tags: **PD**

Physical Design HW3

Problem 1 [Collaborator: None]

Since minimizing the total area xy is a non linear problem, we approximately set the packing x value to the outline width 12, and minimize the y value.

The mixed ILP formulation:

$min\ y$

$sub.\ to$

// x direction fixed outline constraint
$x_1 + r_1 \times 4 + (1 - r_1) \times 2 \leq 12$
$x_2 + r_2 \times 5 + (1 - r_2) \times 3 \leq 12$
$x_3 + r_3 \times 6 + (1 - r_3) \times 4 \leq 12$

// y direction fixed outline constraint
$y_1 + r_1 \times 2 + (1 - r_1) \times 4 \leq y$
$y_2 + r_2 \times 3 + (1 - r_2) \times 5 \leq y$
$y_3 + r_3 \times 4 + (1 - r_3) \times 6 \leq y$

// non overlapping block constraint
$x_1 + r_1 \times 4 + (1 - r_1) \times 2 \leq x_2 + 12(p_{12} + q_{12})$
$x_1 + r_1 \times 4 + (1 - r_1) \times 2 \leq x_3 + 12(p_{13} + q_{13})$
$x_2 + r_2 \times 5 + (1 - r_2) \times 3 \leq x_3 + 12(p_{23} + q_{23})$

$y_1 + r_1 \times 2 + (1 - r_1) \times 4 \leq y_2 + 12(1 + p_{12} - q_{12})$
$y_1 + r_1 \times 2 + (1 - r_1) \times 4 \leq y_3 + 12(1 + p_{13} - q_{13})$
$y_2 + r_2 \times 3 + (1 - r_2) \times 5 \leq y_3 + 12(1 + p_{23} - q_{23})$

$x_1 - r_2 \times 5 - (1 - r_2) \times 3 \geq x_2 - 12(1 - p_{12} + q_{12})$
$x_1 - r_3 \times 6 - (1 - r_3) \times 4 \geq x_3 - 12(1 - p_{13} + q_{13})$
$x_2 - r_3 \times 6 - (1 - r_3) \times 4 \geq x_3 - 12(1 - p_{23} + q_{23})$

$y_1 - r_2 \times 3 - (1 - r_2) \times 5 \geq y_2 - 12(2 - p_{12} - q_{12})$
$y_1 - r_3 \times 4 - (1 - r_3) \times 6 \geq y_3 - 12(2 - p_{13} - q_{13})$
$y_2 - r_3 \times 4 - (1 - r_3) \times 6 \geq y_3 - 12(2 - p_{23} - q_{23})$

// positive coordinate value placing constraint
$x_1, y_1, x_2, y_2, x_3, y_3 \geq 0$

// non overlap constraints inequality encoding paramters
$p_{12}, p_{13}, p_{23}, q_{12}, q_{13}, q_{23}, r_1, r_2, r_3 \in \{0, 1\}$

Problem 2 [Collaborator: None]

a.

$x = \frac{1\times1+3\times24+5\times4+7\times18}{1+3+5+7} = 13.6875$

$y = \frac{1\times21+3\times3+5\times24+7\times6}{1+3+5+7} = 12$

$Q(x) = Q(13.6875) = 14$

$Q(y) = Q(12) = 12$

where $Q(.)$ is the quantizing function.

b.

If the preplaced block is locked (usually a locked cells has higher cost, ex. higher connectivities), then move the selected cell to the nearest vacant location. If the preplaced block is not locked, then move the selected cell to the target location and move the preplaced cell otherwhere by some algorithms such as ripple move and chain move methods.

Problem 3 [Collaborator: None]

a. (2,20)
b. (8,16)
c. (30,6)
d. (12,24)

	L1 norm	**square**
ab	6+4=10	52
ac	28+14=42	980
ad	10+4=14	116
bc	22+10=32	584
bd	4+8=12	80
cd	18+18=36	648

- HPWL wire length:
  $m = (30-2) + (24-6) = 46$
- Minimum cost spanning tree wire length:
  $n = 10 + 12 + 32 = 54$
- Minimum cost steiner tree wire length:
  adding steiner point p (8,20), then
  $p = ap + pb + pd + bc = (6+0) + (0+4) + (4+4) + (22+10) = 50$
- squared euclidean distance wire length:
  $q = (52 + 980 + 116 + 584 + 80 + 648) = 2460$

- Log-sum-exp (gamma=10) wire length:
  $r = 10 \times (ln(exp(2/10) + exp(8/10) + exp(30/10) + exp(12/10))$
  $+ ln(exp(-2/10) + exp(-8/10) + exp(-30/10) + exp(-12/10))$
  $+ ln(exp(20/10) + exp(16/10) + exp(6/10) + exp(24/10))$
  $+ ln(exp(-20/10) + exp(-16/10) + exp(-6/10) + exp(-24/10)))$
  $= 69.75$
- Weighted-average(WA) wire length:
  $s = \frac{2*exp(2/10)+8*exp(8/10)+30*exp(30/10)+12*exp(12/10)}{exp(2/10)+exp(8/10)+exp(30/10)+exp(12/10)}$
  $- \frac{2*exp(-2/10)+8*exp(-8/10)+30*exp(-30/10)+12*exp(-12/10)}{exp(-2/10)+exp(-8/10)+exp(-30/10)+exp(-12/10)}$
  $+ \frac{20*exp(20/10)+16*exp(16/10)+6*exp(6/10)+24*exp(24/10)}{exp(20/10)+exp(16/10)+exp(6/10)+exp(24/10)}$
  $- \frac{20*exp(-20/10)+16*exp(-16/10)+6*exp(-6/10)+24*exp(-24/10)}{exp(-20/10)+exp(-16/10)+exp(-6/10)+exp(-24/10)}$
  $= 26.5637$

Problem 4 [Collaborator: B05901053 吳宥璁]

a.
Two differentiable functions to approximate the minimum function $min\{z_1, \ldots, z_n\}$:
(1) $\{\sum_i (z_i \times exp(-z_i/\gamma))\}/\{\sum_i exp(-z_i/\gamma)\}$
This function is a weighted average function with weight $exp(-z_i/\gamma)$. Since $exp(-x)$ function is a decreasing function, when the minimum term is much smaller than others, its weight is much larger than others, resulting this function a good approximation to minimum function.

(2) $\{\sum_i (z_i \times 1/z_i^n)\}/\{\sum_i 1/z_i^n\} = \{\sum_i 1/z_i^{n-1}\}/\{\sum_i 1/z_i^n\}$
This function is a weighted average function with weight $1/z_i^n$. Same as (1), the weight function is a decreasing function, so it can approximate minimum function if the min. value is much smaller than others. **Note that this function is only continuously decreasing in** $(0, \infty)$ **and non-differentiable at x=0**, if any number in the minimum function is smaller or equal to 0, it might be wrong, in such case we can find any contiously decreasing function as our weight, such as $sigmoid(-x), atanh(-x), \ldots$

b.
Parameters control for more accurate approximations:
(1) Smaller γ can get higher accuracy, since smaller γ emphasize the different between the smallest value and others, causing larger weight to the smallest value.
(2) Larger n can get higher decreasing rate, thus can emphasize more on the weight of the smallest value.

