CAD Contest Problem D
Chip Level Global Router

Group 14

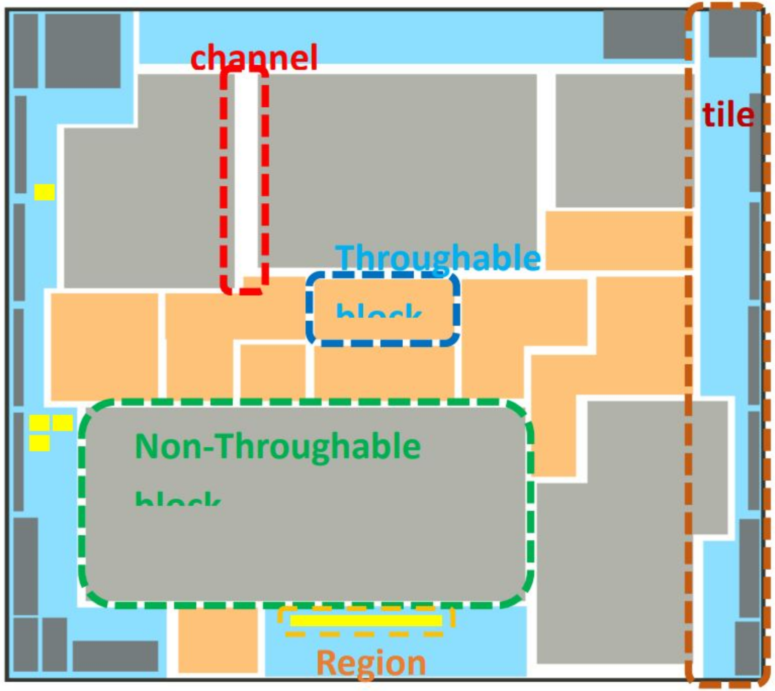
李致頡　霍芷媛　林雋哲

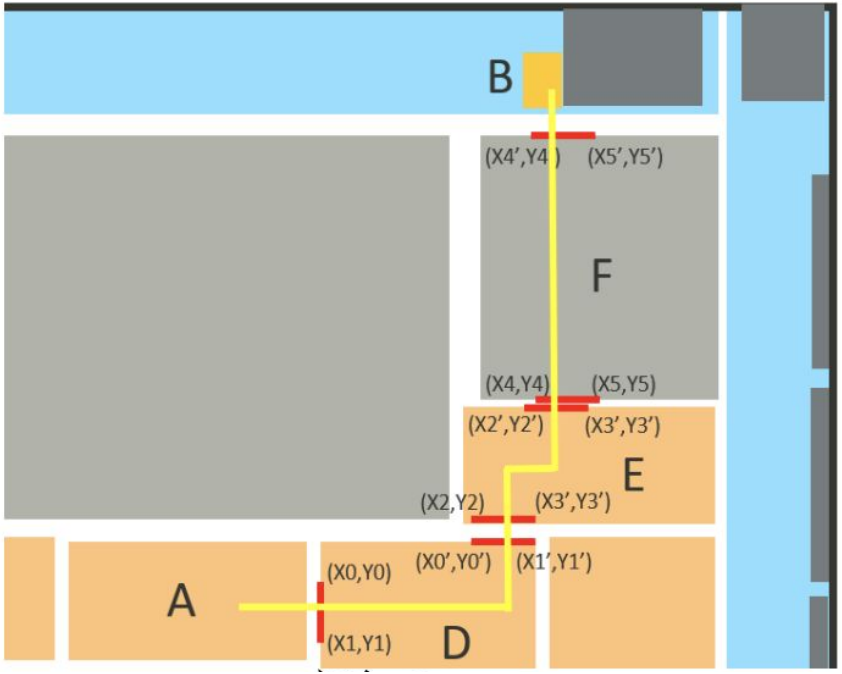
Content

- Background
- Materials and methods
- Preliminary results
- Future work

Background

Background


channel
tile
Throughable
block
Non-Throughable
block
Region


B
(X4',Y4')
(X5',Y5')
F
(X4,Y4)
(X5,Y5)
(X2',Y2')
(X3',Y3')
E
(X2,Y2)
(X3',Y3')
(X0,Y0)
(X0',Y0')
(X1',Y1')
A
(X1,Y1)
D

Materials and Methods

Materials : inputs (.def and .json files)

Methods : Parsing => Partition => Steiner Tree Construction

=> Redundancy removal and Local refinement

Materials and Methods - Input files

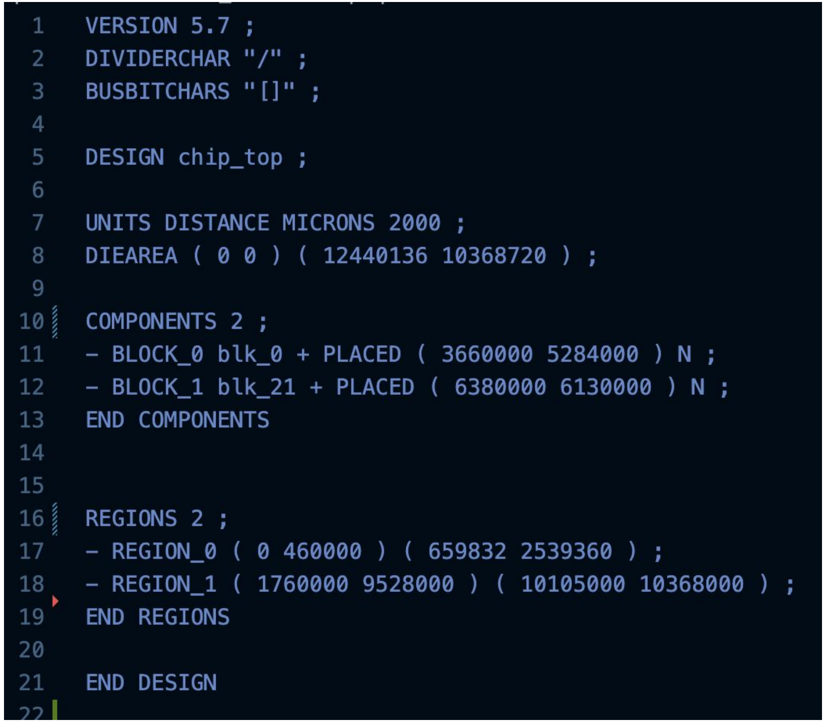
1) .def files : chip_top.def, block.def

2) CFG (.json)

3) Connection matrix (.json)

