An *O*(*nlogn*) Algorithm for Obstacle-Avoiding Routing Tree Construction in the λ-Geometry Plane

Zhe Feng[1], Yu Hu[2], Tong Jing[1], Xianlong Hong[1], Xiaodong Hu[3], Guiying Yan[3]

[1] CST Department
[1] Tsinghua University
Beijing 100084, China

[2] EE Department
[2] UCLA
LA, CA 90095, USA

[3] Institute of Applied Mathematics
[3] Chinese Academy of Science
Beijing 100080, China

Speaker: Zhe Feng

Outline

- Introduction
- Previous work
- Our algorithm
- Experimental results & discussions
- Conclusions

Outline

- Introduction
- Previous work
- Our algorithm
- Experimental results & discussions
- Conclusions

Introduction

- Routing plays an important role in very/ultra large scale integrated circuit (VLSI/ULSI) physical design.
- Obstacle-avoiding Rectilinear Steiner minimal tree (OARSMT) construction is often used as the estimation for wire length and delay throughout the process of routing.
- No polynomial-time algorithm can solve OARSMT problem exactly.
- Most attention has been devoted to λ - geometry routing recently.

Outline

- Introduction
- Previous work
- Our algorithm
- Experimental results & discussions
- Conclusions

Previous work

- Complexity according to the size of routing area
  - Maze routing [C.Y.Lee, 1961; F.Rubin, 1974]
  - Line search routing [Mikami K. And Tabuchi K. 1968; D.W.Hightower, 1969.]
- Complexity according to the size of cases
  - G3S, B3S, and G4S heuristics [J. L. Ganley and J. P. Cohoon, 1994]
  - Based on generalized connection graph [Zhou *et al*, 2003]
  - 2-step heuristic [Y. Yang, Q. Zhu, T. Jing, X. L. Hong *et al*, 2003]
  - FORst [Y. Hu, Z. Feng, T. Jing, X.L. Hong *et al*, JICS2004]
  - An-OARSMan [Y. Hu, T. Jing, X.L. Hong, Z. Feng *et al*, 2005]
  - CDCTree [Y.Y.Shi, T.Jing, L.He *et al*, 2006]

- The solution quality of most existing algorithms of the construction of OARSMT is not good enough.
- The running time is very long, so the capability is limited while the nets and circuits are in large scale.
- The algorithm of the construction of OASMT for λ -geometry is needed, which is already supported by industry.
- We need an efficient λ -OASMT algorithm to get better routing solution quality in a shorter running time.

Outline

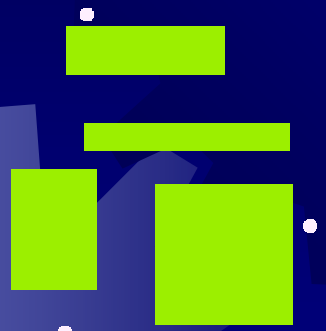
- Introduction
- Previous work
- **Our algorithm**
- Experimental results & discussions
- Conclusions

Primary contribution of this paper

- An $O(n\log n)$ algorithm for OARSMT construction is proposed.
- We extend the algorithm to handle obstacle-avoiding Steiner minimal tree construction problem in λ-geometry plane.
- This is the first work addressing the problem of obstacle-avoiding Steiner minimal tree construction in the λ-geometry plane (λ-OASMT).

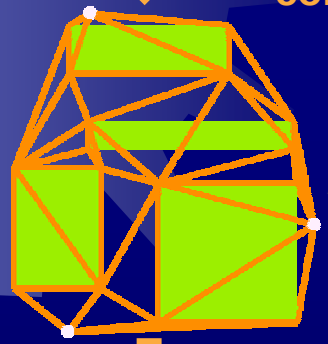
Our algorithm

Step1
Graph construction



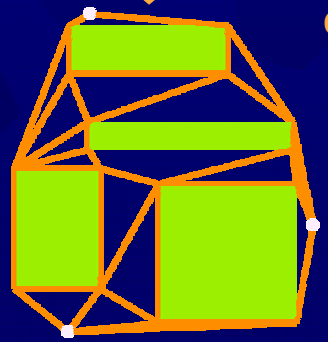
Input: terminals and obstacles

Delaunay triangulation construction



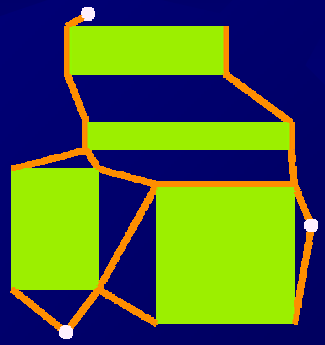
Delaunay triangulation (DT)

