King Fahd University of Petroleum & Minerals
College of Computer Sciences & Engineering
Department of Computer Engineering

Grid Routing

Sadiq M. Sait and Habib Youssef

December 1995

INTRODUCTION

- The phase following cell placement.
- Routing is accomplished using computer programs ($routers$).
- Consists of precisely defining paths that carry electrical signals run.
- Takes up almost 30% of design time and a large percentage of layout area.
- Routing algorithms were first applied to design of PCBs.
- The main application of automatic routers has been in the automated design of VLSI circuits.

Maze Routers

- Consider the layout as a $maze$.
- Finding a path is similar to finding a path in the maze.
- The layout floor is assumed to be made up of a rectangular array of grid cells.
- Functional cells fill up some slots in the grid and constitute the obstacles of the maze.
- The most popular algorithm is the Lee algorithm.
- Running time of maze algorithms is large and memory requirement is high.
- Maze routers based on the Lee algorithm connect a pair of points at a time.
- Multi-pin nets can be connected in a spanning-tree like fashion.

Problem Definition

Given:

- a set of *cells* with ports (inputs, outputs, clock, power and ground pins) on the boundaries,
- a set of *signal nets*,
- locations of cells on the layout floor, and
- geometrical constraints and number of routing layers.

Goal: find suitable paths on the available layout space, that is those paths that minimize the given objective functions, subject to constraints.

Constraints may be:

- imposed by the designer,
- the implementation process, or
- layout strategy and design style.

Examples of constraints are:

- minimum separation between adjacent wires,
- minimum width of routing wires,
- number of available routing layers,
- timing constraints, etc.

Examples of objective functions include:

- reduction of wire length, and
- avoidance of timing problems.

Routing Material

- The connections are made by first uniformly depositing metal on a carrier surface.

- Then unwanted metal is etched away.

- In PCBs this surface is usually fiberglass. While in VLSI it is silicon.

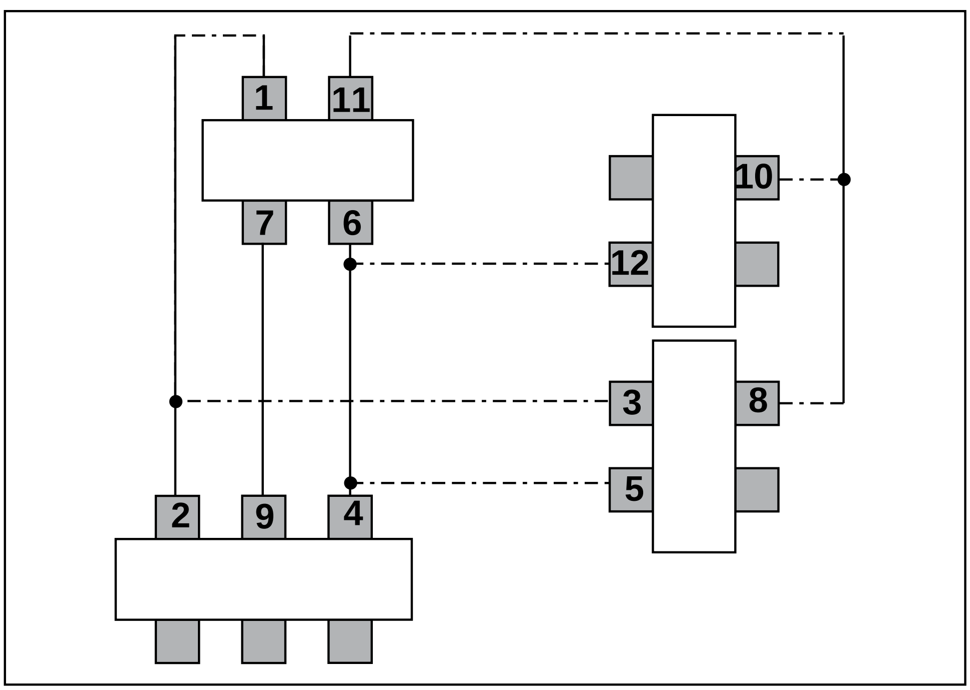
- In VLSI design, polysilicon is also used to carry signals.

- In 2-layer VLSI routing, the 2 layers are separated by an oxide insulating layer.

- Holes in this insulating layer called $contact\text{-}cuts$ (or $vias$) connect conductors between two layers.

- Certain VLSI technologies allow three layers for routing. (2 layers in metal and the 3rd layer in polysilicon)

Illustration of general routing



N_1 ={1,2,3}

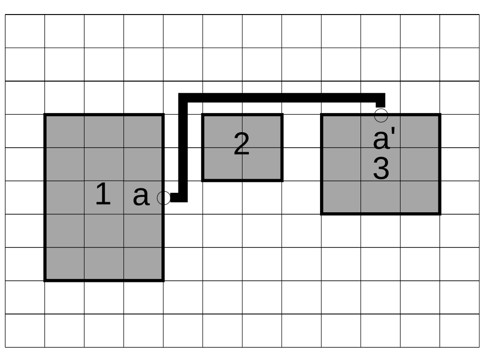
N_2= {4,5,6,12}

N_3= {7,9}

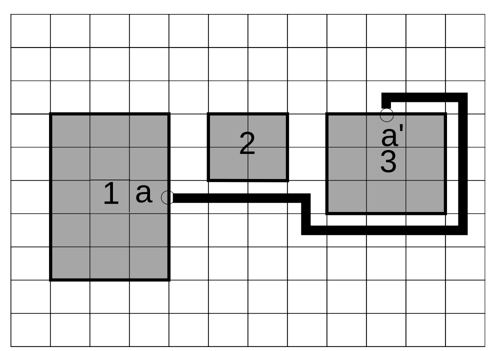
N_4 = {8,10,11}

Cost functions and Constraints

- Reduction of wiring area.
- Improvement in performance.
- Improve yield (by reducing cuts).
- Two possible paths connecting a pair of points are shown below. (a) The shortest path. (b) A longer path with more bends.



(a)



(b)

Geometrical Constraints

- Minimal geometries must be maintained, (minimum width and spacing dictated by the technological process).

- Must be able to consider all geometrical constraints abolishing the need for DRC.

