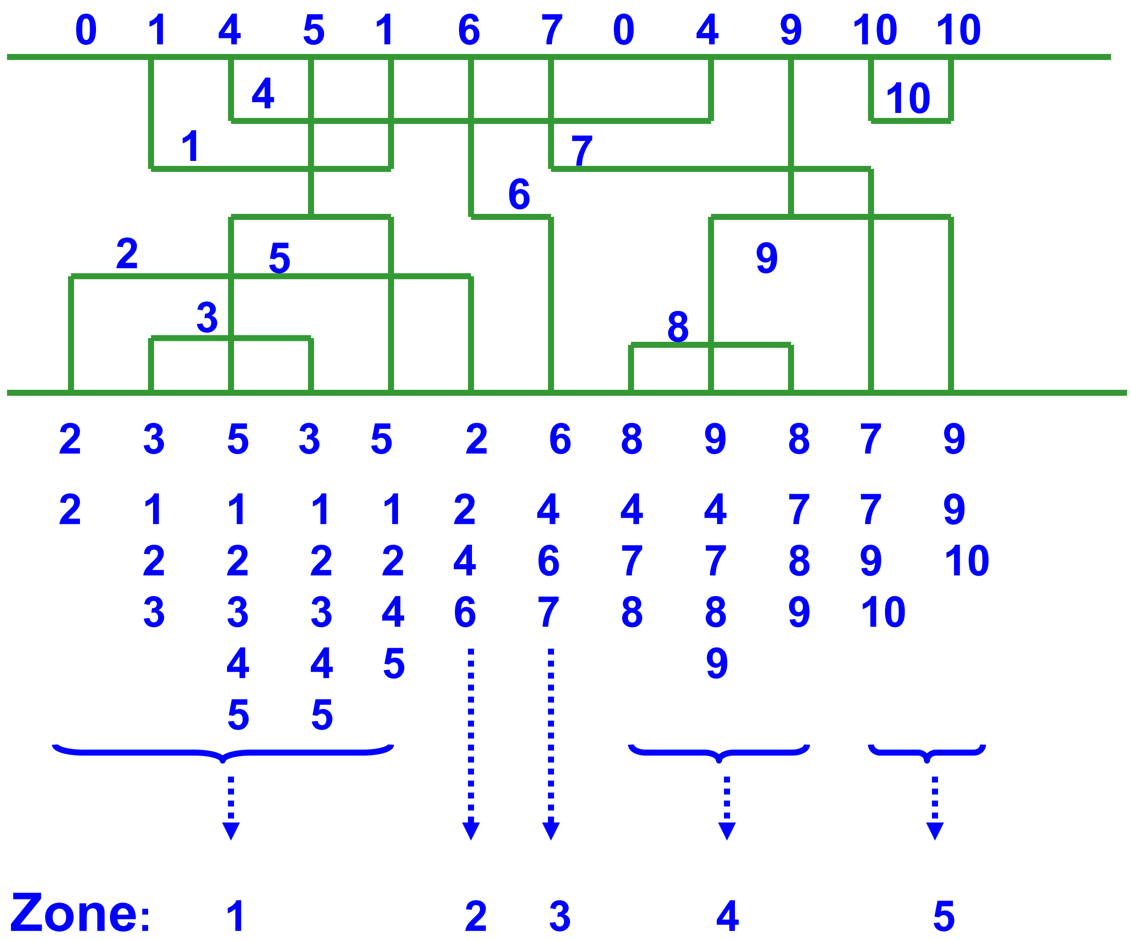
Vertical constraint graph G_v



Horizontal constraint graph

The channel routing problem is completely characterized by the vertical constraint graph and the horizontal constraint graph.

Zone Representation of Horizontal Segments



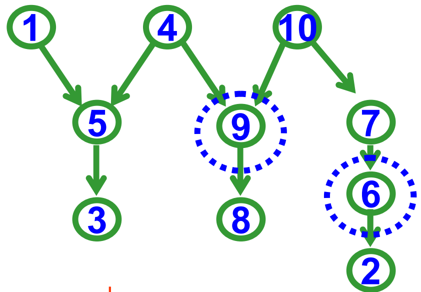
Zone Representation of Horizontal Segments(Cont'd)

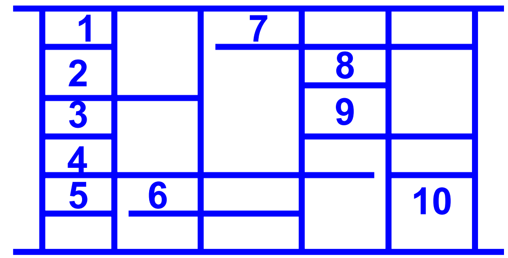
Zone representation

S(i): set of nets intersect column i

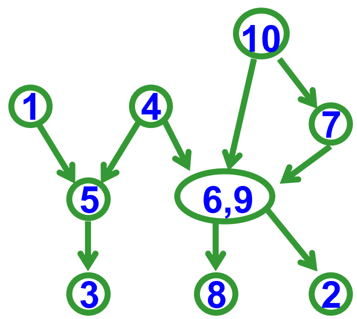
- **we only need to consider those s(i)s which are maximal**
- **Zone $\leftrightarrow$ maximal clique in the horizontal constraint graph**

