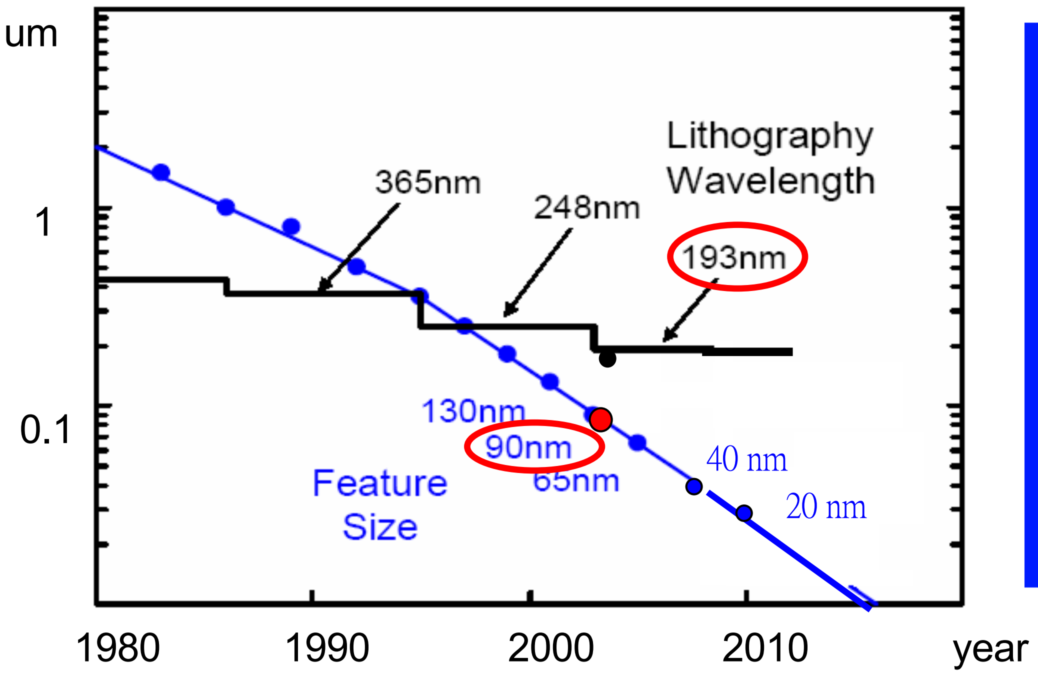
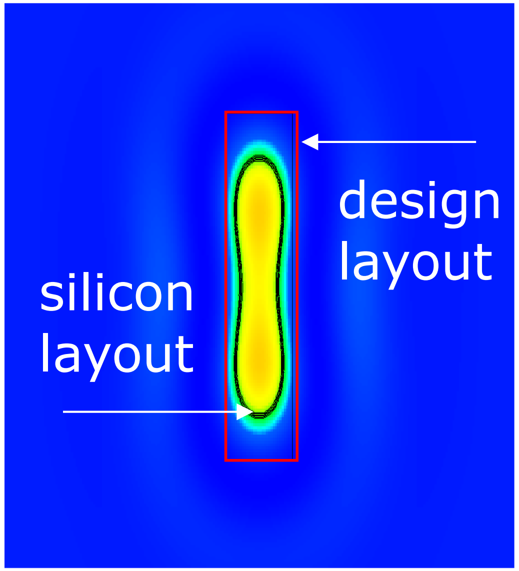
Routing in Advanced Nodes

Challenge 1 – Design Rule Handling

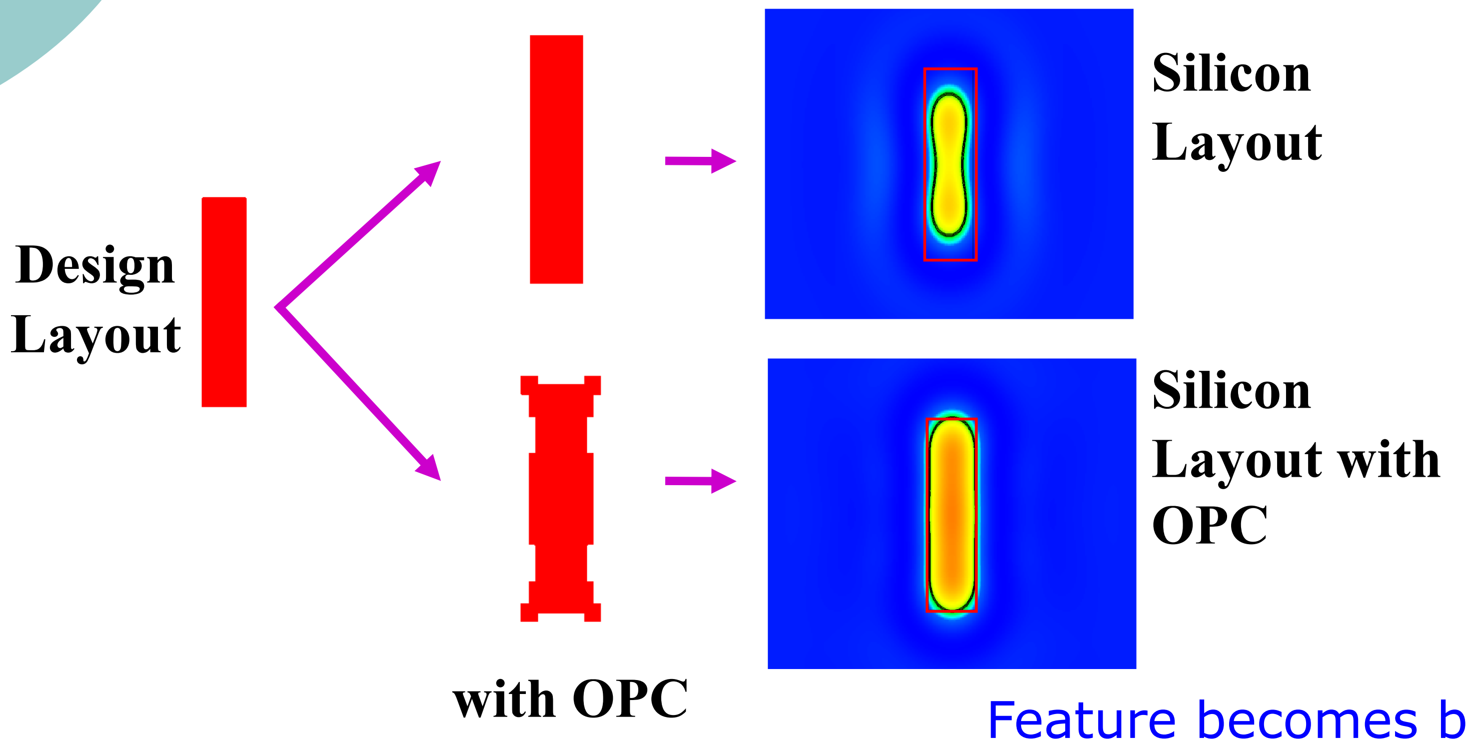
- Process limitations
  - Lithography printability, process variation and etc.
  - What you see is not what you get
    - Impact chip yield