Materials and Methods - Input files

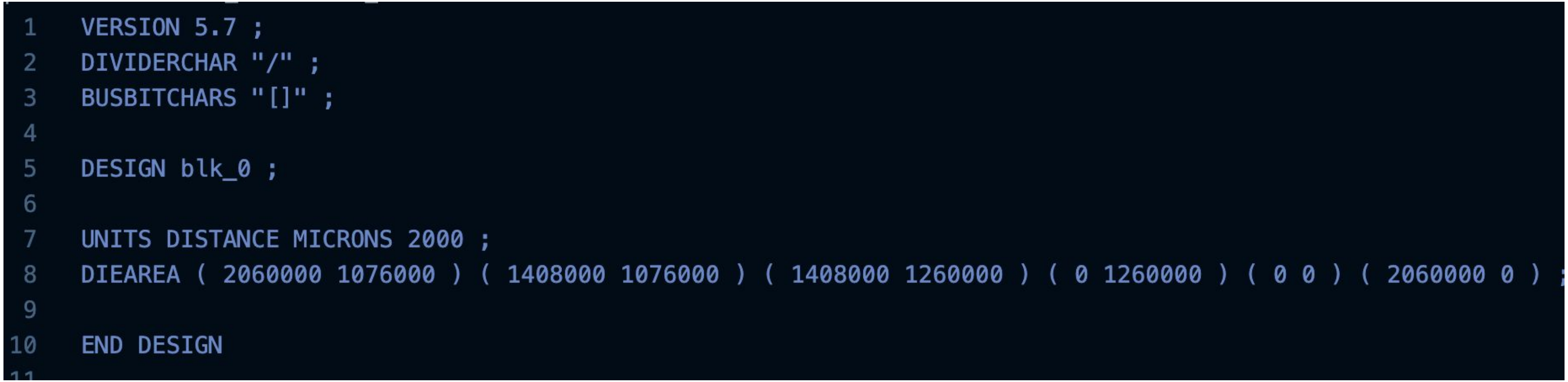
1) .def files : chip_top.def, block.def



```
VERSION 5.7 ;
DIVIDERCHAR "/" ;
BUSBITCHARS "[]" ;

DESIGN chip_top ;

UNITS DISTANCE MICRONS 2000 ;
DIEAREA ( 0 0 ) ( 12440136 10368720 ) ;

COMPONENTS 2 ;
- BLOCK_0 blk_0 + PLACED ( 3660000 5284000 ) N ;
- BLOCK_1 blk_21 + PLACED ( 6380000 6130000 ) N ;
END COMPONENTS


REGIONS 2 ;
- REGION_0 ( 0 460000 ) ( 659832 2539360 ) ;
- REGION_1 ( 1760000 9528000 ) ( 10105000 10368000 ) ;
END REGIONS

END DESIGN
```

Materials and Methods - Input files

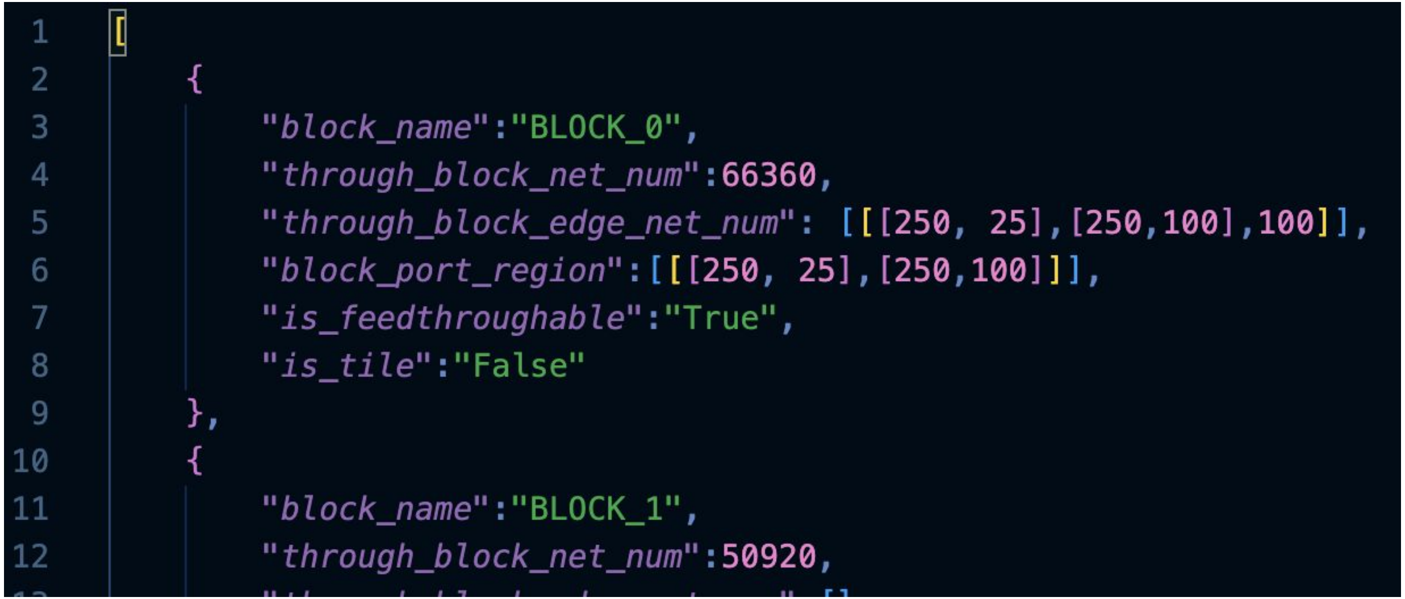
1) .def files : chip_top.def, block.def



```
VERSION 5.7 ;
DIVIDERCHAR "/" ;
BUSBITCHARS "[]" ;

DESIGN blk_0 ;

UNITS DISTANCE MICRONS 2000 ;
DIEAREA ( 2060000 1076000 ) ( 1408000 1076000 ) ( 1408000 1260000 ) ( 0 1260000 ) ( 0 0 ) ( 2060000 0 ) ;

END DESIGN
```