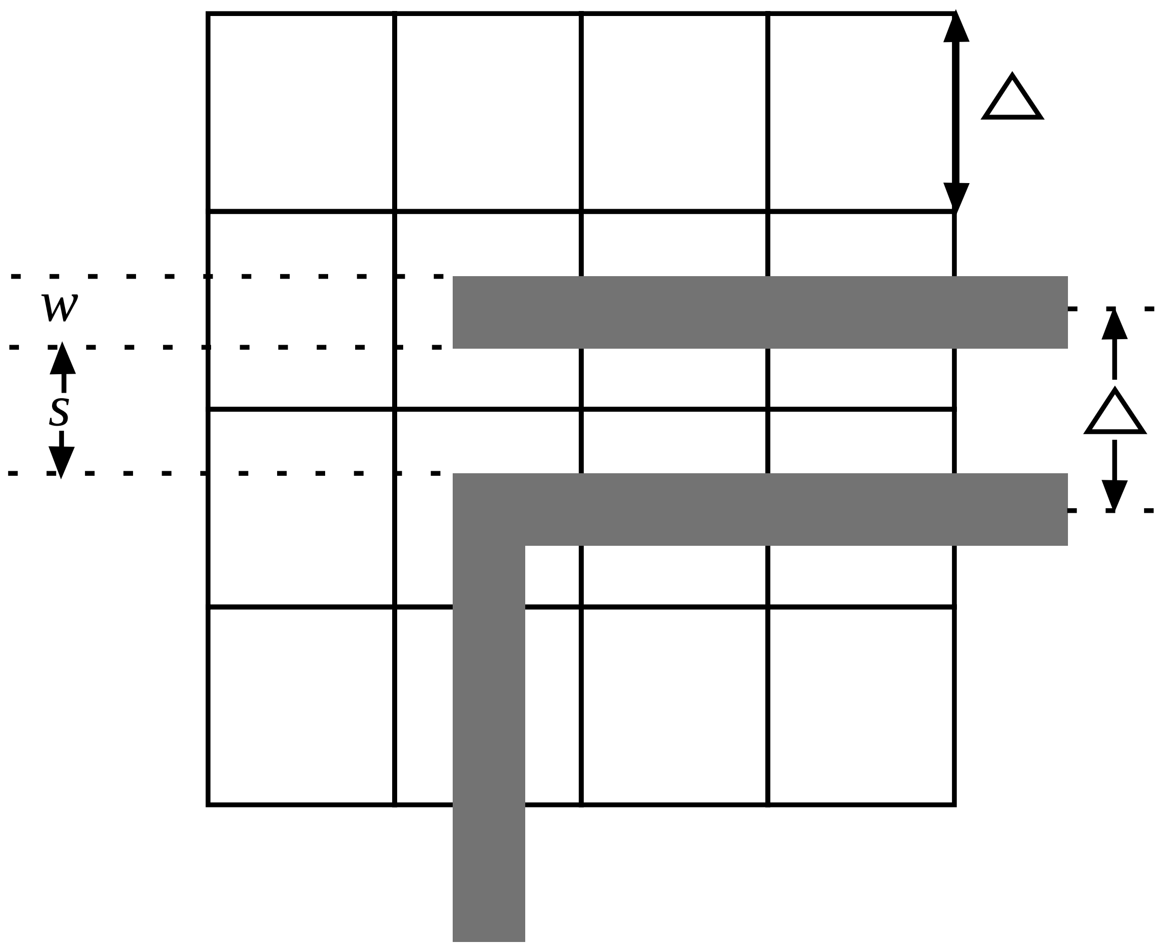
- For routing purposes, only those design rules must be considered which define geometries of wires and contact holes.

- Commonly, this is achieved by using a proper equidistant grid.

- Wires are represented by lines and restricted to grid line positions.

- Wire widths and separation between wires is constant for all nets and design rules are avoided.

Illustration of grid cell size



Maze Routing Algorithms

- The entire routing surface is represented as a rectangular array of grid cells.
- All ports, wires, and edges of bounding boxes that enclose the cells are aligned on the grid.
- Segments on which wires run are also aligned with the grid lines.
- The size of grid cells is defined such that wires belonging to other nets can be routed without violating the width/spacing rules of wires.
- Two points are connected by finding a sequence of adjacent cells from one point to the other.
- Maze routers connect a single pair of points at a time.

Lee Algorithm

- Most widely known maze routing method for finding a path in a maze.

- An excellent characteristic is that if a path exists then it is surely found.

- In addition it is guaranteed to be the shortest available one.

- The algorithm can be divided into three phases.

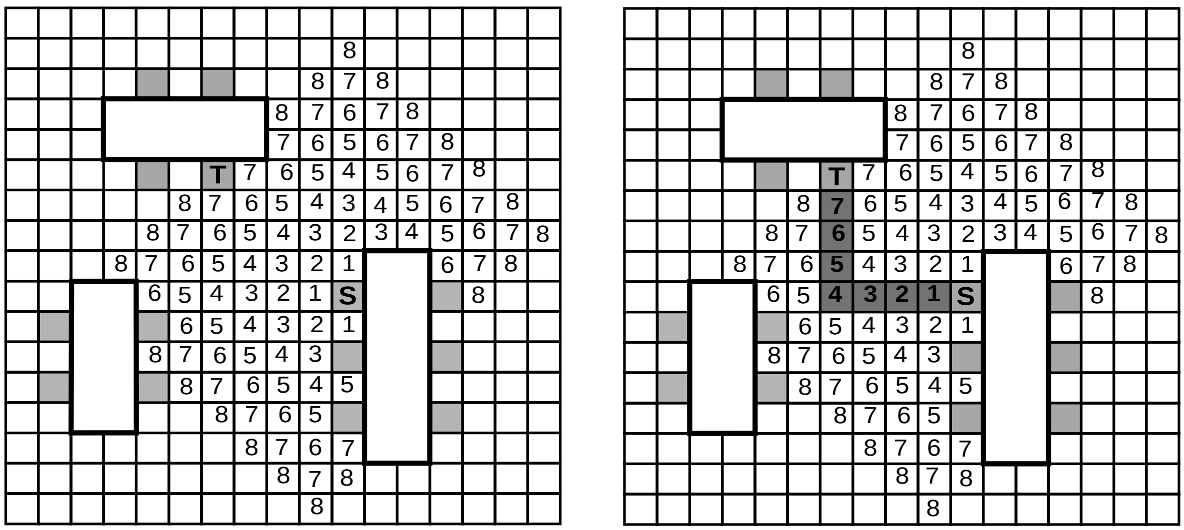
- The first phase consists of labeling the grid, and is called the *filling* or *wave propagation* phase.

- It is analogous to dropping a pebble in a still pond and causing waves to ripple outward.

- The second phase of the algorithm is called the *retrace* phase.

- The final phase is called *label clearance*. In this phase all labeled cells except those used for the path just found are cleared for subsequent interconnections.

Filling in Lee Algorithm



(a) (b)

- The filling phase begins by entering a '1' in all empty cells adjacent to the source cell S.
- Next, 2s are entered in all empty cells adjacent to those containing 1s. Then, 3s are entered adjacent to 2s and so on.
- This process continues and is terminated when one of the three conditions occurs.

Filling & Retrace in Lee Algorithm

Filling continues until:

a. the grid cell T is reached; or

b. T is not reached and at step i there are no empty grid cells adjacent to cells labeled $i-1$; or

c. T is not reached and i equals M, where M is the upper bound on a path length.

The Retrace procedure is the reverse of filling. The actual shortest path is found as follows:

- If grid cell T was reached in step i, then there exists at least one grid cell adjacent to it which contains $i-1$.
- Likewise, a grid cell containing $i-2$ will be adjacent to one containing label $i-1$ and so on.
- By tracing the numbered cells in descending order from T to S, the desired shortest path is found.
- The cells of the retraced path for the filled grid of Figure (a) are shaded in Figure (b).
- Once the desired path is found, the cells used for the route connecting S and T are regarded as obstacles for subsequent interconnections.

