ANALYTICAL PLACEMENT FOR RECTILINEAR BLOCKS

Yasuhiro Takashima
University of Kitakyushu

Outline

1. Background
2. Rectilinear block placement
3. Related works
4. Proposed method
5. Experiments
6. Conclusion remarks

Background

- LSI production method: much improved
  - Large number of elements on one chip
- Design method: not so improved
- Gap between the production and the design: critical

- Using IP modules: much promising
  - Problem: Shape of IP modules may be rectilinear

- Fast rectilinear placement: important

Rectilinear blocks placement

- **[Input]** Set of blocks including Rectilinear blocks, Net-list, Chip outline
- **[Output]** Block placement
- **[Objective]** Minimization of total wire-length
- **[Constraint]** No overlap

Framework of previous works

* Basically, topological relation based
  * SP, BSG, B*-tree, ...
* Utilization of stochastic optimization
  * SA, ...

1. Divide the rectilinear block into a set of rectangles
2. Add constraints to remain the shape in the perturbation
3. Enhance the position calculation to recover the shape

Characteristic of topological relation based method

* **Pros:** No overlap guarantee
  * from the nature of topological relation

* **Cons:** Slow convergence
  * from the nature of stochastic optimization method

Fast placement method: Necessary

Analytical placement is promising

Analytical placement

- Utilization of the gradient of the objective function
  - Requirement: differentiable objective function
    → approximation of max and min functions
- Objective function: non-linear, in general
  - Requirement: no constraint
    → consideration of relaxed problem

Relaxed problem

- **[Input]** Set of blocks including Rectilinear blocks, Net-list, Chip outline
- **[Output]** Block placement
- **[Objective]** Minimization of total wire-length and less overlap

  - To obtain less overlap
    - Density function: widely used
      - hard to capture overlap among rectilinear blocks
    - Direct consideration: **Overlap Removable Length**

Approximation of max and min functions

- Objective function: max and min function included
  - total wire-length
  - overlap removable length
- max and min function: not differentiable
  - need to approximate them to be differentiable
- Log-Sum-Exponential (LSE) :

$$\mathrm{LSE} = t \log \sum_i e^{\frac{x_i}{t}}$$

  - **Pros:** Fast convergence
  - **Cons:** Numerical unstableness
  - Stable LSE [Funatsu, 2009]: Numerical stability

t: smoothing parameter

Stable-LSE (SLSE)

$$\mathrm{SLSE} = X_{\max} + t \log \sum_i e^{\frac{(x_i - X_{\max})}{t}}$$

where $X_{\max}$ is the maximum number of $\{x_i\}$

- for all i, $x_i - X_{\max} \leq 0$
- at least one i, $x_i - X_{\max} = 0$

thus, <u>SLSE is numerical stable</u>

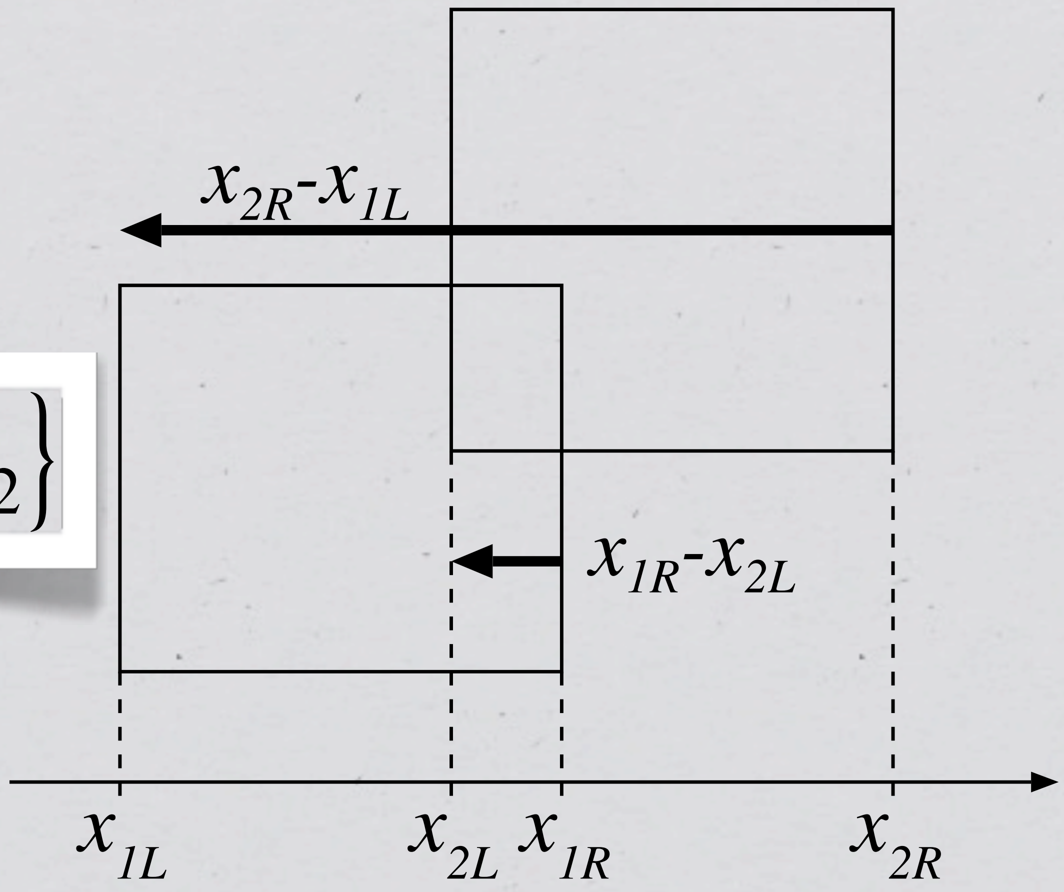
Overlap Removable Length (ORL)