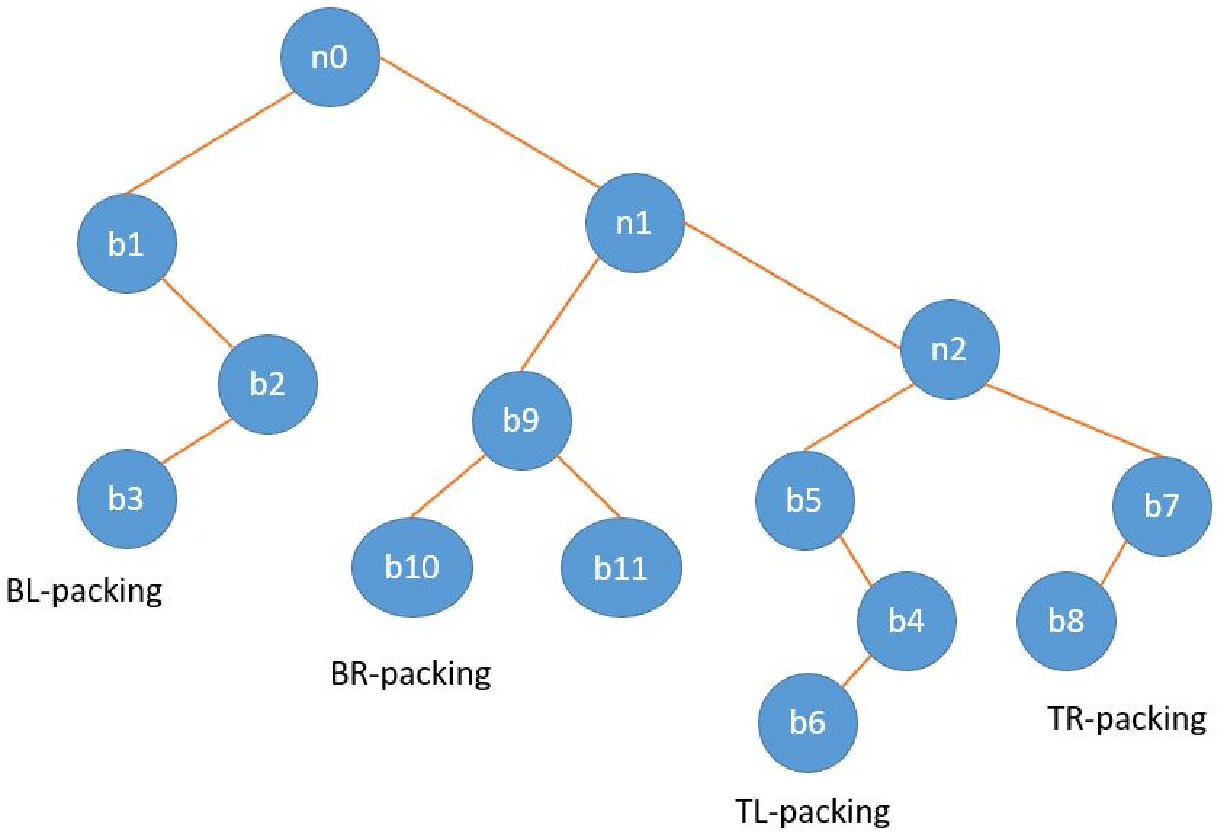
The following is the result of the above two function for finding the min value approximation in number set: 6, 16, 20, 24.

Approximate functions				
(1) $\{\sum_i (z_i exp(-z_i/\gamma))\}/\{\sum_i exp(-z_i/\gamma)\}$	parameters γ	1	5	10
	results	6.0004	8.204	11.68
(2) $\{\sum_i 1/z_i^{n-1}\}/\{\sum_i 1/z_i^n\}$	parameters n	2	3	4
	results	8.931	7.08	6.369

Problem 5 [Collaborator: None]

[Ref. Tung-Chieh Chen, Ping-Hung Yuh, Yao-Wen Chang, Fwu-Juh Huang, and Denny Liu. 2007. MP-trees: a packing-based macro placement algorithm for mixed-size designs. In Proceedings of the 44th Annual Design Automation Conference (DAC '07). ACM, New York, NY, USA, 447-452 (https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4261226)]

a.
The MP-tree of the packing is:



b.
For deriving the packing from the MP-tree. Its x value can be calculate by simply traverse throught the tree and update all x value by the following formula. As for y value, we need to maintain the contour data structures.

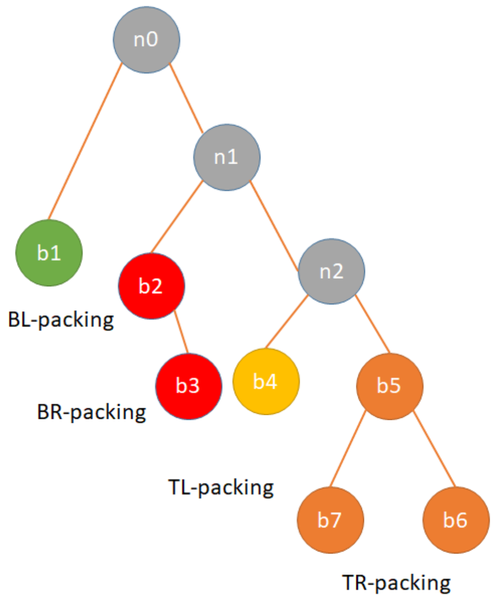
If node n_j is the right child of n_i, the block b_j is

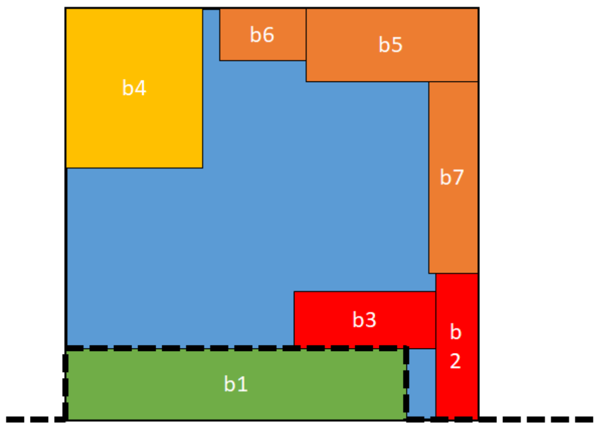
- the lowest adjacent block on the right with $x_j = x_i + w_i$ for BL-packing,
- the highest adjacent block on the right with $x_j = x_i + w_i$ for TL-packing,
- the highest adjacent block on the left with $x_j = x_i - w_j$ for TR-packing, and
- the lowest adjacent block on the left with $x_j = x_i - w_j$ for BR-packing.

If node n_j is the left child of n_i, the block b_j is

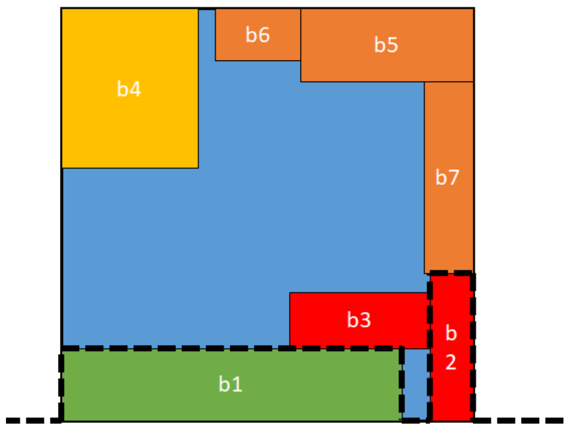
- the first block above b_i with $x_j = x_i$ for BL-packing,
- the first block below b_i with $x_j = x_i$ for TL-packing,
- the first block below b_i with $x_j = x_i + w_i - w_j$ for TR-packing, and
- the first block above b_i with $x_j = x_i + w_i - w_j$ for BR-packing.

We need two horizontal contours for calculating the y value, one represents the bottom countour and the other represents the top contour.
First, we packed the BL-subtree and maintain the contour data structure (a double linked list), then we packed the BR-packing based on the bottom contour. This operation is amortized linear time.
For the top contour, it is similar to the bottom contour. However, if the contour overlaps, that means the MP-tree corresponds to an invalid packing structure. Therefore, we need to compare any new packed block with the bottom contour to avoid overlap. This operation is still an amortized linear time operation.
An example of a MP-tree packing with two contour data structures is shown step by step in the figure below.

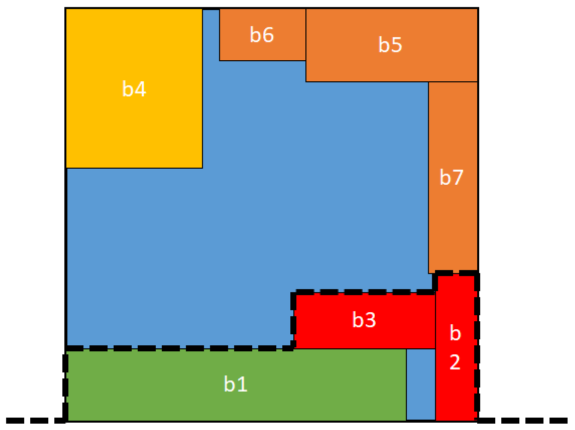




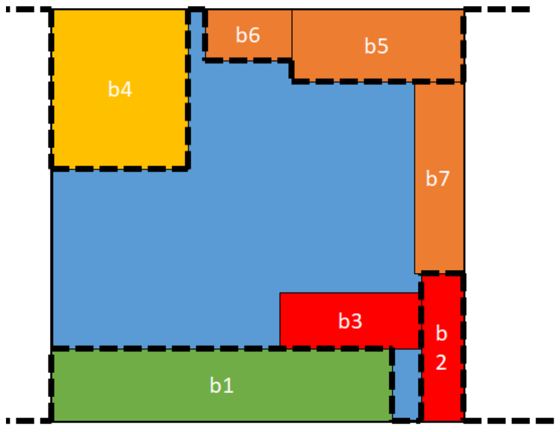
1. Pack b1 in BL subtree to bottom contour



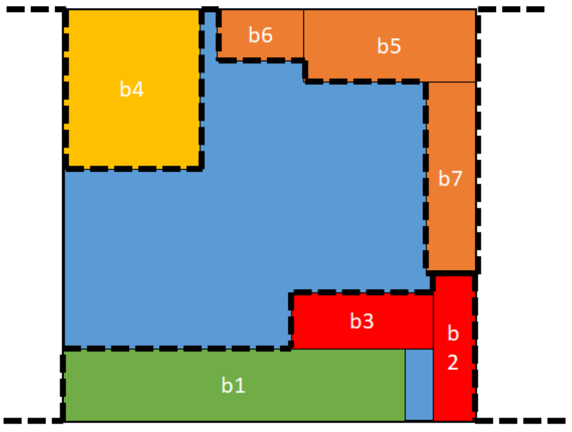
2. Pack b2 in BR subtree to bottom contour



3. Pack b3 in BR subtree to bottom contour, then the bottom contour has done packing



4~6. Similarly pack b4, b5, b6 in TL and TR subtree to top contour



7. Pack b7 in TR subtree to top contour, then the top contour has done packing. Notice that we should compare the two contour every time we pack a new block to avoid contour overlapping. For instance, if b7 has higher height in this case, it will collide with b2, resulting an invalid packing structure.