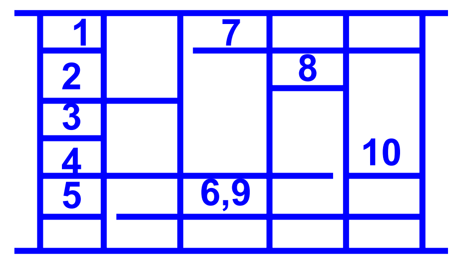
Merging of Nets





? and ⌖ can be merged





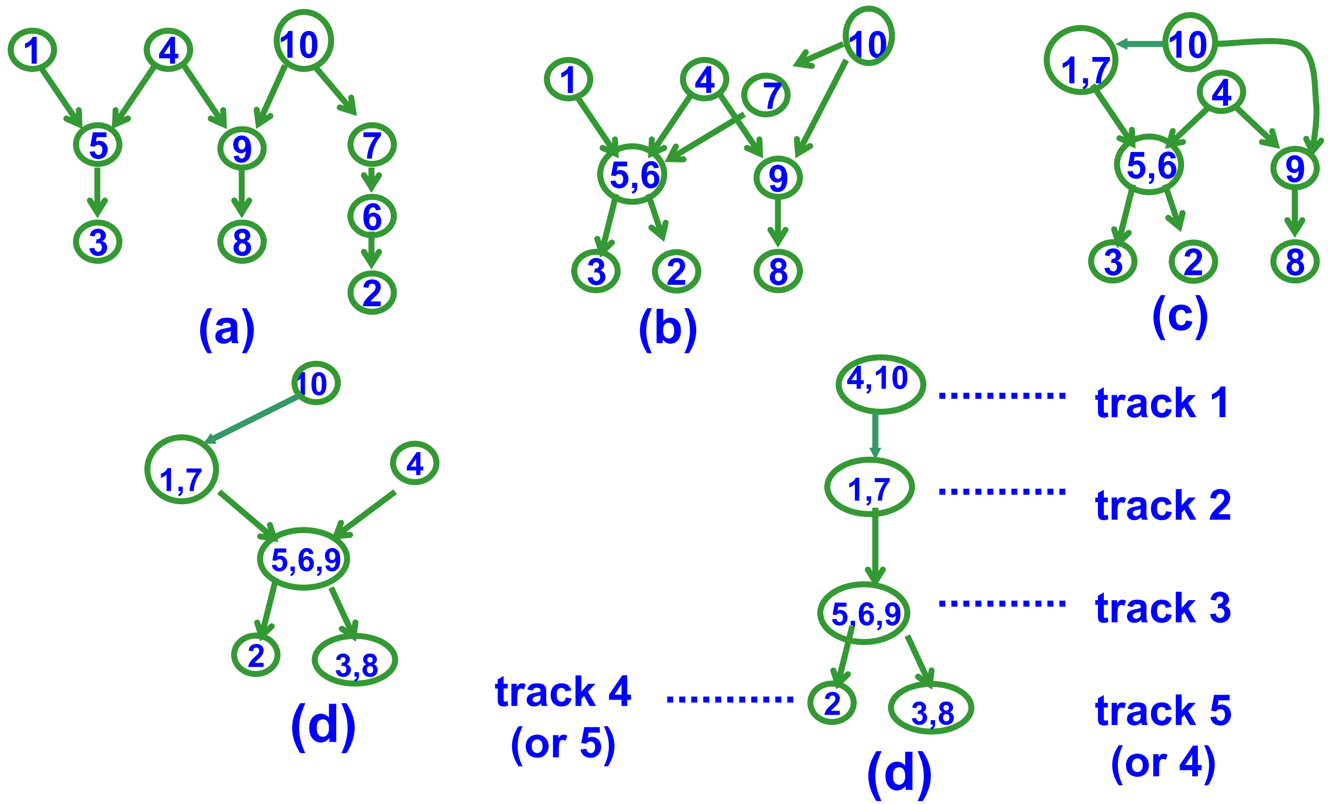
Updated graph and zone rep

Net i and net j can be merged if

(a) there is no path (directed connecting them in VCG;

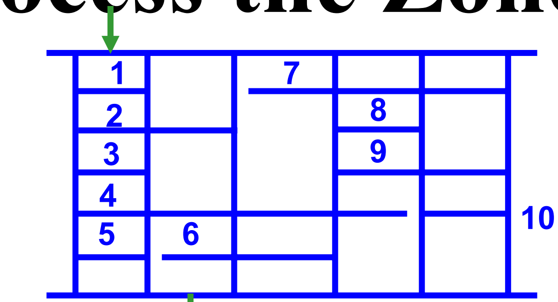
(b) the two nets do not overlap

Change of VCGs



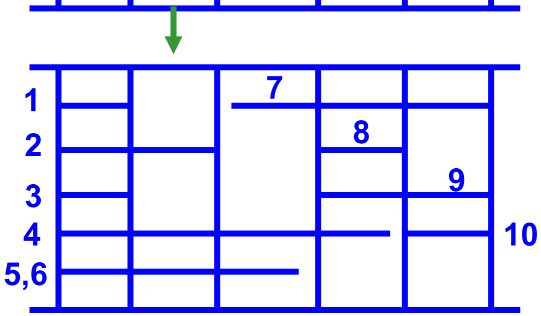
How to choose two feasible nets to merge?
⇒Determine the quality of the solutions

Process the Zones Sequentially



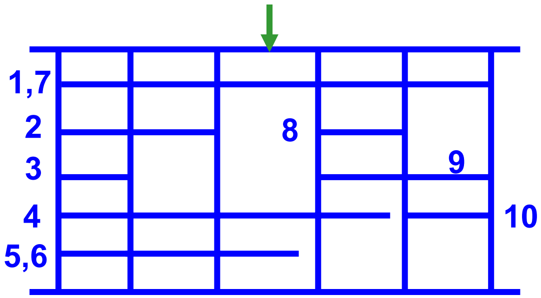
LEFT={1,3,5}

RIGHT={6)



LEFT={1,2,3}

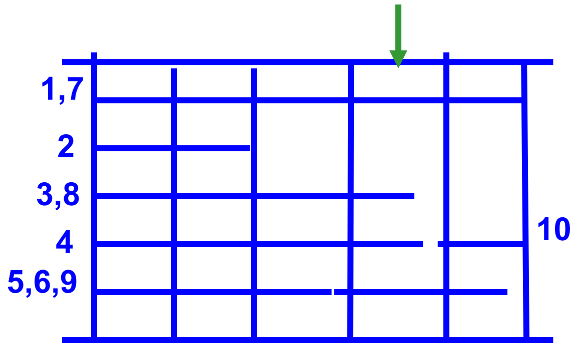
RIGHT={7)



LEFT={2,3,5.6}

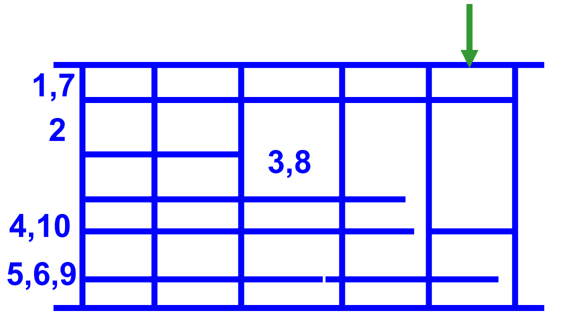
RIGHT={8,9)

Process the Zones Sequentially
(Cont'd)



LEFT={2,3.8,4}

RIGHT={10)