- Calculate the necessary length to remove the overlap
  - for x-coordinate:

    $$\mathrm{ORL}^{X}_{1,2} = \max\{\min\{x_{1R} - x_{2L}, x_{2R} - x_{1L}\}, 0\}$$

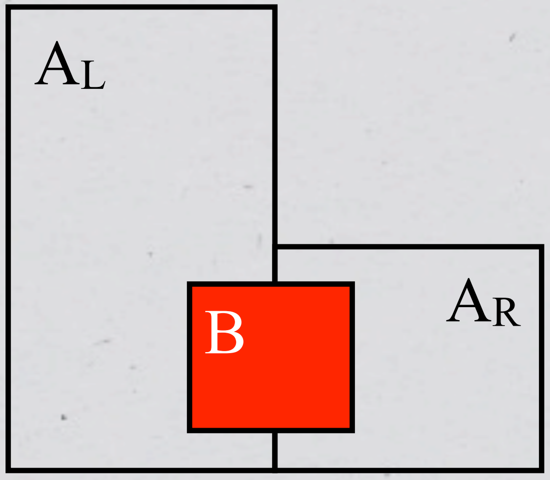
  - similar way for y-coordinate
  - ORL for block 1 and 2:

$$\mathrm{ORL}_{1,2} = \min\left\{\mathrm{ORL}^{X}_{1,2}, \mathrm{ORL}^{Y}_{1,2}\right\}$$



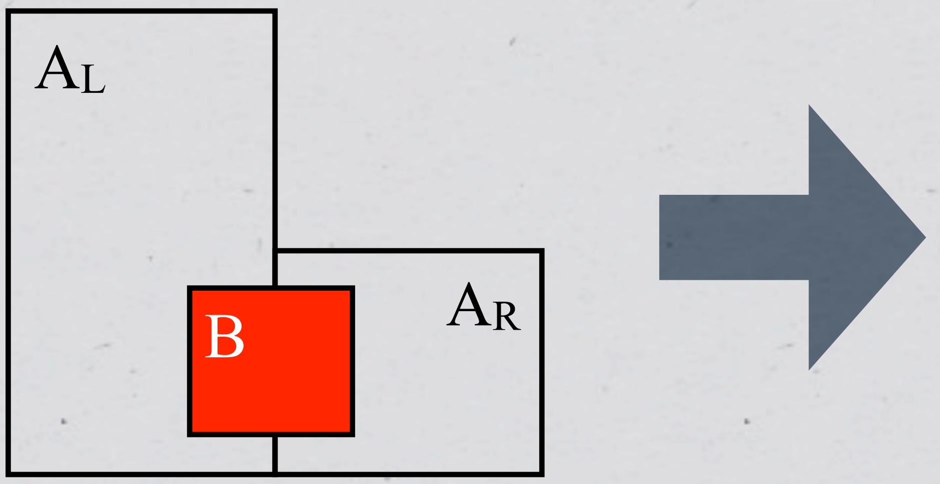
Utilization of ORL: rectangle decomposition

- Similar to topological relation based method
  - Rectilinear block → a set of rectangle blocks
  - seems to be easy to handle them
  - however, there is a main drawback:



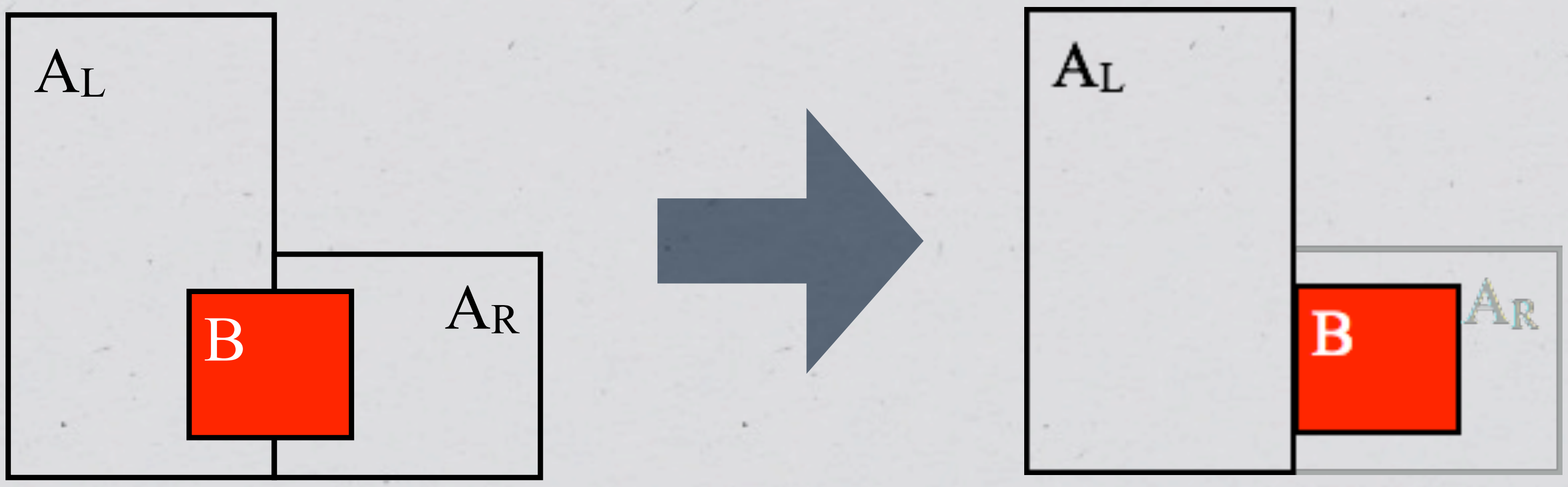
Utilization of ORL: rectangle decomposition

- Similar to topological relation based method
  - Rectilinear block → a set of rectangle blocks
  - seems to be easy to handle them
  - however, there is a main drawback:



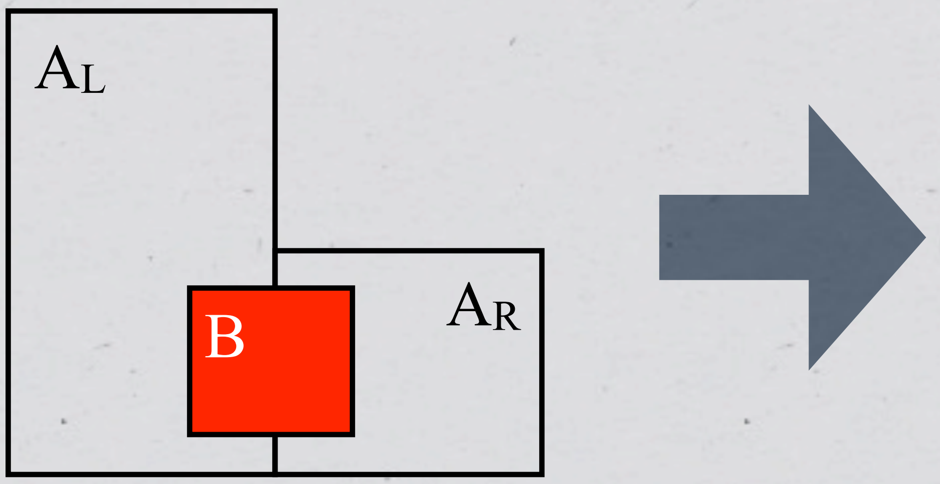
Utilization of ORL: rectangle decomposition

