Obstacle-Aware Longest Path using Rectangular Pattern Detouring in Routing Grids

Jin-Tai Yan, Ming-Ching Jhong and Zhi-Wei Chen

Department of Computer Science & Information Engineering Chung Hua University, Taiwan

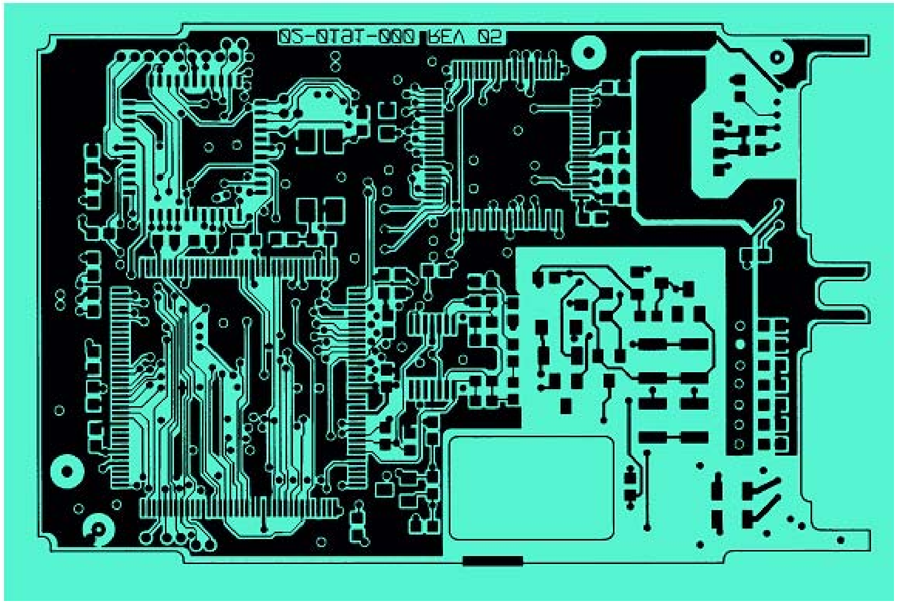
ASPDAC'10

Outline

- Introduction
- Problem Formulation
- Rectangular Pattern Detouring
- Obstacle-Aware Longest-Path in Routing Grids
- Experimental Results
- Conclusions

Introduction

- More than thousands of pins are involved in modern PCBs.
- Routing methods are complex under various design constraints, ex: length matching, coupling control...etc.
- Physical “obstacles” on PCB
  - Component mounting pads: DIP / SMD
  - Pre-route predictions: analog / TTL design reservation



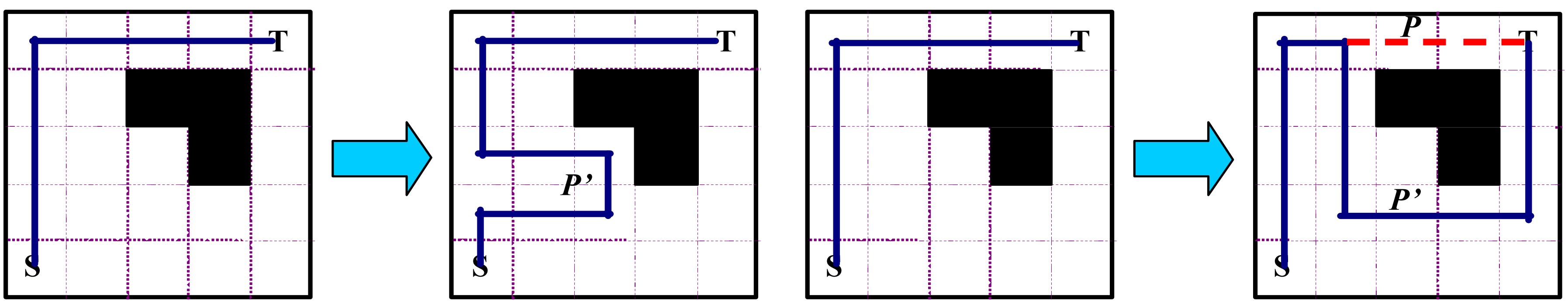
Source from:*www.speedingedge.com*

Motivation & Contributions

- The length controllability of a net decides
  - Routing delay of the net
  - Ability to meet the length-matching constraint
- An effective generation for the longest path gives
  - Grid-based global routing for a single net in PCB
  - Consideration of obstacles on PCB
  - Use of Rectangular pattern detouring for the reduction of the routing time