Challenge 1 – Design Rule Handling

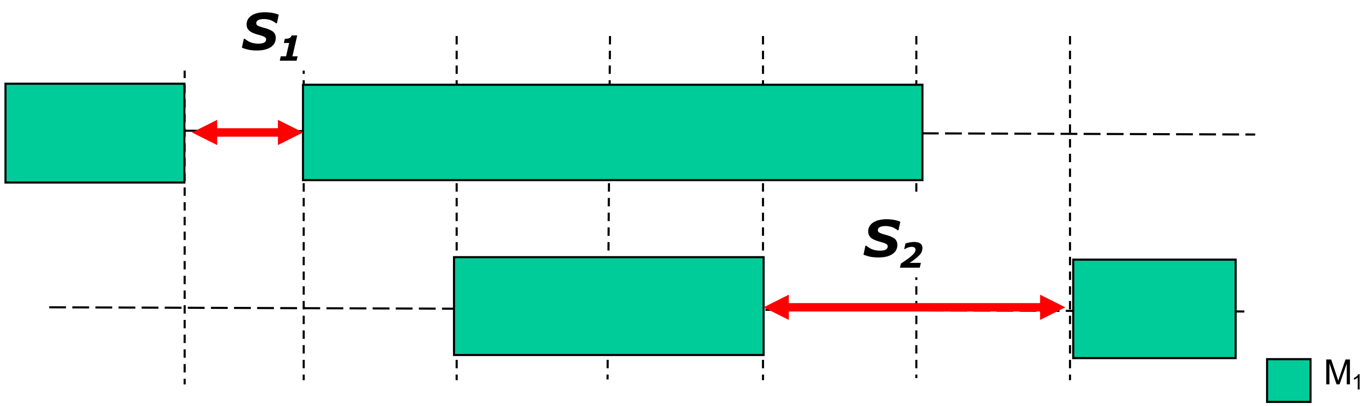
- Optical proximity correction (OPC)
  - Using design rules to specify the required extra spacing



Feature becomes bigger after OPC =>
More spacing is required btn features

End-End Separation Rule

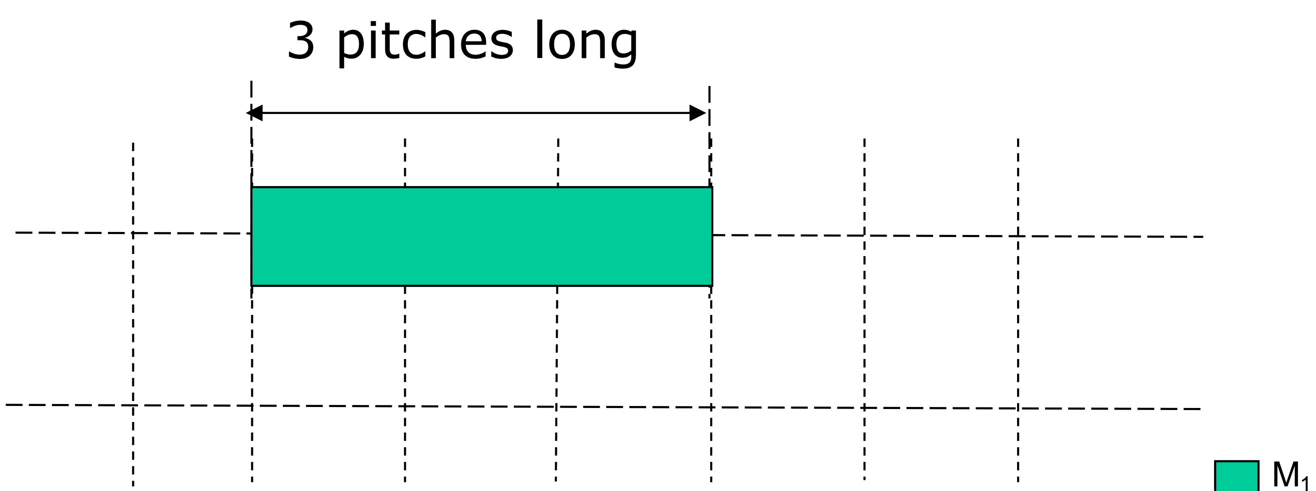
- The end-end separation rule
  - $\geq S_2$: if there is a wire near a wire end on a neighboring track
  - $\geq S_1$: otherwise



End-End Separation

Minimum Length Rule

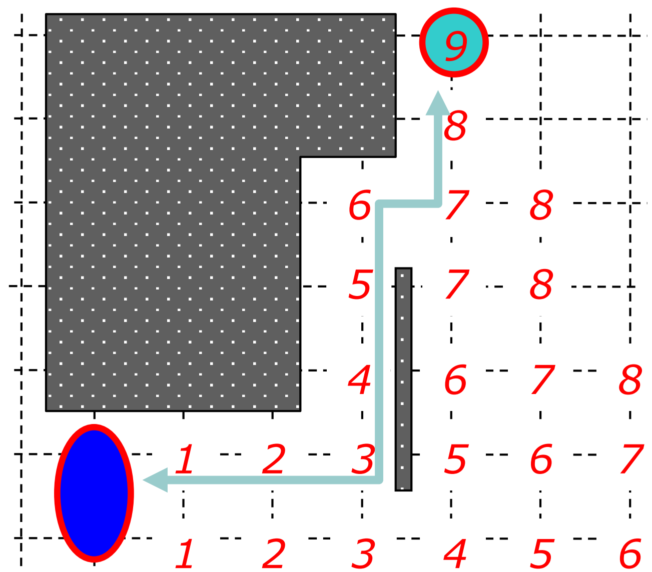
- The minimum required length for each wire segment