- Similar to topological relation based method
  - Rectilinear block → a set of rectangle blocks
  - seems to be easy to handle them
  - however, there is a main drawback:



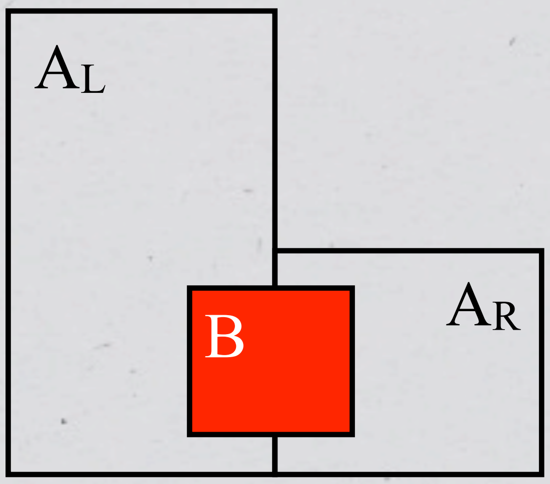
Utilization of ORL: rectangle decomposition

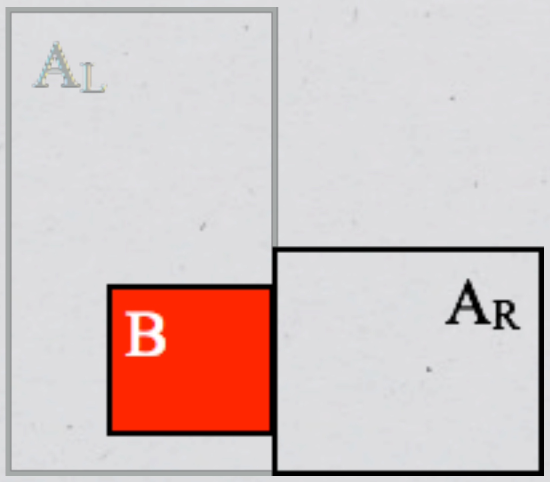
- Similar to topological relation based method
  - Rectilinear block → a set of rectangle blocks
  - seems to be easy to handle them
  - however, there is a main drawback:



Utilization of ORL: rectangle decomposition

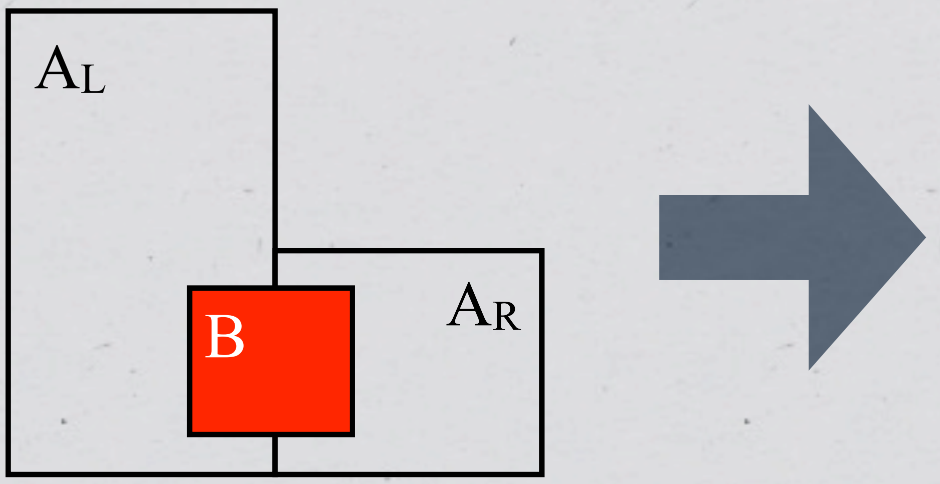
- Similar to topological relation based method
  - Rectilinear block → a set of rectangle blocks
  - seems to be easy to handle them
  - however, there is a main drawback:





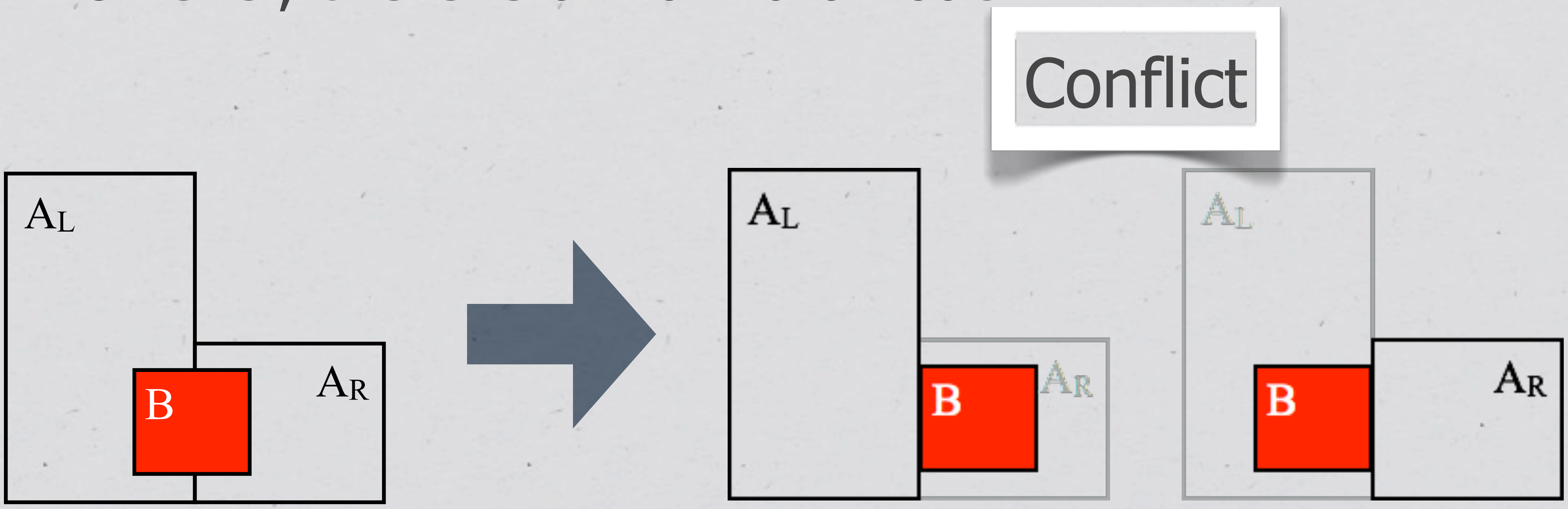
Utilization of ORL: rectangle decomposition