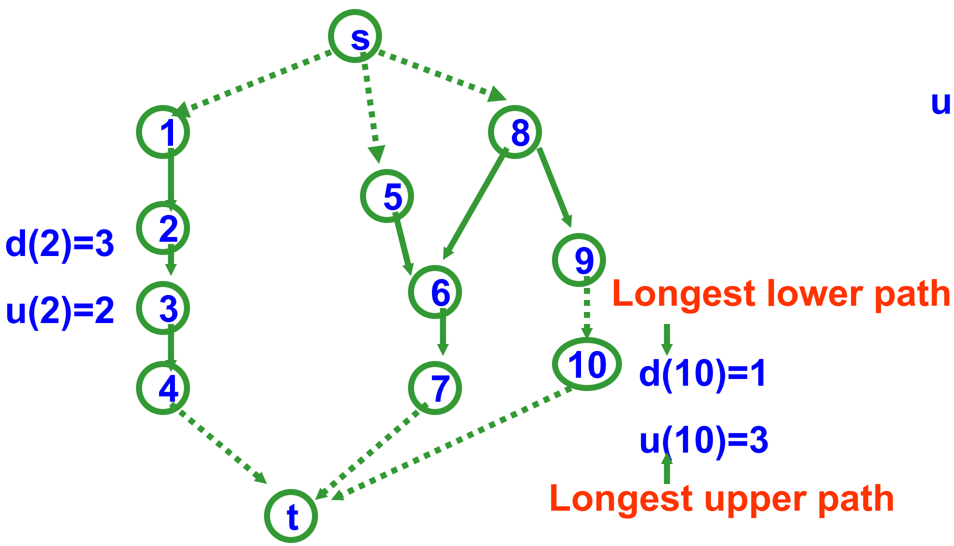


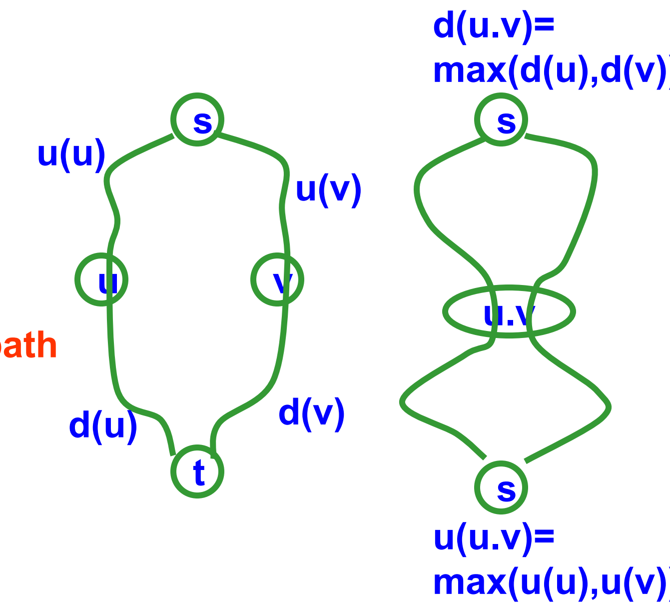
LEFT={1.7, 2, 3.8, 4.10, 5.6.9}

RIGHT= ϕ

First Approach

- **Merge LEFT and RIGHT so as to minimize the increase of the longest path length in the VCG**
- **Heuristic rule to select nets to merge sequentially**





What to Choose from P/Q?

The purpose here is to minimize the length of the longest path after merger. However, it will be too time consuming to find an exact minimum merger, hence a heuristic merging algorithm will be given. Let us introduce some basic intuitive ideas. First, a node $m \in Q$ is chosen, which lies on the longest path before merger; furthermore, it is farthest away from either s or t. Next, a node $n \in P$ is chosen such that the increase of the longest path after merger is minimum. If there are two or more nodes which will result in a minimum increase we choose n such that $u(n) + d(n)$ is maximum or nearly maximum and that the condition $u(m)/d(m) = u(n)/d(n)$ is satisfied or nearly satisfied. These can be implemented by introducing the following:

Heuristic

Merging Algorithm

given P, Q;
    begin;
a1:   while Q is not empty do;
        begin;
a2:   among Q, find m^* which maximizes $f(m)$;
a3:   among P, find n^* which minimizes $g(n, m^*)$, and which is neither ancestor nor descendent of m^*;
a4:   merge n^* and m^*;
a5:   remove n^* and m^* from P and Q, respectively;
        end;
    end;

Formulas

(1) for $m \in Q$

$$f(m) = C_\infty * \{u(m) + d(m)\} + \max\ \{u(m), d(m)\},$$

$$C_\infty \gg 1$$

lies on the longest path before merge, farthest away from *s* or *t*

(2) for $n \in P$, $m \in Q$

$$g(n, m) = C_\infty * h(n, m) - \{\sqrt{u(m) * u(n)} + \sqrt{d(m) * d(n)}\}$$

increase of longest path after merge is minimum, $u(n)+d(n)$ maximized and $u(m)/d(m) = u(n)/d(n)$

where

$$h(n, m) = \max\ \{u(n), u(m)\} + \max\ \{d(n), d(m)\} - \max\ \{u(n) + d(n), u(m) + d(m)\}$$

—the increase of the longest path length passing through n or m, by merging of n and m.

Results

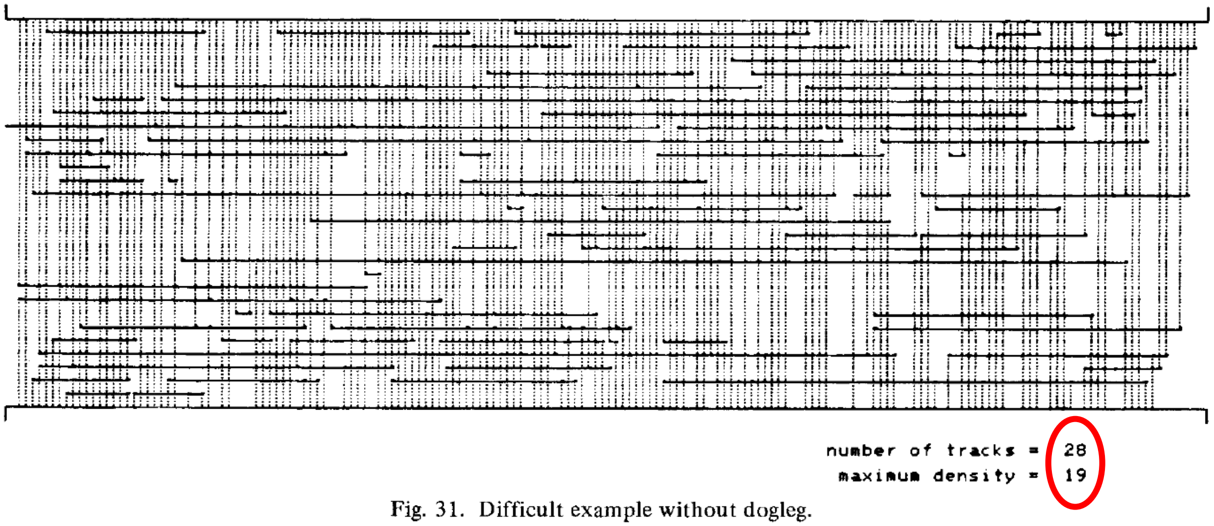


Fig. 31. Difficult example without dogleg.