Materials and Methods - Input files

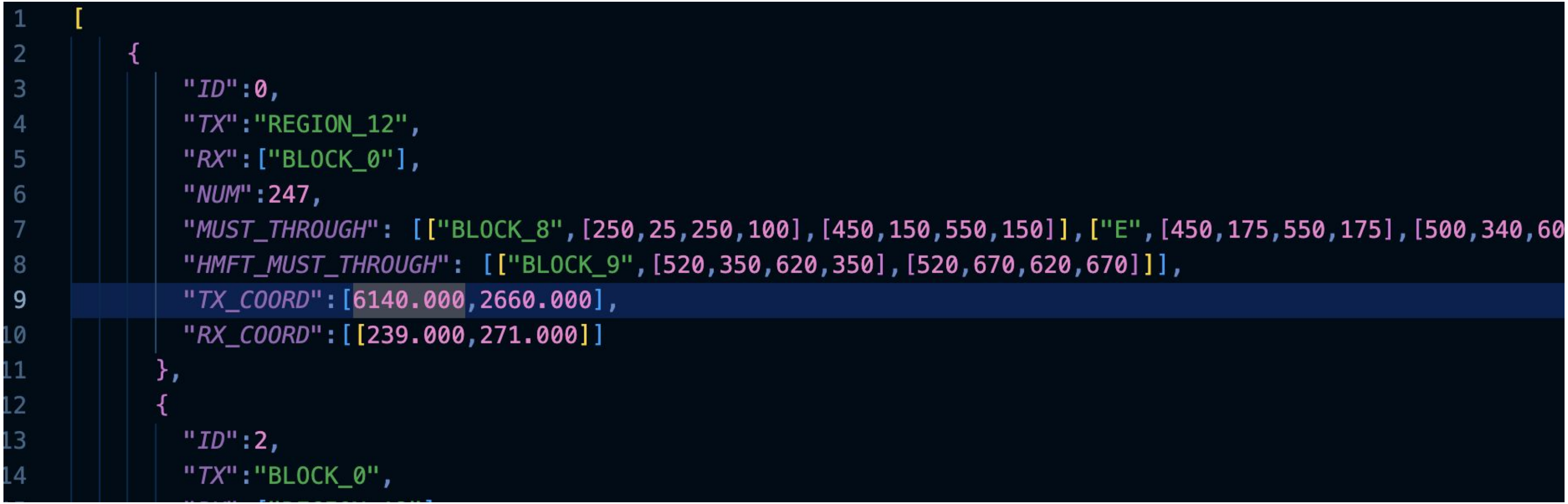
2) CFG (.json)



```
[
    {
        "block_name":"BLOCK_0",
        "through_block_net_num":66360,
        "through_block_edge_net_num": [[[250, 25],[250,100],100]],
        "block_port_region":[[[250, 25],[250,100]]],
        "is_feedthroughable":"True",
        "is_tile":"False"
    },
    {
        "block_name":"BLOCK_1",
        "through_block_net_num":50920,
        "through_block_edge_net_num": []
```

Materials and Methods - Input files

3) Connection matrix (.json)



```
[
    {
        "ID":0,
        "TX":"REGION_12",
        "RX":["BLOCK_0"],
        "NUM":247,
        "MUST_THROUGH": [["BLOCK_8",[250,25,250,100],[450,150,550,150]],["E",[450,175,550,175],[500,340,60
        "HMFT_MUST_THROUGH": [["BLOCK_9",[520,350,620,350],[520,670,620,670]]],
        "TX_COORD":[6140.000,2660.000],
        "RX_COORD":[[239.000,271.000]]
      },
      {
        "ID":2,
        "TX":"BLOCK_0",
```

Materials and Methods

1) Parsing JSON & DEF files into desired format
2) Partitioning the blocks
3) Consider additional properties (feedthroughable, capacity, ...)
4) Build Obstacle-Avoiding Rectilinear Steiner Minimal Tree (OARSMT)
5) Output the final result

Materials and Methods - Partitioning

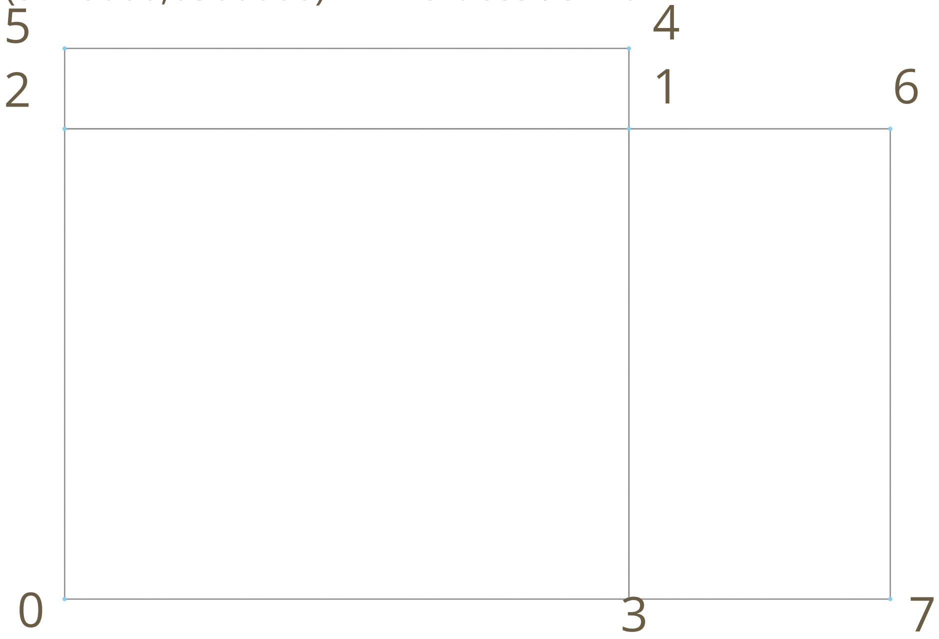
obstacles =

```
obstacleid : 0  start, end : (3660000,5284000) (5068000,6360000)      vertices : 0 2 1 3
obstacleid : 1  start, end : (3660000,6360000) (5068000,6544000)      vertices : 2 5 4 1
obstacleid : 2  start, end : (5068000,5284000) (5720000,6360000)      vertices : 3 1 6 7
```

vertices =

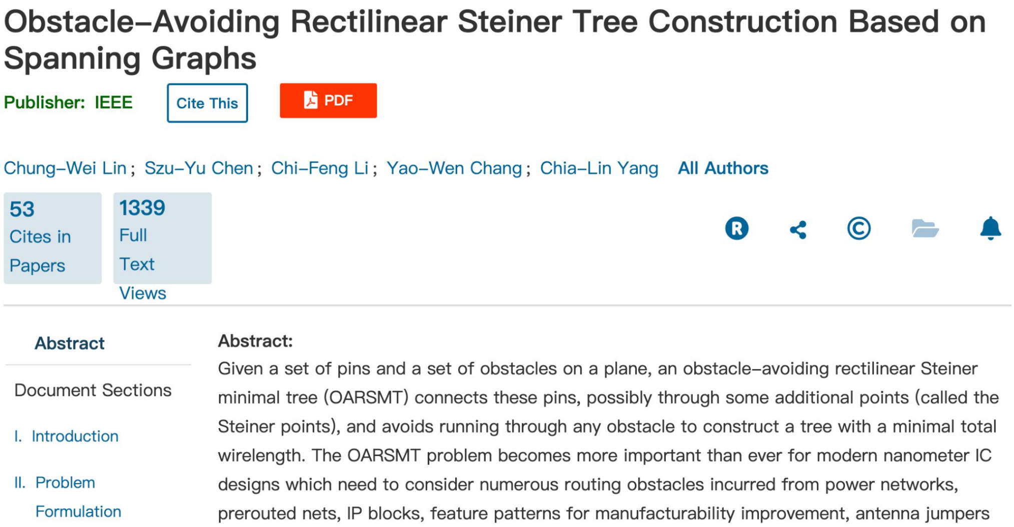
```
{ 0: (3.66e+06,5.284e+06),
1: (5.068e+06,6.36e+06),
2: (3.66e+06,6.36e+06),
3: (5.068e+06,5.284e+06),
4: (5.068e+06,6.544e+06),
5: (3.66e+06,6.544e+06),
6: (5.72e+06,6.36e+06),
7: (5.72e+06,5.284e+06)}
```

cut edges = [(2,1), (3,1)]