- Similar to topological relation based method
  - Rectilinear block → a set of rectangle blocks
  - seems to be easy to handle them
  - however, there is a main drawback:



Utilization of ORL: rectangle decomposition

- Similar to topological relation based method
  - Rectilinear block → a set of rectangle blocks
  - seems to be easy to handle them
  - however, there is a main drawback:



Analysis of ORL

- from the definition of ORL:

$$\mathrm{ORL}_{1,2} = \min\left\{\mathrm{ORL}_{1,2}^{X}, \mathrm{ORL}_{1,2}^{Y}\right\}$$

  - Equivalent to the minimum length to remove overlap for any directions
    - for rectangles, consideration of only x- and y-directions is enough
- For rectilinear blocks, enhance ORL to seek the minimum length for any directions

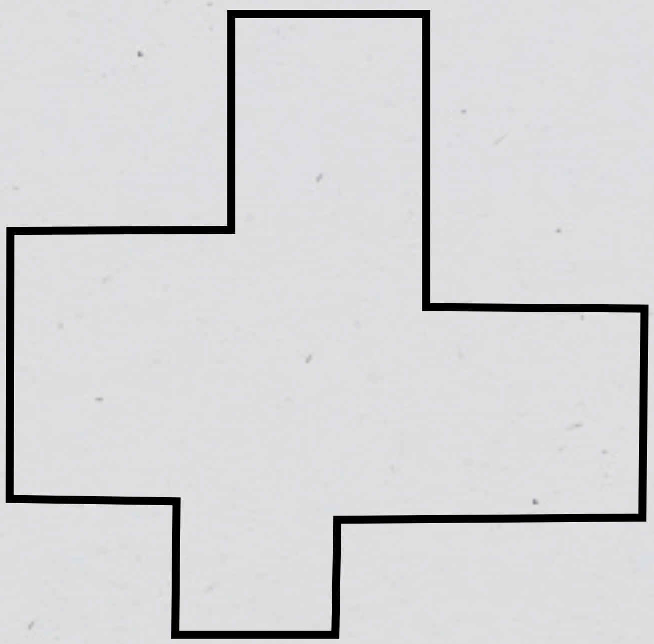
Enhancement of ORL

Pre-process

1. Calculate each side profile of each block
2. For each block pair, enumerate the non-overlap conditions

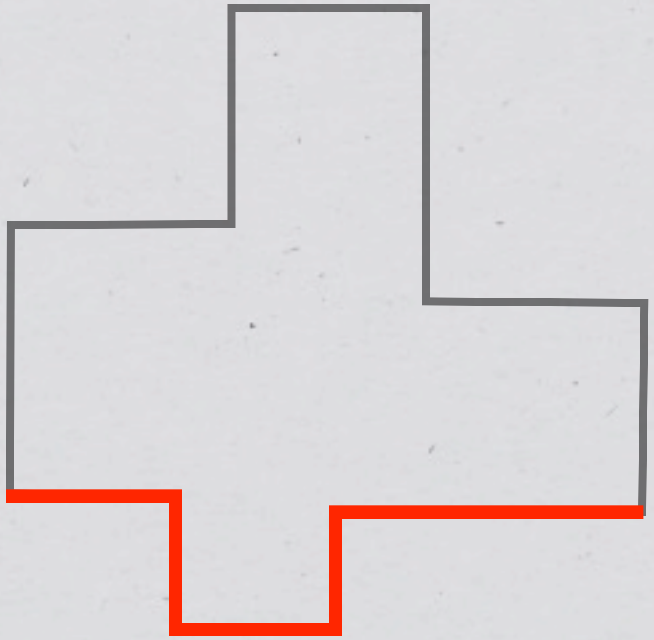
Side profile

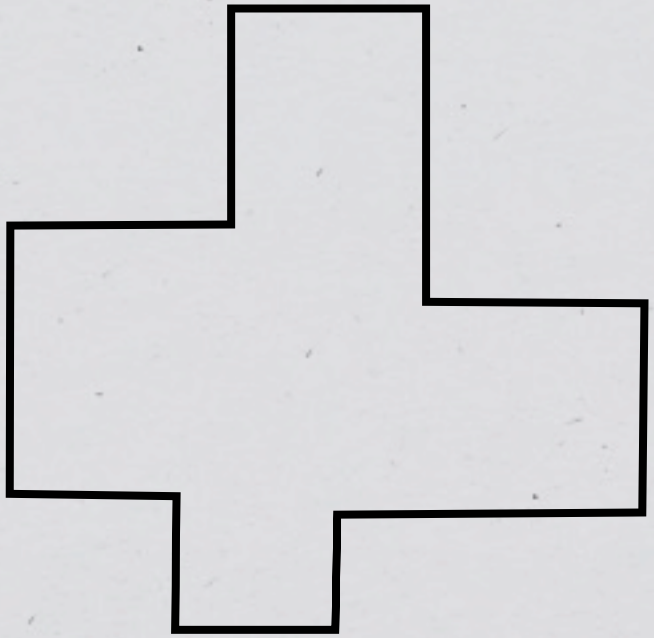
* Outer shape of each side



Side profile

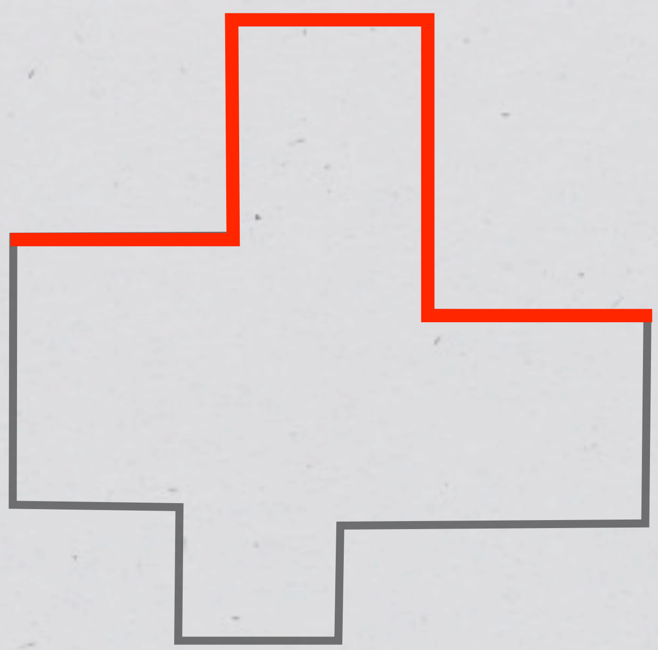
* Outer shape of each side
  * Bottom profile