Delete intersecting edges



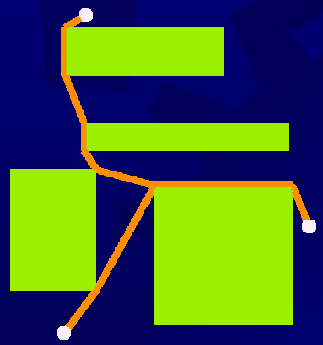
Obstacle-avoiding constrained Delaunay triangulation (OACDT)

Step2
Tree construction



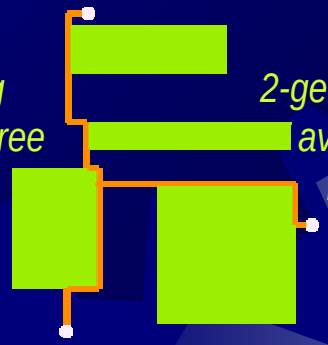
Obstacle-avoiding Minimum spanning tree (OAMST)

Reduction



Full connective tree (FCT)

Step3
Embedding

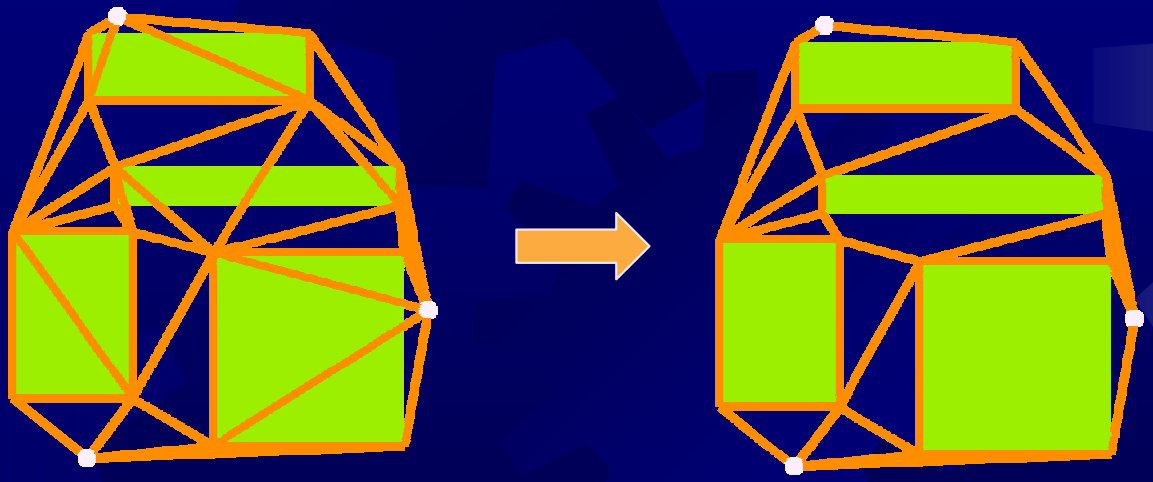


2-geometry obstacle-avoiding Steiner minimal tree (2-OASMT)

4-geometry obstacle avoiding Steiner minimal tree (4-OASMT)

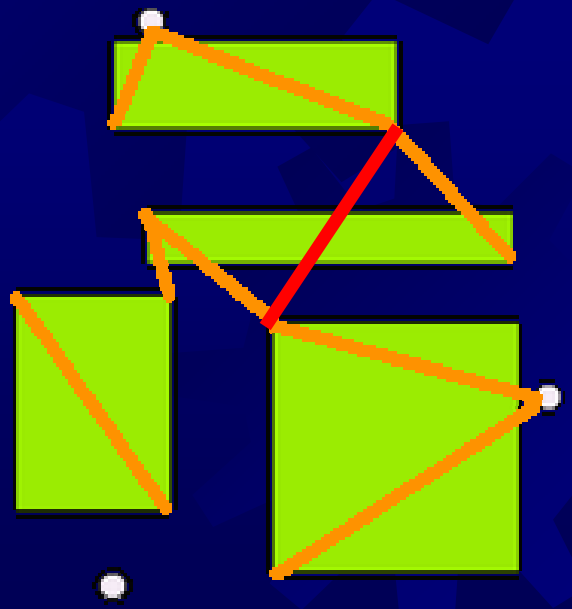
Our algorithm

- **Step 1(1)**
  - We construct a Delaunay triangulation (DT)
  - To delete the edges intersecting with the obstacle, we transform the DT into an Obstacle-avoiding constrained DT by a novel technique, namely *edge deleting*, which is independent of geometry.