Time/Space Complexity of Lee Algorithm

- The processing time for filling is proportional to L^2, (L is the length of the path)

- The processing time for retrace is proportional to L

- Therefore, the algorithm has a time complexity of O(L^2) for each path

- In addition, for an $N \times N$ grid plane, the algorithm requires O(N^2) memory

- Also, some amount of storage is required to store positions of cells on the wavefront

- The worst case running time is also of O(N^2)

- Extensions to reduce running time and storage have been proposed

Coding Schemes to Reduce Memory

- A non-trivial storage problem is that a unit of memory space is needed for every grid cell.

- In a filled grid we observe that for each cell labeled k, all adjacent cells are labeled either $k-1$ or $k+1$.

- Therefore, during retrace, it is sufficient if we can distinguish the *predecessor* cells from the *successor* cells.

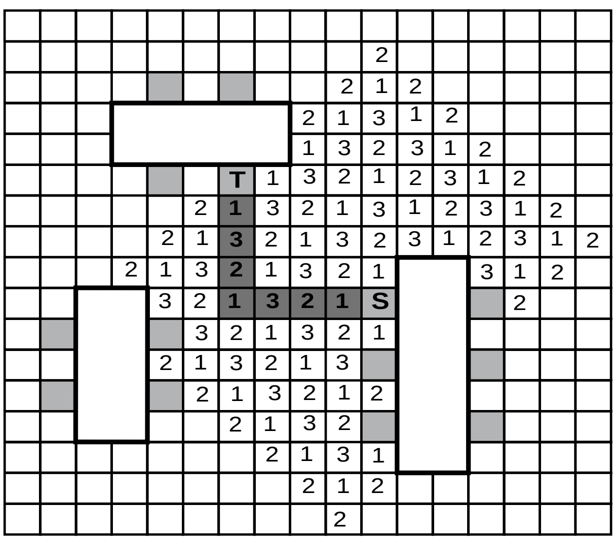
- Labeling schemes based on this idea are widely used. Two are listed below.

Coding Schemes to Reduce Memory ..

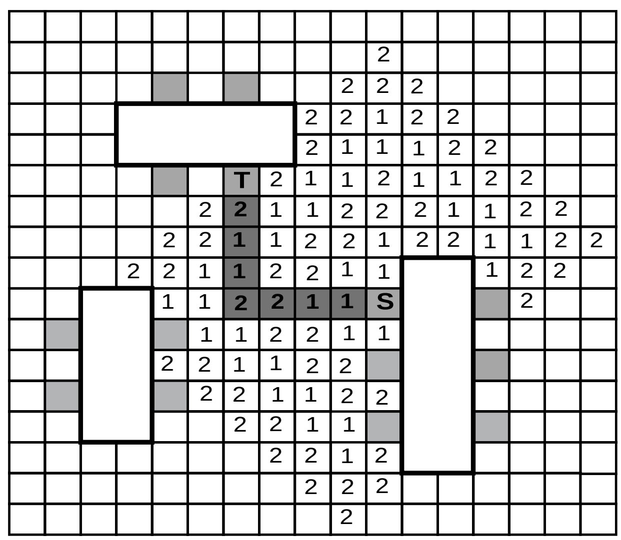
- In the first scheme
  - Labeling sequence is 1,2,3, 1,2,3 . . .
  - Only three bits per memory cell are required since a grid cell may be in one of the 5 states.
- The second scheme proposed by Akers
  - Labeling sequence is 1,1, 2,2, 1,1, 2,2 . . . .
  - This scheme is most economical, since each cell will be in one of the four states: *empty*, *blocked*, labeled with 1, or labeled with 2.
  - Independent of the grid size, two bits per memory cell are sufficient.

Filling Sequences That Reduce Memory Requirement

(a) Sequence 1,2,3, 1,2,3 . . . . (b) Sequence 1,1, 2,2, 1,1, 2,2 . . . .



(a)



(b)

Reducing Running Time

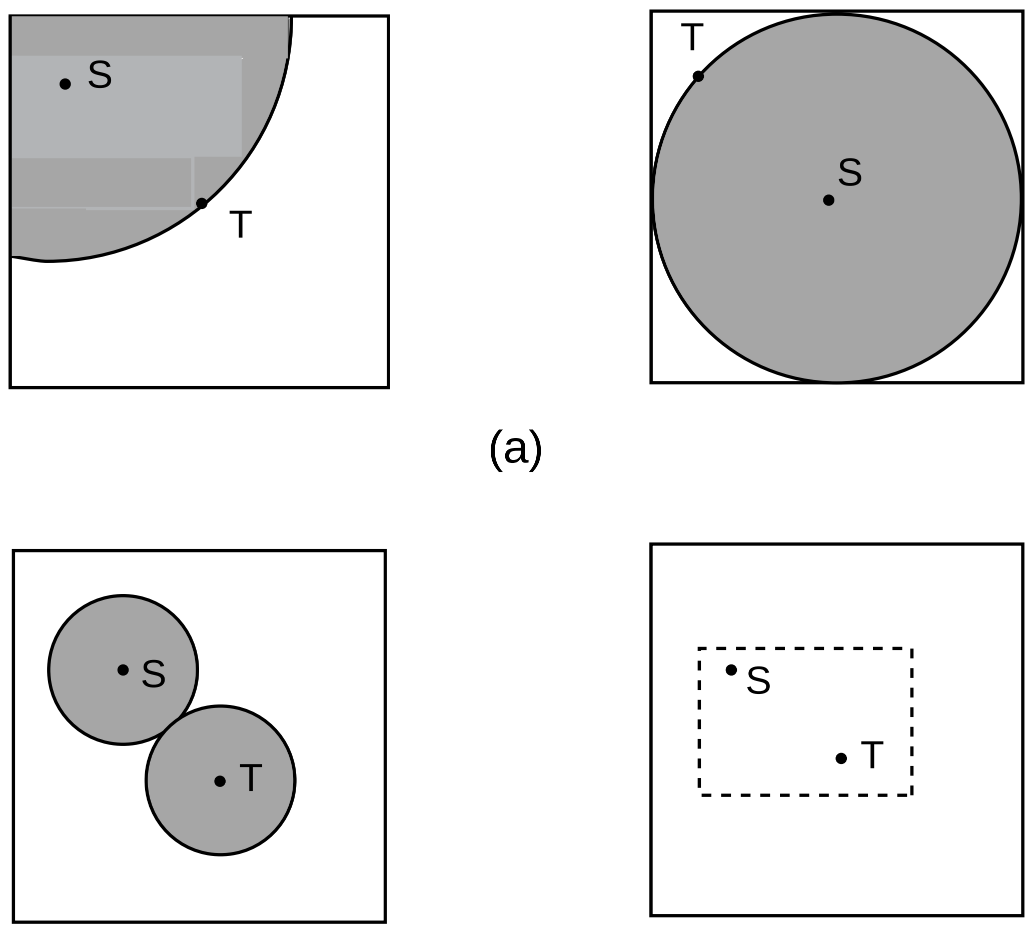
- Running time is proportional to the number of cells searched in the filling phase.