MANA: A Shortest Path MAze Algorithm under Separation and Minimum Length NAnometer Rules

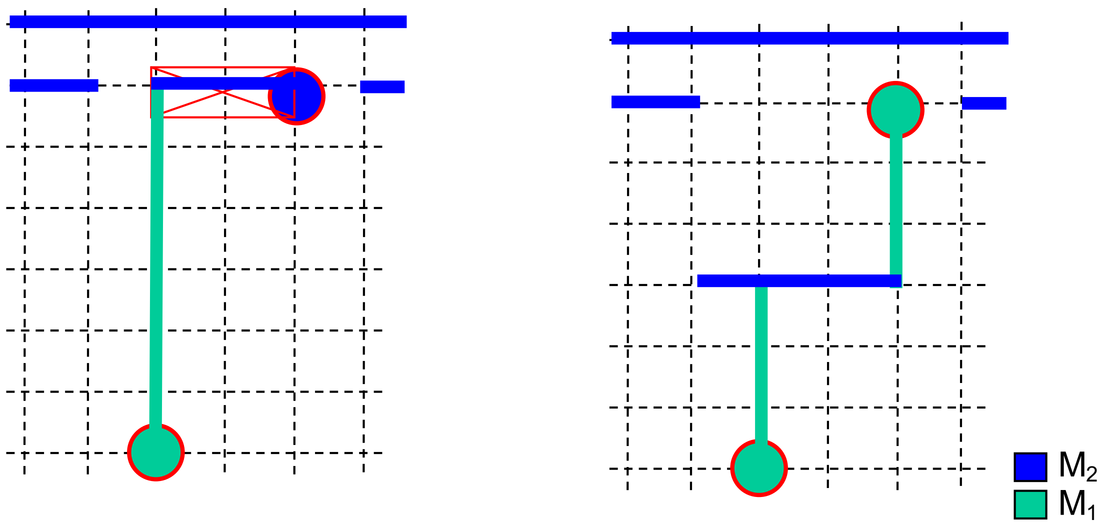
F.Y. Chang, R.S. Tsay, W.K.Mak, S.H. Chen

Conventional Maze Routing



How to handle the New Rules (1/2)

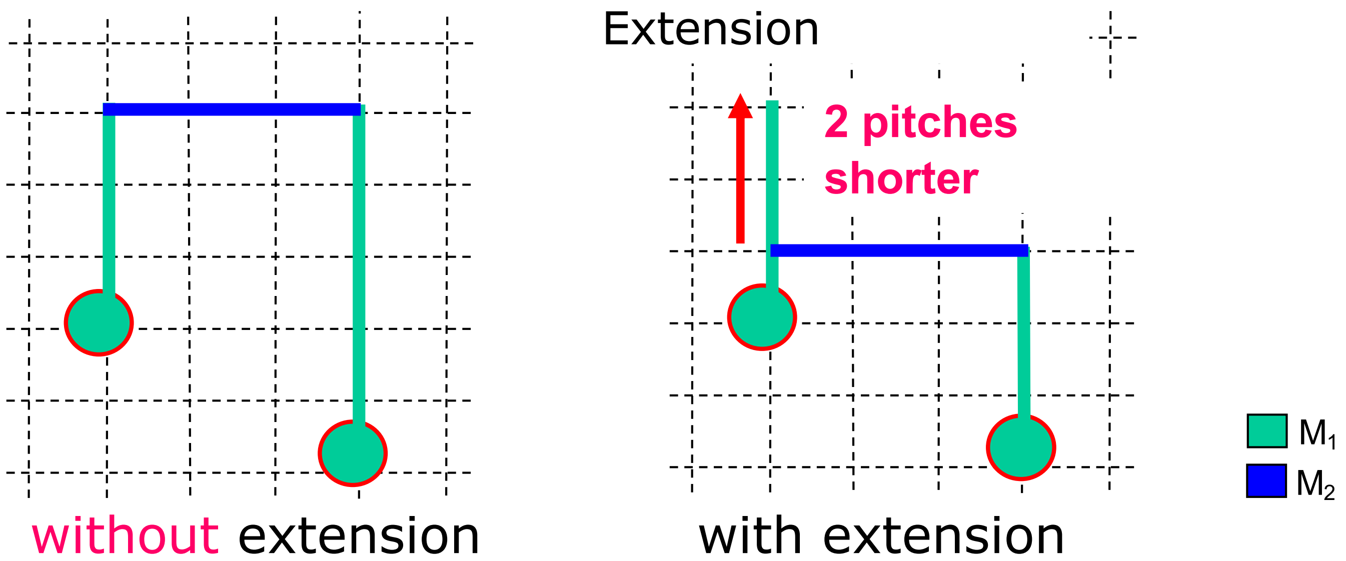
- By post-processing
  - 1. Find shortest paths without considering the above rules using conventional maze routing
  - 2. Extend short wires to meet the required length, rip-up and re-route for end-to-end separation violations



Min-Wire-Length: 3 Pitches

How to handle the New Rules (2/2)

- By enhanced maze routing
  - Find a shortest path where each segment is no less than the required length (DAC 2012)



Min-Wire-Length: 3 Pitches

Problems of Previous Methods

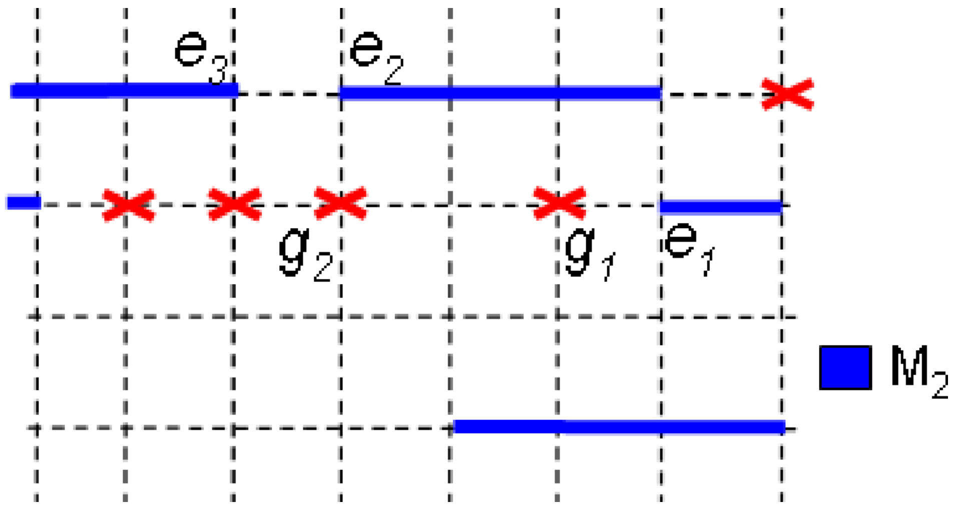
- Paths found by previous approaches may contain violations or are not shortest