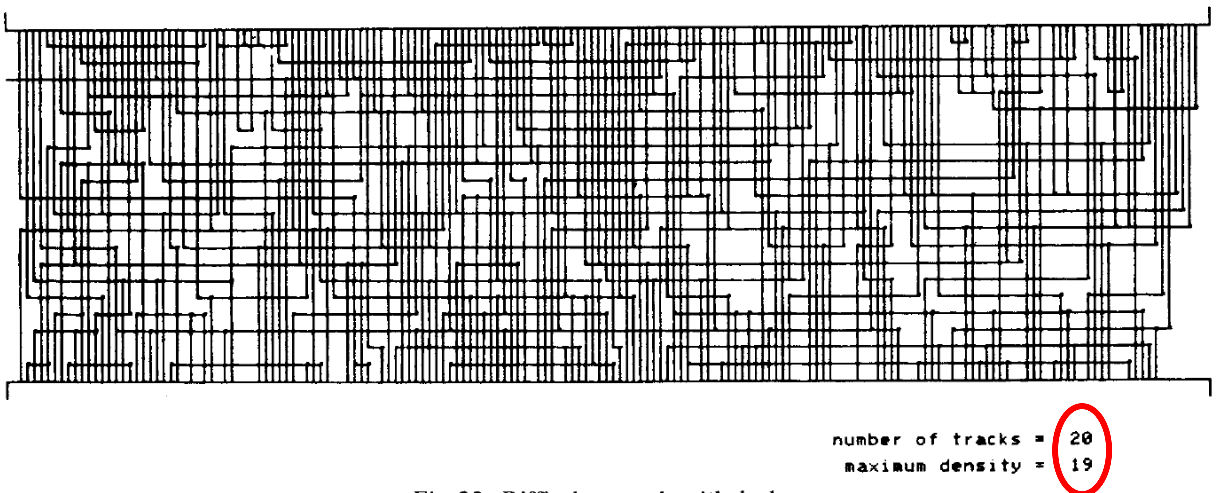
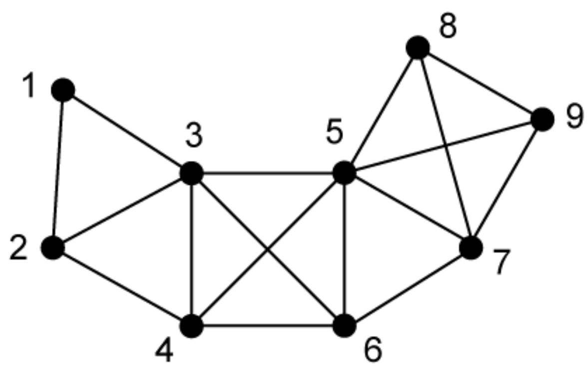


Fig. 32. Difficult example with dogleg.

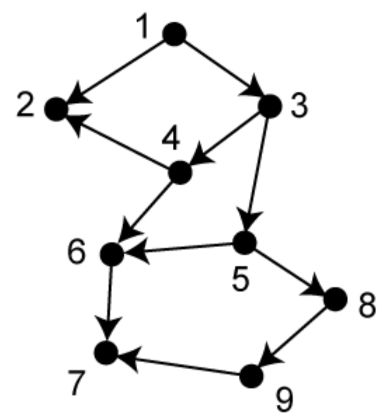
Yoshimura-Kuh Channel Routing

- Perform YK channel routing with K = 100

```
TOP = [1,1,4,2,3,4,3,6,5,8,5,9]
BOT = [2,3,2,0,5,6,4,7,6,9,8,7]
```



