FOARS: FLUTE Based Obstacle-Avoiding Rectilinear Steiner Tree Construction

Gaurav Ajwani and Chris Chu

Iowa State University

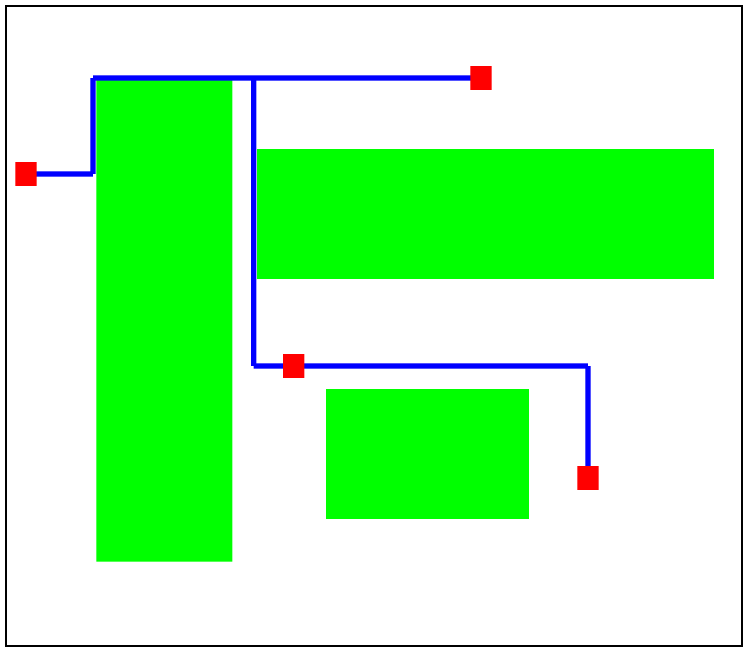
Wai-Kei Mak

National Tsing Hua University

OARSMT Problem Formulation

Obstacle-avoiding rectilinear Steiner minimal tree

- Input
  - A set of pins
  - A set of rectilinear obstacles
- Output
  - A rectilinear Steiner tree
    - Minimizing total wirelength
    - Connecting all pins
    - Avoiding all obstacles

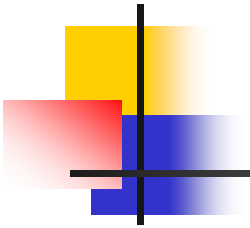


- Applications in routing, wirelength estimation, etc.
- NP-complete
- More than ten heuristics proposed since 2005

Our Contributions

- An OARSMT algorithm called FOARS (FLUTE Based Obstacle-Avoiding Rectilinear Steiner Tree)
  - Outstanding wirelength
  - Efficient
  - Scalable: O(n log n) time
    - where n = # pins + # obstacle corners

- New Ideas:
  - Approach to leverage FLUTE for OARSMT construction
  - An efficient obstacle-aware partitioning technique
  - Algorithm to construct obstacle-avoiding spanning graph with good properties



If There Is No Obstacle

- Rectilinear Steiner Minimal Tree (RSMT) problem
- FLUTE -- Fast LookUp Table Estimation [TCAD 08]
  - Extremely fast and accurate
    - More accurate than BI1S heuristic
    - Almost as fast as minimum spanning tree construction
- Can we leverage FLUTE for OARSMT construction?

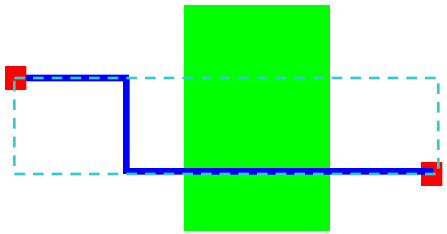
Obstacle-Aware FLUTE (OA-FLUTE)

- // P = set of pins, OB = set of obstacles
  Function OA-FLUTE(P, OB)

  T = FLUTE(P) // ignore obstacles
  If (T overlaps with obstacle)
  Partition into several sub-problems $P_1,\ldots,P_t$
  T = OA-FLUTE(P_1, OB) + ... + OA-FLUTE(P_t, OB)
  Return T

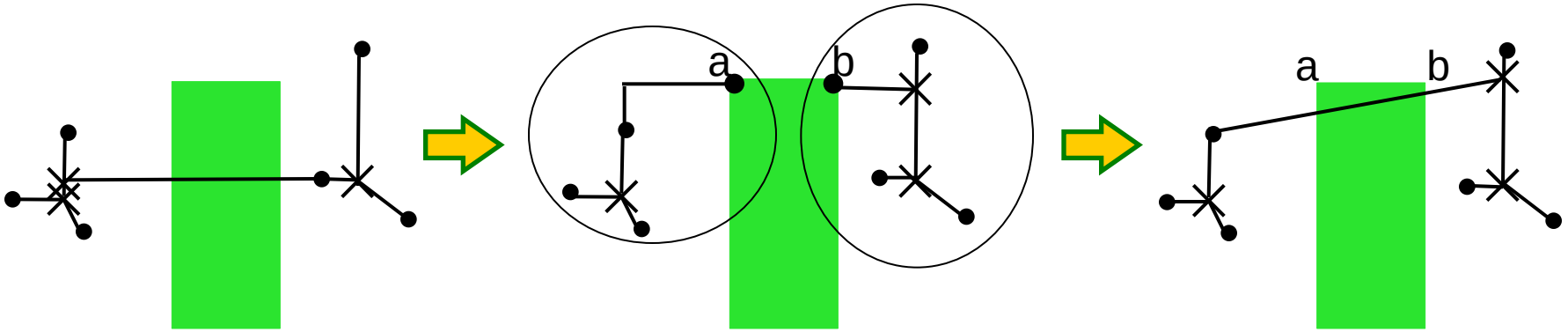
- Two possible types of overlap:

1. An edge is *completely blocked* by an obstacle



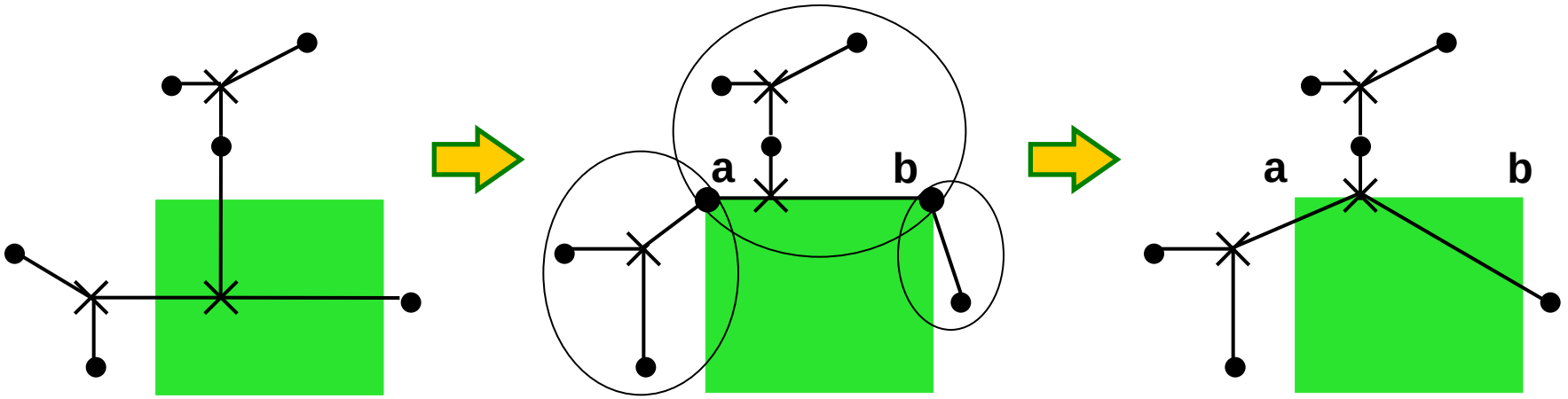
2. A Steiner node is on top of an obstacle

Type 1: Edge over Obstacle



- Partition pins according to the overlapping edge
  - Include obstacle corners
- Apply OA-FLUTE recursively on sub-problems to obtain sub-trees
- Merge sub-trees and exclude corners

Type 2: Steiner Node over Obstacle

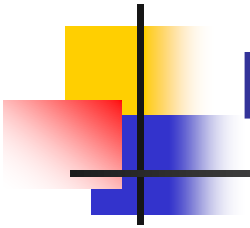


- Partition pins according to the overlapping Steiner node
  - Include obstacle corners
- Apply OA-FLUTE recursively on sub-problems to obtain sub-trees
- Merge sub-trees and exclude corners

Problems with OA-FLUTE

- Does not work well if:
  1. Routing region is too cluttered by obstacles
     Reason: Partitioning based on initial tree which ignores obstacles
  2. There are too many pins
     Reason: Performance of FLUTE starts to deteriorate for more than a hundred pins

- Need a better way to partition the pins
- Then OA-FLUTE can be called to handle each sub-problem

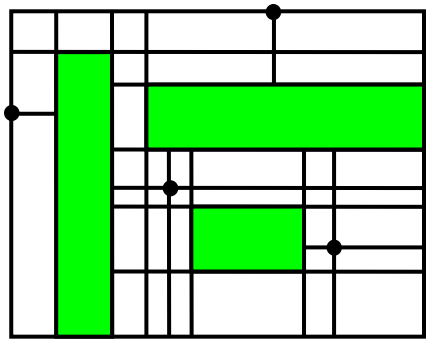


FOARS Overview

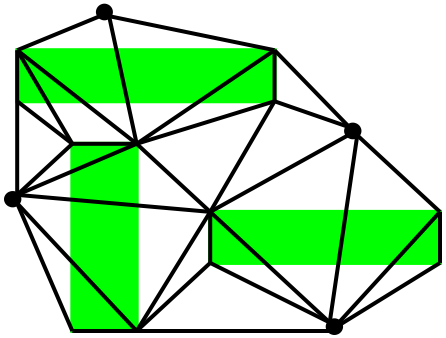
1. Partitioning Pins
   - Obstacle-Avoiding Spanning Graph (OASG)
   - Minimum Terminal Spanning Tree (MTST)
   - Obstacle Penalized Minimum Spanning Tree (OPMST)
   - Partition according to OPMST to obtain sub-problems
2. Fixing tree topology and Steiner node locations
   - Applying OA-FLUTE to Sub-problems
3. Routing edges between Steiner nodes / pins
   - Rectilinearize edges to create OARSMT
   - V-shape refinement

Connection Graphs

- To capture the proximity information amongst pins and obstacle corners
- Previous connection graphs:
  - Escape Graph (Ganley et al. [ISCAS 94])
    - $O(n^2)$ edges



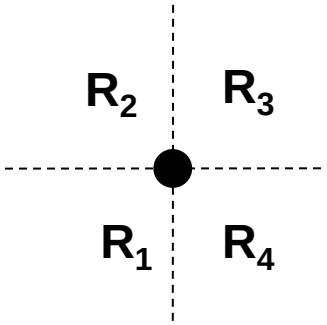
  - Delaunay Triangulation
    - $O(n^2)$ edges

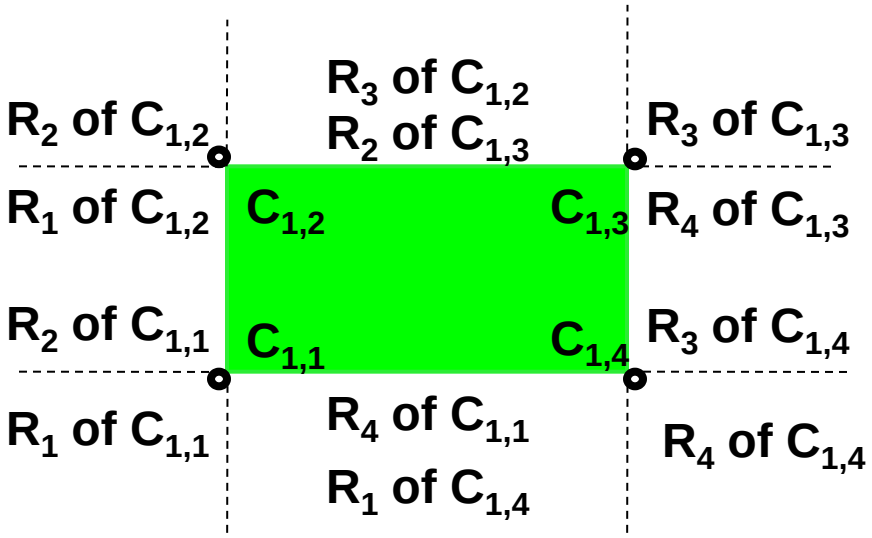


  - Obstacle-Avoiding Spanning Graph (OASG)
    - Extension of spanning graph (Zhou et al. [ASPDAC 01])
    - O(n) edges