The Proposed Algorithm

- MANA: a shortest path MAze algorithm under NAnometer Rules
  - Nanometer design rules
    - End-end separation rules
    - Minimum length rule
  - Polynomial time complexity

End-End Separation Rule Handling

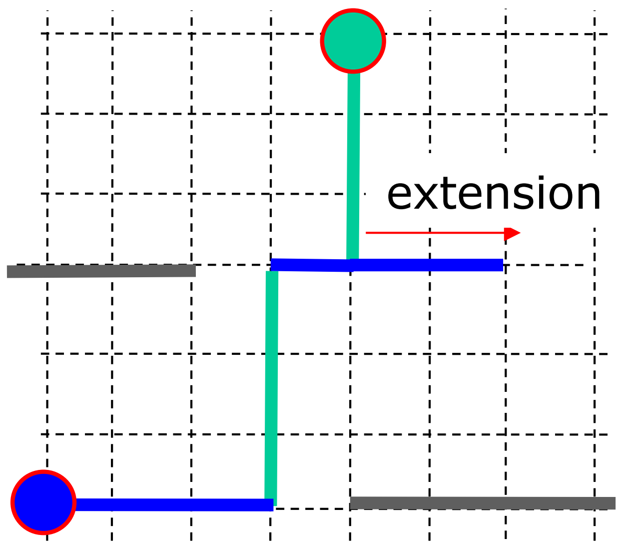
- End-end separation rule handling
  - Pre-filter out infeasible grid points
    - For each grid point, check if any end-end separation rule is violated when a wire passes the point



Separation = 2 pitches, if there is a wire near a wire end

The Minimum Length Handling

- Minimum length rule handling
  - During maze routing, check if there is enough resource for wire extension



Min-Wire-Length: 3 Pitches

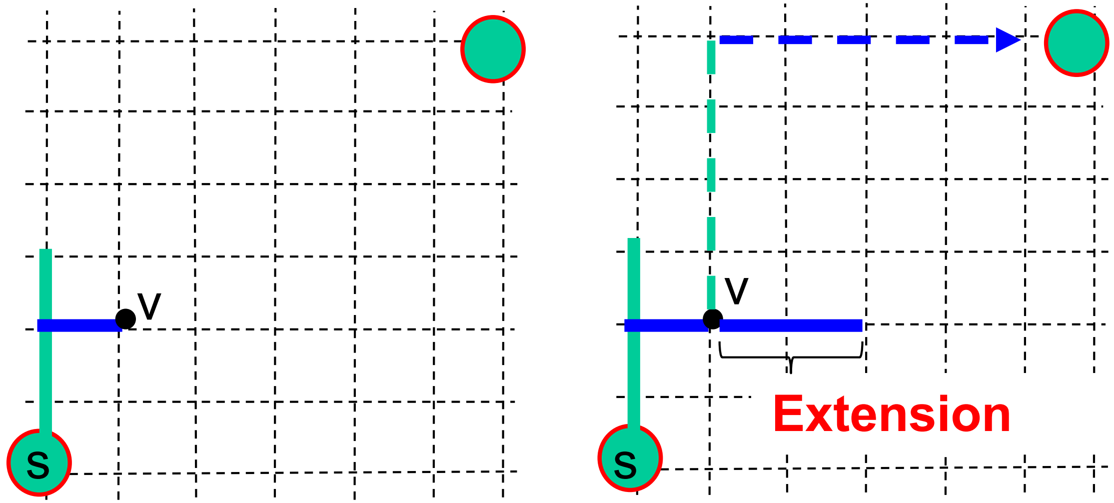
Find a Shortest Path (1/2)

- Length of a partial path
  - Min length : minlen(P) denotes total length of all wires on partial path P without extension of the last wire segment
  - Max length: maxlen(P) denotes total length of all wires on partial path P with extension of the last wire segment

e.g. Consider a partial path from s to v below

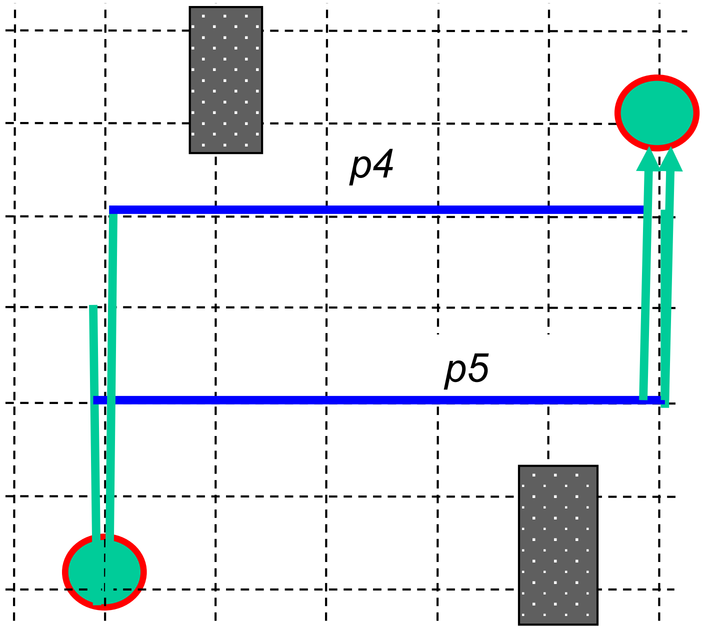
Min length: 4 pitches

Max length: 6 pitches



Min-Wire-Length: 3 Pitches

Find a Shortest Path (2/2)