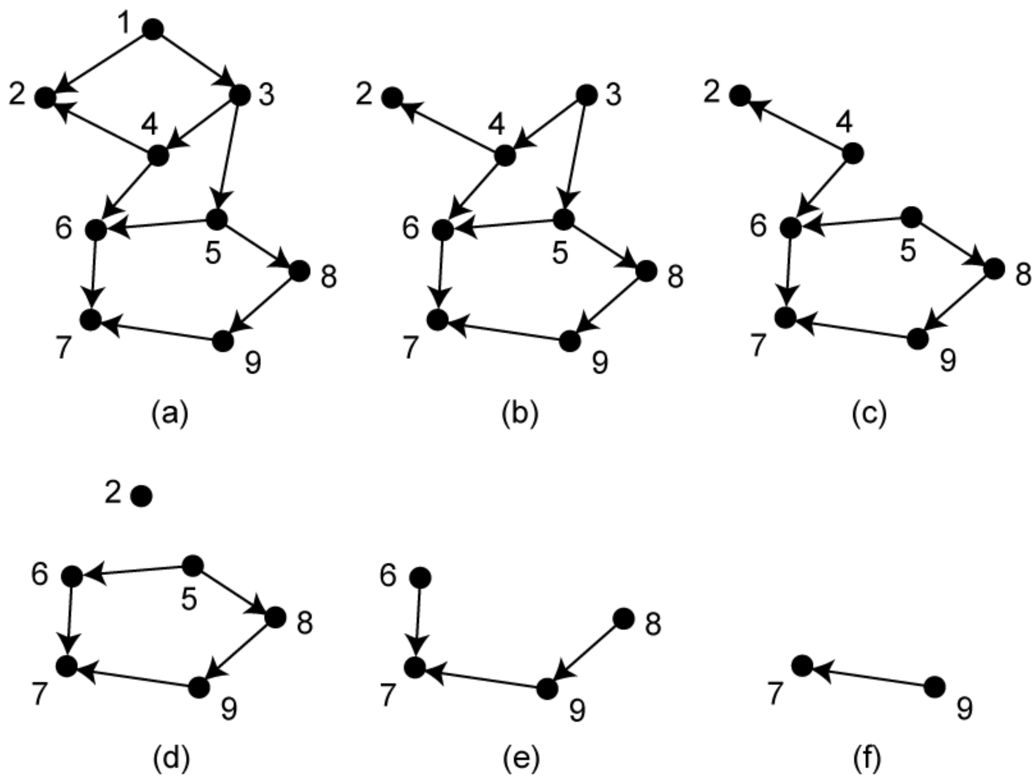
HCG

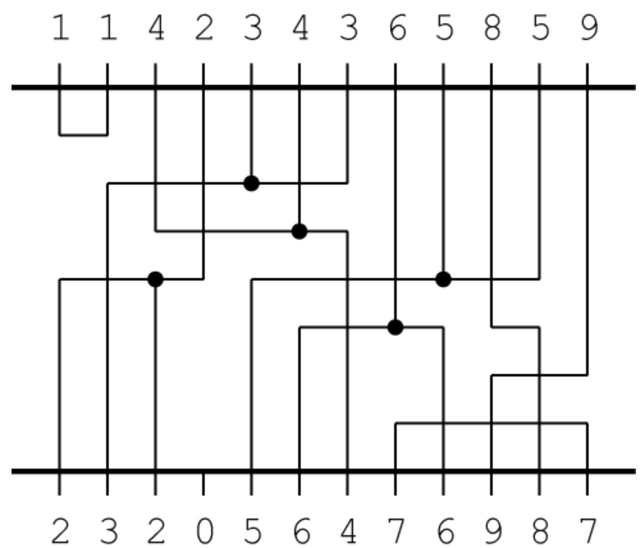


VCG

Constrained Left-Edge Algorithm

- First perform CLE on original problem (for comparison)
  - Assign VCG nodes with no incoming edge first
  - Use tracks top-to-bottom, left-to-right





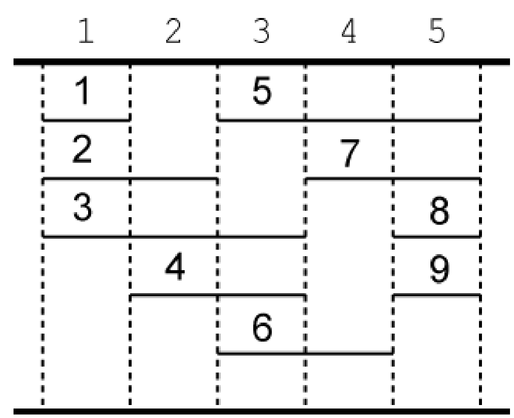
Zone Representation

- Horizontal span of the nets and their zones

TOP = [1,1,4,2,3,4,3,6,5,8,5,9]

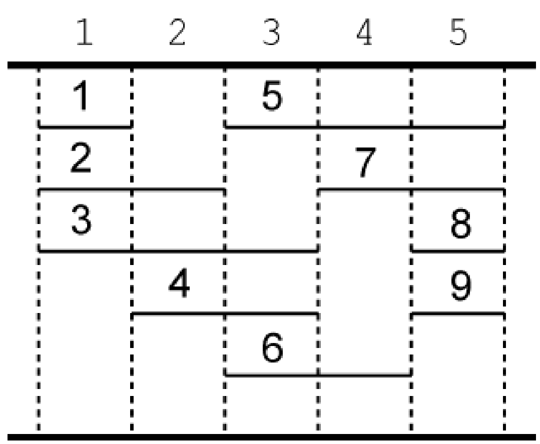
BOT = [2,3,2,0,5,6,4,7,6,9,8,7]

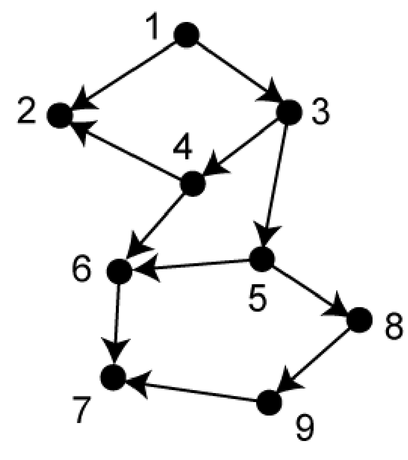
	zone 1		zone 2		zone 3			zone 4		zone 5		
net	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12
1	1	1										
2	2	2	2	2								
3		3	3	3	3	3	3					
4			4	4	4	4	4					
5					5	5	5	5	5	5	5	
6						6	6	6	6			
7								7	7	7	7	7
8										8	8	
9										9	9	9



Net Merging: Zone 1 and 2

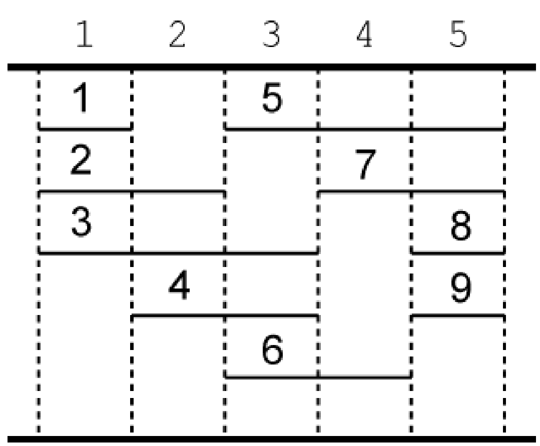
- We compute
  - $L = \{1\}$ and $R = \{4\}$
  - Net 1 and 4 are on the same path in VCG: no merging possible