Materials and Methods - building routing steiner tree

Obstacle-Avoiding Rectilinear Steiner Tree Construction Based on Spanning Graphs



Obstacle–Avoiding Rectilinear Steiner Tree Construction Based on Spanning Graphs

Publisher: IEEE

Chung–Wei Lin ; Szu–Yu Chen ; Chi–Feng Li ; Yao–Wen Chang ; Chia–Lin Yang

53 Cites in Papers

1339 Full Text Views

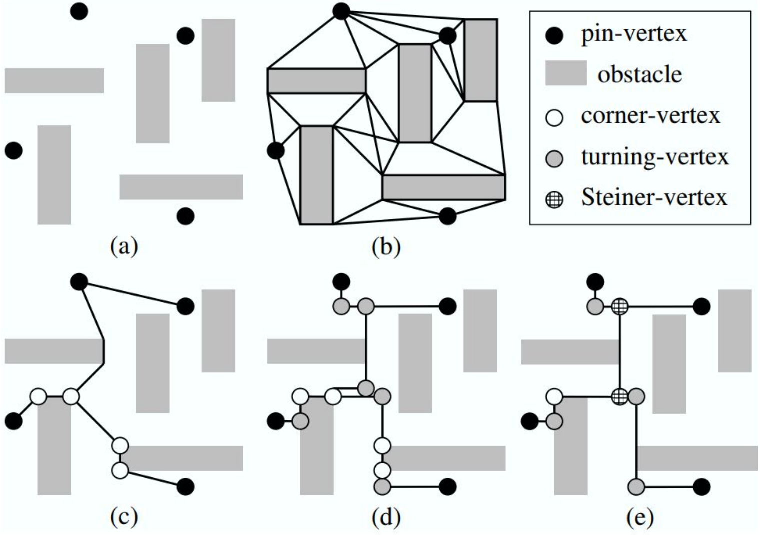
Abstract:

Given a set of pins and a set of obstacles on a plane, an obstacle–avoiding rectilinear Steiner minimal tree (OARSMT) connects these pins, possibly through some additional points (called the Steiner points), and avoids running through any obstacle to construct a tree with a minimal total wirelength. The OARSMT problem becomes more important than ever for modern nanometer IC designs which need to consider numerous routing obstacles incurred from power networks, prerouted nets, IP blocks, feature patterns for manufacturability improvement, antenna jumpers

C. -W. Lin, S. -Y. Chen, C. -F. Li, Y. -W. Chang and C. -L. Yang, "Obstacle-Avoiding Rectilinear Steiner Tree Construction Based on Spanning Graphs," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 4, pp. 643-653, April 2008, doi: 10.1109/TCAD.2008.917583.

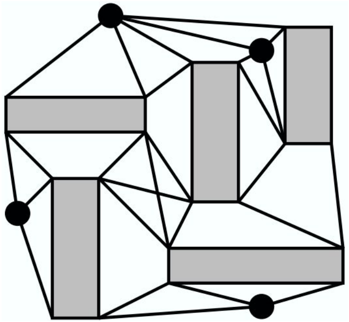
Materials and Methods: Steps of building OARSMT

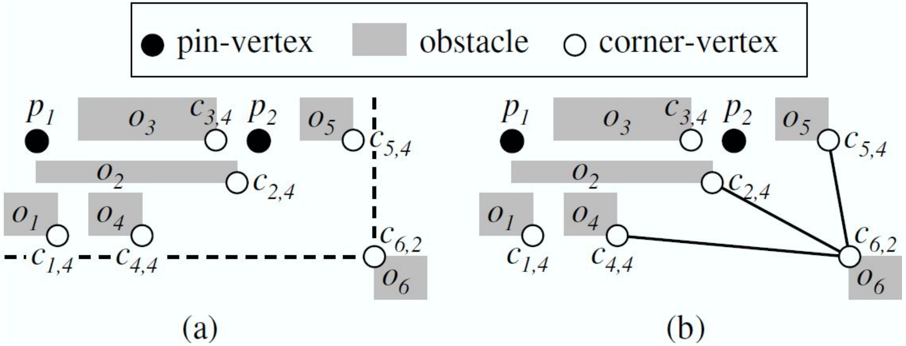
1) Obstacle-Avoiding Spanning Graph (OASG)
2) Obstacle-Avoiding Spanning Tree (OAST)
3) Obstacle-Avoiding Rectilinear Spanning Tree (OARST)
4) Obstacle-Avoiding Rectilinear Steiner Minimal Tree (OARSMT)



Materials and Methods - Step1: OASG

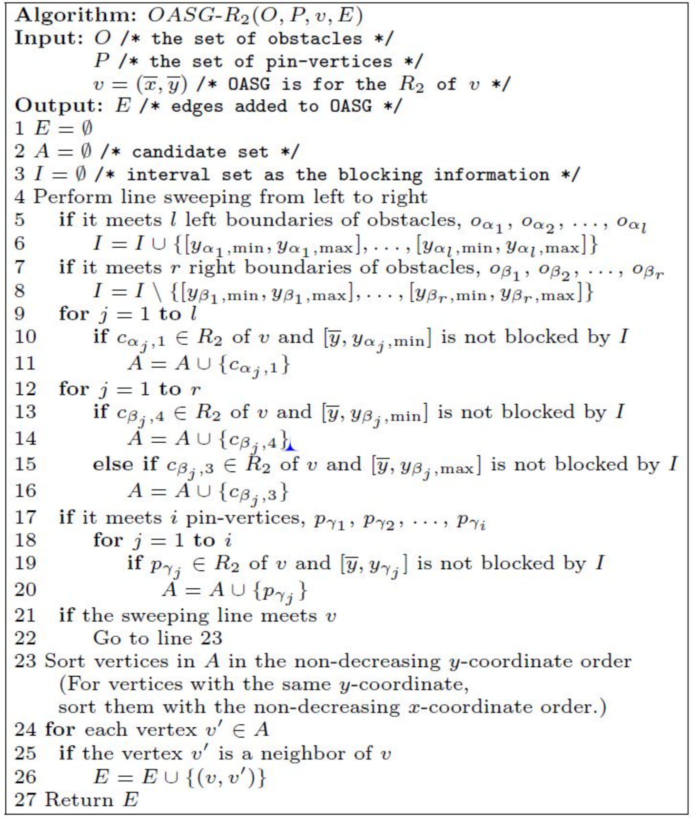
For each vertex, perform this algorithm on all 4 quadrants to find vertices close to it.