Preliminary

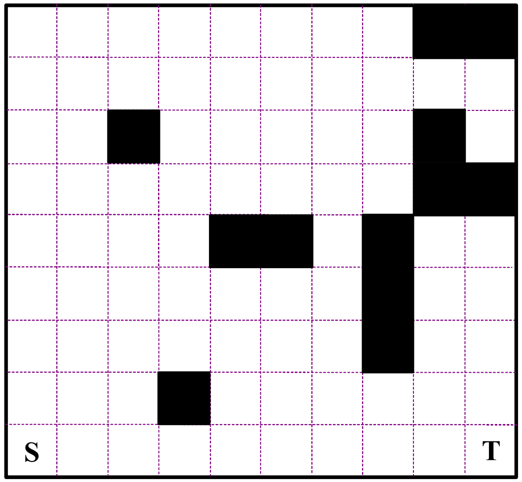
- Routing path
  - Two grids are specified as the locations of the start and target terminals, ***S*** and ***T***
  - A **path** connects ***S*** and ***T*** by using horizontal or vertical segments and passes each available grid at most once
  - The **length** of a connecting path is defined as the number of passed grids in the routing path
- Flip operations for path detouring
  - R-flip: A partial path can be detoured by using adjacent routing grids and inserting a detouring path, P'
  - C-flip: A partial path, P, can be replaced by using another partial path, P', where the length of P' is larger than that of P



Problem Formulation

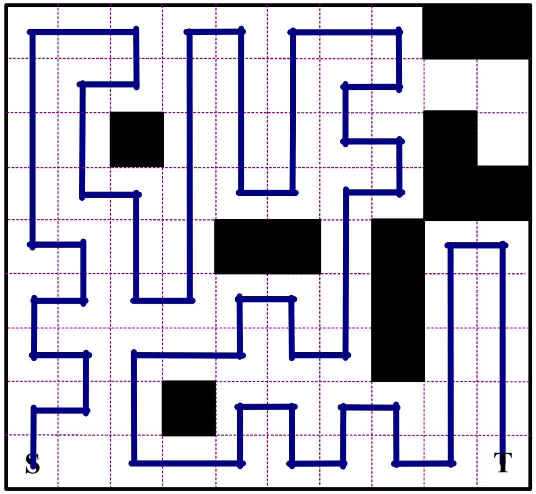
[Input]

- A single routing layer in routing grids
- A start terminal, ***S***, and a target terminal, ***T***, in **m**x**n** routing grids
- A set of obstacles



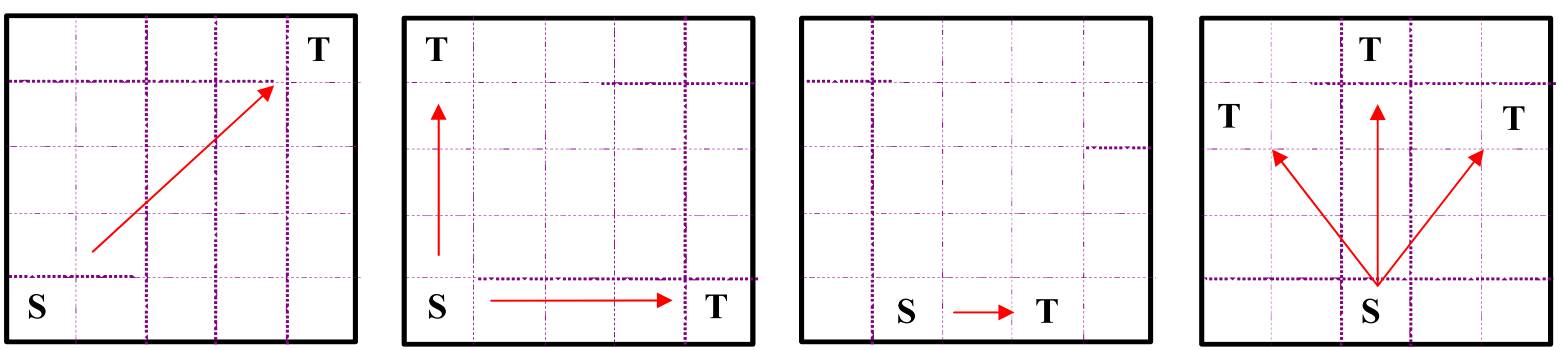
[Output]

- The longest path that passes the available routing grids from ***S*** to ***T***