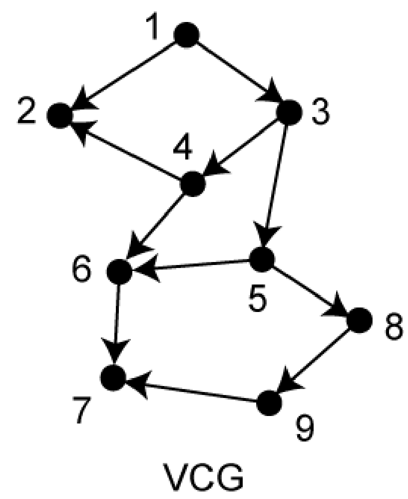




Net Merging: Zone 2 and 3

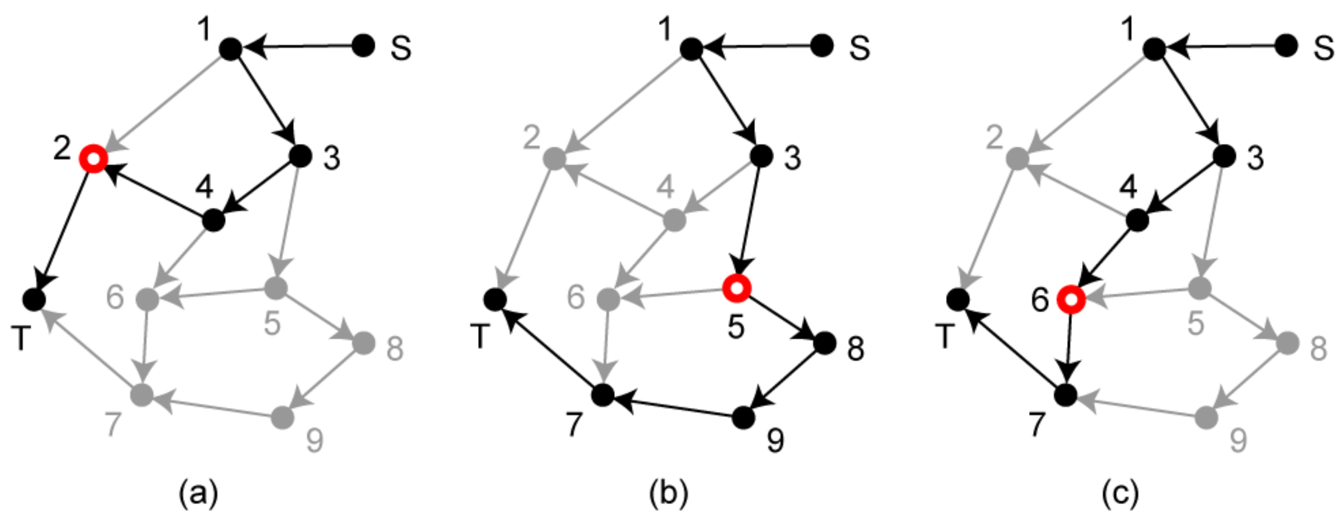
- We compute
  - L = {1,2} and R = {5,6} (= net 1 inherited from last step)
  - Merge-able pairs: (2,5) and (2,6) (= not on the same path in VCG)





Net Merging: Zone 2 and 3 (cont)

- Choose the "best" pair between (2,5) and (2,6)
  - We form P = {5,6} and Q = {2} and choose best from each set
  - We compute
    - $u(2) = 4$, $d(2) = 1$, $u(5) = 3$, $d(5) = 4$, $u(6) = 4$, $d(6) = 2$
  - Only 1 element in Q, so m^* = net 2 trivially



Net Merging: Zone 2 and 3 (cont)

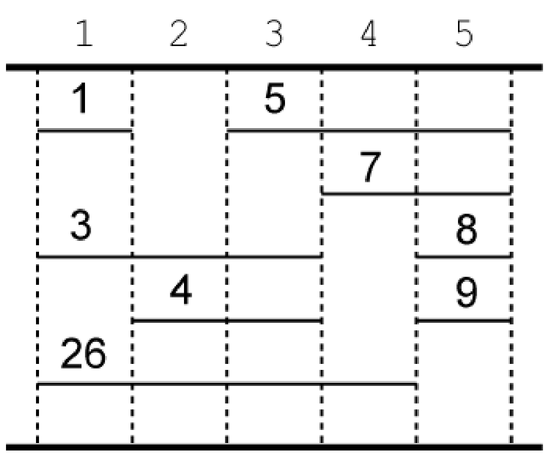
- Now choose "best" from *P*
  - We compute $g(5,2)$ and $g(6,2)$ using $K = 100$

$$
\begin{aligned}
h(5,2) &= \max\{u(5), u(2)\} + \max\{d(5), d(2)\} \\
&\quad - \max\{u(5) + d(5), u(2) + d(2)\} = 1 \\
h(6,2) &= \max\{u(6), u(2)\} + \max\{d(6), d(2)\} \\
&\quad - \max\{u(6) + d(6), u(2) + d(2)\} = 0 \\
g(5,2) &= 100 \cdot h(5,2) - \{\sqrt{u(2) \cdot u(5)} + \sqrt{d(2) \cdot d(5)}\} \\
&= 94.5 \\
g(6,2) &= 100 \cdot h(6,2) - \{\sqrt{u(2) \cdot u(6)} + \sqrt{d(2) \cdot d(6)}\} \\
&= -5.4
\end{aligned}
$$

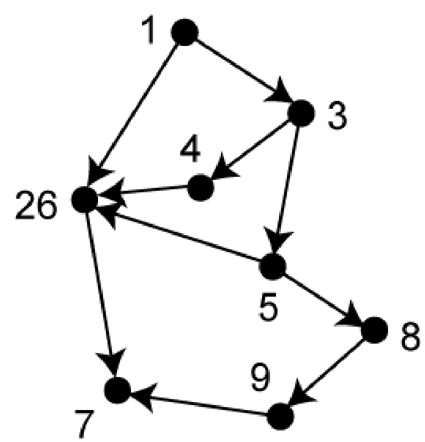
  - Since $g(5,2) > g(6,2)$, we choose $n^* =$ net 6
  - We merge $m^* = 2$ and $n^* = 6$
    - Likely to minimize the increase in the longest path length in VCG

Net Merging: Zone 2 and 3 (cont)

- Merged net 2 and 6
  - We had $P = \{5,6\}$ and $Q = \{2\}$, and need to remove 2 and 6
    - Q is empty, so we are done with zone 2 and 3
  - We had $L = \{1,2\}$ and $R = \{5,6\}$, and need to remove 2 and 6
    - We keep $L = \{1\}$
  - Updated zone representation and VCG



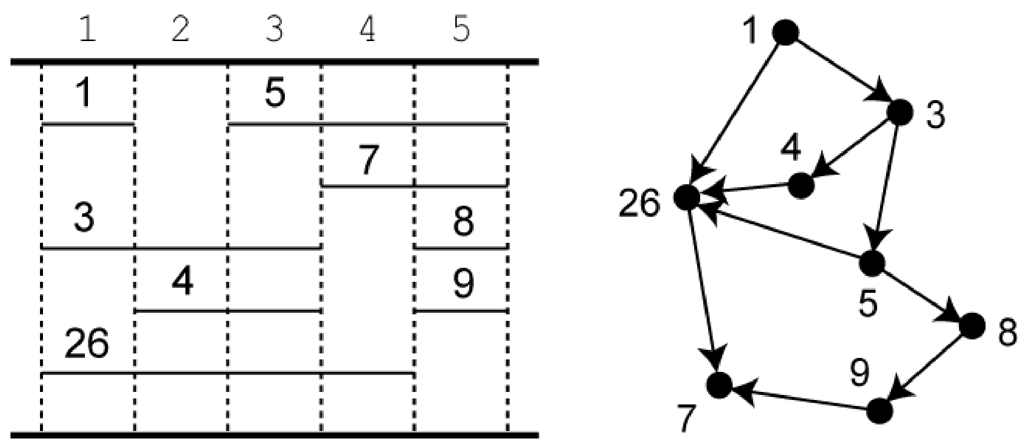
(a)



(b)