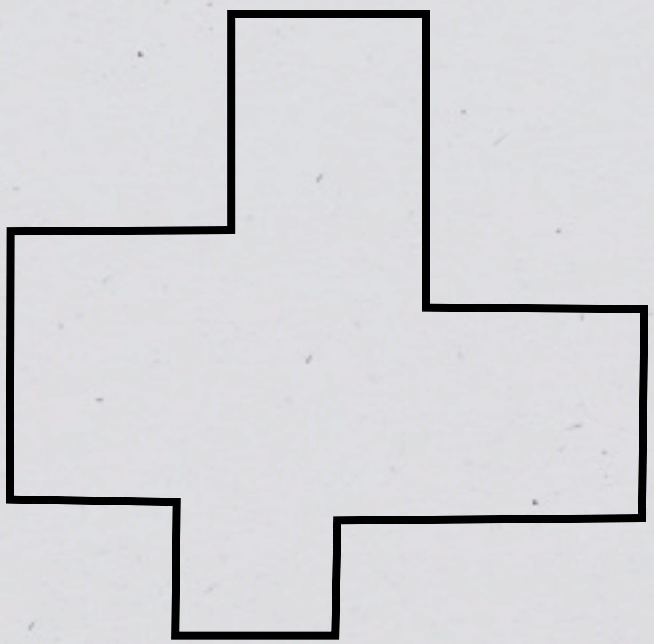


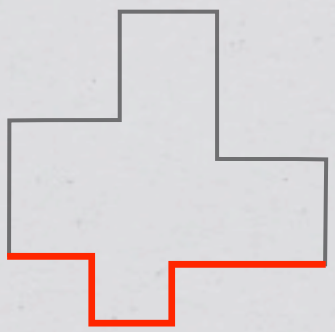


Side profile

- Outer shape of each side
  - Bottom profile
  - Top profile

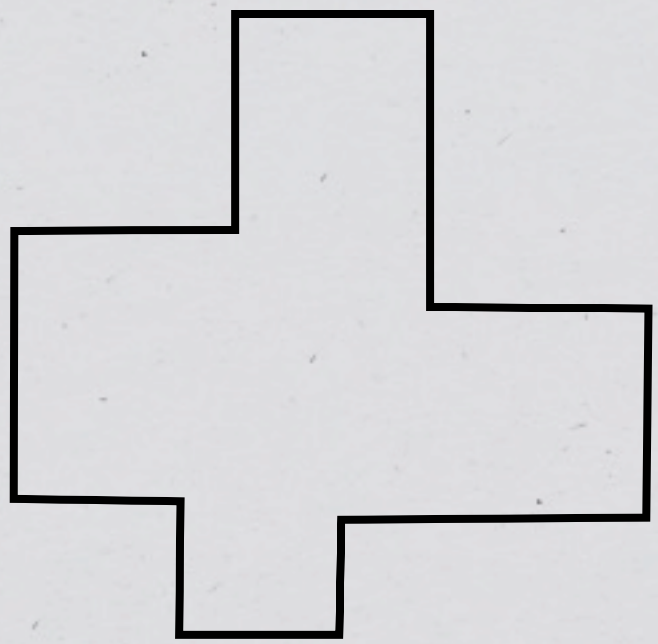


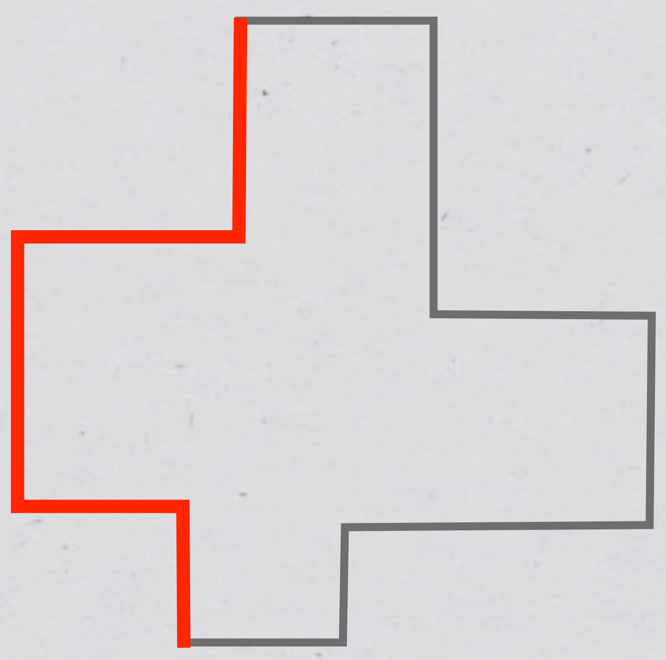


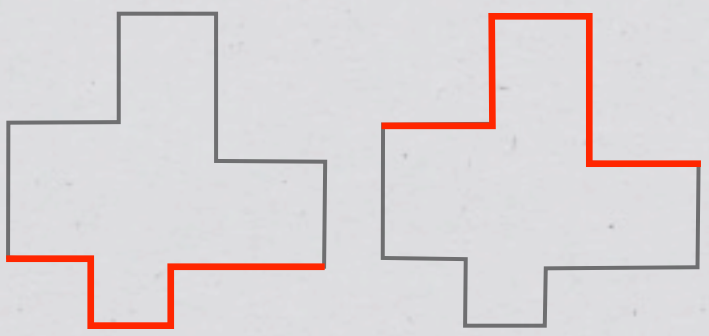


Side profile