Problem with Previous OASG

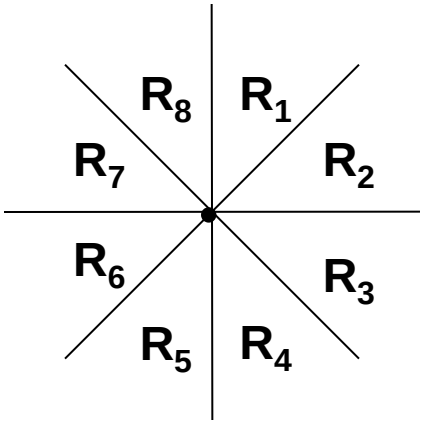
- Previous OASG Approaches:
  - Shen et al. [ICCD 05]
  - Lin et al. [ISPD 07]
    - Adding "essential edges" ➔ $O(n^2)$ edges
  - Long et al. [ISPD 08]
- All considered quadrant partition
  - May not contain RMST even in the absence of obstacle

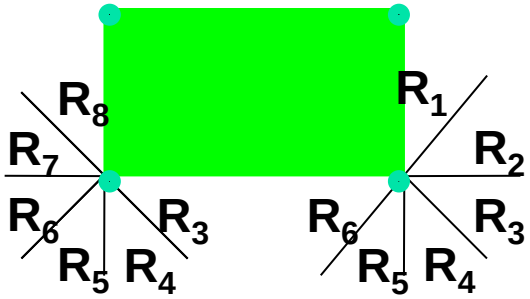




Our OASG Approach

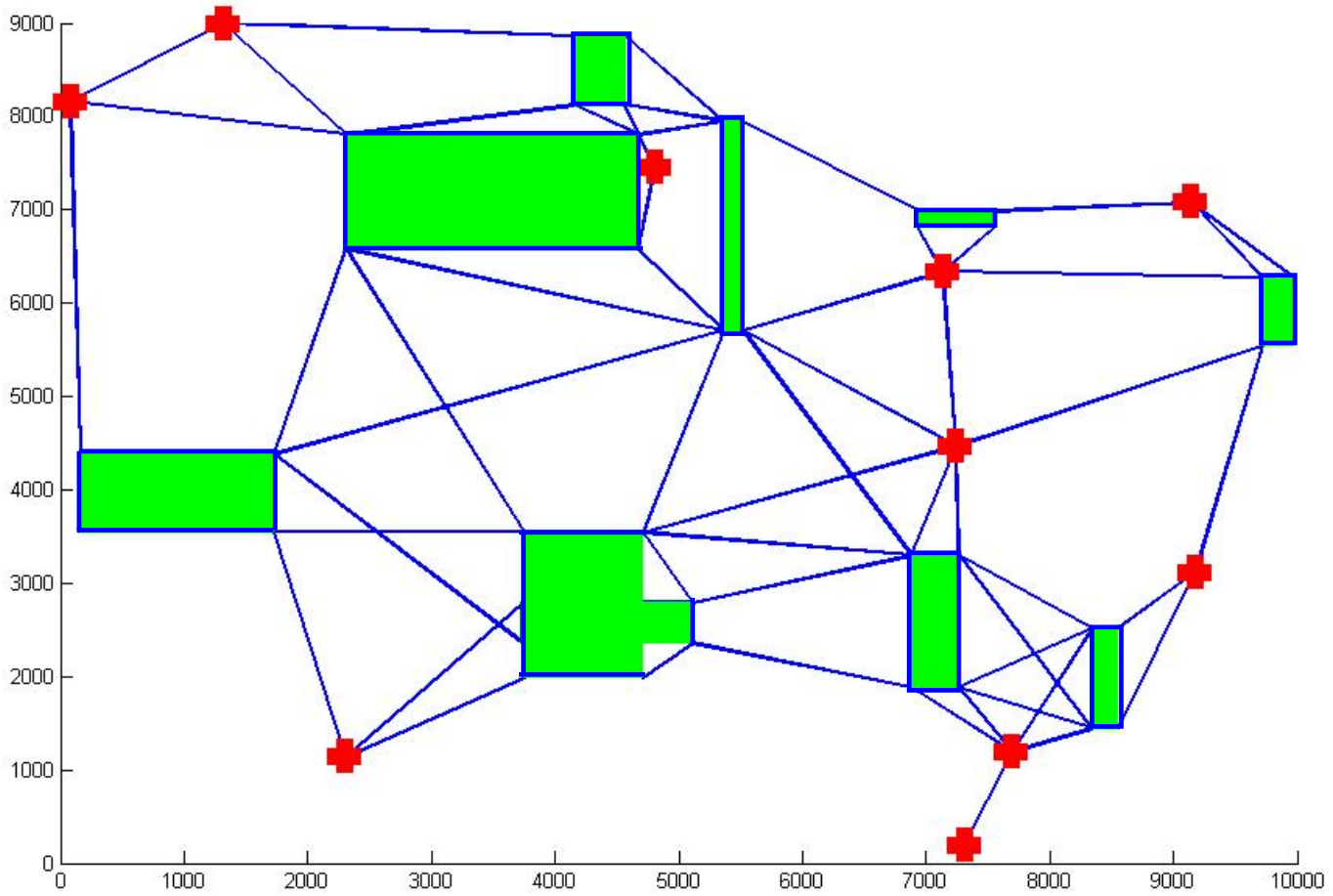
- Generalization of Zhou's Approach
  - If no obstacle, same as Zhou's original algorithm, i.e., presence of RMST guaranteed
- Octant partition