Net Merging: Zone 3 and 4

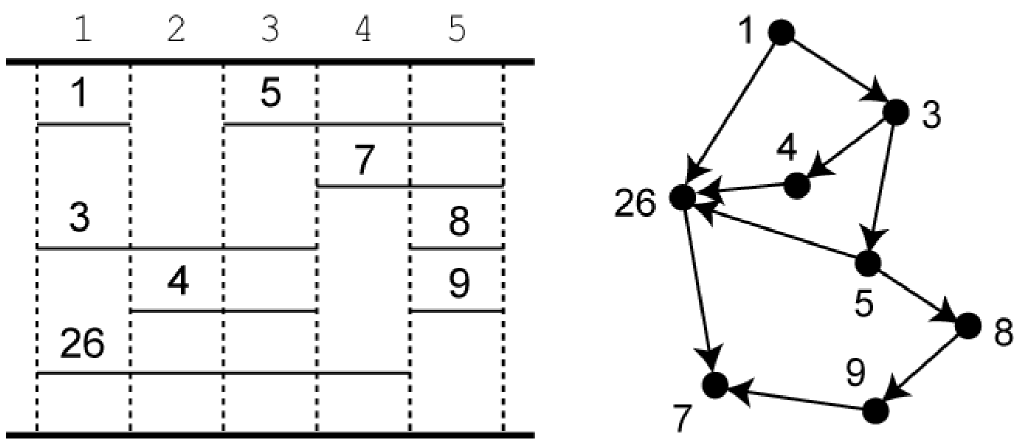
- We compute
  - L = {1,3,4} and R = {7} (= net 1 inherited from last step)
  - All nets in L and R are on the same path in VCG
    - no merging possible



(a) (b)

Net Merging: Zone 4 and 5

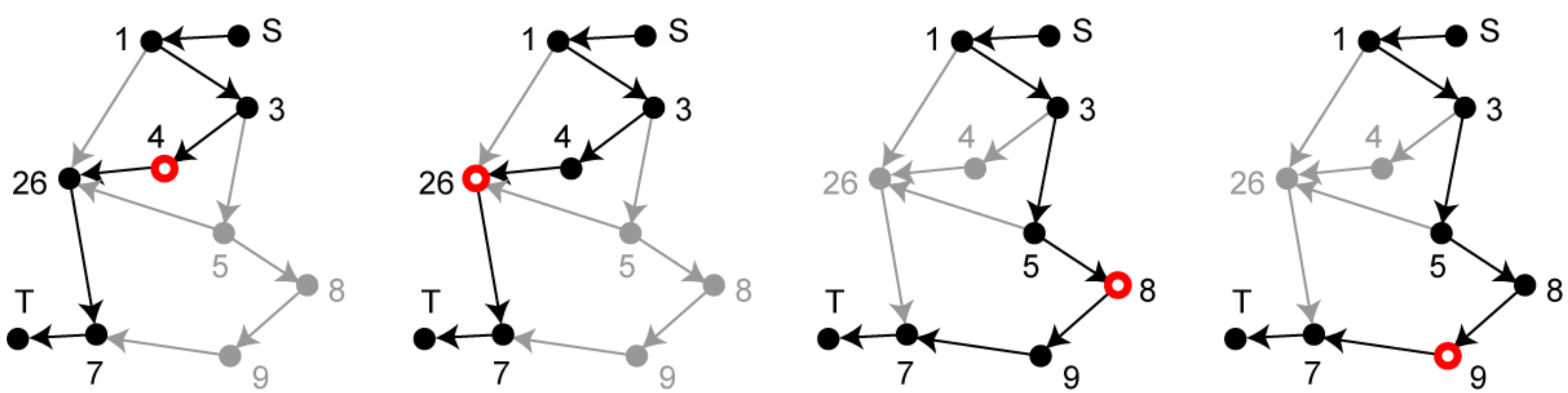
- We compute
  - $L = \{1,3,4,26\}$ and $R = \{8,9\}$
  - Merge-able pairs: (4,8), (4,9), (26,8), (26,9)



(a) (b)

Net Merging: Zone 4 and 5 (cont)

- Choose m^* from Q
  - We form P = {4,26} and Q = {8,9}
  - We compute
    - $u(4) = 3$, $d(4) = 3$, $u(26) = 4$, $d(26) = 2$, $u(8) = 4$, $d(8) = 3$, $u(9) = 5$, $d(9) = 2$



(a) (b) (c) (d)

Net Merging: Zone 4 and 5 (cont)

- Choose m^* from Q (cont)
  - We find m^* from Q that maximizes
    - $f(8) = 100 \cdot \{u(8) + d(8)\} + \max\{u(8), d(8)\} = 704$
    - $f(9) = 100 \cdot \{u(9) + d(9)\} + \max\{u(9), d(9)\} = 705$
  - So, $m^* = 9$

Net Merging: Zone 4 and 5 (cont)

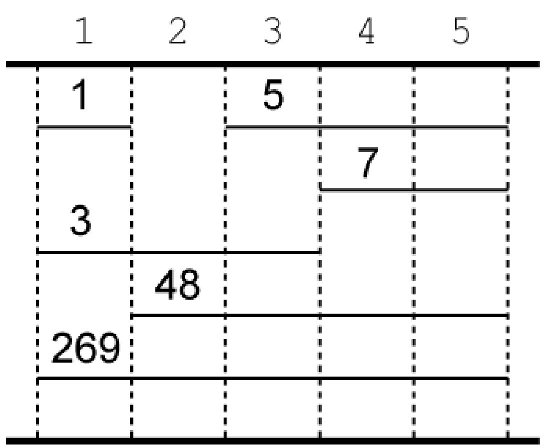
- Choose n^* from P
  - We compute $g(4,9)$ and $g(26,9)$ using $K = 100$

$$\begin{aligned}
h(4,9) &= \max\{u(4), u(9)\} + \max\{d(4), d(9)\} \\
&\quad - \max\{u(4) + d(4), u(9) + d(9)\} = 1 \\
h(26,9) &= \max\{u(26), u(9)\} + \max\{d(26), d(9)\} \\
&\quad - \max\{u(26) + d(26), u(9) + d(9)\} = 0 \\
g(4,9) &= 100 \cdot h(4,9) - \{\sqrt{u(9) \cdot u(4)} + \sqrt{d(9) \cdot d(4)}\} \\
&= 93.7 \\
g(26,9) &= 100 \cdot h(26,9) - \{\sqrt{u(9) \cdot u(26)} + \sqrt{d(9) \cdot d(26)}\} \\
&= -6.5
\end{aligned}$$

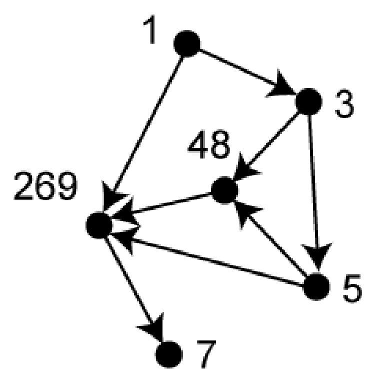
  - Since $g(4,9) > g(26,9)$, we get n^* = net 26
  - We merge $m^* = 9$ and $n^* = 26$