From the above discussion, we can see that the complexity of MP-tree packing is amortized linear time.

c.
The strengths and weakness of MP-tree macro placement.

	Strengths	**Weakness**
1	Compared with four packing trees, it condisers more global information.	It can only packed macros in the corner, which may not be the optimal design. For instance, the dead space in macro placment is not a good place to place standard cells due to routability issues, resulting area utilization loss. Mix-size placers may be able to solve these problems, finding a better solution.
2	Experiments show that it is robust in finding legal placements and routable results.	It has many limitations; for example, it is much harder to deal with the region constraints that cross different regions.

Problem 6 [Collaborator: None]

Given $err = \gamma \times ln(\sum_{i=1 \sim n} e^{z_i/\gamma}) - max(z_1, \ldots, z_n)$
Express its upper/lower bound as functions of n and γ

- upper bound:
  $max_{z_i}\ err$
  $= max_c\{max_{z_i,\ max(z_1,\ldots,z_n)=c}\ err\}$
  $= max_c\{\gamma \times ln(\sum_{i=1 \sim n} e^{c/\gamma}) - c\}$
  $= max_c\{\gamma \times ln(ne^{c/\gamma}) - c\}$
  $= max_c\{\gamma \times ln(n)\}$
  $= \gamma \times ln(n)$

- lower bound:
  $min_{z_i}\ err$
  $= min_c\{min_{z_i,\ max(z_1,\ldots,z_n)=c}\ err\}$
  $= min_c\{\gamma \times ln(e^{c/\gamma}) - c\}$
  $= min_c\{c - c\}$
  $= 0$

Problem 7 [Collaborator: None]

point index	**1**	**2**	**3**	**4**	**5**	**6**
x	x1	x2	x3	10	0	50
y	y1	y2	y3	2	40	30

edge with nets: 12, 13, 14, 24, 25, 36

Optimization problem:

$$\phi = (x1-x2)^2 + (x1-x3)^2 + (x1-10)^2 + (x2-10)^2 + (x2-0)^2 + (x3-50)^2$$
$$+ (y1-y2)^2 + (y1-y3)^2 + (y1-2)^2 + (y2-2)^2 + (y2-40)^2 + (y3-30)^2$$

Obviously, it is a convex optimization problem. Therefore, the optimal solution will satisfy the KKT condition:

$$\frac{\phi}{\partial x1} = 2(x1-x2) + 2(x1-x3) + 2(x1-10) = 0$$
$$\frac{\phi}{\partial x2} = -2(x1-x2) + 2(x2-10) + 2x2 = 0$$
$$\frac{\phi}{\partial x3} = -2(x1-x3) + 2(x3-50) = 0$$
$$\frac{\phi}{\partial y1} = 2(y1-y2) + 2(y1-y3) + 2(y1-2) = 0$$
$$\frac{\phi}{\partial y2} = -2(y1-y2) + 2(y2-2) + 2(y2-40) = 0$$
$$\frac{\phi}{\partial y3} = -2(y1-y3) + 2(y3-30) = 0$$

Solving the above linear system, we can get:

$x1 = 17.6923, x2 = 9.2308, x3 = 33.8462, y1 = 14.3077, y2 = 18.7692, y3 = 22.1538$

Problem 8 [Collaborator: B05901053 吳宥璁 (DIY problem)]

a.

pins: (0,0) (16,0) (16,8) (32,0)

a(HPWL wire length cost) = $(32-0) + (8-0) = 40$

b(minimum Steiner tree cost) = 32+8 = 40

error rate |40-40|/40*100% = 0%

b.

pins: (0,0) (16,0) (16,8) (32,0) (8,8)

a(HPWL wire length cost) = $(32-0) + (8-0) = 40$

b(minimum Steiner tree cost) = 32+8+8 = 48

error rate |40-48|/48*100% = 16.67%

c.

pins: (0,0) (16,0) (16,8) (32,0) (8,8) (24,8)

a(HPWL wire length cost) = $(32-0) + (8-0) = 40$

b(minimum Steiner tree cost) = 32+8+8+8 = 56

error rate |40-56|/56*100% = 28.57%

d.

From **a.**, **b.**, **c.**, we can see that for high degree nets, if there are pins in the bounding box, the HPWL measure would not increase at all, resulting in HPWL underestimating the actual wire length.

A better approximation for high degree nets is quadratic clique net models such as GordainL; however, it may over estimate in most cases. An even better approach is finding the rectlinear minimum spanning tree (RMST), and it is proved to be a 1.5-approximate algorithm to the minimum rectilinear steiner tree (MRST), but RMST requires $O(ElgV)$ complexity, which is really high.

We proposed a new hierachical-HPWL wire length that can calculate in $O(VlgV)$ time and get higher wire length precision then HPWL.

For a number of nodes, we first find the max and min of x and y. Then, we aim to partition the nodes into two clusters. We find the range of x and y, and partition the direction with larger ranges. The algorithm for partition is (1) find the value of the median (linear time for applying BFPRT algorithm), (2) nodes that have value smaller then the median is partitioned in one group, and nodes with higher value then the median is in another.
Once we have solved the two subproblem, the merging criterion is to add the direction of the cut only once (O(1) time). Therefore, the overhead of merging the subproblem is linear time.
And the terminate condition is untill the number of nodes is small enough, say 3, so that HPWL can get quite good approximation.
The pseudo code for our hierachical-HPWL is as follows:

```
hier_HPWL(vector<node> nodes)
    vector<double> x_value = nodes' x value
    vector<double> y_value = nodes' y value
    if(number of nodes <= 3)
        return HPWL_x, HPWL_y; // the conventional HPWL
    else if(range(y_value) > range(x_value)){
        // cut y
        // find median, linear time
        med = find median of y_value
        vector<node> group_1 = nodes' with y_value < med
        vector<node> group_2 = nodes' with y_value >= med
        // find adjacent nodes, linear time
        vector<node> group_1_adj_x = all nodes' with max x in group_1
        vector<node> group_2_adj_x = all nodes' with min x in group_2
        // find adjacent nodes, linear time
        group_1_adj_min, group_1_adj_max = min & max y group_1_adj_x
        group_2_adj_min, group_2_adj_max = min & max y group_2_adj_x
        // get subproblem solution
        HPWL_x1 = hier_HPWL(group_1)
        HPWL_x2 = hier_HPWL(group_2)
        // compute current solution
        return  HPWL_x1 + HPWL_x2 + max(group_2_adj_min-group_1_adj_max,0) +
            max(group_2_adj_max-group_1_adj_min,0)
    }
    else{
        ...
        // similar to the blockc above
    }
```

The complexity of this algorithm can be calculate by master's thm:
$T(n) = 2T(n/2) + O(n)$
Notice that partition by median can get balance parition.

A simple comparison table is shown below.