(a)

(b)



```
Algorithm: $OASG\text{-}R_2(O, P, v, E)$
Input: $O$ /* the set of obstacles */
       $P$ /* the set of pin-vertices */
       $v = (\overline{x}, \overline{y})$ /* OASG is for the $R_2$ of $v$ */
Output: $E$ /* edges added to OASG */
1 $E = \emptyset$
2 $A = \emptyset$ /* candidate set */
3 $I = \emptyset$ /* interval set as the blocking information */
4 Perform line sweeping from left to right
5   if it meets $l$ left boundaries of obstacles, $o_{\alpha_1}, o_{\alpha_2}, \ldots, o_{\alpha_l}$
6     $I = I \cup \{[y_{\alpha_1,\min}, y_{\alpha_1,\max}], \ldots, [y_{\alpha_l,\min}, y_{\alpha_l,\max}]\}$
7   if it meets $r$ right boundaries of obstacles, $o_{\beta_1}, o_{\beta_2}, \ldots, o_{\beta_r}$
8     $I = I \setminus \{[y_{\beta_1,\min}, y_{\beta_1,\max}], \ldots, [y_{\beta_r,\min}, y_{\beta_r,\max}]\}$
9   for $j = 1$ to $l$
10    if $c_{\alpha_j,1} \in R_2$ of $v$ and $[\overline{y}, y_{\alpha_j,\min}]$ is not blocked by $I$
11      $A = A \cup \{c_{\alpha_j,1}\}$
12  for $j = 1$ to $r$
13    if $c_{\beta_j,4} \in R_2$ of $v$ and $[\overline{y}, y_{\beta_j,\min}]$ is not blocked by $I$
14      $A = A \cup \{c_{\beta_j,4}\}$
15    else if $c_{\beta_j,3} \in R_2$ of $v$ and $[\overline{y}, y_{\beta_j,\max}]$ is not blocked by $I$
16      $A = A \cup \{c_{\beta_j,3}\}$
17  if it meets $i$ pin-vertices, $p_{\gamma_1}, p_{\gamma_2}, \ldots, p_{\gamma_i}$
18    for $j = 1$ to $i$
19      if $p_{\gamma_j} \in R_2$ of $v$ and $[\overline{y}, y_{\gamma_j}]$ is not blocked by $I$
20        $A = A \cup \{p_{\gamma_j}\}$
21  if the sweeping line meets $v$
22    Go to line 23
23 Sort vertices in $A$ in the non-decreasing $y$-coordinate order
   (For vertices with the same $y$-coordinate,
   sort them with the non-decreasing $x$-coordinate order.)
24 for each vertex $v' \in A$
25   if the vertex $v'$ is a neighbor of $v$
26     $E = E \cup \{(v, v')\}$
27 Return $E$
```