Net Merging: Zone 4 and 5 (cont)

- Merged net 26 and 9
  - We had P = {4,26} and Q = {8,9}, and need to remove 26 and 9
    - Q is not empty, so we repeat the whole process
  - Updated P = {4} and Q = {8}
    - Trivial to see that m^* = 8 and n^* = 4, so we merge 8 and 4
  - Updated zone representation and VCG



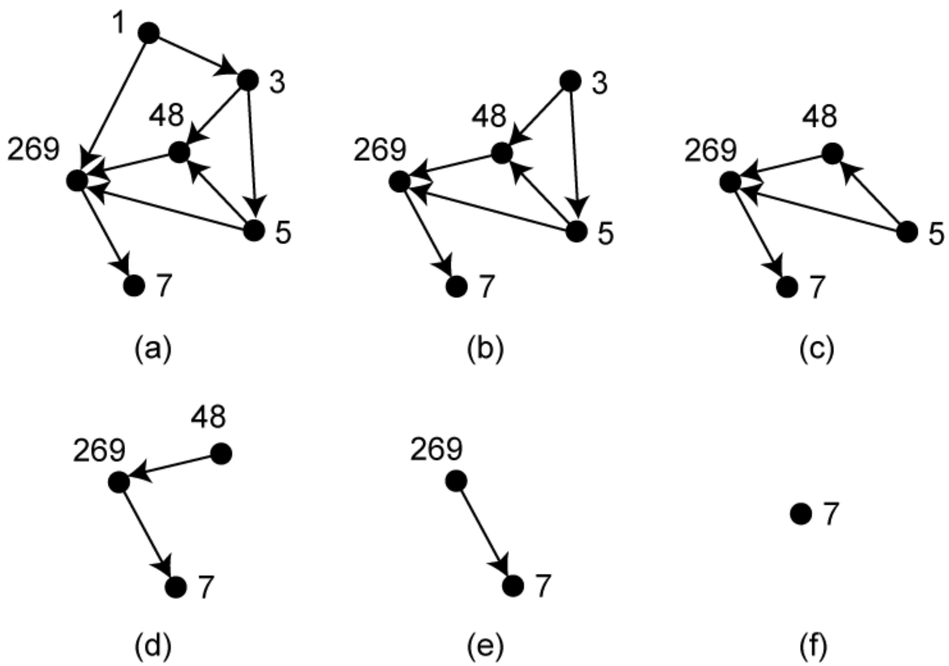
(a)

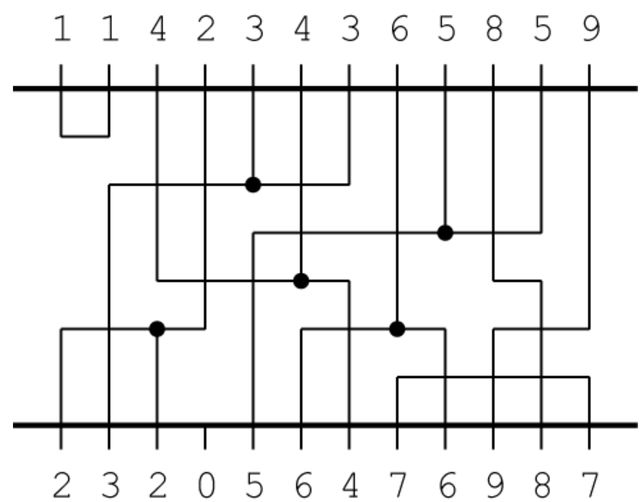


(b)

Routing with Merged Nets

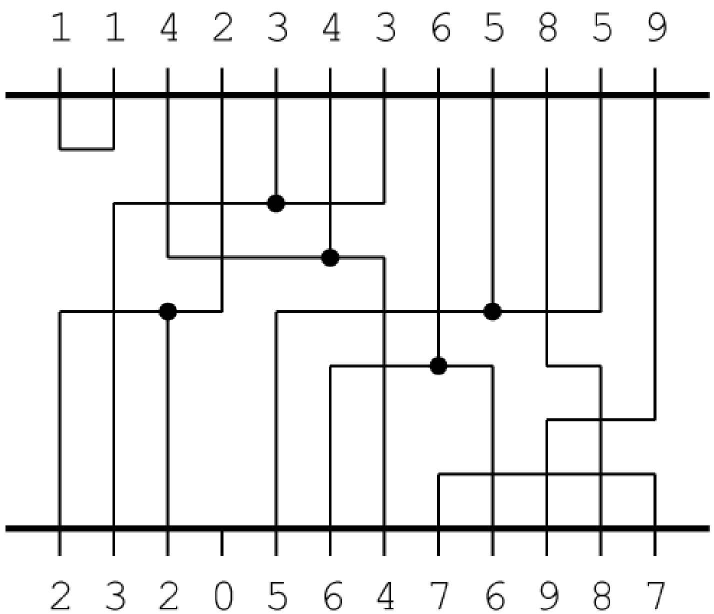
- Perform CLE on merged netlist
  - Use tracks top-to-bottom, left-to-right



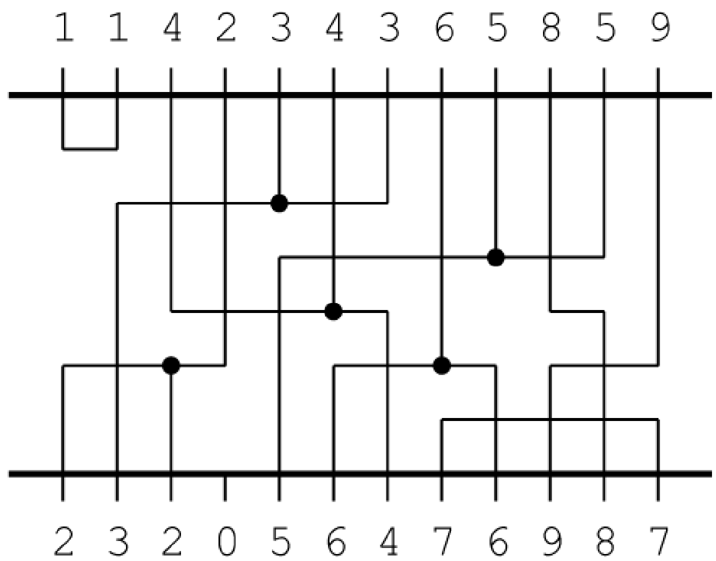


Comparison

- Net merging helped
  - Reduce channel height by 1



without net merging



with net merging