- Outer shape of each side
  - Bottom profile
  - Top profile
  - Left profile

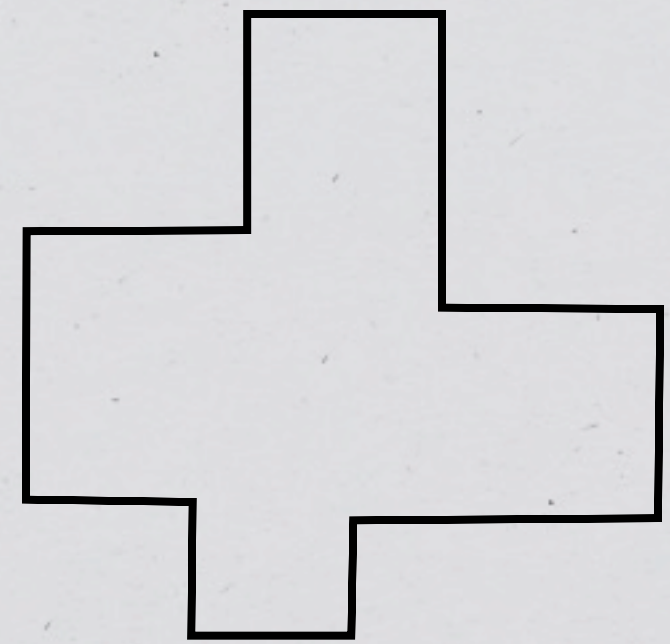


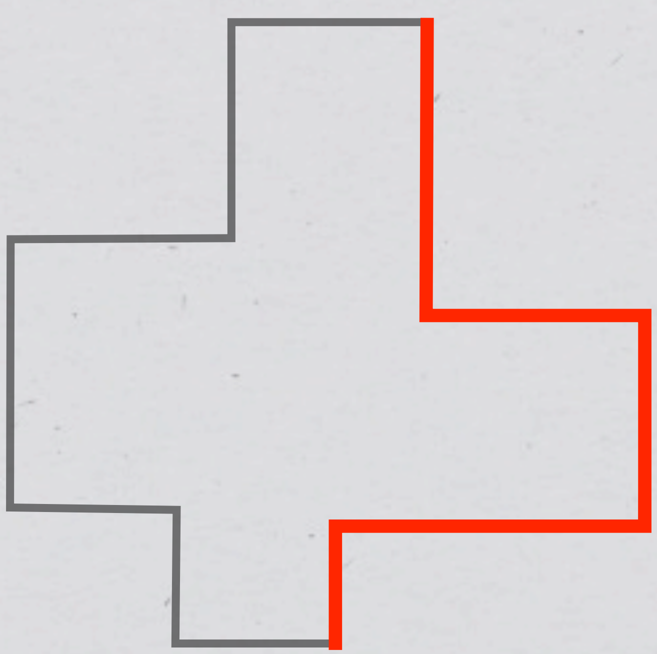


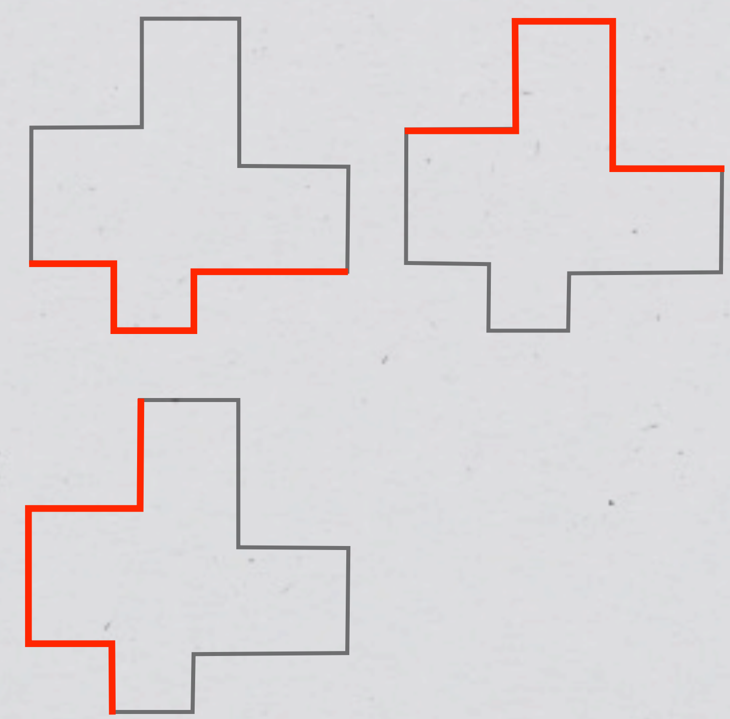


Side profile

- Outer shape of each side
  - Bottom profile
  - Top profile
  - Left profile
  - Right profile