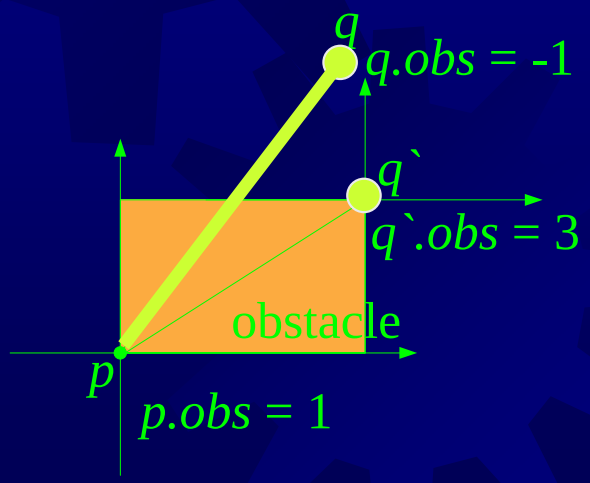
Our algorithm

- **Step 1(2)**
  - **Edge deleting**
    - case 1: one end point of the edge is also a corner point of the intersecting obstacle.
    - case 2: both end points of the edge are non-corner points of the intersecting obstacle.



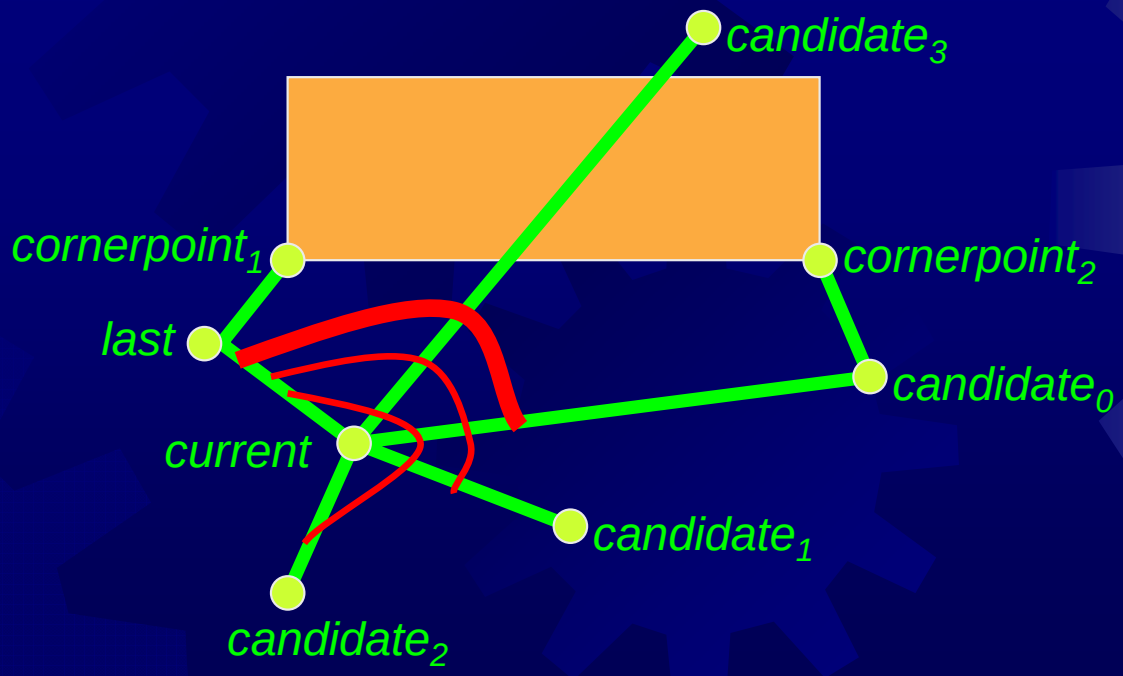
Our algorithm

- Step1(3)
  - For the first case
    - We maintain a domain *obs* for each point, which indicates whether the point is a terminal or a corner point of an obstacle.
    - After that, each edge will be checked to know whether it intersects with the obstacle. If so, the intersecting edge will be deleted.