min length	
p4	*9*
p5	*10*

Length of complete path expanded from *p4*: 12

Length of complete path expanded from *p5*: 11

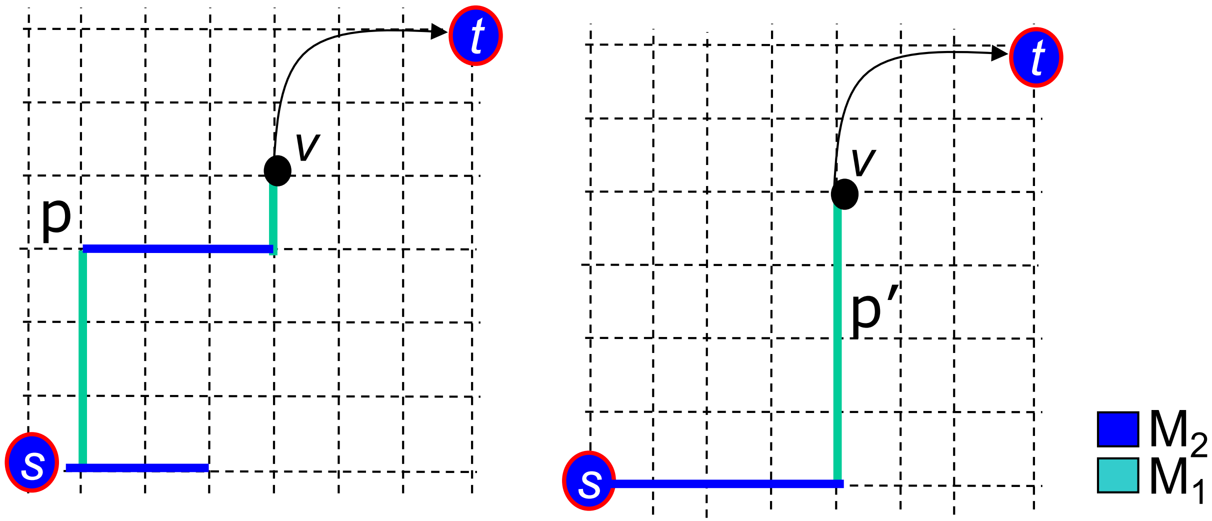
Min-Wire-Length: 3 Pitches

- Key
  - Expand legal partial paths from the source until finding a complete path with length no longer than the min-length of any un-expanded partial path

Polynomial time complexity

- Polynomial time complexity
  - By pruning unnecessary partial paths
    - We can prune p and keep p′. (Why?)

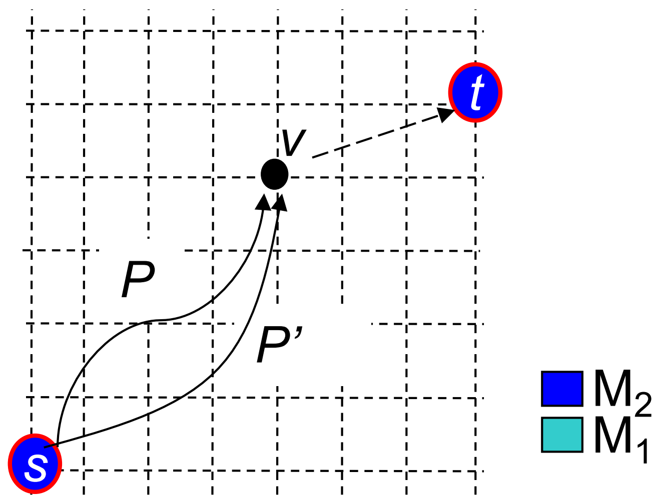
Min-Wire-Length:
3 Pitches



Pruning Strategies (1/2)

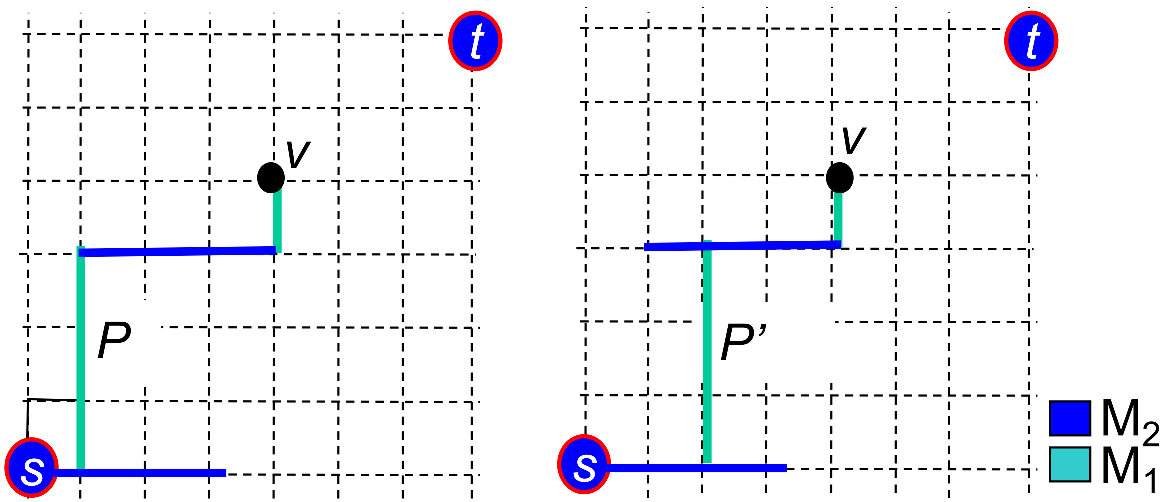
Let P and P' be two partial paths reaching v from the same direction.