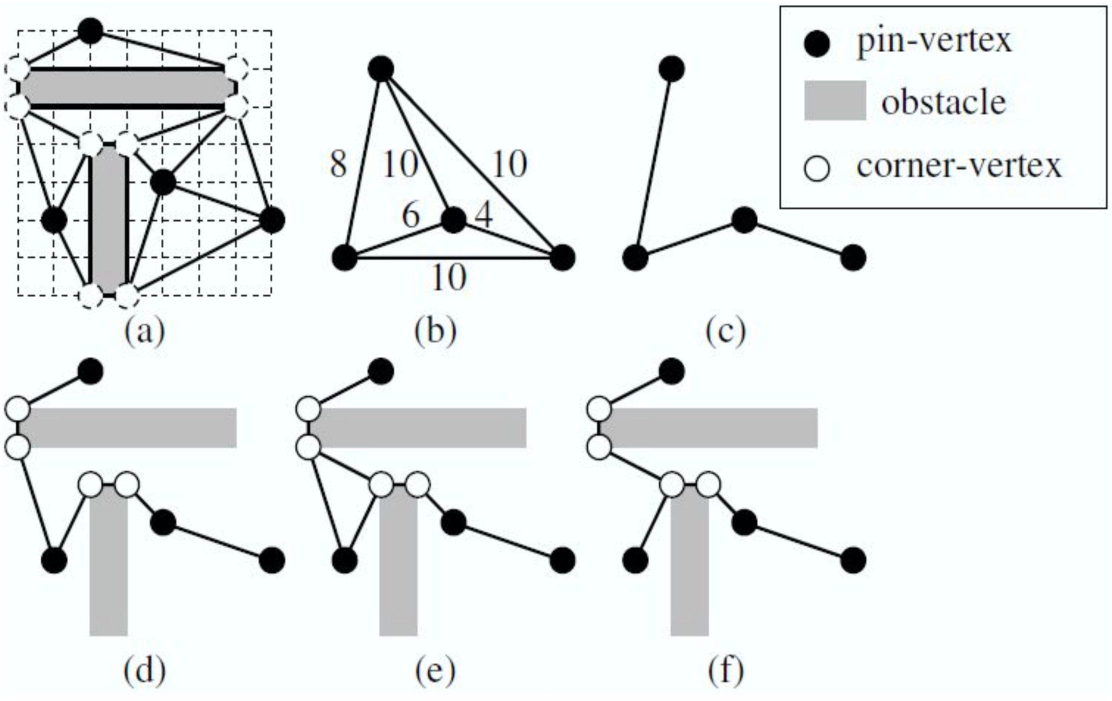
Materials and Methods - Step2: OAST

1) Pin and vertex shortest path computation

2) Initial OAST construction

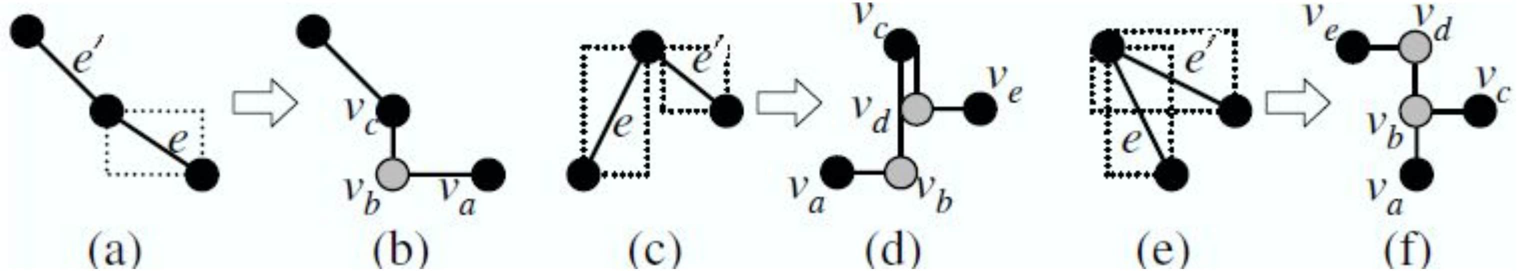
3) Local refinement

Materials and Methods - Step2: OAST



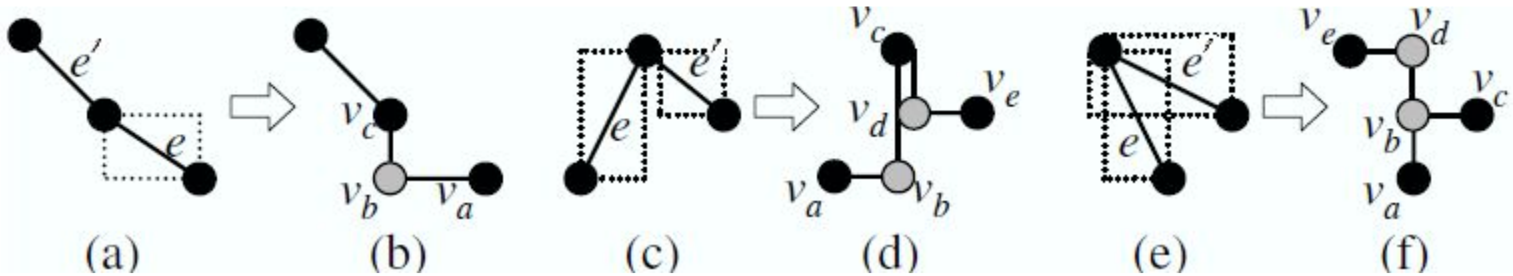
Materials and Methods - Step3: OARST

We transform each slant edge of the given OAST into vertical and horizontal edges to obtain the obstacle-avoiding rectilinear steiner tree.

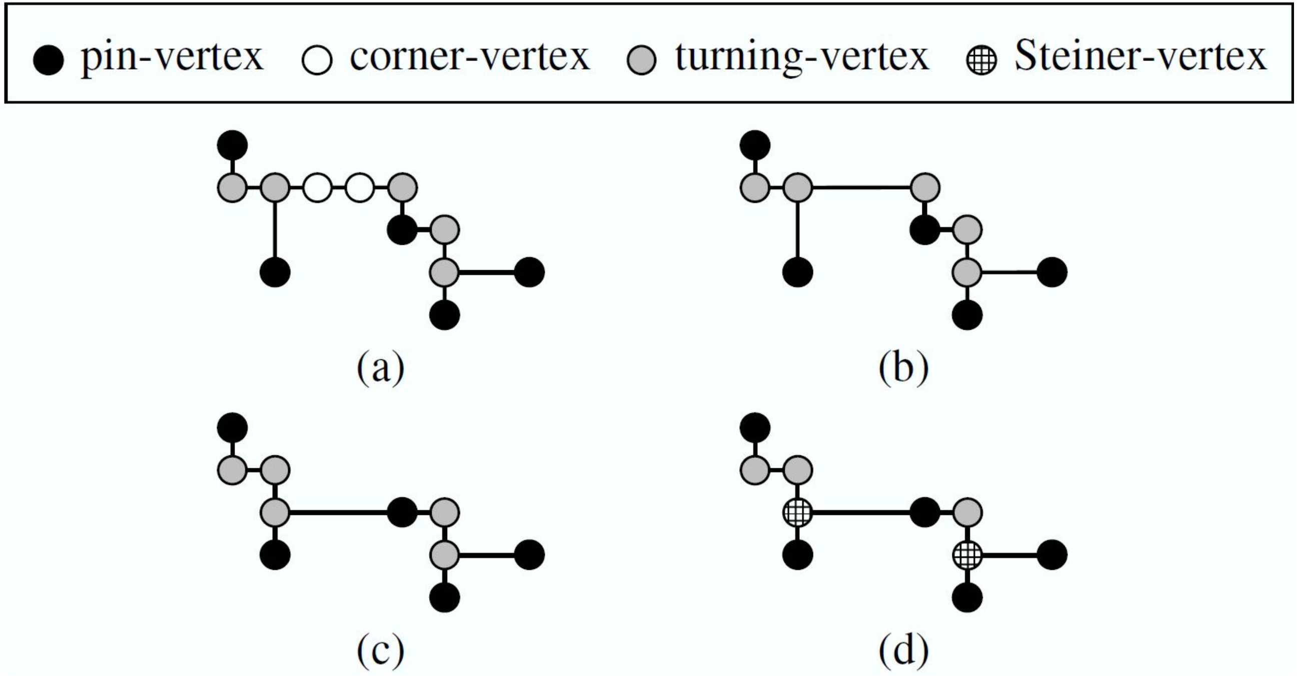


Materials and Methods - Step3: OARST

- STEP1: select the longest edge
- STEP2: select a adjacent edge with longest sharing path
- STEP3: Based on coordinates, choose the best strategy



Materials and Methods - Step4: OARSMT



(a)

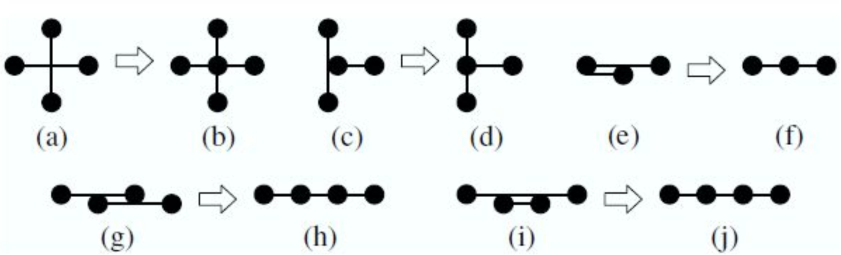
(b)

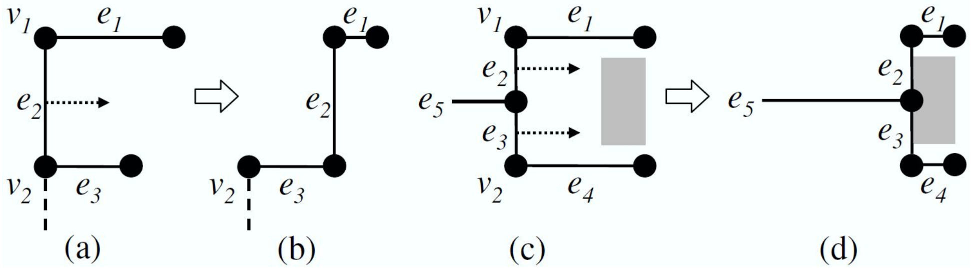
(c)