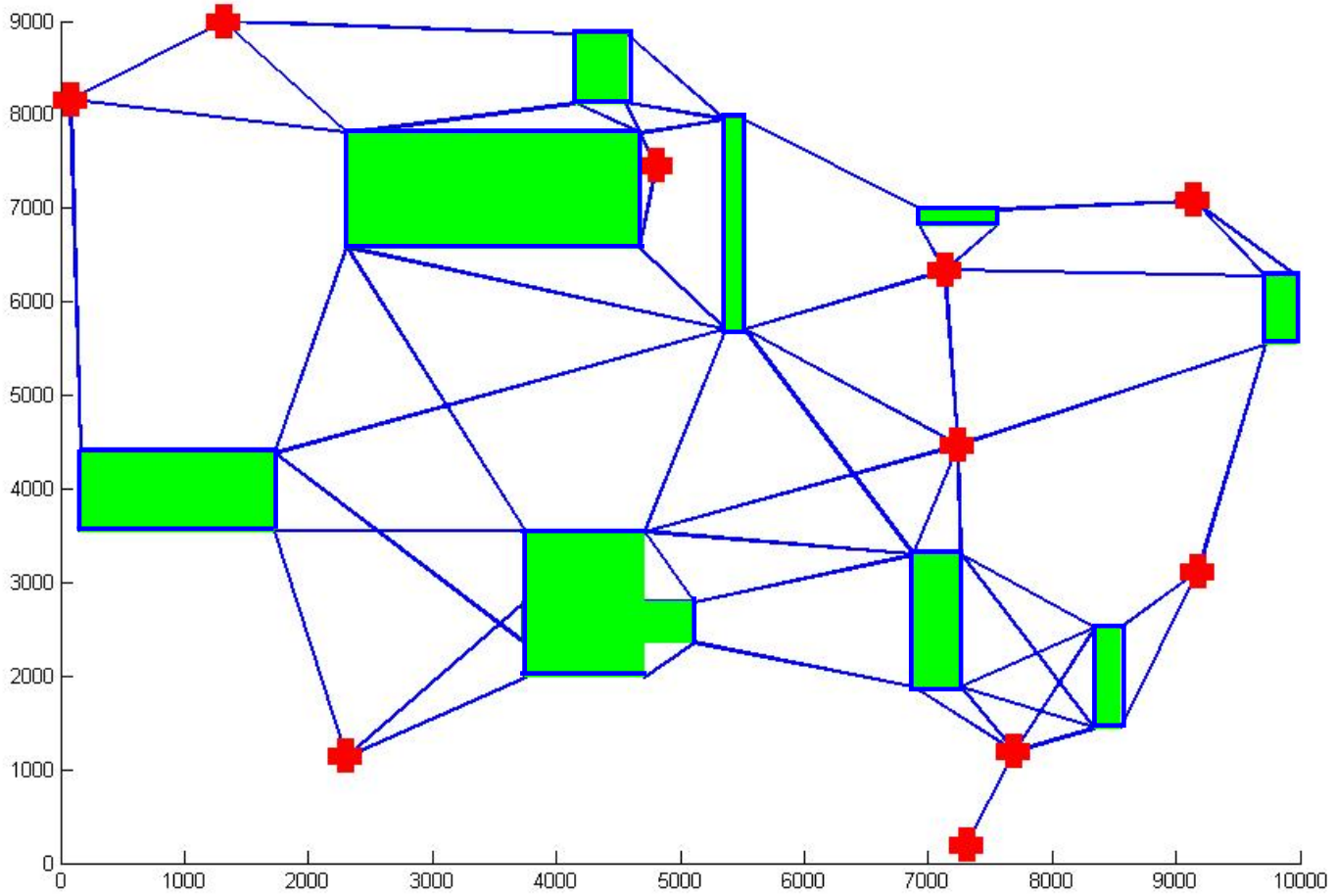
- O(n) edges

OASG Example



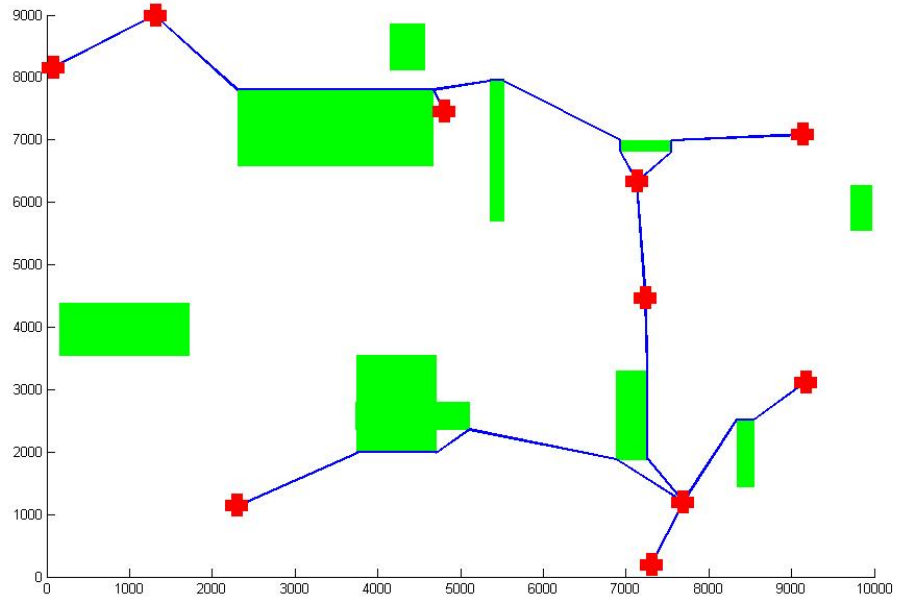
Minimum Terminal Spanning Tree (MTST)

- Use the technique proposed by Wu et al. [ACTA INFORMATICA 86]