	Complexity	**Accuracy**
Qudratic	V^2	low (over estimate)
HPWL	V	low (under estimate)
RMST	$V^2 lgV$	highest of all
hier-HPWL	$VlgV$	high

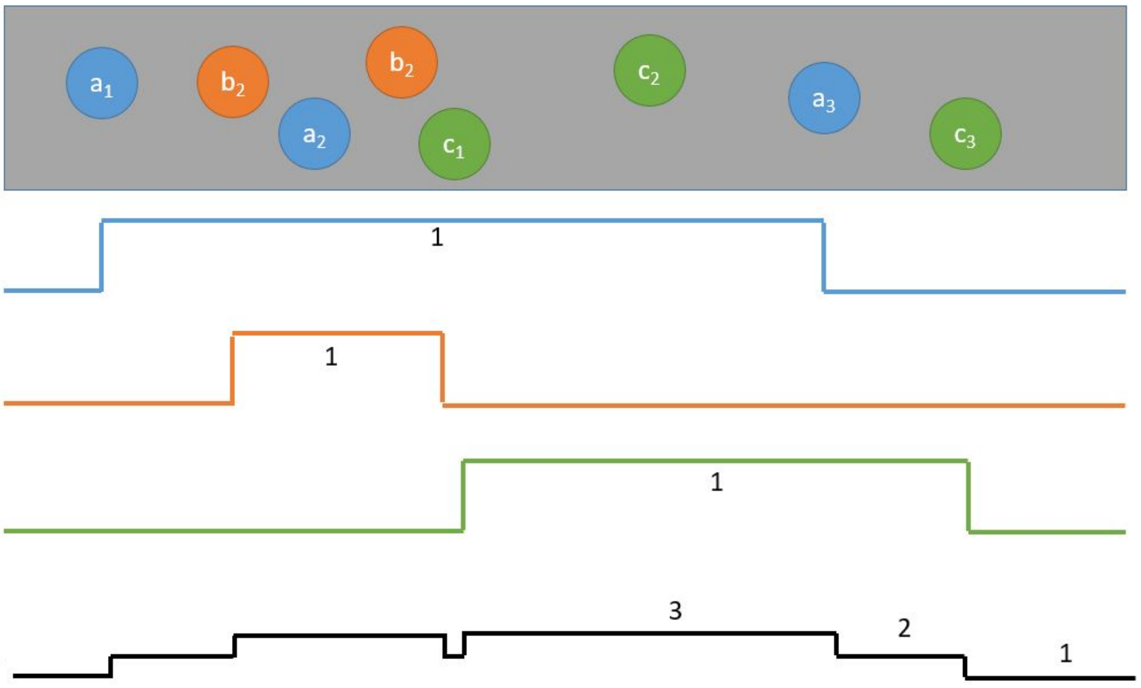
Our hierachical HPWL algorithm can get exact wire length in the cases of problem a, b, c.

Problem 9 [Collaborator: B05901053 吳宥璁]

In the ultra thin chip floorplan, most interconnections fun horizontally. Therefore, the most important point is that the number of wires passing through each V cut line can not be too much; otherwise, the routability would be poor.

The exact number of wires passing throught the V cut line at x is $n(x) = \sum_{e \in E} 1\{min_{v_i \in e}(v_i) \leq x \leq max_{v_i \in e}(v_i)\}$, where $1\{.\}$ is the indicator function.

An simple example for calculate the $n(x)$ function is shown below.



Therefore, the optimization problem for a ultra thin chip can be formulate by:
$min\ HPWL,\ n(x) \leq M_x$
where M_x is the maximum allowed wire number (or wire density) in the V cut line at x.

We set the constriant problem to an unconstrainted form by add the constriant to the objective function as an regularize term.
$min\ HPWL + \lambda(n(x) - M_x)^2$, where λ is the regularized parameter. Since the function $HPWL$ is not differentiable, we apply the WA wire length model.

However, one problem for solving this optimization problem is that the function $n(x)$ is not differentiable.Therefore, we aim to find a differtiable function $\hat{n}(x)$ to approximate this function.

We use sigmoid function as an approximation of the indicator function $1\{.\}$ (see the formula below), and weighted-average (WA) as an approximation of the max and min function.

$$f(l,x,u) = \begin{cases} 1, & \text{if } l < x < u \\ 0, & \text{otherwise} \end{cases} \xrightarrow{p(t) = \frac{1}{1+e^{-\alpha t}}} f(l,x,u) \cong p(x-l)p(u-x)$$

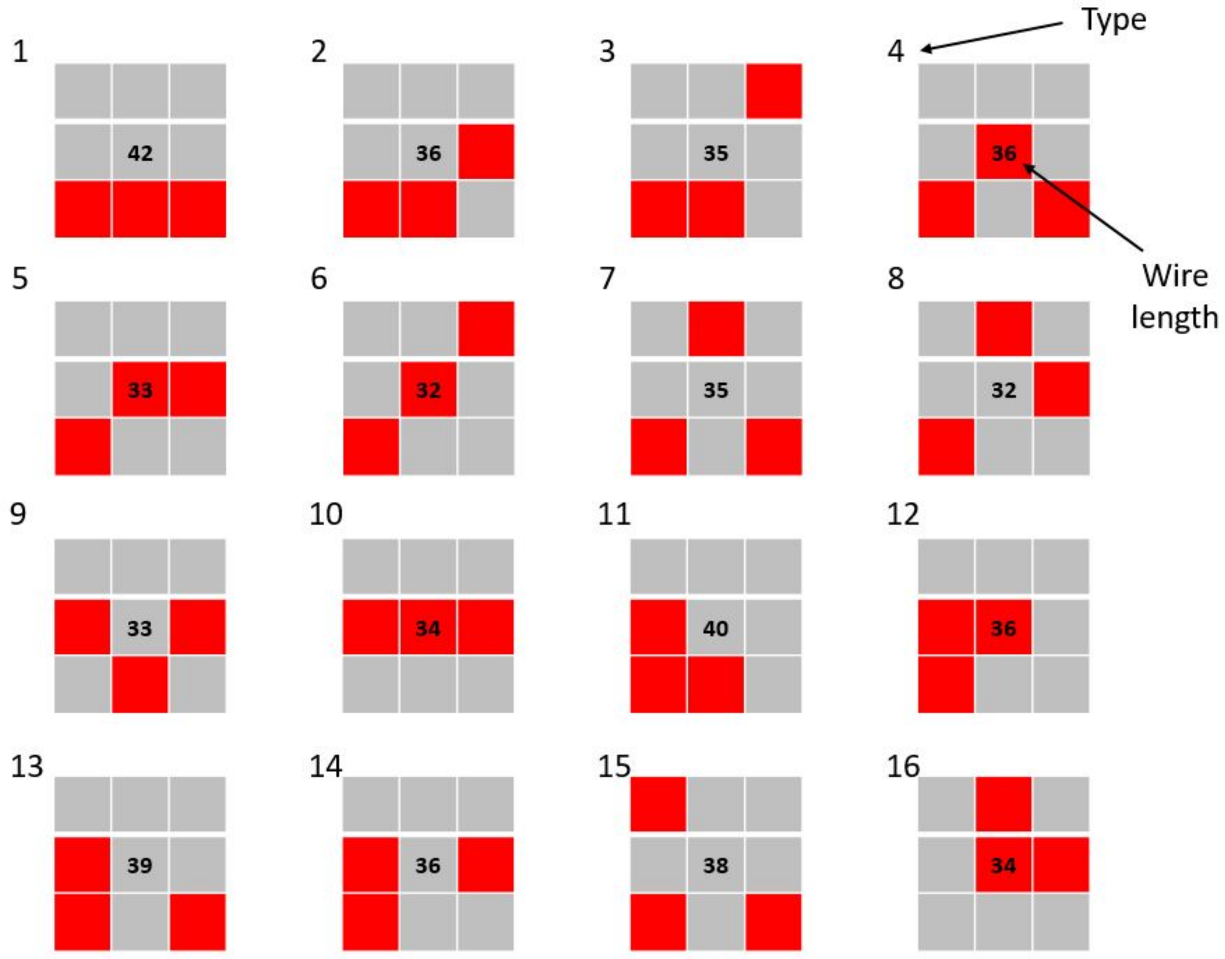
Therefore the approximated differentible function is:
$\hat{n}(x) = \sum_{e \in E} \{p(x - \frac{\sum_{v_i \in e} x_i exp(-x_i/\gamma)}{\sum_{v_i \in e} exp(-x_i/\gamma)}) \times p(\frac{\sum_{v_i \in e} x_i exp(x_i/\gamma)}{\sum_{v_i \in e} exp(x_i/\gamma)} - x)\}$

where p is the sigmoid function and γ is a user specified constant (smaller γ has smaller error).

The main difference between a normal placement and a ultra thin placement is that the prior focus on the bin density of the module to avoid overlaps, and the latter focus on wire density on every V cut line to achieve good routability.