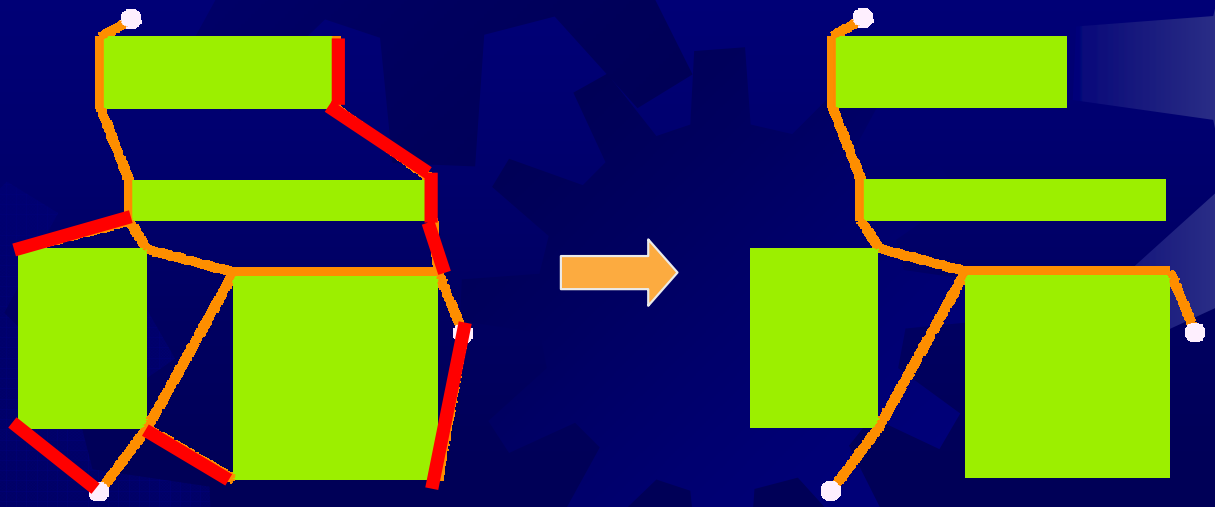
Our algorithm

- Step1(4)
  - For the second case
    - For each obstacle, we construct the shortest path connecting two corner points of each edge.
    - Then, we delete the edges which connect with the shortest path, while they also intersect with the obstacle.



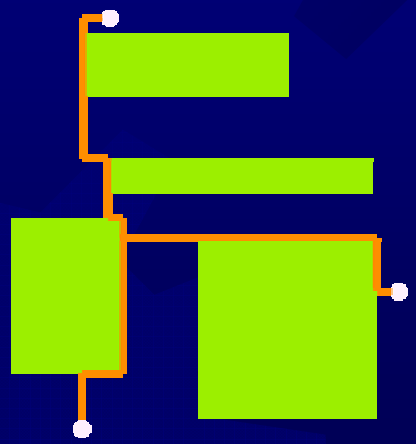
Our algorithm

- **Step 2**
  - An obstacle-avoiding minimum spanning tree (OAMST) is generated on OACDT by Prim algorithm.
  - We delete all the non-terminal leaves of the OAMST to generate the FCT.
  - Both the two stages are independent of geometry.



Our algorithm

- **Step 3(1)**
  - We embed an FCT into a λ-OASMT according to their geometry. Based on a novel method, namely *zonal combination*.
  - Firstly, we traverse the FCT. For each node, we allocated all its child nodes into the phases according to their relative position to it.
  - Secondly, we choose the phase including the already constructed edge as a start-up. Then, the rest phases are handled by the counter-clockwise order. In each phase, all the child nodes are connected to their parent node by new edges.