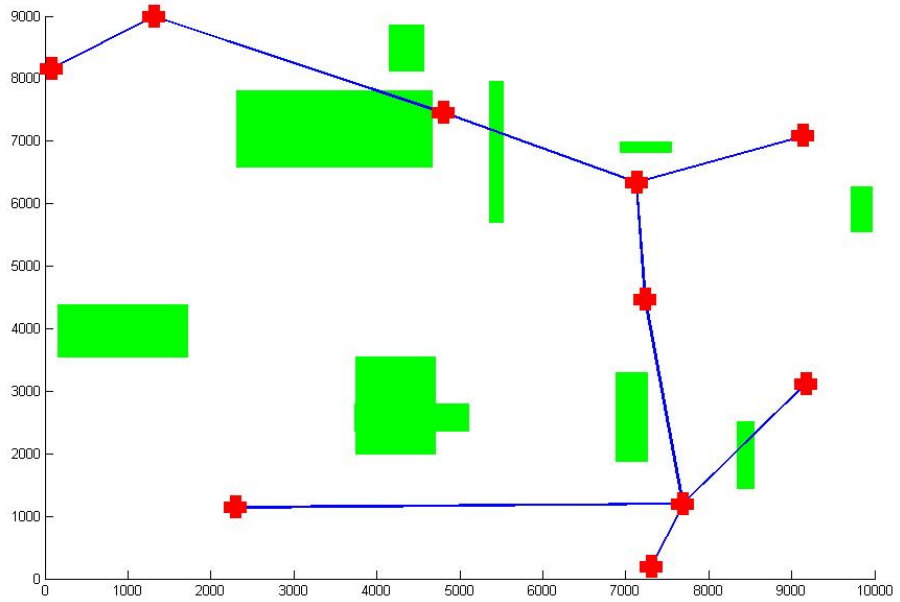


Obstacle Penalized Minimum Spanning Tree

MTST

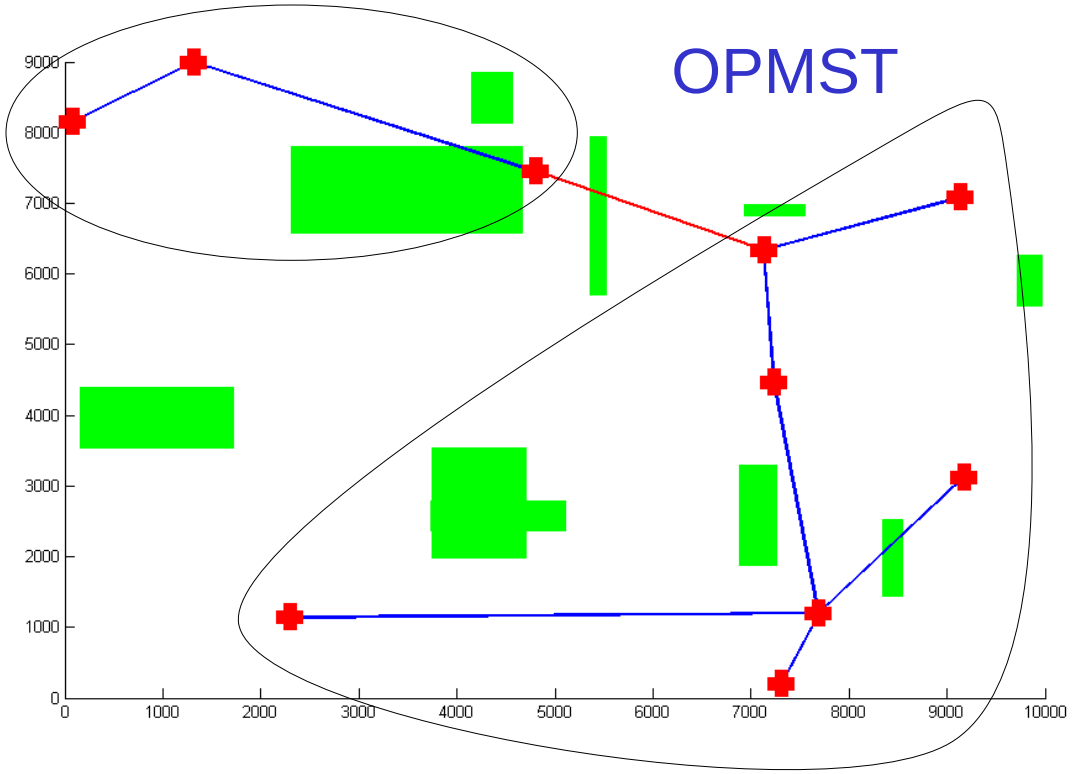


OPMST



Edge weight
= Wirelength considering detour

Partitioning Pins

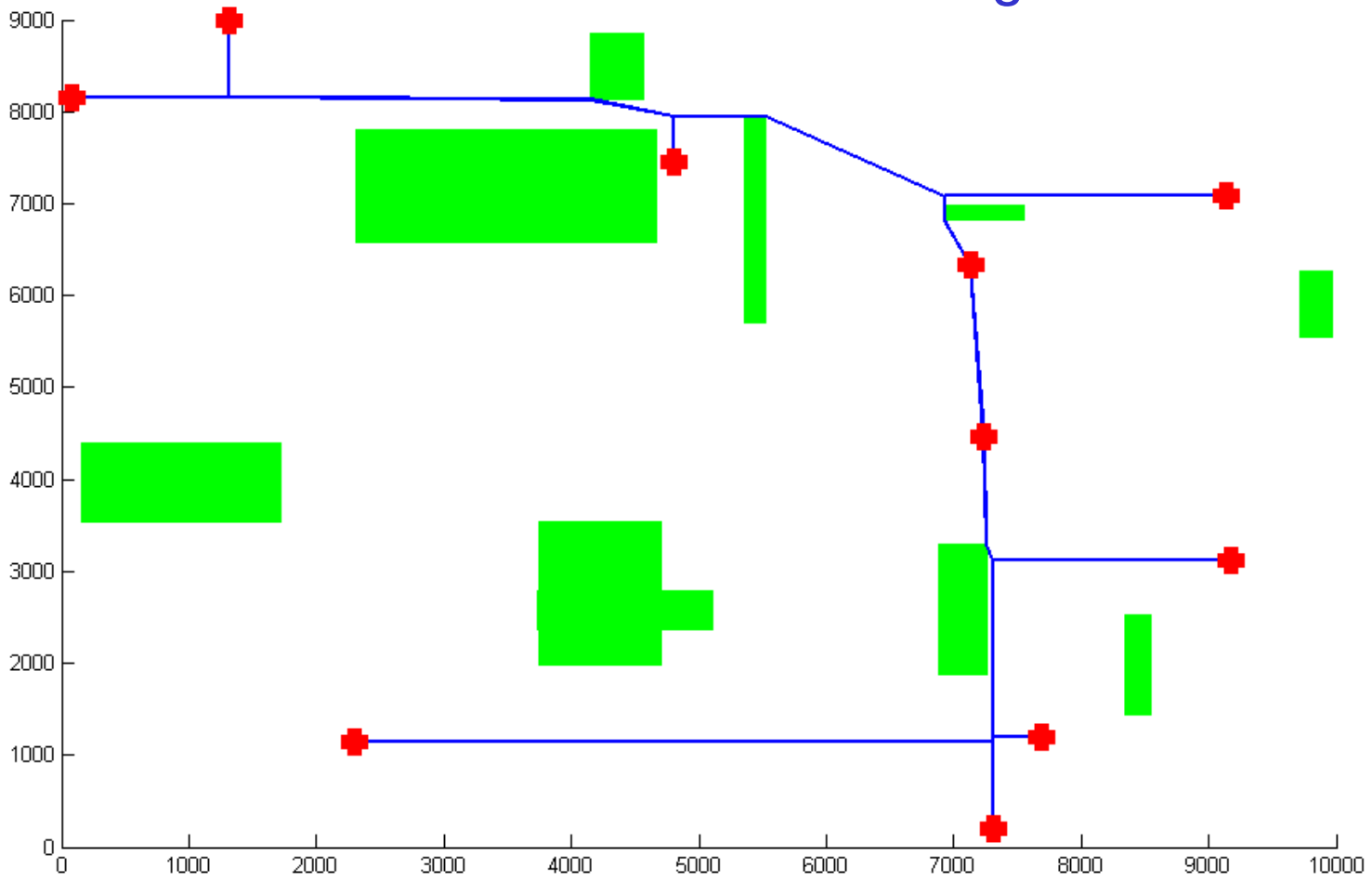


Apply OA-FLUTE to each sub-problem

- Partition:
  1. If an edge is completely blocked by an obstacle
  2. If # pins in sub-tree > 20