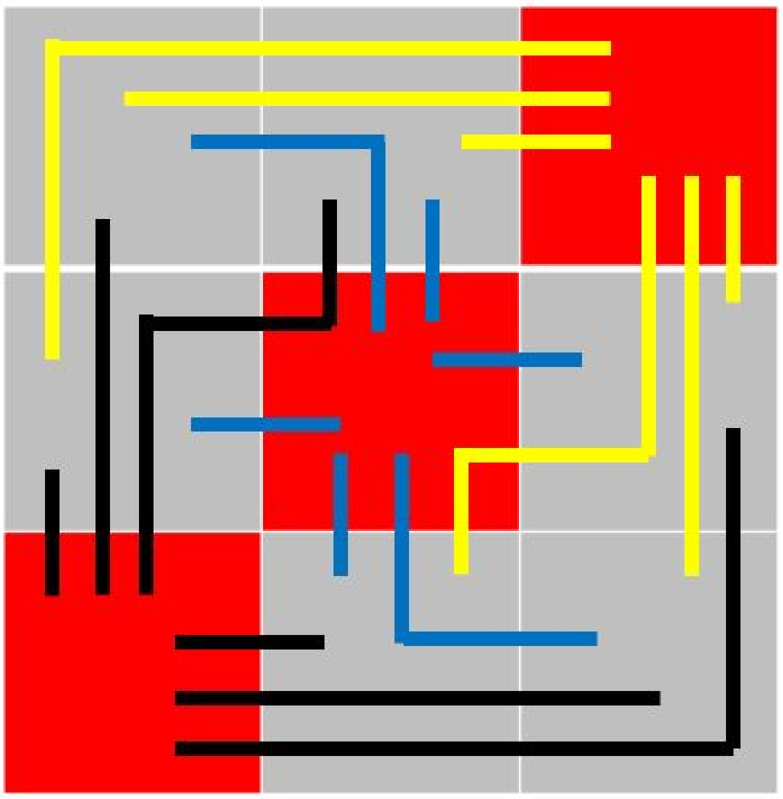
Problem 10 [[Collaborator: B05901053 吳宥璁]

a.
Wire length optimization
For placing a 3 layer neural network, we only need to consider the middle layer (the 2nd layer) and the side layers (the 1th & 3rd layer).
For 3x3 grids there are only 16 types of placing the 3 neuron in the middle layer (other situations is equivalent to one of them after rotating or flipping).
For listing and calculating wire length of whole 16 types, we can get the minimum wire length 32 (happens at type 6 and 8).



Routability optimization
There are 12 conceptual grid segments in a 3x3 grid, and we have shown that the minimum wire length is 32. Therfore, the minimax of the grid segments wire crossing (GSWC) must be greater then $\lceil 32/12 \rceil = 3$.

The following shows a condition that the max GSWC is 3, so the minimax of GSWC is 3 for a 3x3 grid geometry.



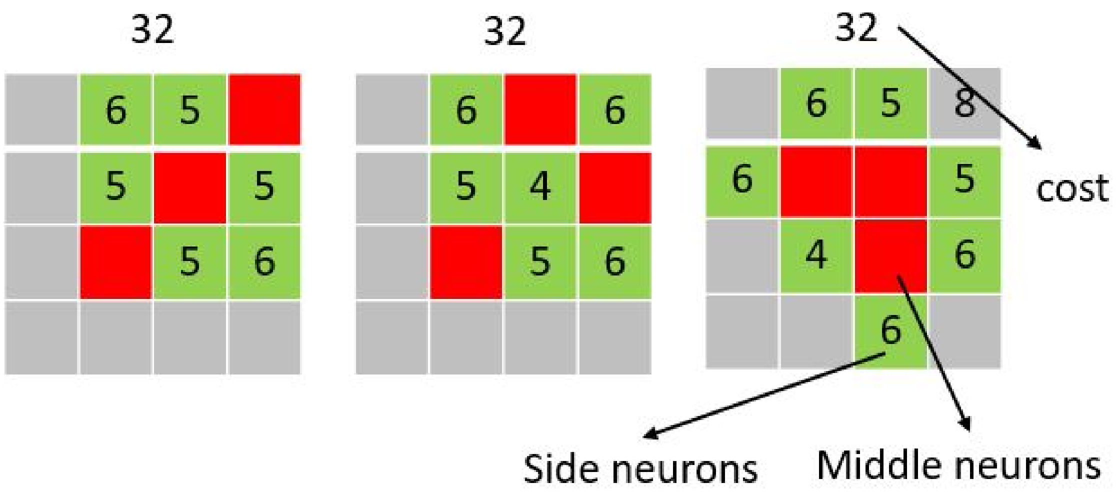
b.

Wire length optimization

Similar to the methods in **a.**, but we only consider the topology of the 3 neurons in the middle layer, instead of putting them into the 3x3 grids.

We can calculate the cost of grids that are adjancent to them. Then, find the 6 minimum values to put the side neurons.

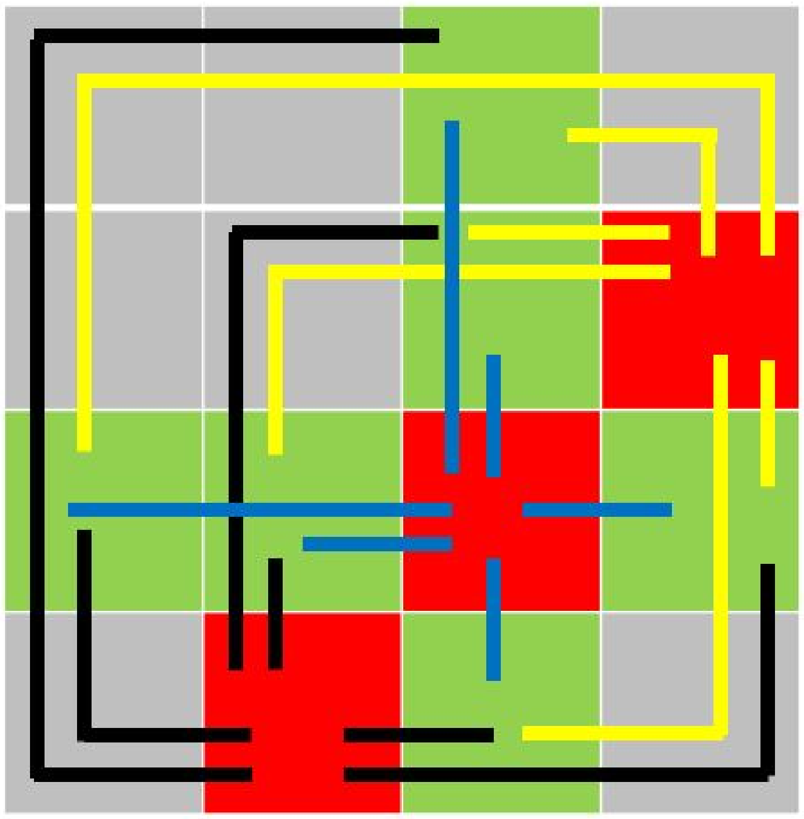
The following shows three topologies to achieve the optimal wire length 32.



Routability optimization

For a neuron in the middle layer, it is connected to 6 wires and each grid has 4 grid segments at max. Therfore, hte minimax GSWC must be greater then $\lceil 6/4 \rceil = 2$.

The following shows a condition that the max GSWC is 2, so the minimax of GSWC is 2 for a 4x4 grid geometry.

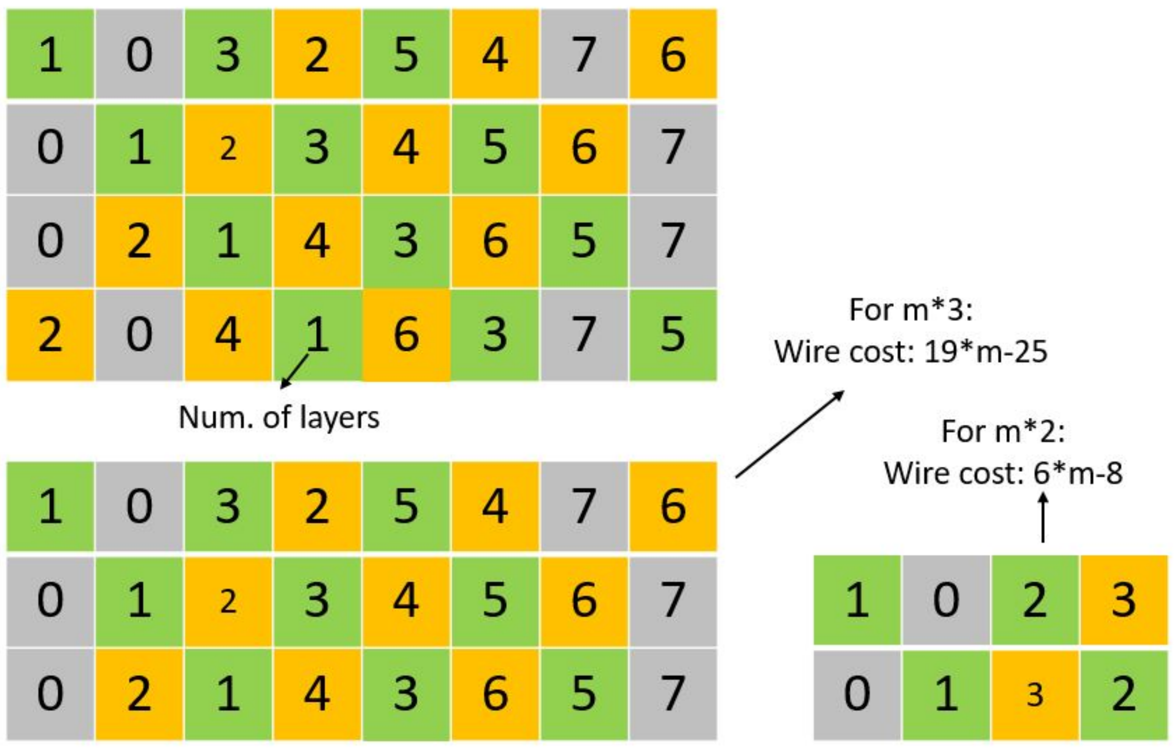


c.