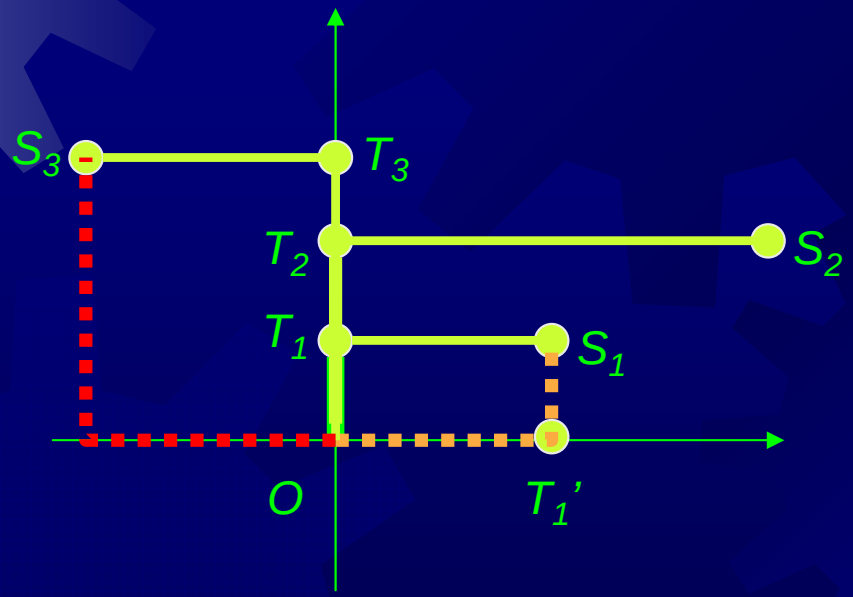


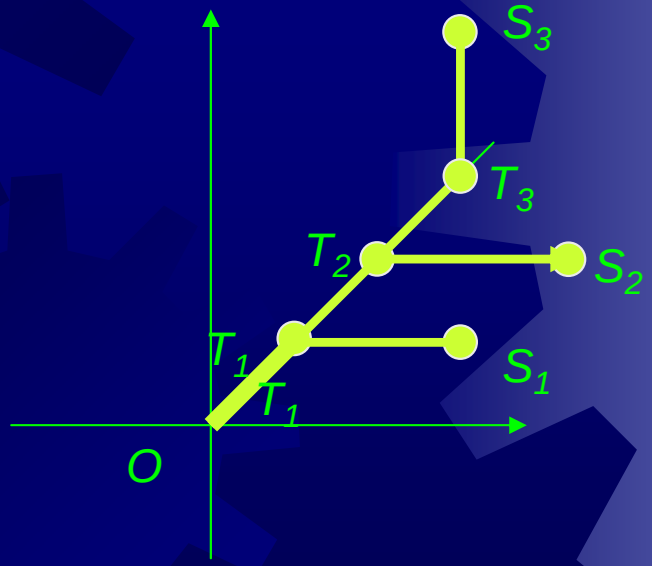
2-OASMT

4-OASMT

Our algorithm

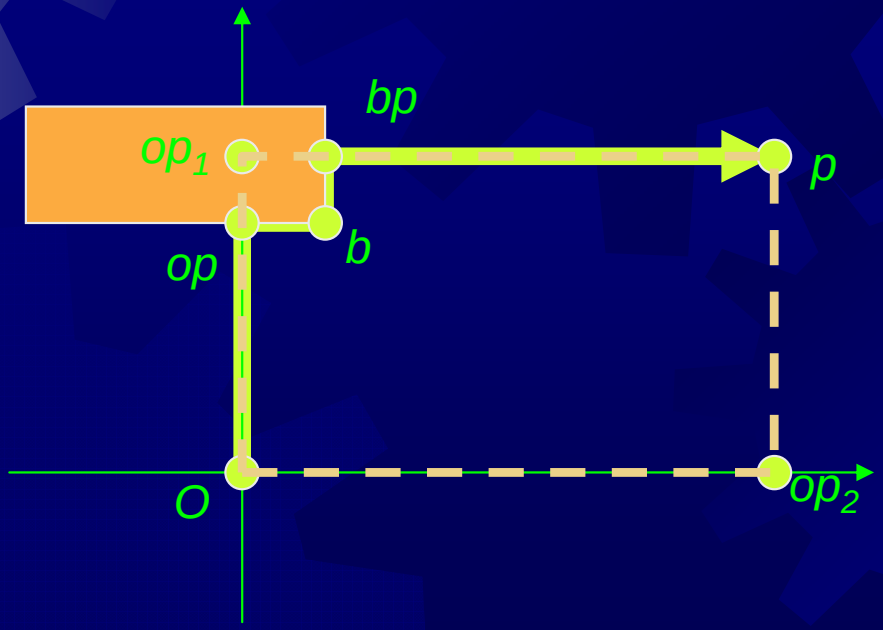
- Step3(2)
  - How to make full use of shared edges to keep the wire length of the λ-OASMT as short as possible? There are two cases.
    - The child nodes to be connected to the node are in the same phase.
    - The child nodes are in the neighbor phases.