Rectangular Pattern Detouring

- Given a set of m×n routing grids without any obstacle, all the possible detouring results are considered as follows:
  - Both *S* and *T* are located on corners
    1. (Cond. I) In different sides
    2. (Cond. II) In the same side
  - *S* and *T* are located on boundaries but not on corners
    1. (Cond. III) In the same side
    2. (Cond. IV) In different sides



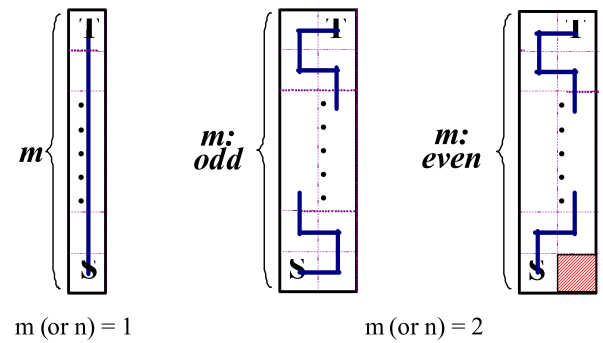
Cond. I: Corners in Different Sides

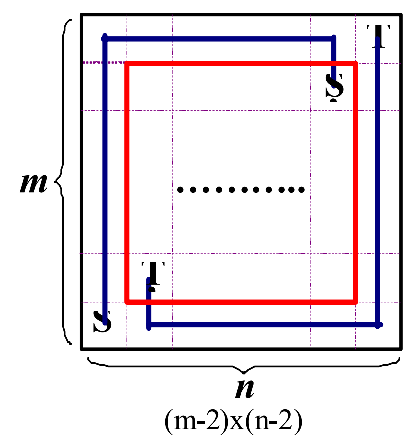
If (m or n = 1) ➔ straight path

If (m or n = 2) ➔ snaking path

If (m and n >= 3)

- mxn routing grids for corners in different sides
  - Two L-type routing paths, and
  - (m-2)x(n-2) routing grids for corners in different sides



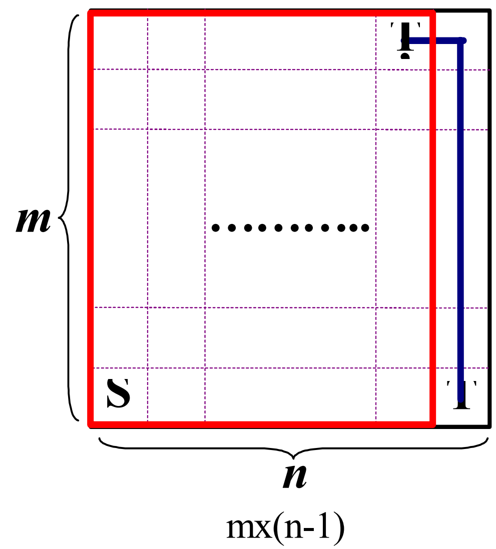


Detouring Length	#Unreachable grids	Conditions
mn	0	(1) m or n is odd
mn-1	1	(1) m and n are even

Cond. II: Corners in the Same Side

If (n >= 2)

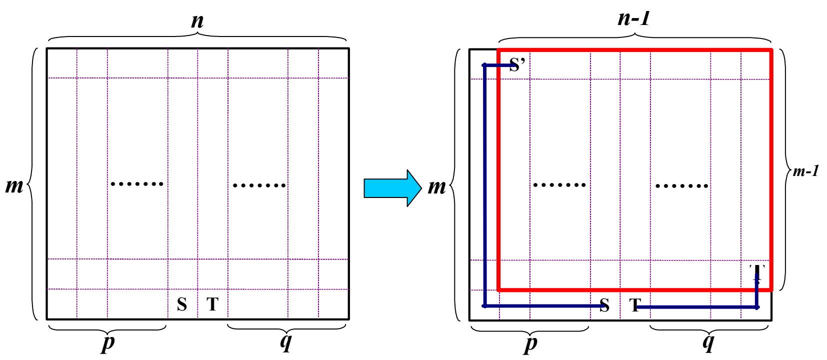
- mxn routing grids for corners in the same side
  - One L-type routing path, and
  - mx(n-1) routing grids for corners in different sides



Detouring Length	#Unreachable Grids	Conditions
mn	0	(1) (m is odd) or (n is even)
mn-1	1	(1) (m is even) and (n is odd)