(d)

Materials and Methods - Step4: OARSMT

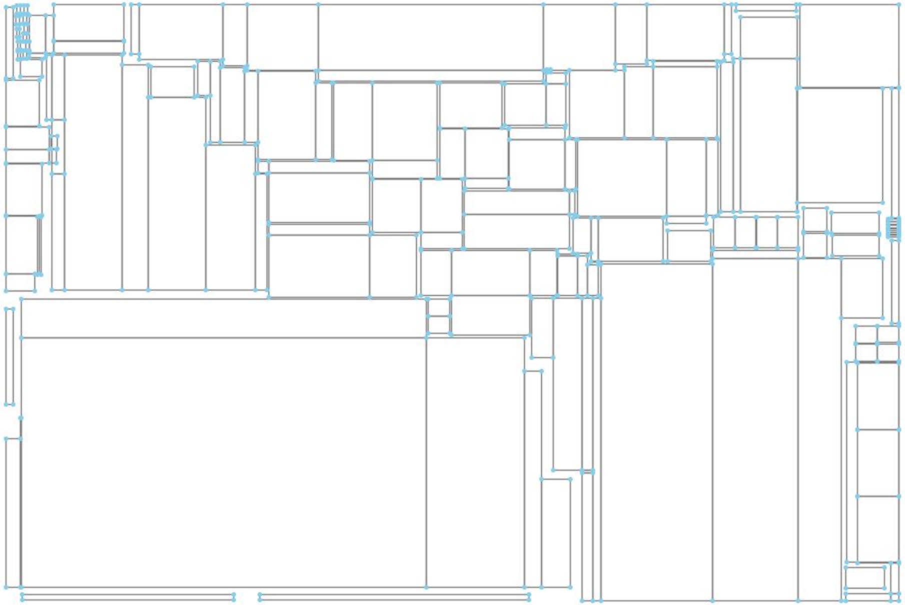
- Overlapping edge removal
- Redundant vertex removal
- U-shaped pattern refinement