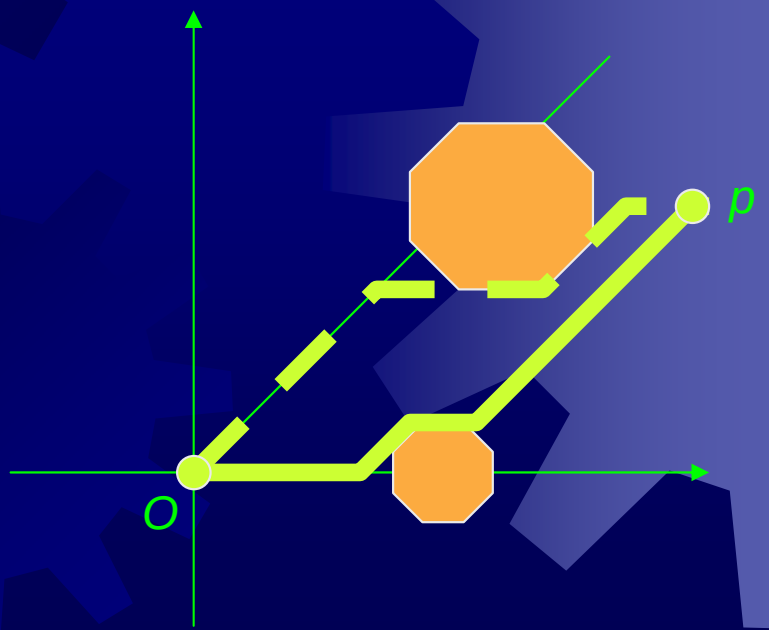




Our algorithm

- Step3(3)
  - How to get new edges connecting the point and its child nodes avoiding obstacles?
    - Firstly, we have to find a bent.
    - Secondly, we have to avoid the bent by some technique.
  - Definition(Bent)
    - During the process of embedding, if a new generated path intersects with a boundary of an obstacle, the path has to be altered in order to avoid the obstacle. The event is called bent. The new point, added into the path to alter it, is also called bent.





Outline

- Introduction
- Previous work
- Our algorithm
- Experimental results & discussions
- Conclusions

Experimental results & discussions(1)

- Experiment environments:
  - Hardware: Sun V880 fire workstation (755MHz CPU and 4GB memory)
  - Software: gcc2.9.1, Solaris5.8
- Benchmark data:
  - We randomly generate some terminals among rectilinear polygon obstacles in a 32767*32767 plane.
  - Compare our algorithm with typical existing algorithms: FORst, An-OARSMan

Experimental results & discussions(2)

- Comparison on wire length
  - FORst [1]
  - An-OARSMan [2]
  - 2-OASMT
  - 4-OASMT

Term #	Obs#	Wire Length						
		[1]	[2]	2-OASMT		4-OASMT		
		Wire length	Wire length	Wire length	Improve to [1]/[2] (%)	Wire length	Improve to [1]/[2] (%)	Improve to 2-OASMT (%)
10	10	29630	27840	30410	-9.23	27279	2.02	10.30
30	10	56880	43090	45640	-5.92	41222	4.34	9.68
50	10	64020	63250	58570	7.34	52432	17.10	10.48
70	10	69450	66310	63340	4.48	57699	12.99	8.91
100	10	84590	82320	83150	-1.01	73090	11.21	12.10
500	100	223415	-	198010	11.37	176497	21.00	10.86
1000	100	285821	-	250570	12.33	222758	22.06	11.10

Experimental results & discussions(3)

- Comparison on running time
  - FORst [1]
  - An-OARSMan [2]
  - 2-OASMT
  - 4-OASMT