- Strategy 1
  - If minlen(P) $\geq$ maxlen(P')
    - Prune P



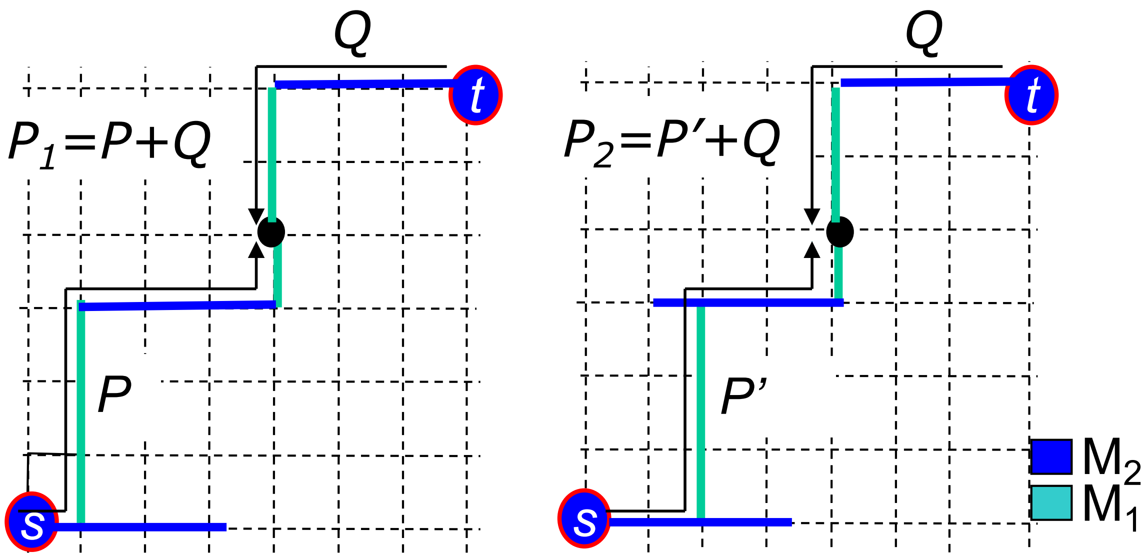
Pruning Strategies (2/2)

- Strategy 2
  - If maxlen(P) = maxlen(P') and minlen(P) = minlen(P')
    - Prune either P or P'



Equal Path Length Theorem (1/4)

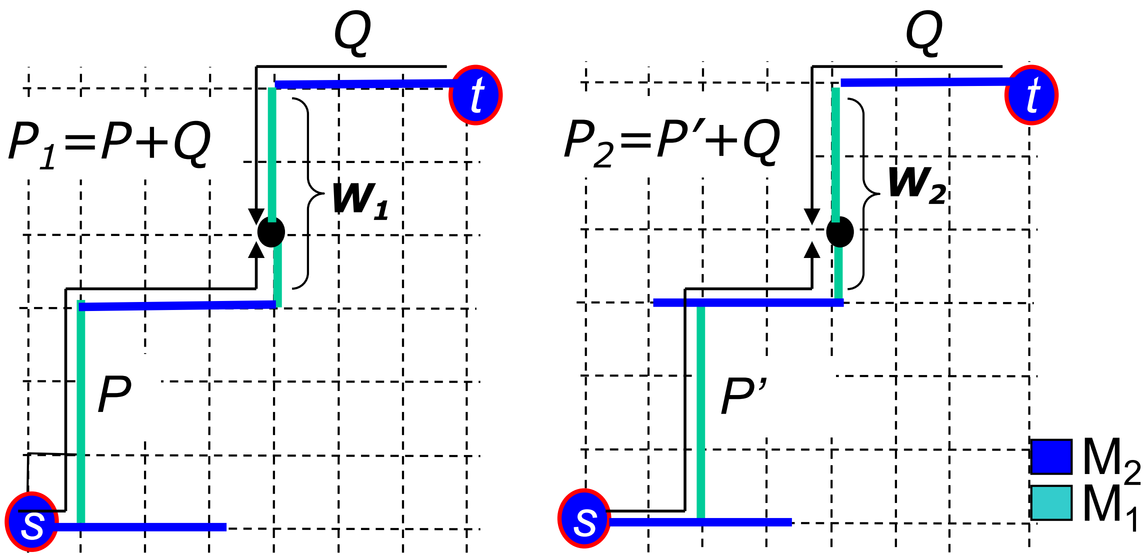
- **Theorem:** Suppose P and P' reach v from the same direction. If maxlen(P)=maxlen(P') *and* minlen(P)= minlen(P'), then len(P_1) = len(P_2) where P_1 and P_2 are two complete paths expanded from P and P', respectively, with the same subsequent subpath Q to the target.



Equal Path Length Theorem (2/4)

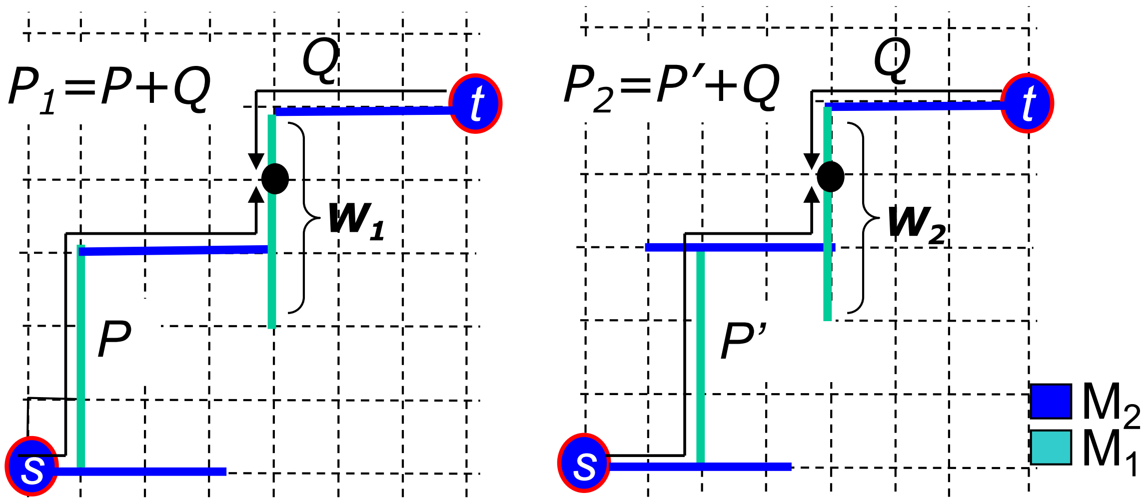
Proof. Assume wire segments W_1 of P_1 and W_2 of P_2 contain the last segments of P and P', resp.