- The following speed-up techniques are used.

  (a) Starting point selection.

  (b) Double fan-out.

  (c) Framing.



(a)

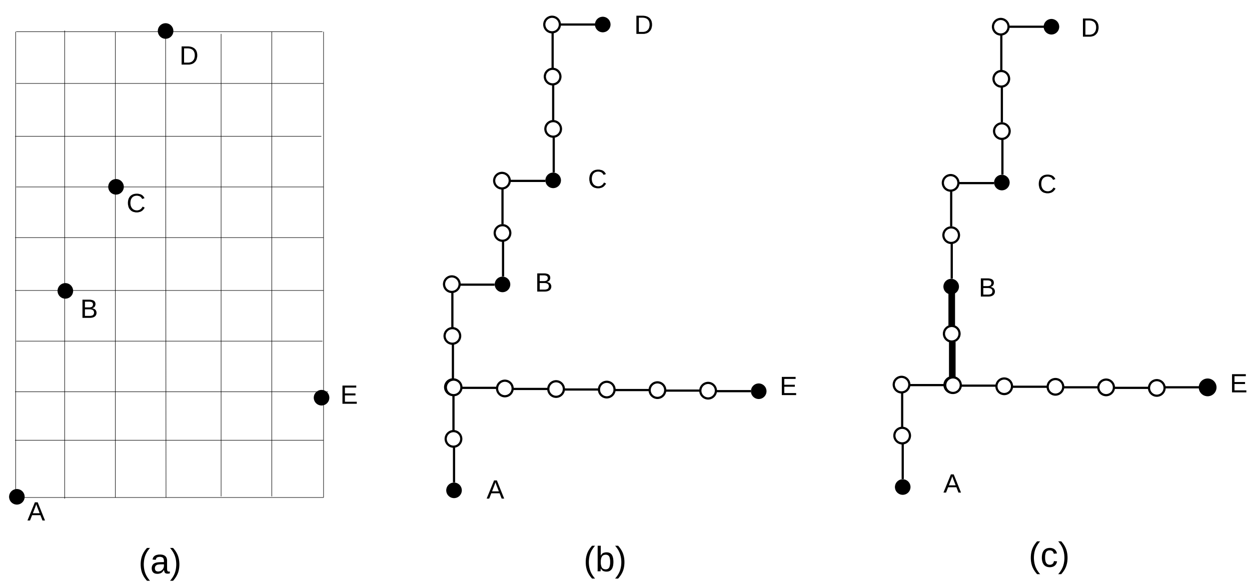
(b) (c)

Connecting Multi-point Nets

- Lee algorithm as seen above connects two pins
- A multi-pin net consists of three or more pins to be connected.
- The optimal connection of these pins resulting in the least wirelength is termed as the *Steiner tree problem.*
- This problem has been proven to be NP-hard
- A sub-optimal solution to connect a multi-point net can be obtained using Lee algorithm

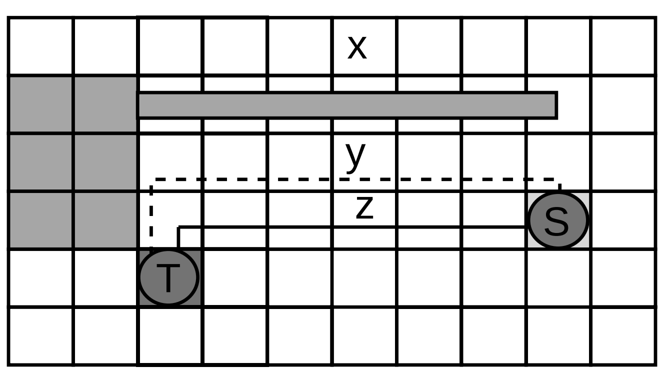
Example of Routing a Multipoint Net

(a) Five points of a net. (b) Interconnection tree found by repeated application of modified Lee algorithm. (c) A shorter interconnection found by deleting an edge and re-routing.



Finding More Desirable Paths

- Often practical situations require a more desirable path, not necessarily the shortest
- An example is of finding a path that will cause least amount of difficulty for subsequent paths
- The filling phase of the Lee Algorithm can be modified to accommodate such constraints.
- The requirement of any filling phase is that the *desired* path be unambiguously traced back.
- Akers observed that a path running along obstructions would leave more room for subsequent ones
- Suppose that a net x has been routed as shown below. The standard Lee Algorithm will result in the shortest path z. while the longer path y could be more preferable.



Finding More Desirable Paths

- If the objective is to accomplish the desired path such as y, then the required path selection can be accomplished by preparing a weighted array as shown in Figure below.

- The desired path may be generated by routing a net so as to minimize the total weight of used cells.

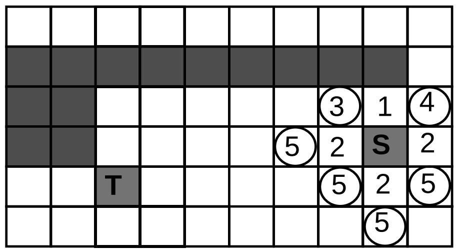
- For path y this weight is 13, and for path z it is 15.

- The wave propagation phase in Lee Algorithm is modified to minimize the total weighted sum of grid points.

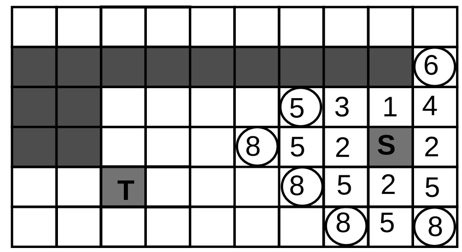
- The modified procedure is shown in the following slide with an example.

2	2	2	2	2	2	2	2	2	3
									2
		1	2	2	2	2	2	1	3
		1	3	3	3	3	2	**S**	2
2	1	**T**	2	3	3	3	3	2	3
3	3	2	3	3	3	3	3	3	3

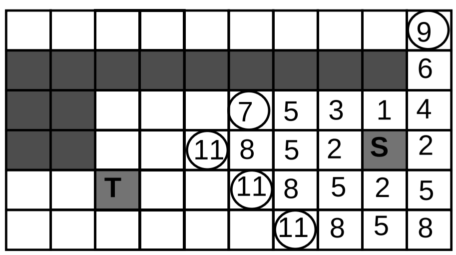
Example of Finding Desirable Paths



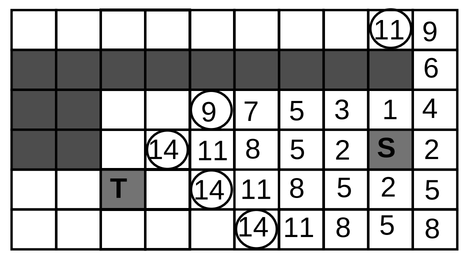
(a)