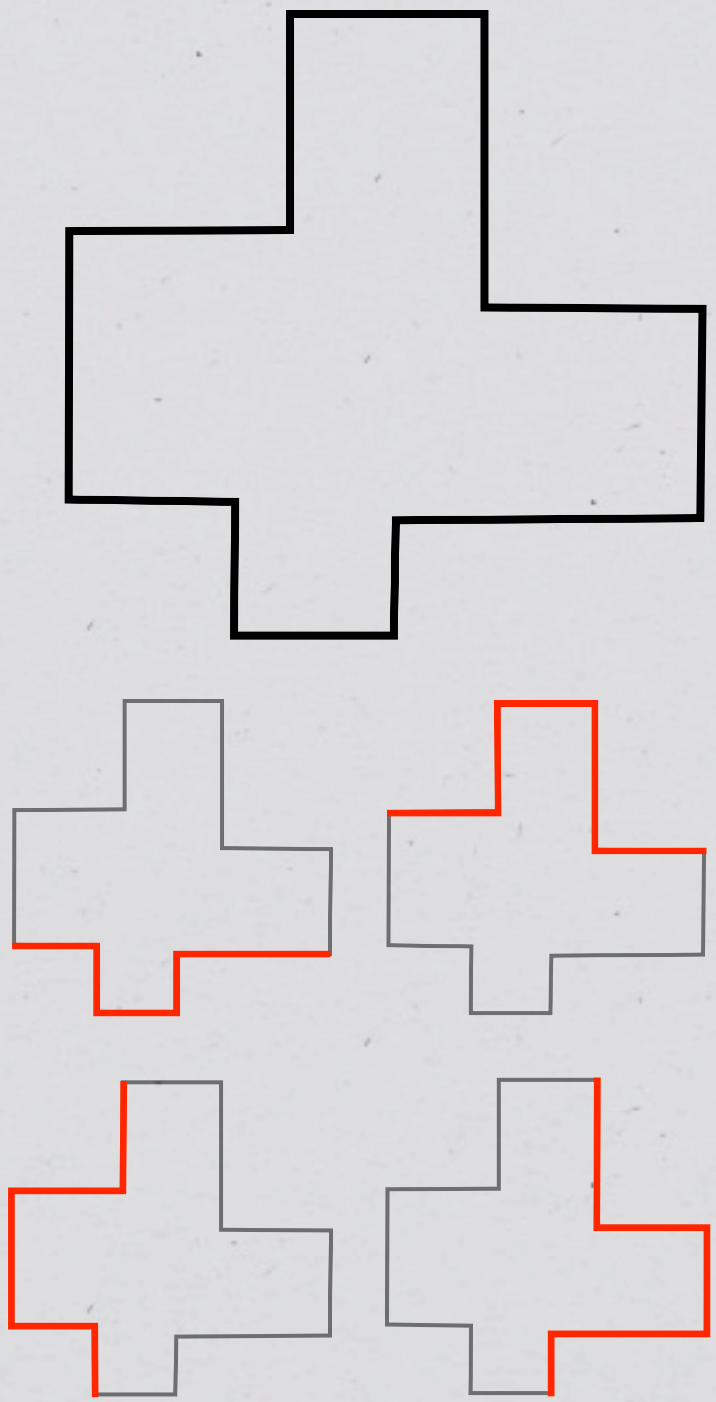




Side profile

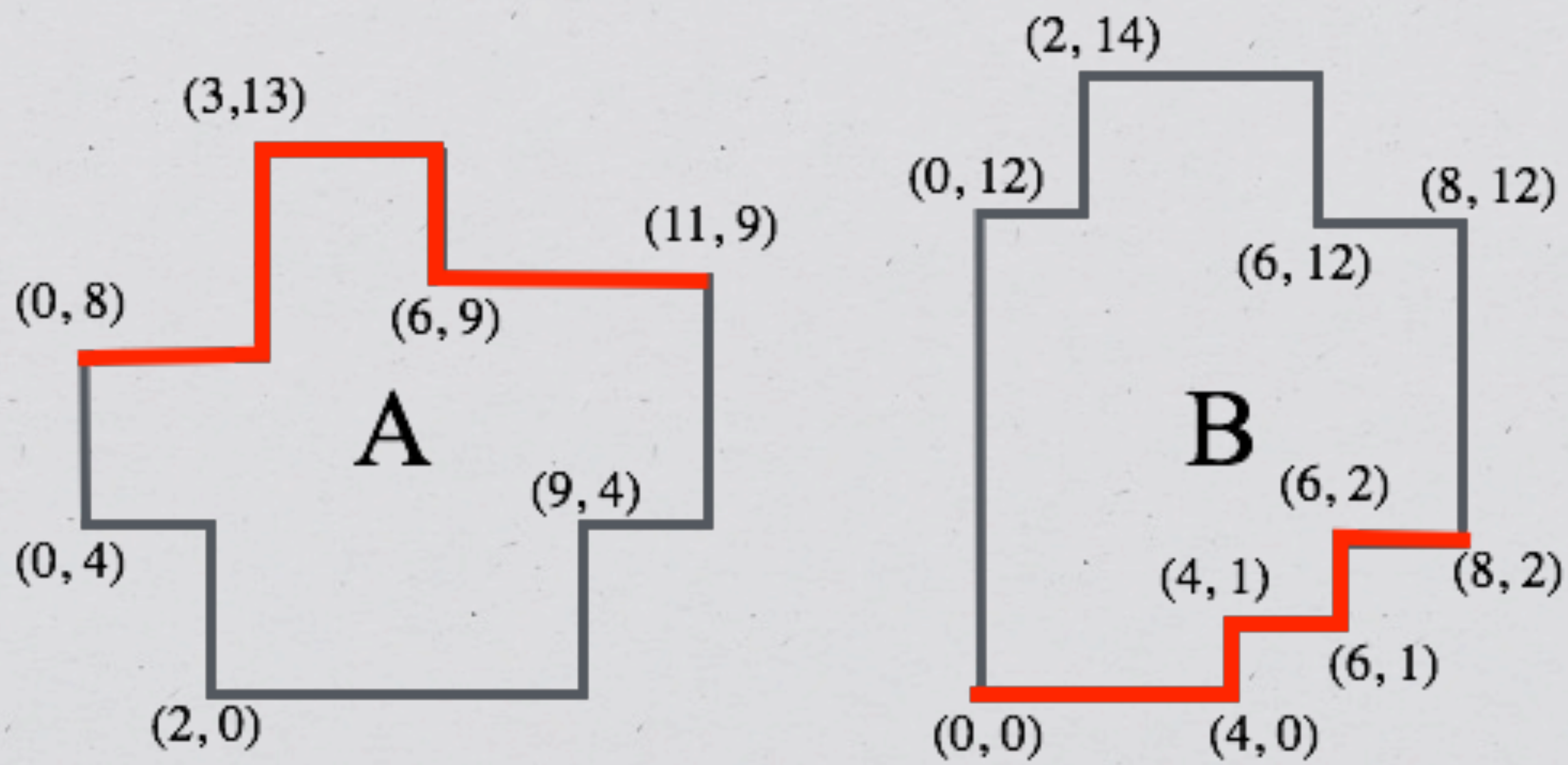
* Outer shape of each side
  * Bottom profile
  * Top profile
  * Left profile
  * Right profile