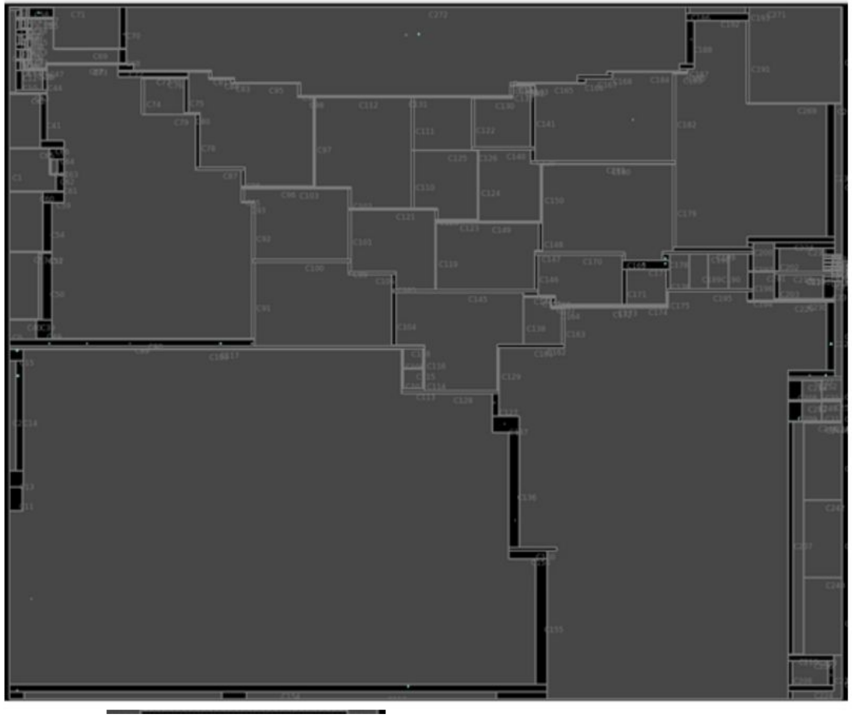




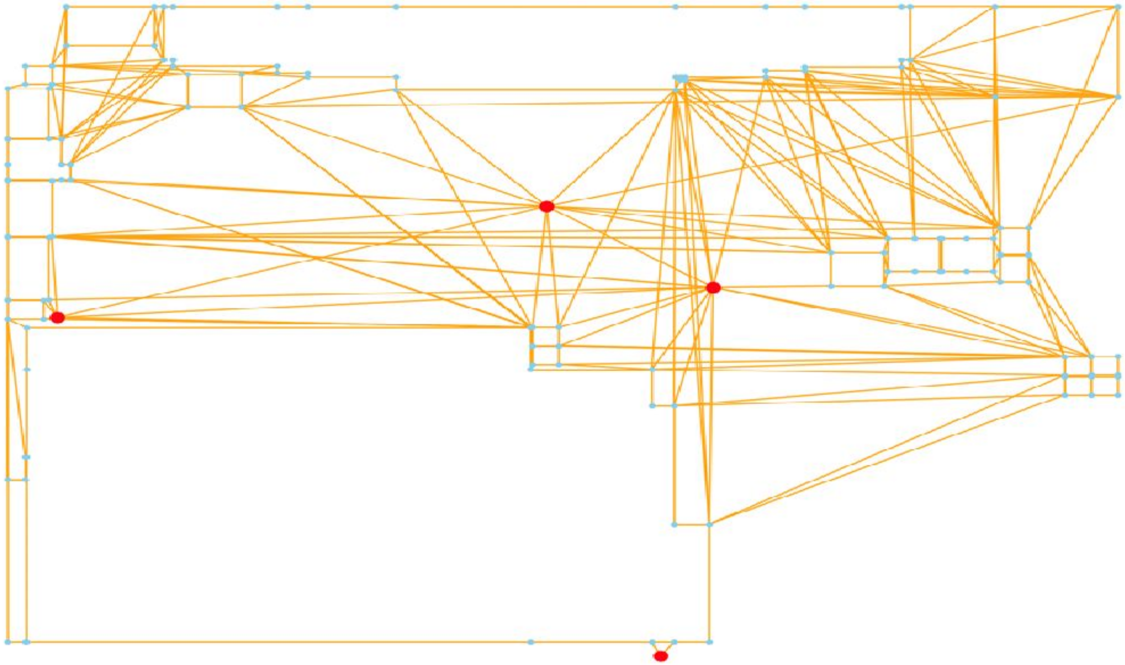
Preliminary result

Preliminary result - Partition

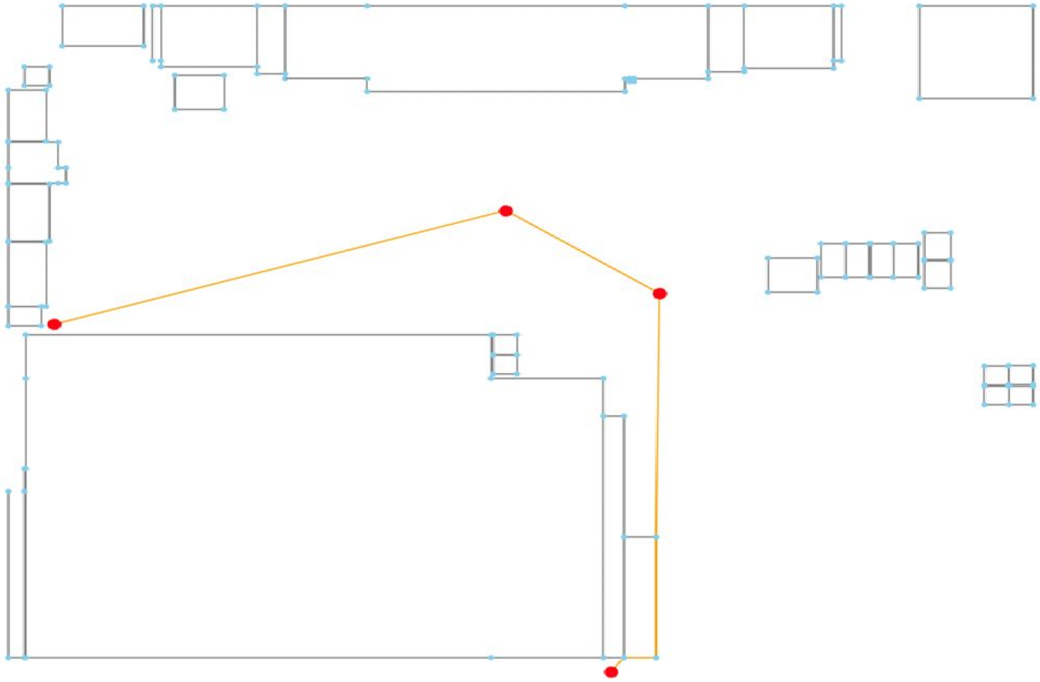




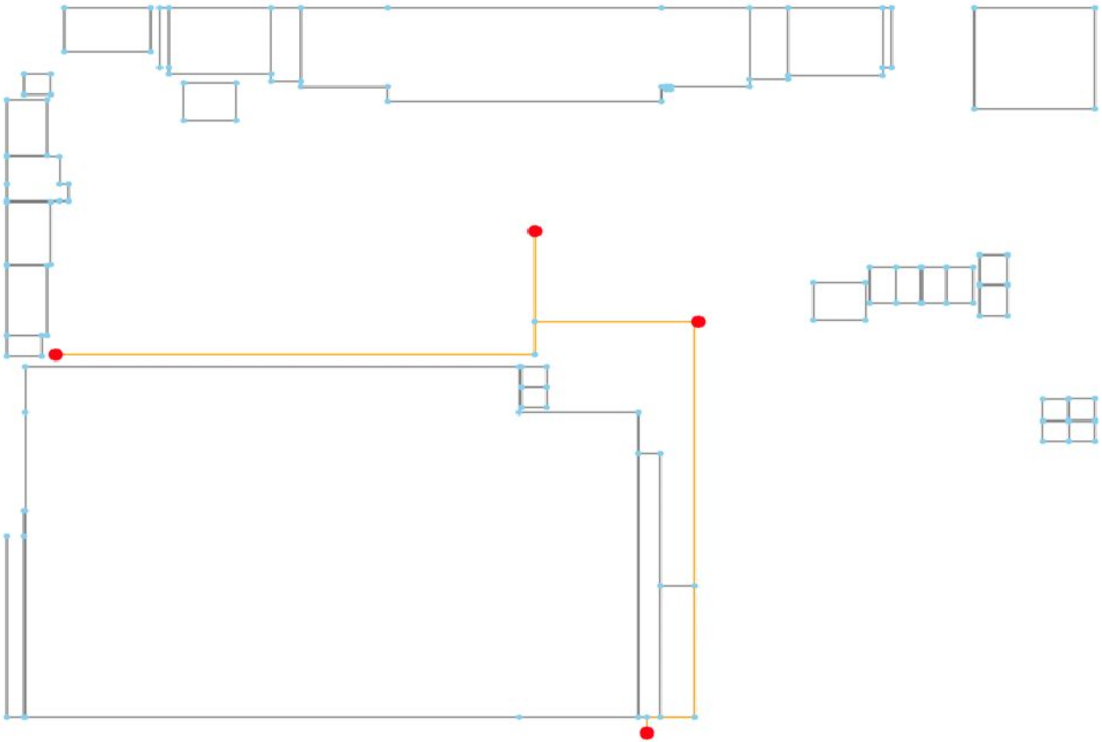
Preliminary result - OASG



Preliminary result - OAST

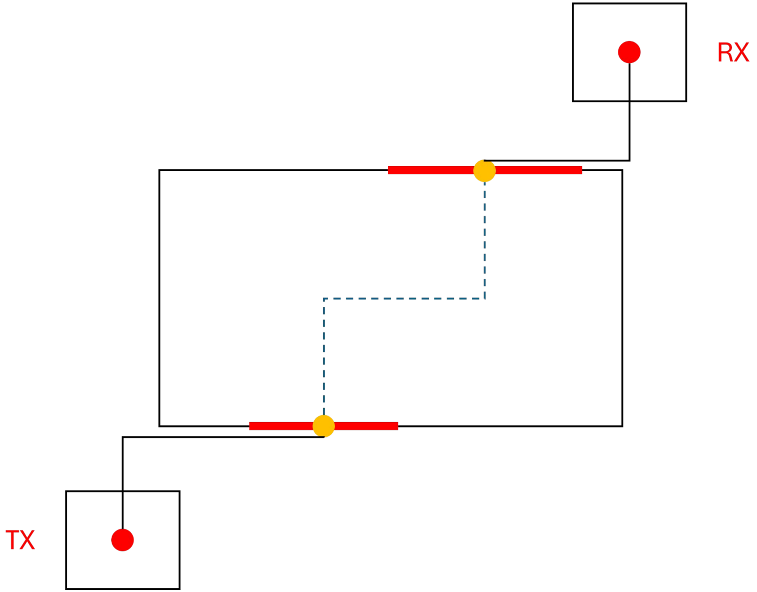


Preliminary result - OARST



Future work

- MustThrough blocks / through edge : add pseudo pins on block edges



Future work

- through_block_net_num of Feedthroughable blocks: memorize how many tracks through the block, place back the block until exceed the limit
- Congestion mitigation: apply rip-up and reroute to change several net to a alternative path to balance load across the grid.
-

THANK YOU