- Case 1: W_1 and W_2 need no extension
  - len(P_1) = minlen(P) + minlen(Q)
  - len(P_2) = minlen(P') + minlen(Q)
  - As minlen(P)=minlen(P'), we get len(P_1)=len(P_2)



Equal Path Length Theorem (3/4)

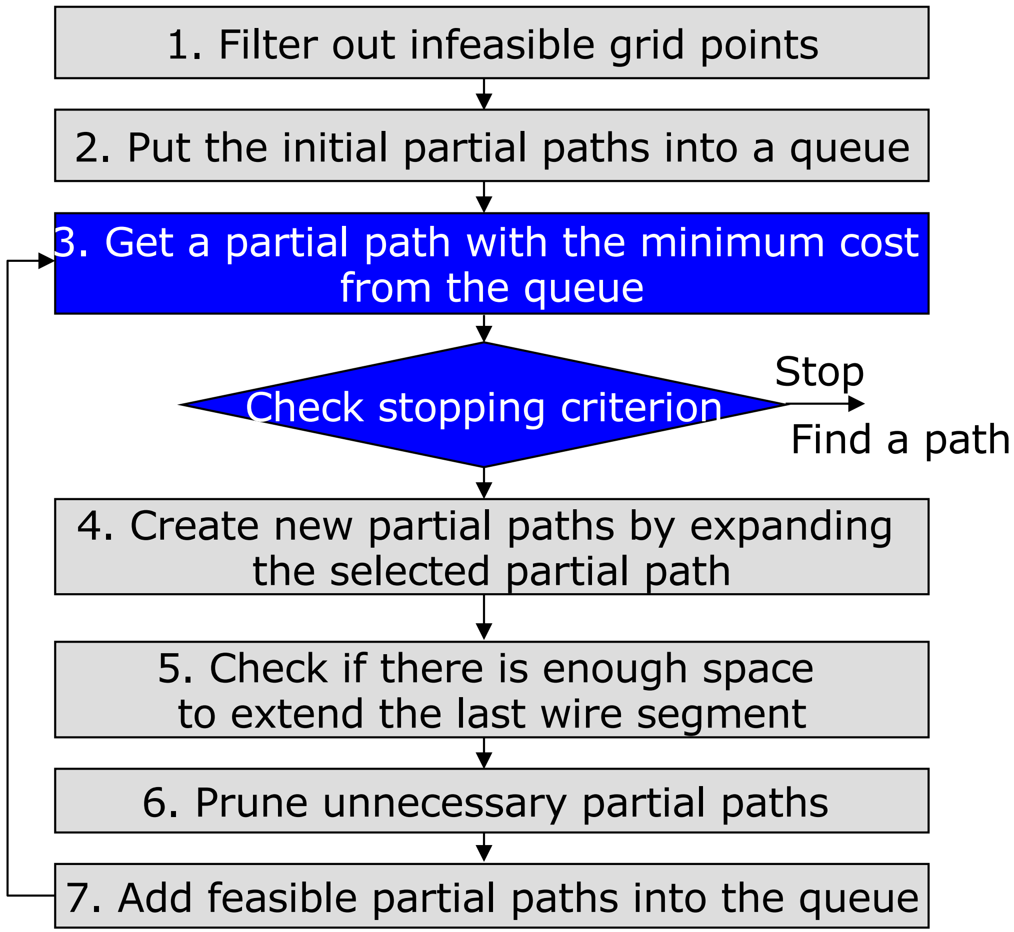
- Case 2: W_1 and W_2 both need extensions
  - By the assumption, the last segments of P and P' must be identical which implies W_1 and W_2's minimum extensions are identical, so len(P_1) = len(P_2) again



Equal Path Length Theorem (4/4)

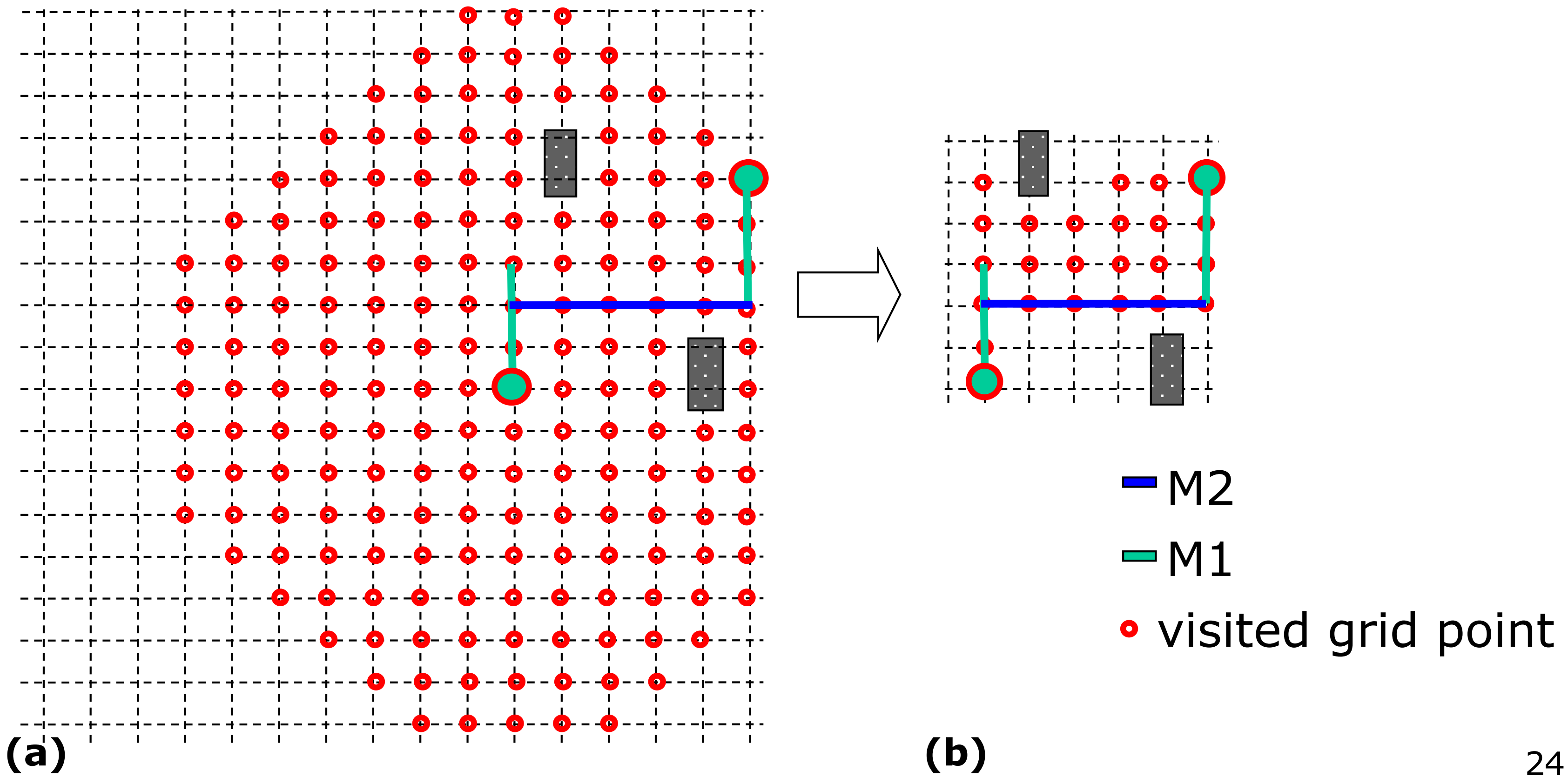
- Case 3: Only one of W_1 and W_2 contains extensions
  - Non-existing case