Wire length optimization

For a mxn grid placements it is hard to list all the consequences. In order to place the middle layers near to its adjacent layers, our strategy is to place the cells in a interleaving scheme. Also, a interleaving placing strategy can make the structure more regular.

The following figure shows some examples. And the cost of the interleaving strategy can be easily computed into a formula.



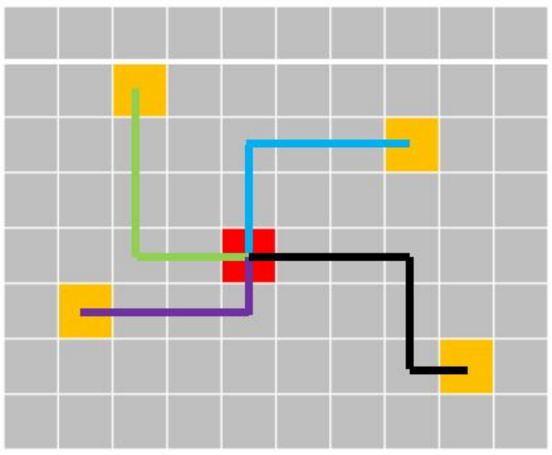
Routability optimization
Also, an interleaving placing strategy can also achieve great routablility due to its regularity.

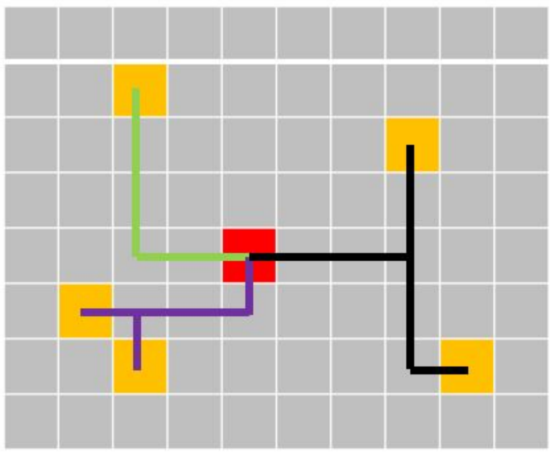
d.
Wire length optimization
Our strategy is to place them in the interleaving scheme in $m \times n$ grids, and leave the $m \times (n-m)$ empty.

Routability optimization
Place the adjacent layer agjancent in the grid, and seperate as far as we can.

Problem 11 [Collaborator: B05901053 吳宥璁]

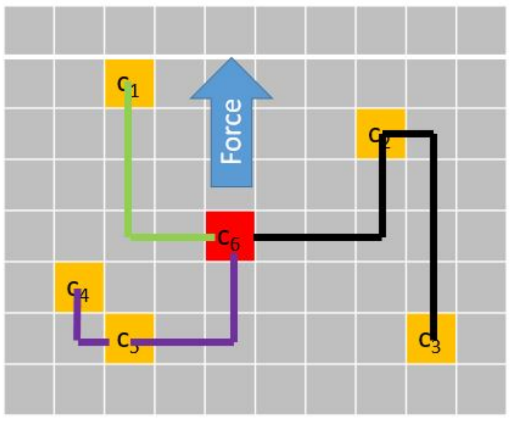
In the Force-Directed Method each movable cell is assumed to be directly connected to a pin (see the left figure below), but in real cases, a net can contain several pins (see the right figure below).


No multiple pin nets

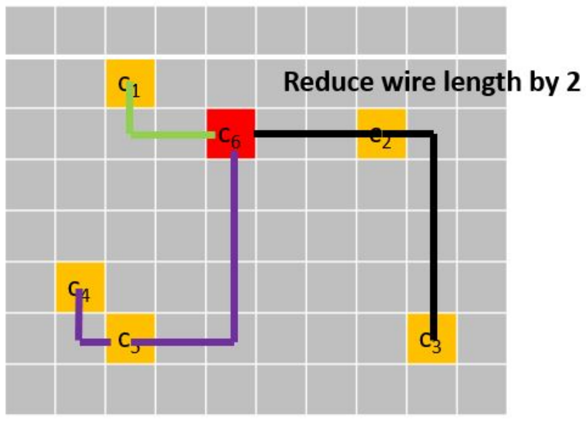

Exists multiple pin nets
Forced-Directed can't be applied

In this problem, we aim to find an algorithm to solve this problem efficienlty. First, all movable cells is randomly placed and the initial route is given by running a simple minimum spanning tree or Spanning-Graph-Based Steiner Minimal trees (Wire can we overlapped in this stage!). Notice that each movable cell can be connected to several nets, so moving a movable cell can increase or decrease wirelength for each connecting nets.

The following figure shows an example of our idea to modified the force directed method. In this example, c_4, c_5, c_6 is in the same net and c_2, c_3, c_6 is in the same net. These nets are initially connected by a minimum reclinear spanning tree solution and cell c_6 is the movable cell.



Step1.
Randomly placed the movable cell c_6
& connected each net with a RMST solution



Step2.
Apply Modified-Force-Directed method
to further reduce the wire length

Problem Abstraction:

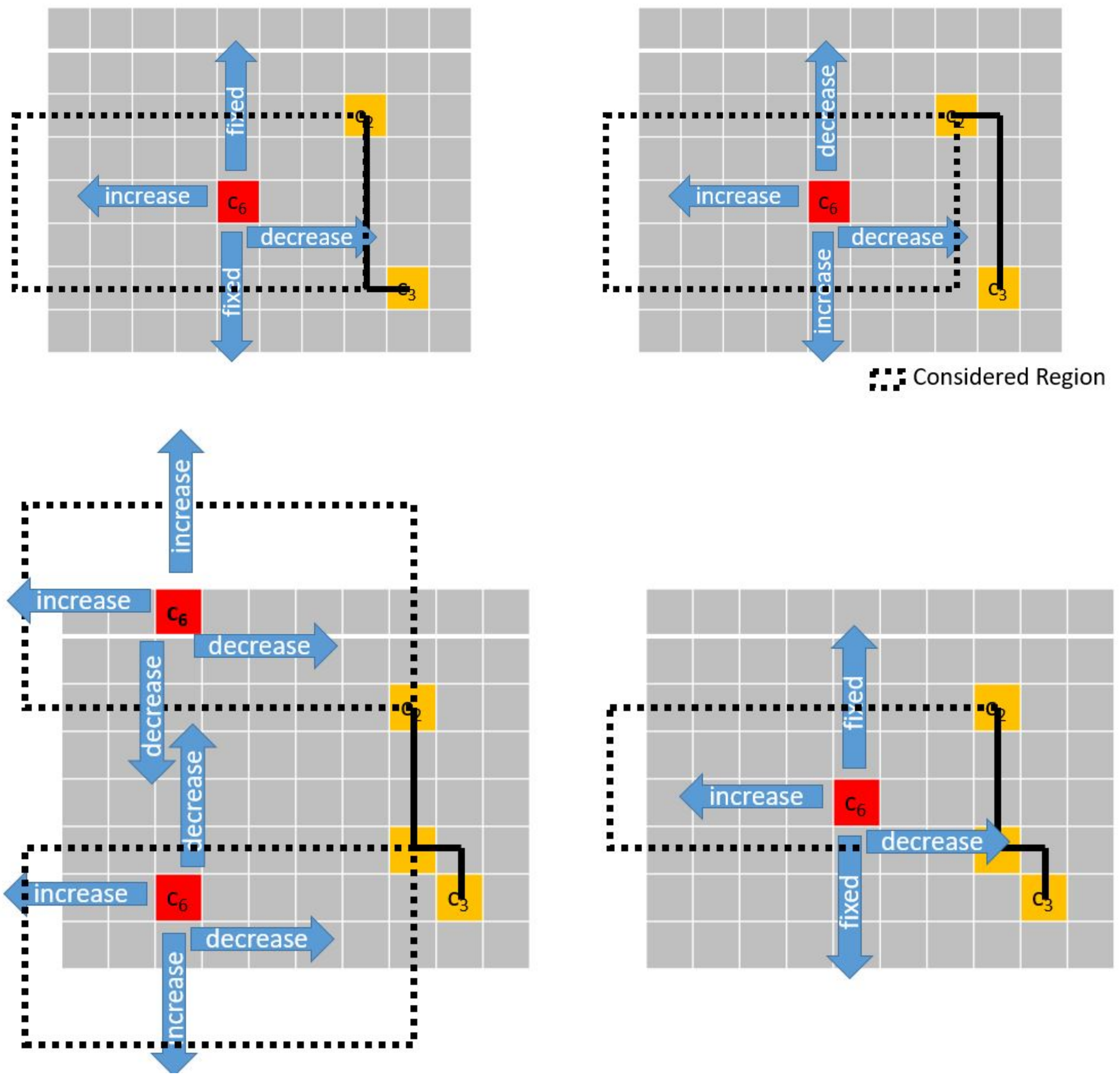
Given:
Fixed cells or pins: $c_1, c_2, ..., c_p$
Movalble cells: $m_1, m_2, ..., m_q$
Nets: $n_1, n_2, ..., n_r$

Notice that:
Each net is connected to multiple fixed or movable cells or pins.
Each cell or pin is connected to multiple nets.