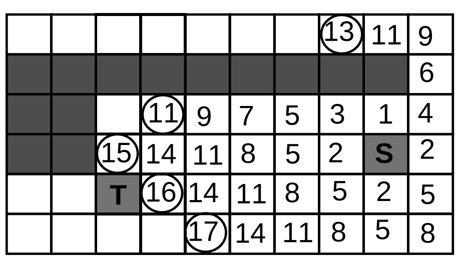
(b)



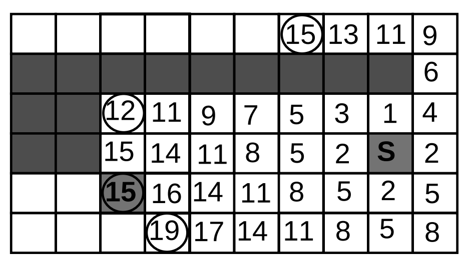
(c)



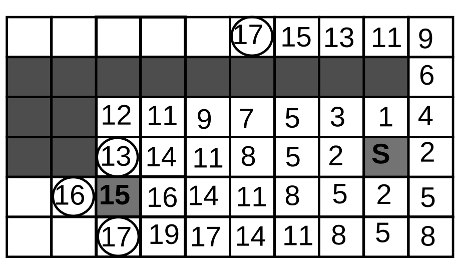
(d)



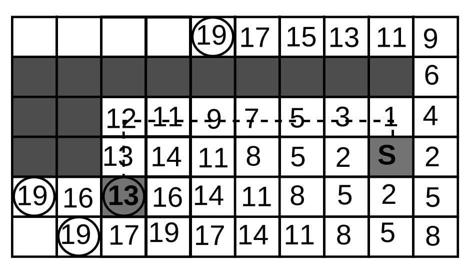
(e)



(f)



(g)



(h)

Further Speed Improvements

- The $filling$ phase of Lee Algorithm is similar to the *breadth first search*

- It can also be thought of as construction of a tree with each node having at most 4 children.

- the running time for a particular instance of source-target pairs is proportional to the number of cells being searched until the target is reached.

- One idea behind speed-up techniques is to advance the wavefront with a higher priority towards the target direction.

- Two techniques are presented in the following slides; Hadlock's algorithm and Soukup's algorithm.

Detour Numbers

- Hadlock's algorithm is a shortest path algorithm with a new method for cell labeling called *detour numbers*

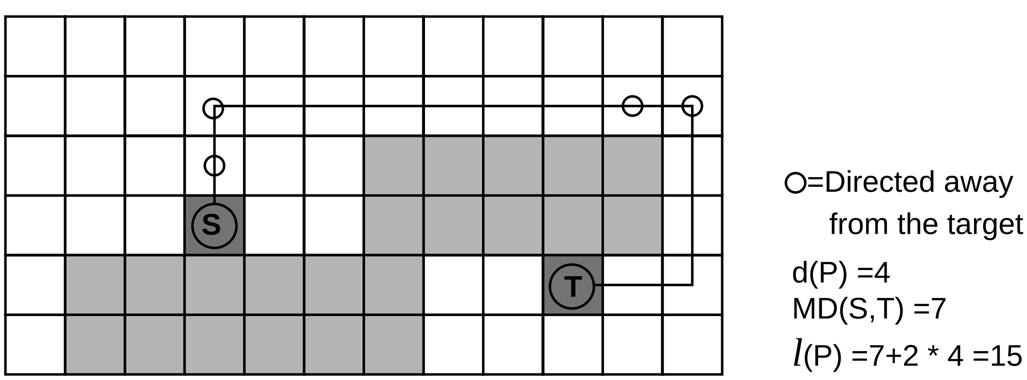
- It is a goal directed search method.

- The detour number $d(P)$ of a path P connecting two cells S and T is defined as the number of grid cells directed away from its target T.

- If $MD(S,T)$ is the Manhattan distance between S and T, then it can be proved that the length $l(P)$ of a path P is given by

$$l(P) = MD(S,T) + 2 \times d(P). \tag{1}$$

- An example of path length and detour numbers is shown in Figure below.



Hadlock's Algorithm

- Figure in previous slide illustrates how path length is represented by the detour number.

- Note that in Eqn (1) $MD(S,T)$ is fixed, independent of the path connecting S and T.

- Based on this idea, the filling phase of the Lee Algorithm is modified as follows:

  a. Instead of filling a cell with a number equal to the distance from the source, the detour numbers with respect to a specified target are entered.
  b. Cells with smaller detour numbers are expanded with higher priority.

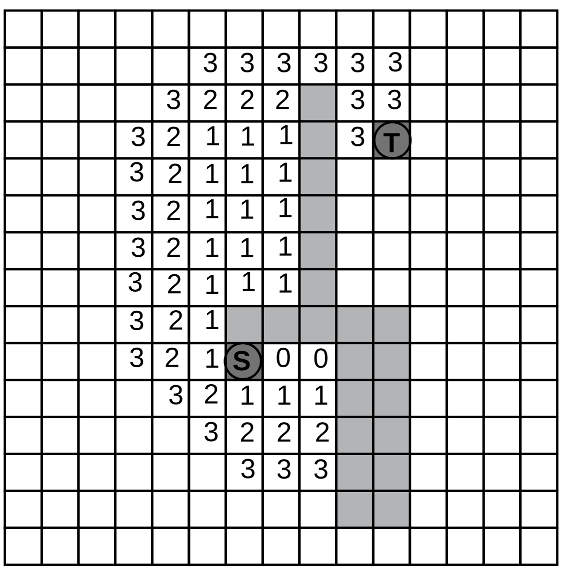
- Figure (a) shows the filling of a grid. Observe that for any cell filled with i, if the adjacent cell is towards the target, then it is filled with the same number, and if it is away from the target then it is filled with $i+1$.

- Path retracing is slightly different from the standard Lee algorithm

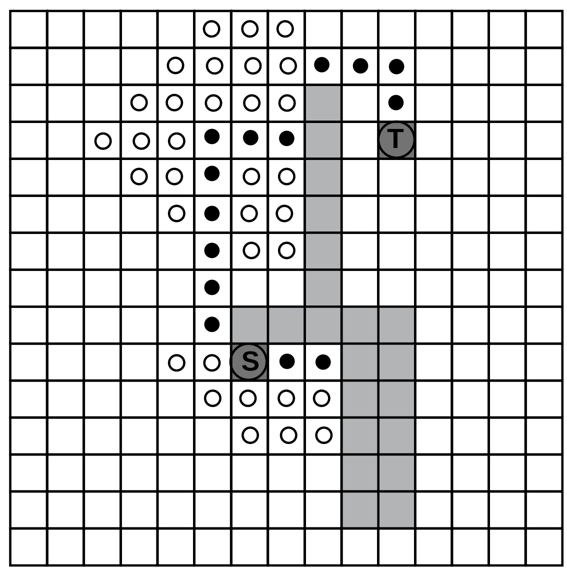
- The number of grid units filled is considerably smaller than in Lee algorithm.