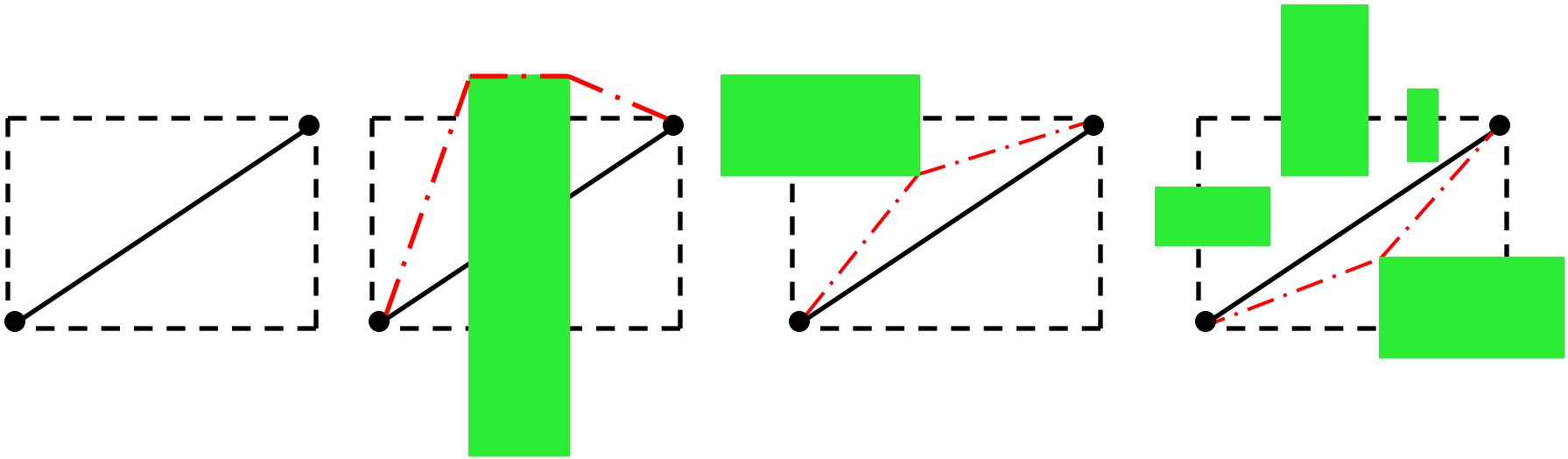
Tree After OA-FLUTE

Wirelength 25980



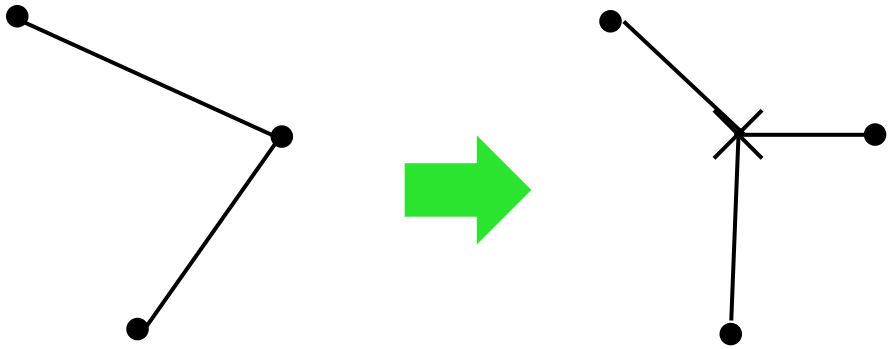
Rectilinearization of Slanted Edges

- Four possible cases for any slanted edge
  1. Both L-shape paths are obstacle free
  2. Both L-shape paths are blocked by one obstacle
  3. One L-shape path is blocked and other is free
  4. Both L-shape path are blocked but by different obstacles