Cond. III: Boundary in the Same Side

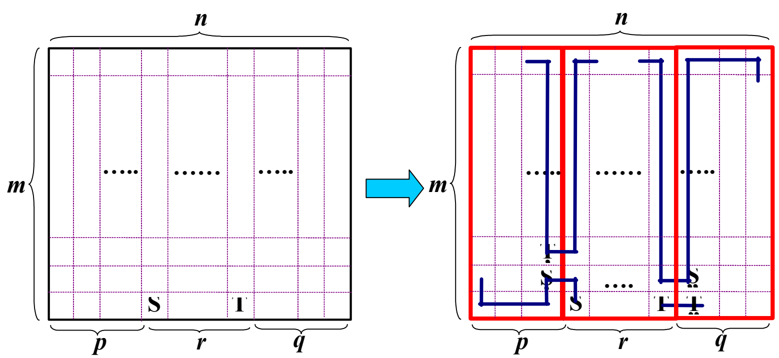
- S and T are located on adjacent grids
  - p(q) are the number of columns on the left(right) sides of S(T)
  - mxn routing grids for boundary in the same side
    - Two L-type routing paths, and
    - (m-1)x(n-1) routing grids for corners in different sides



Detouring Length	#Unreachable Grids	Conditions
mn	0	(1) m or n is even
mn-1	1	(1) m and n are odd

Cond. III: Boundary in the Same Side

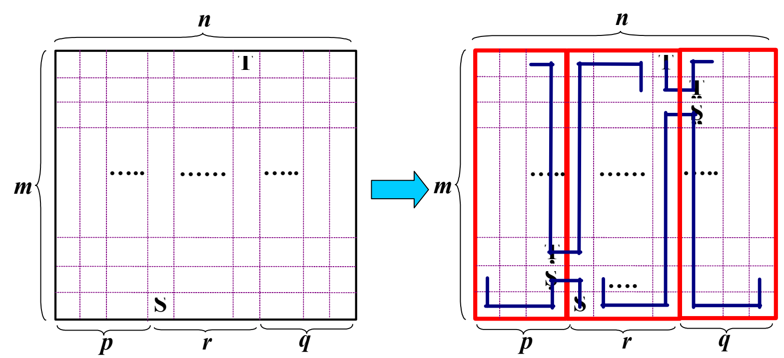
- S and T are not located on adjacent grids
  - p(q) are the number of columns on the left(right) sides of S(T); r is the number of columns between S and T
  - mxn routing grids for boundary in the same side
    - mxr routing grids for corners in the same side
    - mxp and mxq routing grids for boundary in the same side



Detouring Length	#Unreachable Grids	Conditions
mn	0	(1) m and r are even (2) (m is odd) and (p and q are even)
mn-1	1	(1) (m is even) and (r is odd) (2) (m is odd) and (p + q is odd)
mn-2	2	(1) m, p and q are odd

Cond. IV: Boundaries in Different Sides

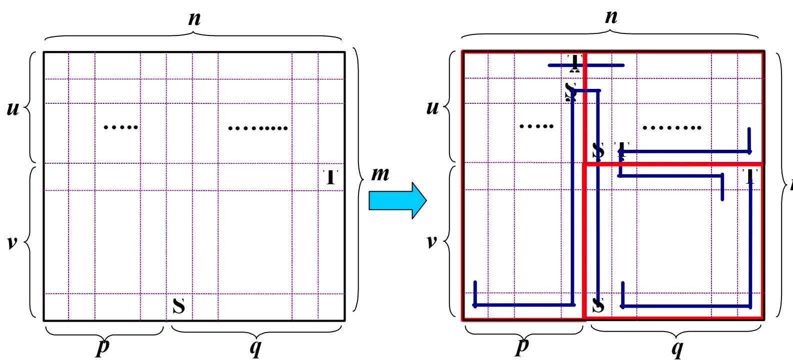
- S and T are located on opposite sides
  - p(q) are the number of columns on the left(right) sides of S(T); r is the number of columns between S and T
  - mxn routing grids for boundaries in different sides
    - mxr routing grids for corners in different sides, and
    - mxp and mxq routing grids for boundary in the same side



Detouring Length	#Unreachable Grids	Conditions
mn	0	(1) (m is even) and (r is odd) (2) (m is odd) and (p and q are even)
mn-1	1	(1) m and r are even (2) (m is odd) and (p+q is odd)
mn-2	2	(1) m, p and q are odd