- Therefore speed improvement is remarkable.

Filling in Hadlock's and Soukup's Algorithms



(a)



(b)

Soukup's Algorithm

- The previous algorithm performed filling in a breadth first manner.
- Soukup suggested adding $depth$ to the search.
- In Soukup's algorithm a line segment from the source is initially extended toward target.
- The cells on this line segment are searched first. The line segment is extended without changing direction unless it is necessary.
- When the line hits an obstacle, Lee algorithm is applied to search around the obstacle.
- During the search, once a cell in the direction of the target is found, another line segment starting from there is extended toward target.
- The darkened circles in the figure indicate the cells directed towards the target.
- Soukup's algorithm finds a path if one exists, but does not guarantee that it is the shortest.
- Its disadvantage is that it generates sub-optimal paths (both in terms of length and # of bends).
- However, it is extremely fast, especially when the routing space is not congested.
- It is claimed that it is 10-50 times faster than the Lee Algorithm on typical two-layer routing problems.

Line Search Algorithms

- Line search algorithms overcome the drawback of Lee algorithm.

- The idea is to draw lines passing through S (source) and T (target). The two lines when intersect give a Manhattan path between S and T.

- Line search algorithms perform a depth-first search.

- Because of their depth-first nature, line search algorithms do *not* guarantee finding the shortest path, and may need several backtrackings.

- Produce completion rates similar to Lee algorithm, with the difference that both memory requirements and execution times are considerably reduced.

- This is because the entire routing space is *not* stored as a matrix.

- The routing space and paths are represented by a set of line segments.

- Line search algorithms were first proposed by Mikami-Tabuchi and Hightower.

Mikami-Tabuchi's Algorithm

- Let S and T be a pair of terminals of a net located on some intersection of an $imaginary$ grid.

- The first step is to generate four lines (two horizontal and two vertical) passing through S and T.

- These lines are extended until they hit obstructions (a placed cell for example) or the boundary of the layout.

- If a line generated from S intersects a line generated from T then a connecting path without any bend or with one bend has been found.

- If the four generated lines do not intersect, then they are identified as *trial lines* of level $zero$ and stored in temporary storage.

- Then at each iteration i the following operations are done.

(1). Trial lines of level i are picked and along each of its grid points ($base\text{-}points$) and traced.

Starting from these $base\text{-}points$ new *trial lines* are generated perpendicular to trial line i.

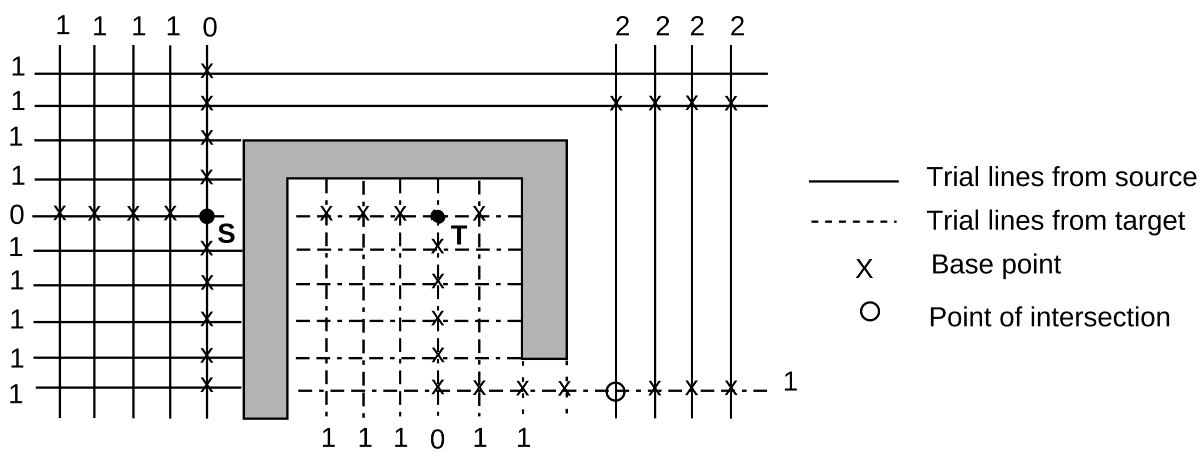
Let the generated line segments be identified as *trial lines* of level $i+1$.

(2). If trial line of level $(i+1)$ intersects a trial line (of any level) originated from the other terminal point, then the required path is found by backtracking from the point of intersection to both points S and T.

Otherwise all trial lines of level $(i+1)$ are added to the temporary storage and the procedure is repeated from Step 1.

- The above algorithm guarantees to find a path if one exists.

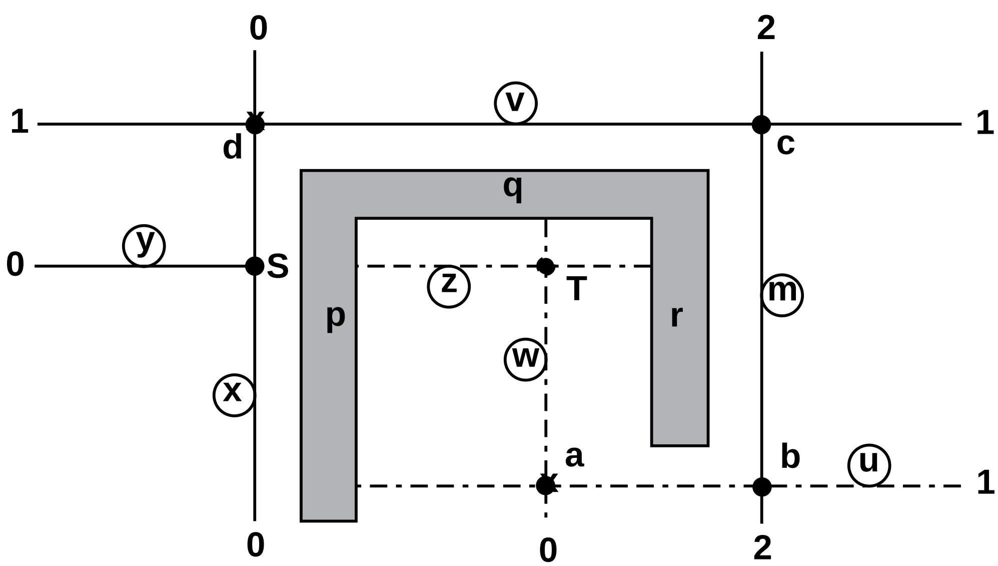
Mikami-Tabuchi's Algorithm ...



- In this example the trial line of level 1 originating from T intersects a trial line of level 2 generated from S.

Hightower's Algorithm

- It is similar to the Mikami-Tabuchi's algorithm.
- The difference is that instead of generating all line segments perpendicular to a trial line, Hightower algorithm considers only those lines that are extendable beyond the obstacle which blocked the preceding trial lines.
- Example.



- The shaded regions *p*, *q*, and *r* constitute obstacles around which the path is to be found. The procedure begins by constructing horizontal and vertical lines from the source and target.