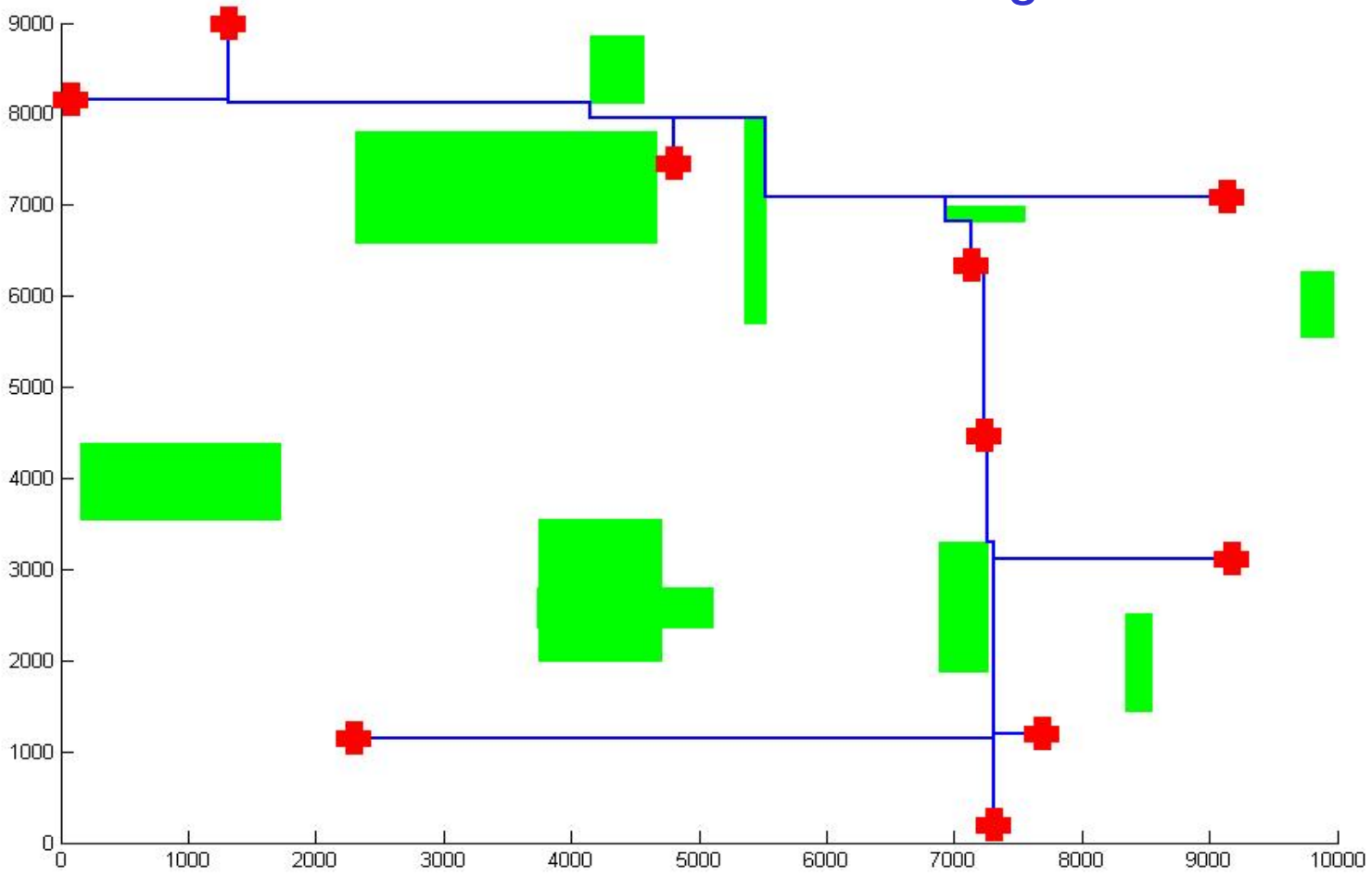
V-Shape Refinement

- Replace any two adjacent edges with a Steiner tree
- Improve wirelength by 1-2%



After Rectilinearization & Refinement

Wirelength 25290



Experimental Results

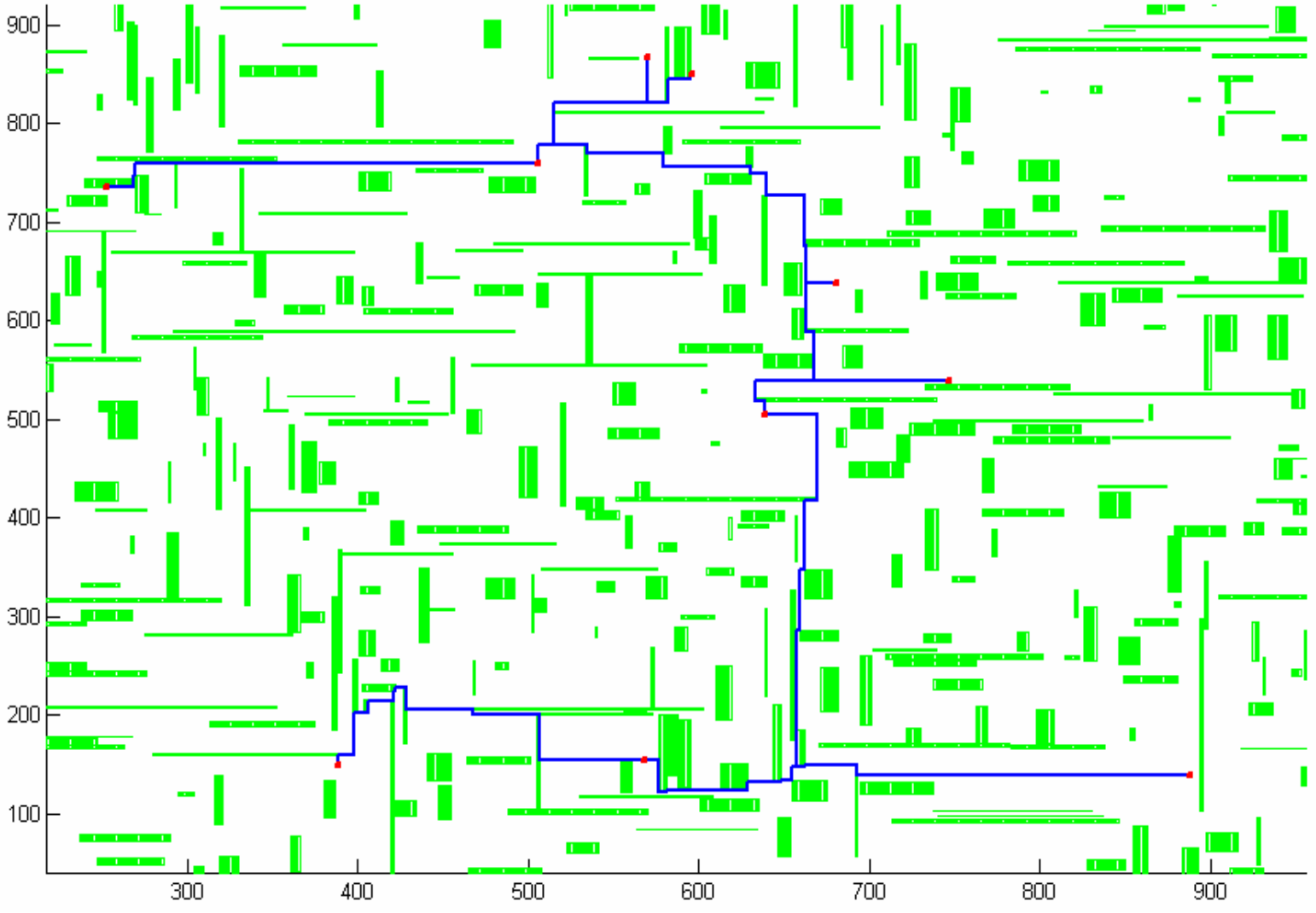
- Algorithm implemented in C
- Comparison with latest binaries from:
  - Lin et al. [ISPD 07]
  - Long et al. [ISPD 08]
  - Li et al. [ICCAD 08]
- All experiments were performed on a 3GHz AMD Athlon 64 X2 Dual Core Machine (use only 1 core)
- Four sets of benchmarks, 27 benchmark circuits
  - RC01-RC12: randomly generated by Feng et al. [ISPD 06]
  - RT01-RT05: randomly generated by Lin et al. [ISPD 07]
  - IND1-IND5: Synopsys industrial testcases from Synopsys in Lin et al. [ISPD 07]
  - RL01-RL05: larger testcases randomly generated by Long et al. [ISPD 08]