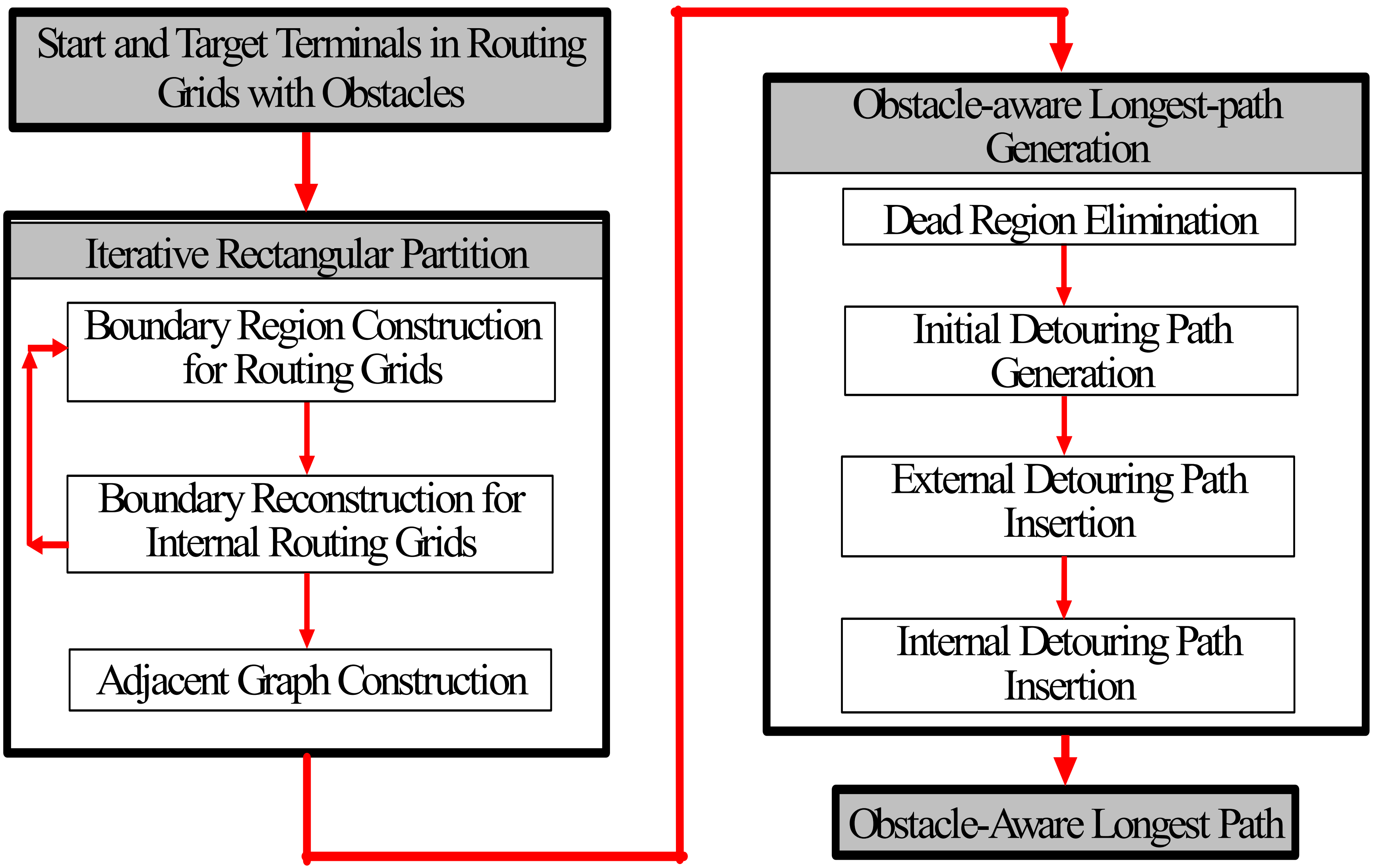
Cond. IV: Boundaries in Different Sides

- S and T are located on adjacent sides
  - p is the number of columns on the left sides of S; q is the number of columns between S and T; v is the number of rows between S and T; u is the number of rows on the top of T
  - mxn routing grids for boundaries in different sides
    - vxq routing grids for corners in different sides, and
    - uxq and pxm routing grids for boundary in the same side