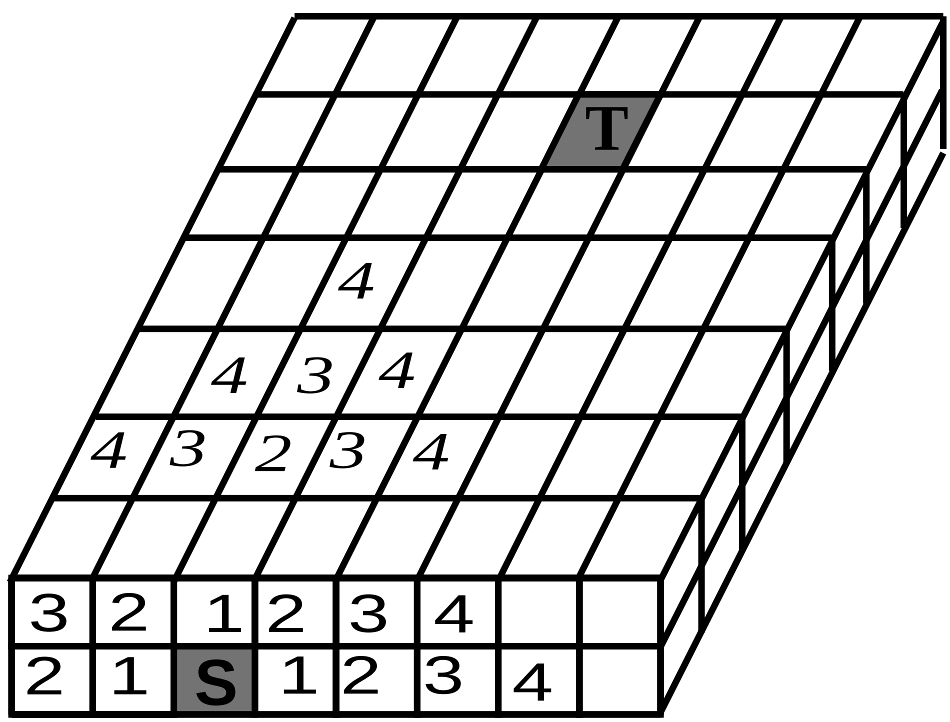
Notes

- When the routing area is not congested, the above algorithms are expected to run fast.
- Particularly, Hightower algorithm is expected to run in time proportional to the number of bends.
- A conservative estimate of running time in a complicated maze is $O(N^4)$
- Thus the memory saving in line search algorithm is dramatic, but the running time does not improve very much.
- We might also need to backtrack from dead ends (resulting from bad sequences of trial lines).

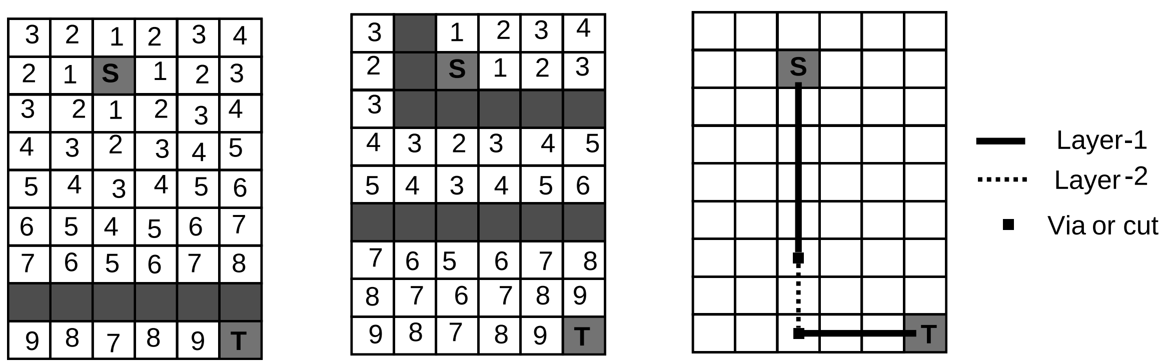
Other Issues

- Multi Layer Routing
- Three-Dimensional Grid

  Three dimensional cellular array for two layer routing is shown below.

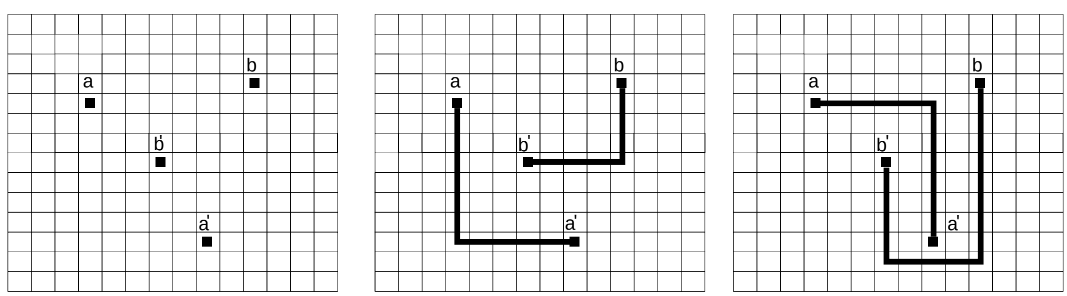


- Two Planar Arrays. Figure below illustrates two layer routing using two arrays. (a) Layer-1. (b) Layer-2. (c) Retrace path.

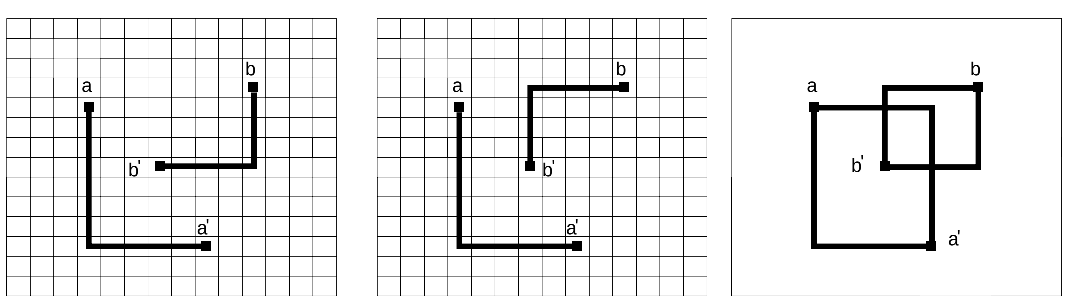


(a) (b) (c)

- Ordering of Nets.