Wirelength and Runtime Comparison

	Lin et al. ISPD 07	Long et al. ISPD 08	Li et al. ICCAD 08	FOARS
Normalized Wirelength	1.023	1.027	0.995	1
Normalized Runtime	78.45	1.20	29.36	1

OARSMT for RT10

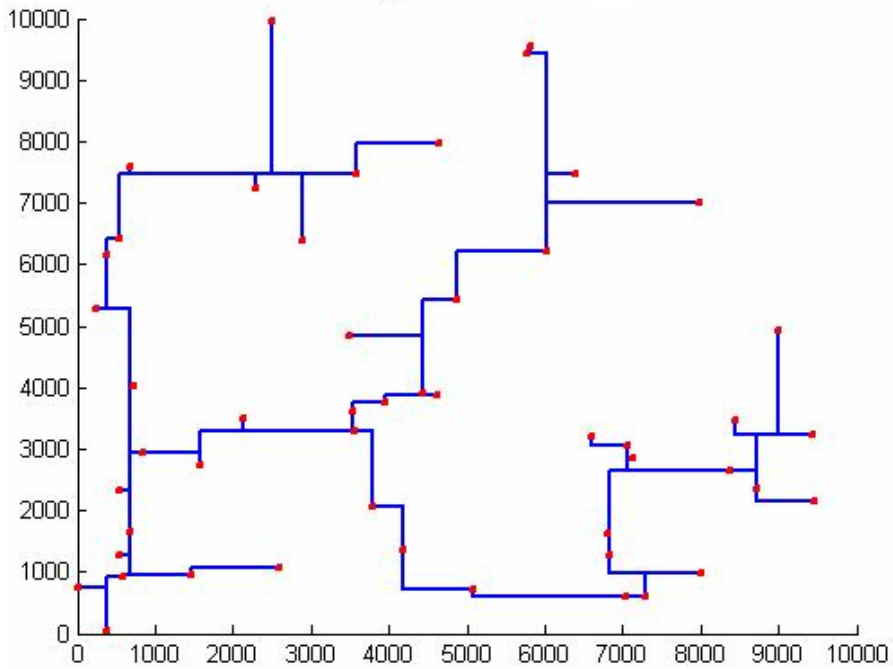
10 Pins, 500 Obstacles



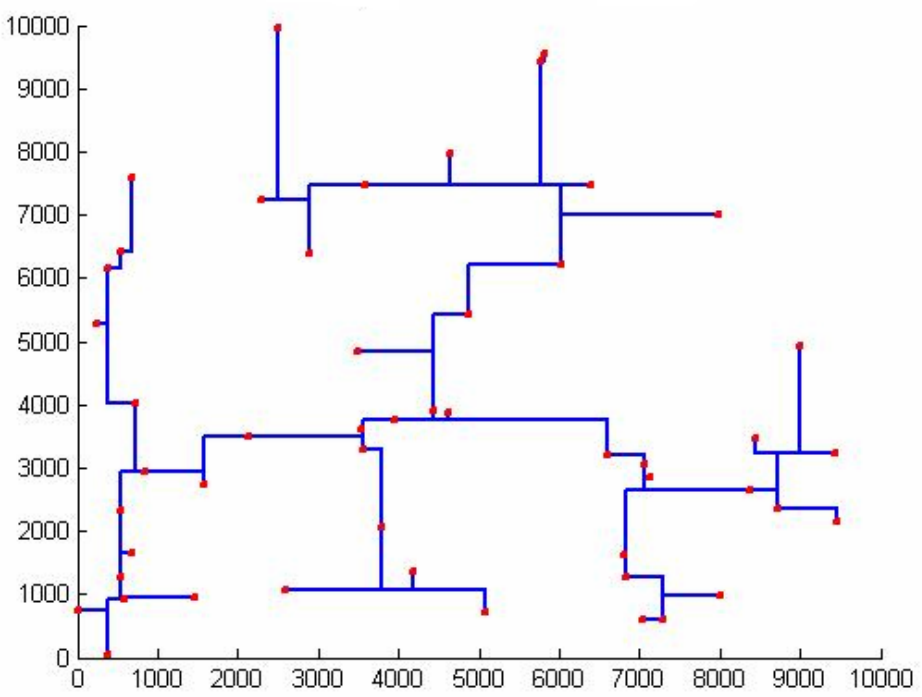
Obstacle-Free Testcase

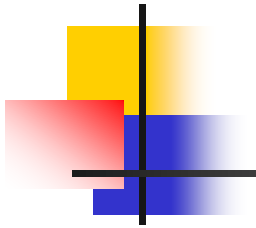
RC03 without obstacles, 50 Pins

FOARS Wirelength: 53050



FLUTE-2.5 Wirelength: 53400





THANK YOU