Term#	Obs#	Runtime (s)					
		[1]	[2]	2-OASMT		4-OASMT	
		Time	Time	Time	Speedup	Time	Speedup
10	10	0.095	0.164	0.002	47x	0.003	32x
30	10	0.584	1.075	0.003	195x	0.003	195x
50	10	0.868	3.504	0.004	217x	0.004	217x
70	10	2.759	10.552	0.004	690x	0.005	552x
100	10	6.312	26.974	0.004	1578x	0.005	1262x
500	100	559.000	-	0.026	21500x	0.033	16939x
1000	100	1240.00	-	0.037	33514x	0.045	27556x
					Average speedup 8249x		

Experimental results & discussions(4)

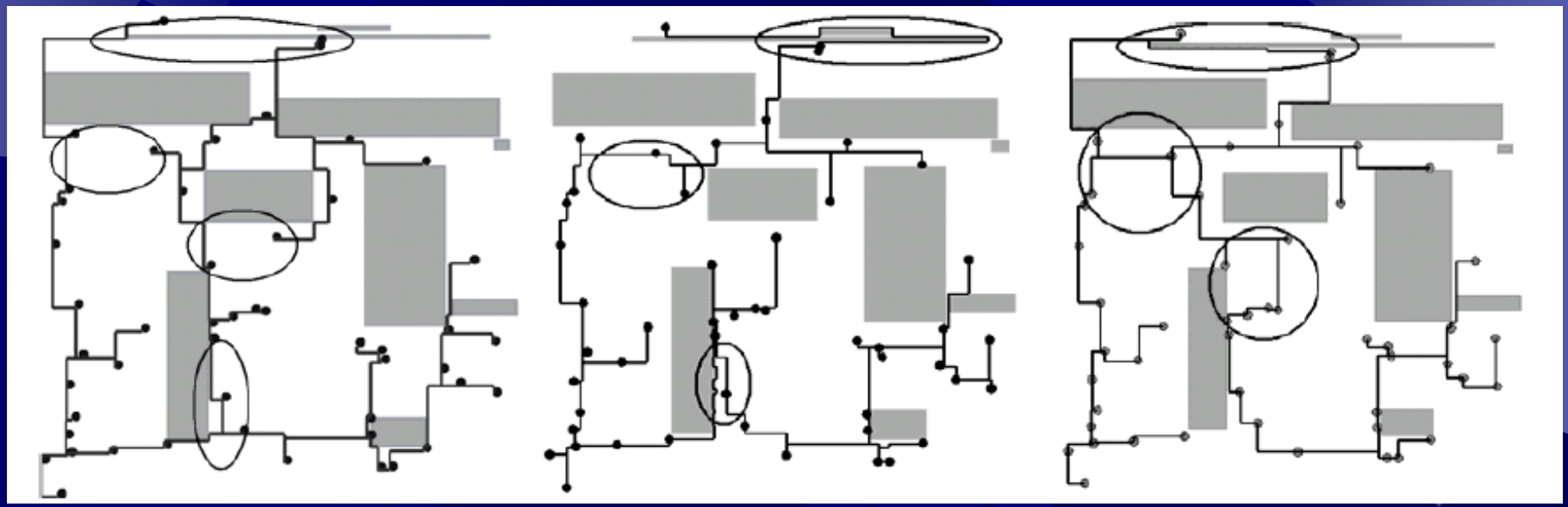
Testing on cases with more obstacles

Term#	Obs#	Wire length		Runtime (*s*)	
		2-OASMT	4-OASMT	2-OASMT	4-OASMT
100	500	149725	135454	0.057	0.070
200	500	181470	162762	0.062	0.075
200	800	202741	182056	0.095	0.119
200	1000	214850	193228	0.129	0.154
1000	10000	1723990	1564170	2.823	3.030