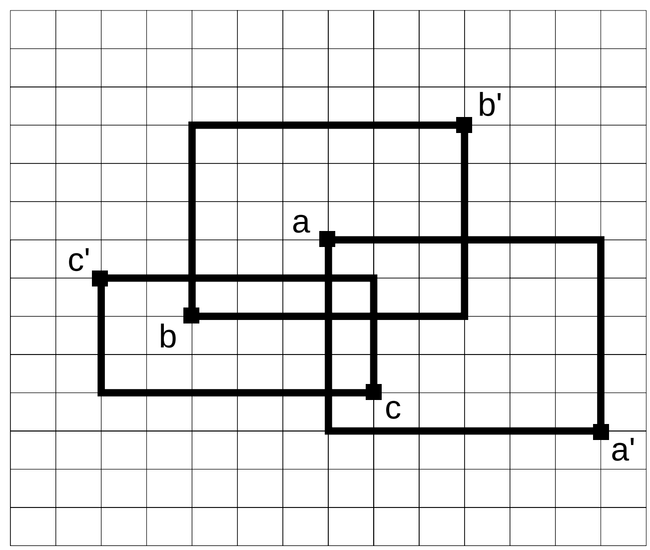


(a) (b) (c)

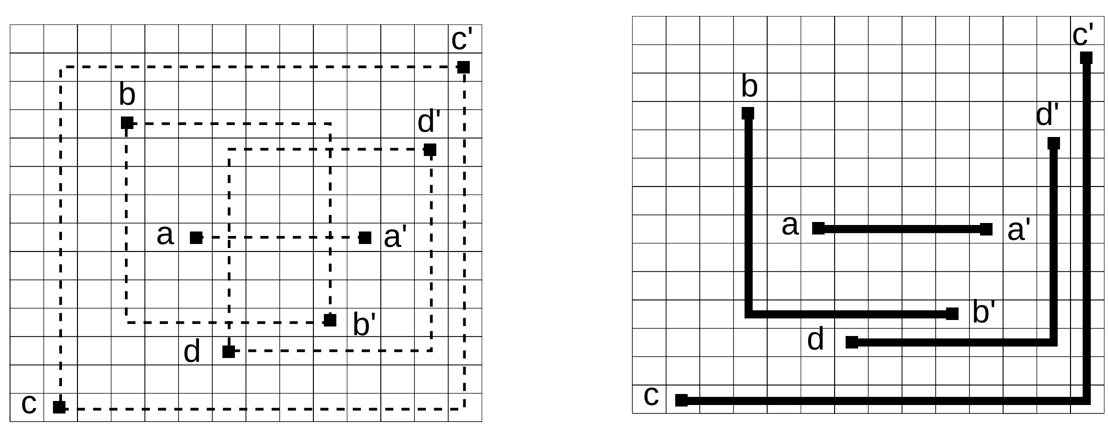


(d) (e) (f)

- Non-transitivity of three 2-point nets.

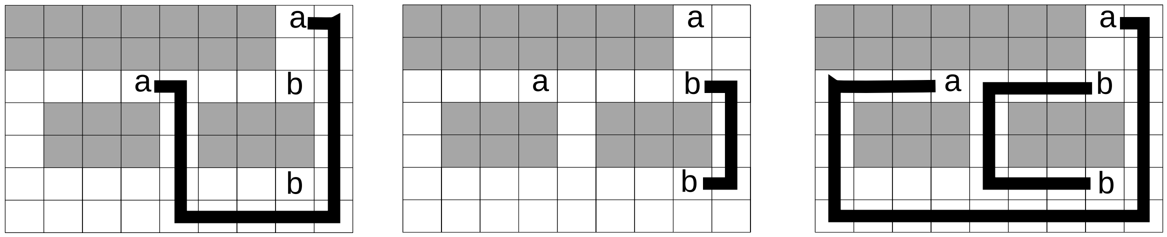


- Four 2-point nets to be ordered.



(a) (b)

- (a) Optimal routing of a prevents routing of b.
  (b) Optimal routing of b prevents routing of a.
  (c) Non-optimal routing of nets a and b.

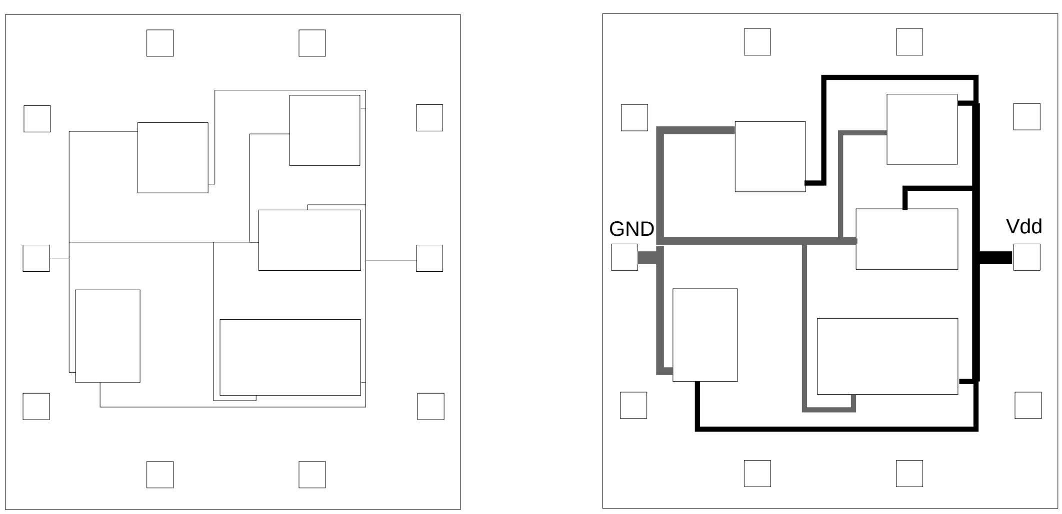


(a) (b) (c)

- Rip-up and Rerouting

- Power and Ground Routing

  (a) Topological trees for power and ground nets. (b) Actual widths of routing layers.



(a) (b)

Summary

- In this chapter we examined two types of grid routers, the maze router and line-search router.
- The maze router uses a physical grid, and line search routers use an imaginary grid.
- The basic grid router that uses Lee algorithm has a large memory requirement and also may require a large amount of running time.
- Techniques to reduce the running time and memory requirement were discussed in detail.
- Other algorithms that modify the filling phase of Lee Algorithm to reduce the running time are Hadlock's algorithm and Soukup's algorithm.
- Their techniques were illustrated with examples. Line search algorithms overcome the high memory requirement of Lee algorithm.
- Two line search heuristics, one due to Mikami and Tabuchi, and the other due to Hightower were presented.
- Maze running algorithms guarantee finding a shortest path if one exists, even if it is the most expensive in terms of the number of vias.
- Line search algorithms guarantee finding a path if one exist; (not necessarily the shortest).

- But they may require several backtracks for all dead ends that are reached.
- In practice however line search algorithms can be significantly faster than maze running algorithms.
- The major advantage for which maze running algorithms are preferred over line search algorithms is that the former are *grid-cell* oriented.
- This gives more flexibility to the weighting of routing area of the chip.
- This is of extreme importance since proper weighting of cells enables finding superior routes.
- Both the maze router and line-search router connect a single net at a time.
- Modifications to the basic routing technique to accommodate multi-point nets are needed.

Algorithm Constraint_Graph_Compaction;

1. Construct the constraint graph $G(V, E)$;
2. Apply the critical path algorithm and find for each ve
3. Move each element to within its range of tolerance;

End.