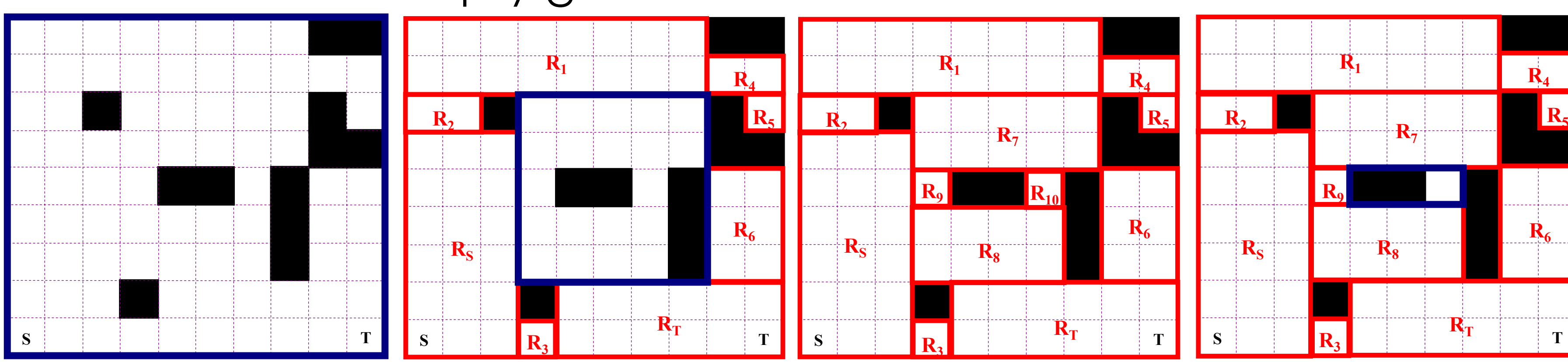
Detouring Length	#Unreachable Grids	Conditions
mn	0	(1) (q is even) and (u and v are odd) (2) (p and u are even) and (v is odd) (3) (q is odd) and (u and v are even)
mn-1	1	(1) p, q and v are odd (2) (p is even) and (q and u are odd) (3) (p is odd) and (q and r are even) (4) p, q and v are even
mn-2	2	(1) (p and u are odd) and (v is even)

Design Flow for Obstacle-aware Longest Path Generation



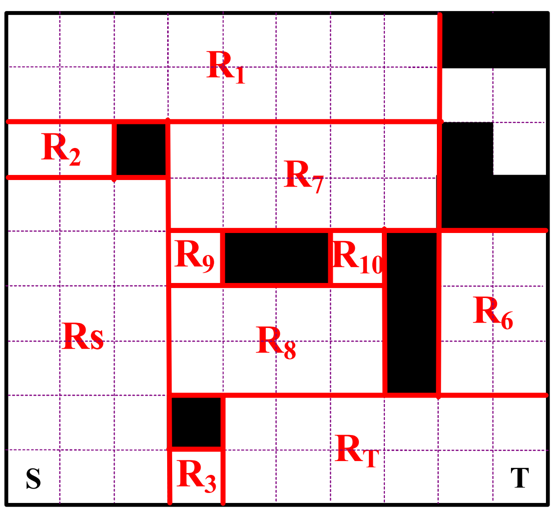
Iterative Rectangular Partition

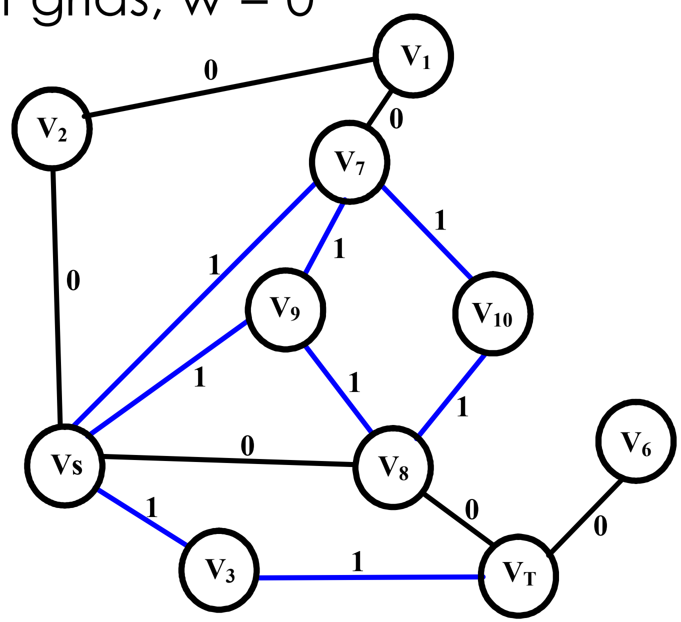
- Construction of Outer Boundary : Minimum covering rectangular boundaries
- Definition of Corner Grids : the perpendicular sides of any routing grid are
  - Outer boundary / Obstacles / Partitioned regions
- Construction of Rectangular Regions: Maximum rectangular region for corner grids
- Iterative Rectangular Partition
  - Construction of Outer Boundary-> Definition of Corner Grids -> Construction of Rectangular Regions
  - Until all of empty grids are included