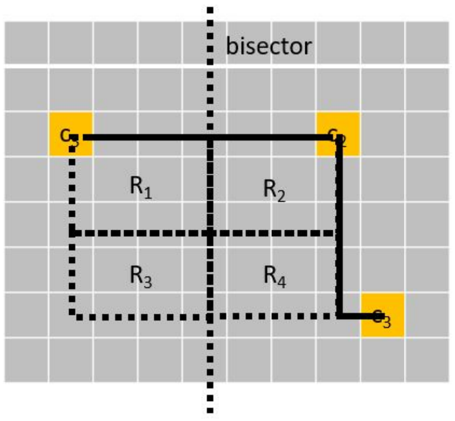
Find a heurictic method similar to Forced-directed's idea to **minimize the actual wirelength**.

Problem Restriction 1: (Only one movable cells in a net)

For the movable cell, we can determine its increment or decrement amount of wirelength by simply analysis its neighborhood geometry, we called this region the Considered Region (CR). Some simple example is shown in the following.



However, if the CR is simply divide by the pins location, there might be some problem for analyzing the wirelength. Therefore, we need to further divide the CR into 4 CRs by two bisectors of two directions (see the figure below).



Region	Up	Down	Left	Right
R_1	-	+	x	x
R_2	-	+	-	+
R_3	-	+	-	+
R_4	x	x	+	-

Therefore, for all CRs we can determine its increment and decrement amount of wirelength in four directions. Notice that the increment/decrement amount can be seen as a 2D surface.

Notice that directly divide the whole plane by the x and y value of all cells forms $(n_{nodes} - 1)^2$ region, which may result in high complexity. Instead, we divide the plane into only $(n_{nodes} * 3 + 1)$ regions by heuristic. Through this division, the optimal force directed will not be guaranteed.

The psuedo code for solving this restricted problem is as follows:

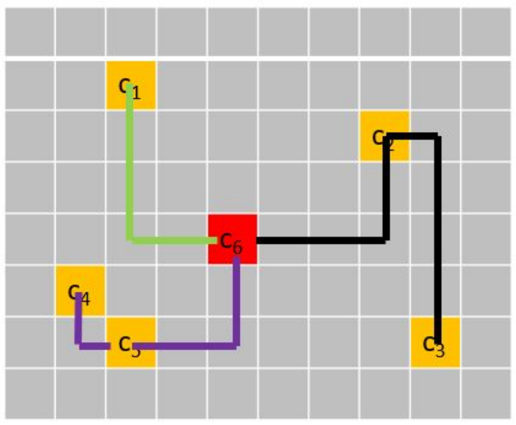
```
1. Sort all pins in increasing x and increasing y direction.
2. Divide the regions into CR regions by first divide the region in x direction
order, then divide the y direction by the pins that has the first smaller and
larger x values (by this method we divide the plane into (n_node*3+1) regions)
3. Further divide each CR into 4 CRs by its bisector (so (n_node*3+1)*4 regions)
4. The CRs is stored by a 2D linked-list structure


5. Determine the four direction increment/decrement amount or the
true wirelength
```

Problem Restriction 2: (Only one movable cells in multiple nets)

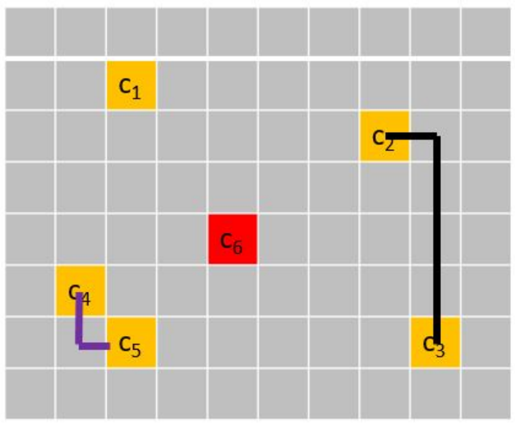
From problem restriction 1, we can see that for each net, we can create a 2D linked list to store its increment/decrement values.
Therfore, for mulitple nets problem, we only need to combine its 2D linked-list structured into one.
The method for combining is simply traverse the 2D list is alternatively switching between different lists in increasing c order (linear time).

The best moving cell is to move the cell to the place that has minimum wirelength, we can find it by simply traversing through the combined 2D linked list.

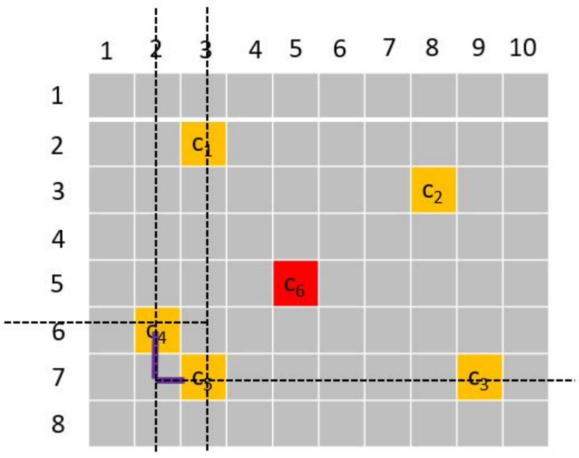
An exmaple is shown in the figure below:



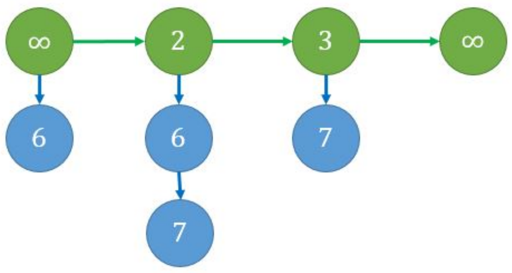
Step1.
Randomly placed the movable cell c_6
& connected each net with a RMST solution