Complete Algorithm



Best Cost-First Expansion (A*-Search)

- Expand partial path with least cost first where cost(P) = maxlen(P)+Manhattan dist. from last wire on partial path P w/ extension to target

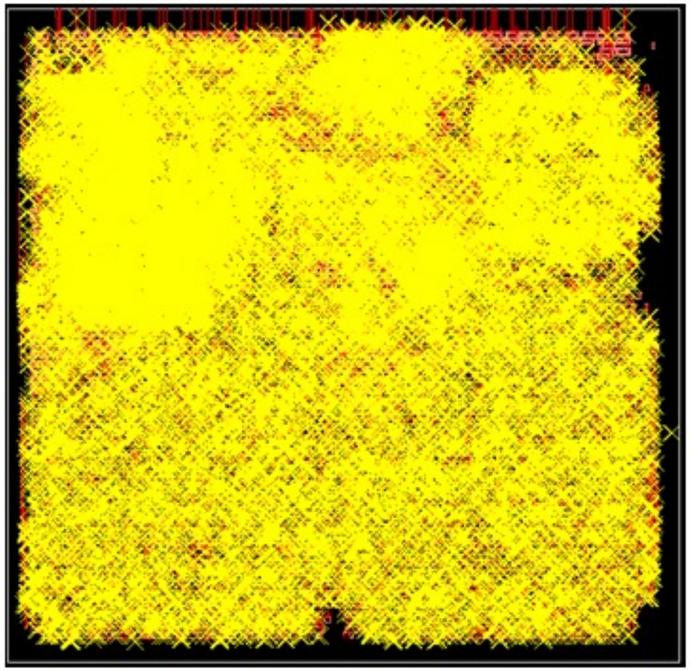
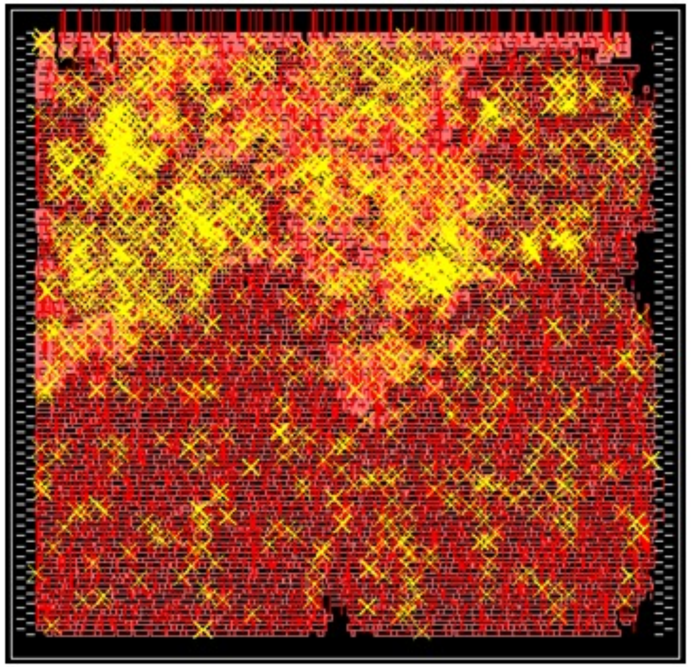


7 Test Cases

Cases	*#inst*	*#net*	*Process*
C1	68,472	12,752	65 nm
C2	1,558	1,775	40 nm
C3	8,971	11,210	40 nm
C4	2,221	407	40 nm
C5	9,984	1,837	40 nm
C6	9,984	1,837	28 nm
C7	11,894	11,210	28 nm

Design Rule Violation Prevention

C3

Original Router	**Original Router with MANA**
	
32,313 violations	**1,816 violations** **(reduce 94%)**

Experimental Results

- Average total runtime reduction
  - More than 2 times
- Wiring quality
  - Wirelenth and #vias are slightly reduced

	Original router				Router with MANA				Comparison		
Case	*time(sec)*	*W.L.(mm)*	*#via*	*#vio*	*time(sec)*	*W.L.(mm)*	*#via*	*#vio*	*t-r*	*w-r*	*v-r*
C1	293	237,084	142,541	0	150	230,178	138,384	0	2.0	1.03	1.03
C2	16	32,943	14,356	45	7	31.983	14,220	**0**	2.3	1.03	1.01
C3	148	111,359	90,504	0	75	108,115	88,742	0	2.0	1.03	1.02
C4	105	9,796	3,373	12	31	9,618	3,357	**0**	3.4	1.02	1.00
C5	492	27,294	17,201	0	101	26,500	17,045	0	4.9	1.03	1.01
C6	188	17,295	15,924	38	170	16,950	15,761	4	1.1	1.02	1.01
C7	209	88,563	74,553	0	183	85,951	74,625	0	1.1	1.03	1.00
			Success	**57%**			**Success**	**86%**	**Avg.** **2.4**	1.03	1.01

t-r = original router time/ MANA time

Contribution

- A shortest path algorithm under end-end separation and minimum length rules
  - Reduce 94% violations
  - Polynomial time complexity
    - Equal length path theory

Appendix: A*-Search Routing

- The maze search is also called **blind search** since it searches the routing region in a blind way.
- A*-search is also called the **best-first search**
  - uses function $f(x) = g(x) + h(x)$ to evaluate the cost of a path x
  - $g(x)$: the cost from the source to the current node of x
  - $h(x)$: the *estimated* cost from the current node of x to the target
- A*-search first searches the routes that are most likely to lead towards the target.
  - BFS is a special case of A*-search where $h(x) = 0$ for all x
- Good property:
  - if $h(x)$ is *admissible* (never overestimates the actual cost from the current node to the target), then A*-search is optimal

Reference

- “MANA: A Shortest Path MAze Algorithm under Separation and Minimum Length NAnometer Rules”, IEEE TCAD, Oct. 2013.