Construction of an Adjacent Graph

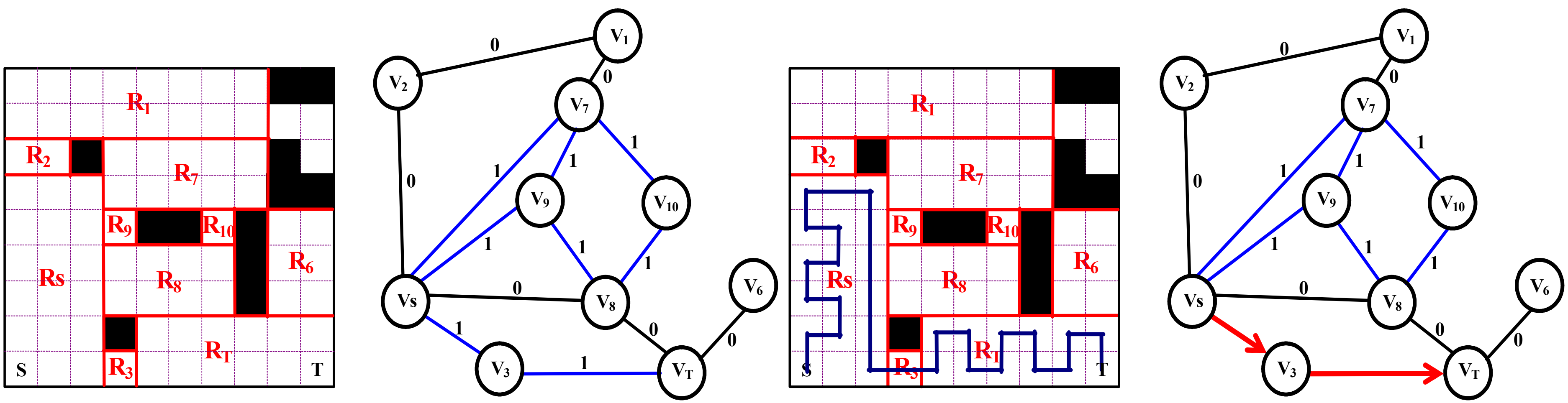
- Given a set of partitioned regions, with R_S, R_T and a set of empty regions
- Adjacent graph, G(V, E)
  - An undirected edge-weighted graph
  - v_i, $v_i \subseteq V$, represents the partitioned region, R_i
  - $e_{i,j}$, $e_{i,j} \subseteq E$, represents the adjacent relation between two partitioned regions, R_i and R_j
  - Edge weight (w)
    - Only an adjacent grid, w = 1
    - More than two adjacent grids, w = 0