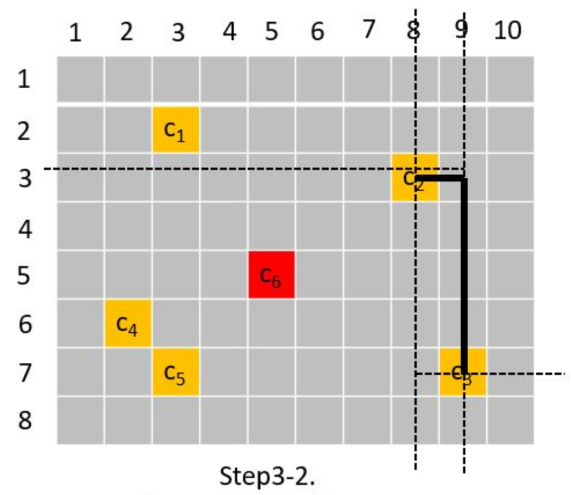


Step2.
Remove all wire connected to the movable cell

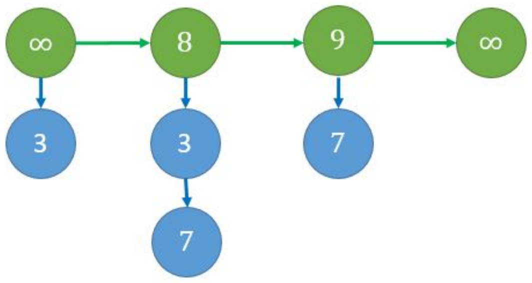


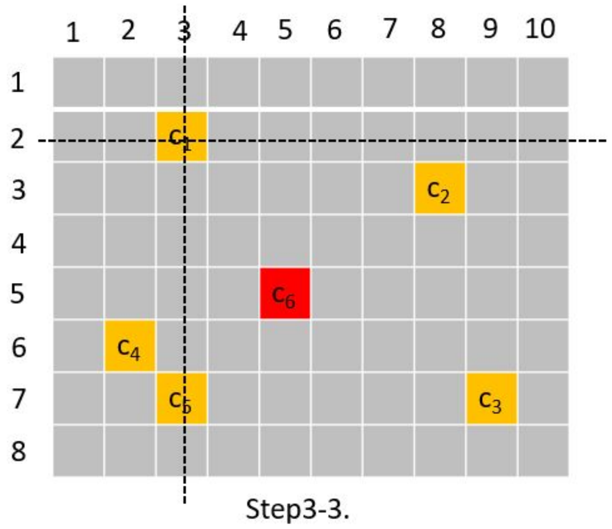
Step3-1.
Forms 2D list of 3 nets



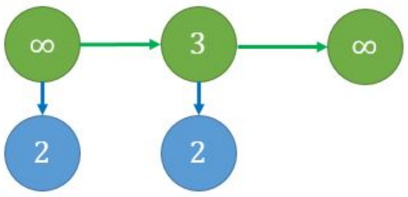


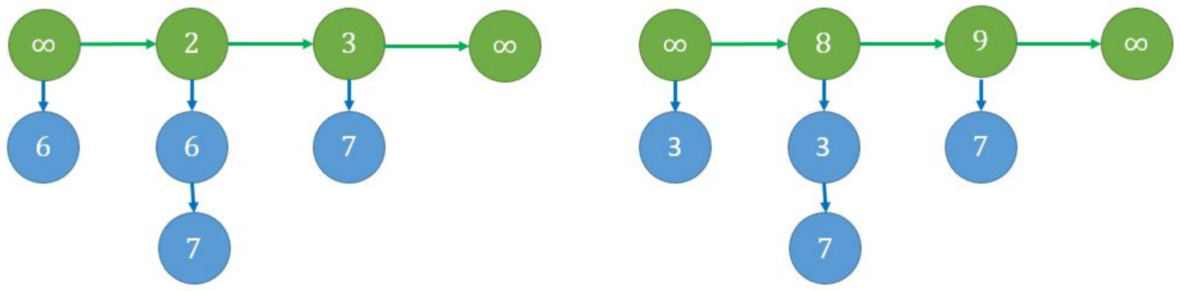
Step3-2.
Forms 2D list of 3 nets

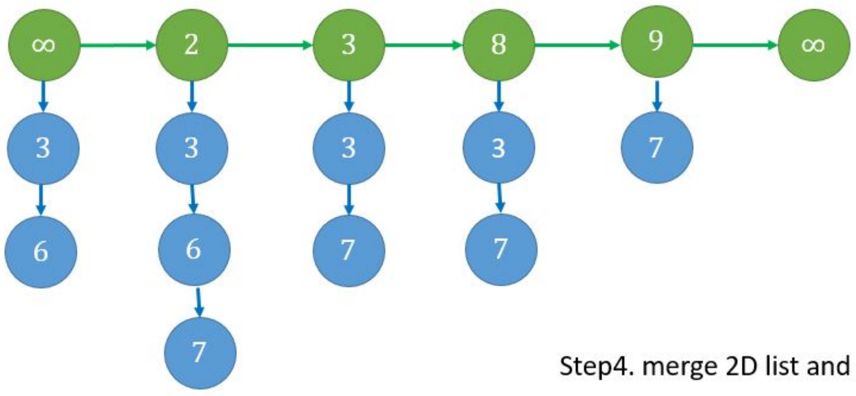




Step3-3.
Forms 2D list of 3 nets







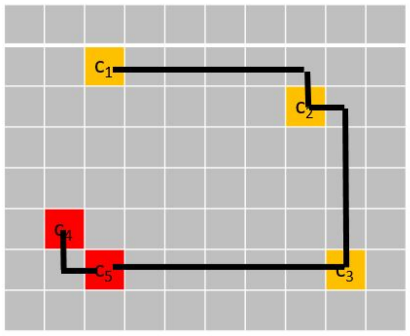
Step4. merge 2D list and combine its data

The psuedo code for solving this restricted problem is as follows:

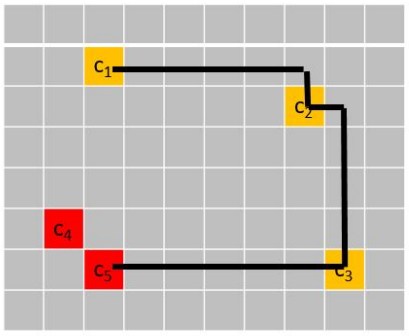
```
1. Apply algorithm in problem restriction 1 for all nets that contain
the movable cell
2. Merge the 2D linked list in an alternative switching scheme (the value in
the new node is the addition of the value all merge nodes)
```

Problem Restriction 3: (Multiple movable cells in multiple nets)

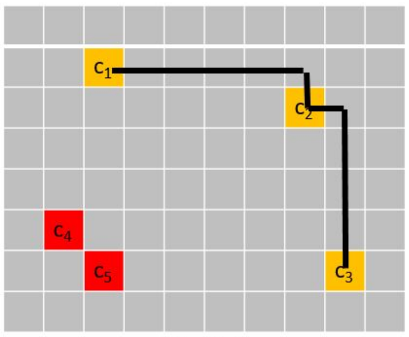
Directly applied algorithm from Restriction 2 for all movable cells may not ba a good idea, since jointly moving 2 or more cells may have larger decrement in the wirelength. For the example below, we can see that if we move c_4 close to the $\{c_1, c_2, c_3\}$ net, the wire length will decrease, but wirelength will not decrease as much since c_4 will be far away from c_5.



Step1.
Randomly placed the movable cell c_4, c_5 & connected each net with a RMST solution

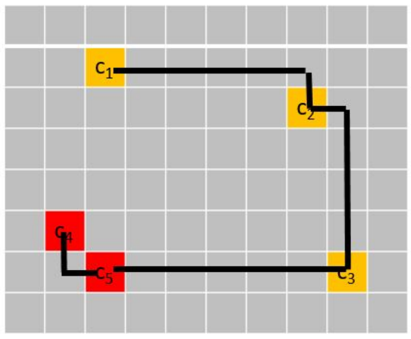


Step2-1.
Remove wire to c_4 & apply algorithm from problem restriction 2 to find the min. wire length

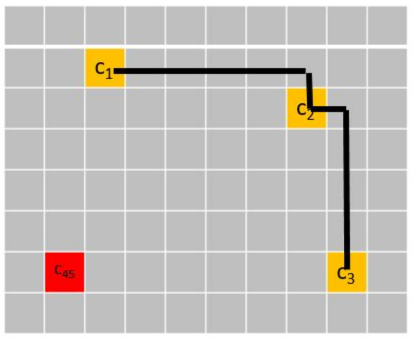


Step2-2.
Remove wire to c_5 & apply algorithm from problem restriction 2 to find the min. wire length

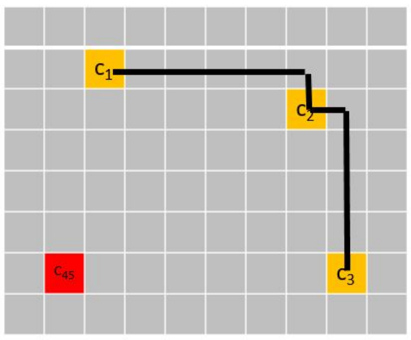
We proposed to tie multiple nodes together, and represent this cluster of nodes by a new aggregate node with higher wirelength decrement/increment weight. Moving the aggregate node can be see as moving the whole cluster. And the number of tying will be smoothly decrease in the iterative process. For the same example, we first merge nodes that are close into aggregate nodes. After few iterations, we seperate the aggregate node into the original one. Notice that the wirelength for the aggregate node will be weight by the number of nodes it represents.



Step1.
Randomly placed the movable cell c_4, c_5 & connected each net with a RMST solution



Step2.
Find movable cells that are close and tie them into an aggregate node



Step3.
Apply algorithm from problem restriction 2 to find the min. wire length of c_{45}, after several iterations, the aggregate node will then separate them into the original nodes

Without this tying process, a large number of movable cells that are close won't be able to move by the algorithm in problem restriction 2.

The psuedo code for solving this restricted problem is as follows:

```
1. Set paramters: max_aggregate_node_size, aggregate_node_size_decrease_rate,
   tying_distance, tying_distance_decrease_rate
2. Tie nodes based on tying_distance and max_aggregate_node_size
3. Apply algorithm in problem restriction 3 for all nets that contain movable cells
4. Determine the min k wirelength and move them
5. max_aggregate_node_size *= aggregate_node_size_decrease_rate
   tying_distance *= tying_distance_decrease_rate
6. Back to 2. if not converge
```