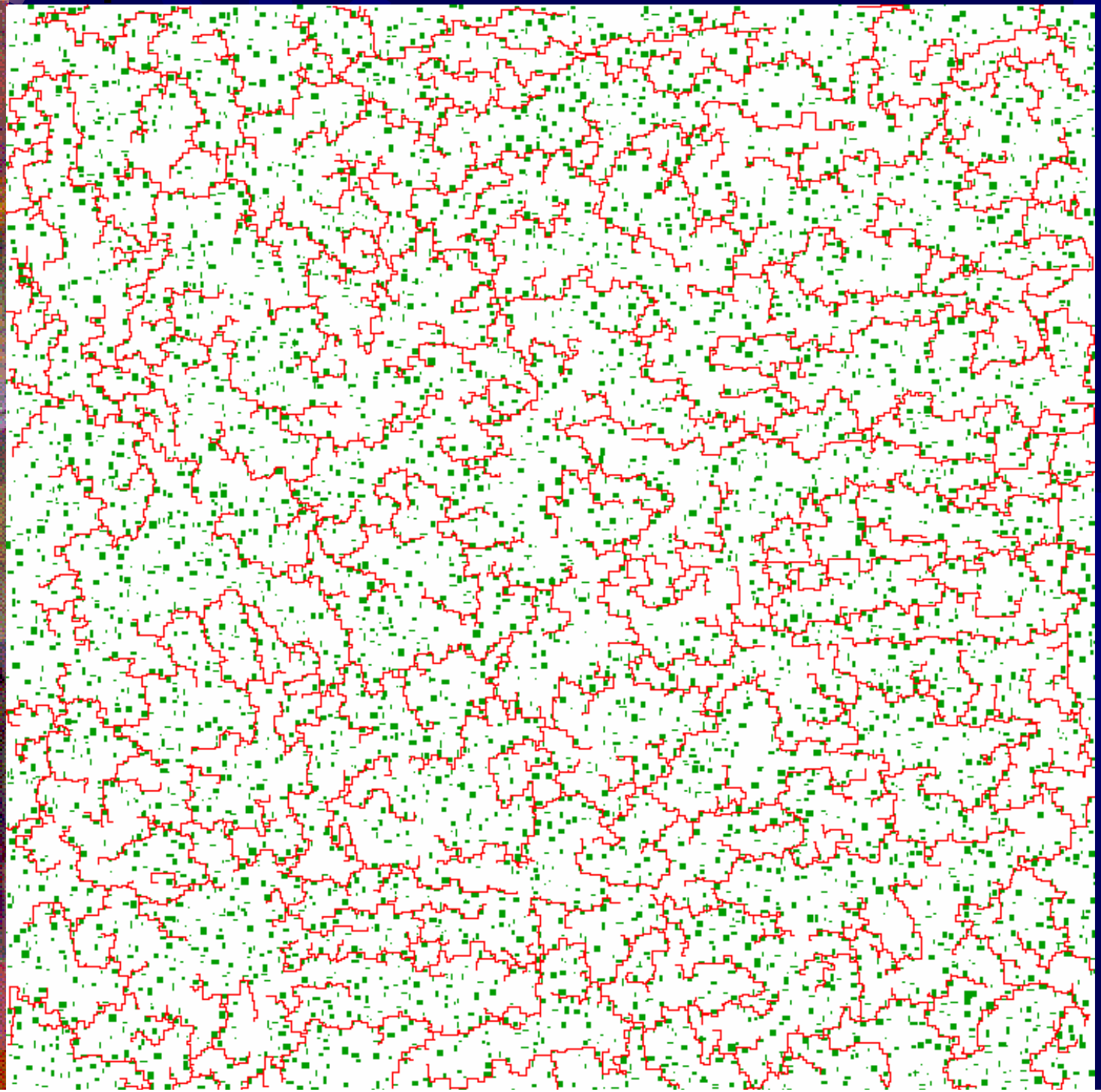
Experimental results & discussions(5)

- Show the topology differences on the test case with 50 terminals and 10 obstacles.
  - 2-OASMT
  - FORst
  - An-OARSMan

2-OASMT FORst An-OARSMan



Experimental results & discussions(6)



1000 terminals in the presence of 10000 rectangular obstacles in 2-geometry

Outline

- Introduction
- Previous work
- Our algorithm
- Experimental results & discussions
- **Conclusions**

Conclusions

- An $O(n\log n)$ algorithm for obstacle-avoiding Steiner minimal tree construction in the λ-geometry plane (λ-OASMT) is proposed.
- The run time is good enough without the loss of the solution quality. Compared with two most recent works on the OARSMT problem, λ -OASMT obtains up to 30Kx speedup with an even better solution quality.
- The algorithm still works well for the nets and circuits in large scale. We tested the randomly generated cases with up to 1K terminals and 10K rectilinear obstacles within 3 seconds on a Sun V880 workstation (755MHz CPU and 4GB memory).
- The high efficiency and accuracy of λ-OASMT make it practical in the routing phase, as well as interconnect estimation in the process of floorplanning and placement.


THANK YOU