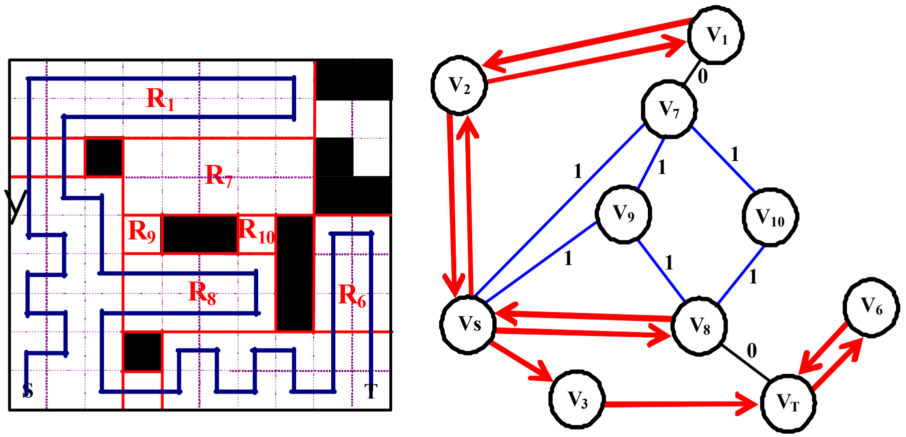
Initial Detouring Path Generation

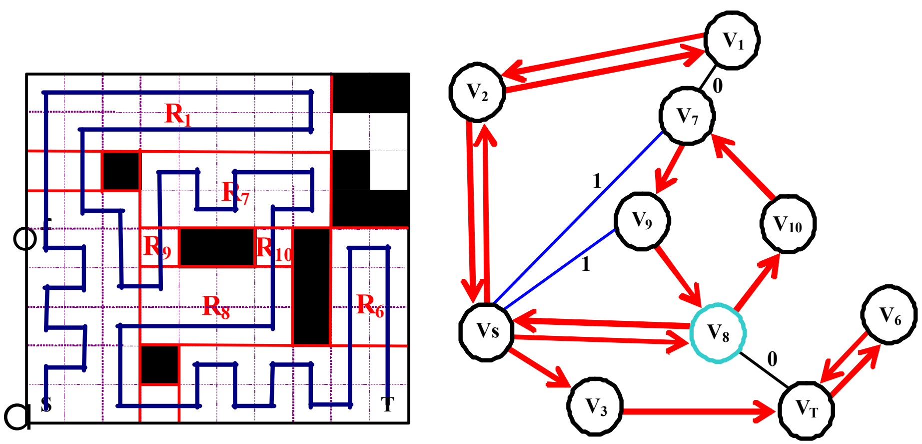
- Dead region elimination
  - Dead region: v_i is not v_s and v_T, the degree is 1 and the edge weight is 1
  - Elimination of dead regions due to no help for path generation
- Initial detouring path generation
  - Based on breadth-first search and rectangular pattern detouring, the initial detouring path between v_s and v_T is determined
- Reference vertices
  - The vertices in the adjacent graph are covered by the current detouring path



External / Internal Detouring Path Insertion

- Adjacent relation with reference vertices
  - Adjacent relation can be obtained by using a BFS algorithm
  - W= "0": for a R-flip operation
  - W= "1": for a C-flip operation possibly
- Selection policy
  1. A vertex is included for a R-flip operation, excluded by a C-flip operation
     - Directly involve the vertex for a R-flip operation
  2. A vertex is both included for a R-flip and C-flip operations
     - Choose the results with min. number of unreachable grids
- Internal detouring path
  - The region is generated by running a C-flip operation and routed by running a R-flip operation





Analysis of Time Complexity

Given a set of m×n routing grids

- Iterative rectangular partition: O(*mn***log**(*mn*))
  - Sorting the locations of all the obstacles: O(*mn***log**(*mn*))
  - Selection of all the partitioned regions: O(*mn*)
- Obstacle-aware longest path generation: O(*mn*)
  - Dead region elimination: O(*mn*)
  - Initial detouring path by BFS algorithm: O(*mn*)
  - External / Internal detouring insertions: O(*mn*)

Obstacle-aware longest path generation: O(*mn***log**(*mn*))

Experimental Results

- Implementation Environment
  - PC: Pentium QuadCore 2.66GHz w/ 2G memory
  - Standard C++ language
  - Ten tested cases, Data01-Data10, from Y. Kohira, et al.[5]

Circuits	*Area*	*#Grid*	*#Dead Grids*	*US Routing*		*Our Routing*	
				Wirelength (#Grid)	*Time(s)*	*Wirelength (#Grid)*	*Time(s)*
Data01	***10x9***	***83***	***0***	***80(96.4%)***	***0.004***	***82(98.8%)***	***<0.001***
Data02	***11x13***	***121***	***0***	***119(98.3%)***	***0.005***	***119(98.3%)***	***<0.001***
Data03	***11x13***	***110***	***0***	***107 (97.3%)***	***0.005***	***107 (97.3%)***	***<0.001***
Data04	***11x13***	***106***	***0***	***103(97.2%)***	***0.005***	***103(97.2%)***	***<0.001***
Data05	***11x13***	***98***	***0***	***95(96.9%)***	***0.005***	***95(96.9%)***	***<0.001***
Data06	***11x13***	***116***	***0***	***85(73.3%)***	***0.005***	***85(73.3%)***	***<0.001***
Data07	***11x13***	***117***	***0***	***113(96.6%)***	***0.005***	***113(96.6%)***	***<0.001***
Data08	***11x13***	***118***	***0***	***115(97.5%)***	***0.005***	***115(97.5%)***	***<0.001***
Data09	***20x20***	***297***	***21***	***251(90.9%)***	***0.010***	***261(94.6%)***	***0.002***
Data10	***70x100***	***6654***	***4***	***6626(99.6%)***	***4.760***	***6636(99.8%)***	***1.181***

Conclusions

- PCB Routing,
  - The length controllability of a net decides the routing delay of the net and skew
  - The consideration of obstacles is necessary
- Longest Path Generation
  - Analysis of unreachable grids for rectangular pattern detouring for the reduction of the routing time
  - Based on the rectangular partition in routing grids, an efficient approach is proposed to generate the grid-based longest path for PCB routing
- Experimental Results
  - Compared with US routing, our proposed approach can achieve longer paths for tested examples in less CPU time

Thanks Your Kind Attention!