  * Complexity: $O(m)$
    * m: # corners



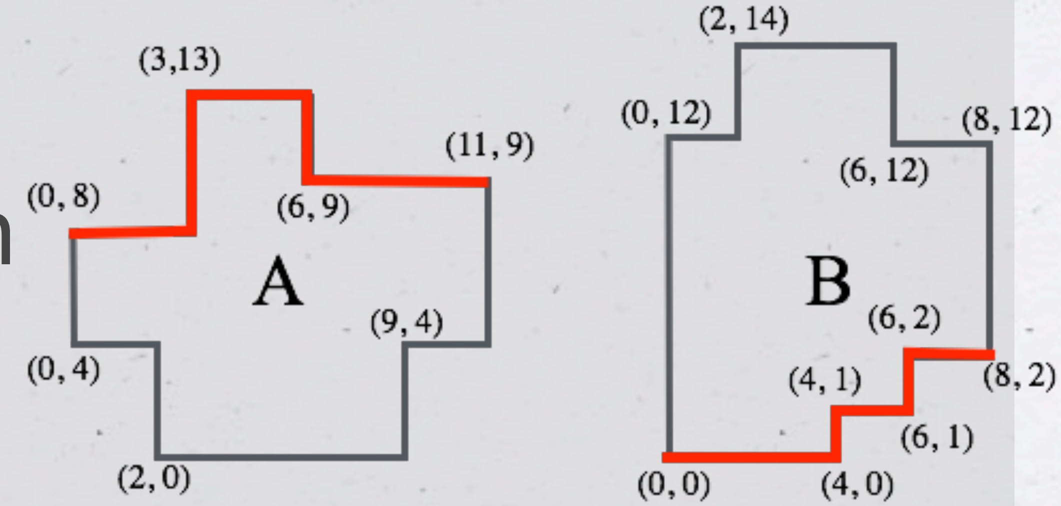
Enumerate Conditions

- Non-overlap conditions:
  - represented by relative position



Enumerate Conditions

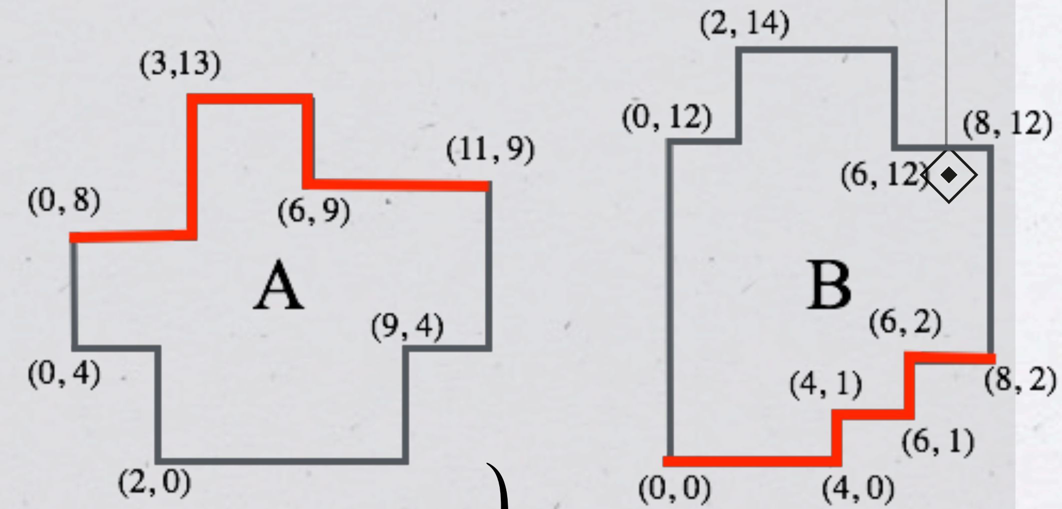
- Non-overlap conditions:
  - represented by relative position



$$
\begin{aligned}
\Delta_x &< -8, \\
-8 \leq \Delta_x &< -6, \Delta_y \geq 6, \\
-6 \leq \Delta_x &< -5, \Delta_y \geq 7, \\
-5 \leq \Delta_x &< -3, \Delta_y \geq 11, \\
-3 \leq \Delta_x &< -1, \Delta_y \geq 12, \\
-1 \leq \Delta_x &< 6, \Delta_y \geq 13, \\
6 \leq \Delta_x &< 11, \Delta_y \geq 9, \\
11 \leq \Delta_x &
\end{aligned}
$$

$$
\text{where } \begin{cases} \Delta_x = x_B - x_A, \text{ and} \\ \Delta_y = y_B - y_A \end{cases}
$$

ORL



$$
\min \left\{ \begin{array}{l}
\max\{\Delta_x + 8, 0\}, \\
\max\{\min\{-\Delta_x - 8, \Delta_x + 6\}, 0\} + \max\left\{-\Delta_y + 6, 0\right\}, \\
\max\{\min\{-\Delta_x - 6, \Delta_x + 5\}, 0\} + \max\left\{-\Delta_y + 7, 0\right\}, \\
\max\{\min\{-\Delta_x - 5, \Delta_x + 3\}, 0\} + \max\left\{-\Delta_y + 11, 0\right\}, \\
\max\{\min\{-\Delta_x - 3, \Delta_x + 1\}, 0\} + \max\left\{-\Delta_y + 12, 0\right\}, \\
\max\{\min\{-\Delta_x - 1, \Delta_x - 6\}, 0\} + \max\left\{-\Delta_y + 13, 0\right\}, \\
\max\{\min\{-\Delta_x + 6, \Delta_x - 11\}, 0\} + \max\left\{-\Delta_y + 9, 0\right\}, \\
\max\{\min\{-\Delta_x - 8, \Delta_x + 6\}, 0\} + \max\left\{\Delta_y + 8, 0\right\}, \\
\max\{\min\{-\Delta_x - 6, \Delta_x - 7\}, 0\} + \max\left\{\Delta_y + 14, 0\right\}, \\
\max\{\min\{-\Delta_x + 7, \Delta_x - 11\}, 0\} + \max\left\{\Delta_y + 8, 0\right\}, \\
\max\{-\Delta_x + 11, 0\}
\end{array} \right\}
$$

Optimization framework

- Objective function: $\sum \mathrm{WL} + \alpha\left(\sum \mathrm{ORL}^2 + \sum \mathrm{ORL\_wcb}\right)$

  where
  - WL: wire length with HPWL
  - ORL: Overlap removable length
  - ORL_wcb: Overlap removable length with chip boundary

Optimization flow

1. Construct an initial placement randomly.
2. Set the smoothing parameter $t = 10$.
   2.1. Optimize the placement with $\alpha = 0$.
   2.2. Optimize the placement with larger α, iteratively.
3. Set the smoothing parameter $t = 0.01$.
   3.1. Optimize the placement with larger α, iteratively.

Experiments

* CPU: Intel Core i5-4570, 3.2GHz
* Memory: 4GB
* OS: LinuxMint 17 (qiana)
* gcc: version 4.8.2
* Non-linear programming solver: liblbfgs 1.10
* Benchmarks

name	#blocks	#rectilinear blocks	#nets
n100	100	10	885
n200	200	156	1585
n300	300	243	1893

Experimental results

* Average of 100 trials
  * WL Comparison: Rectangle placement with B*-tree
    * n100: 32.06, n200: 58.33, n300 71.00

name	ORL	WL	Runtime [sec]
n100	0.204	32.44	5.98
n200	0.248	64.05	56.95
n300	0.770	73.25	132.41

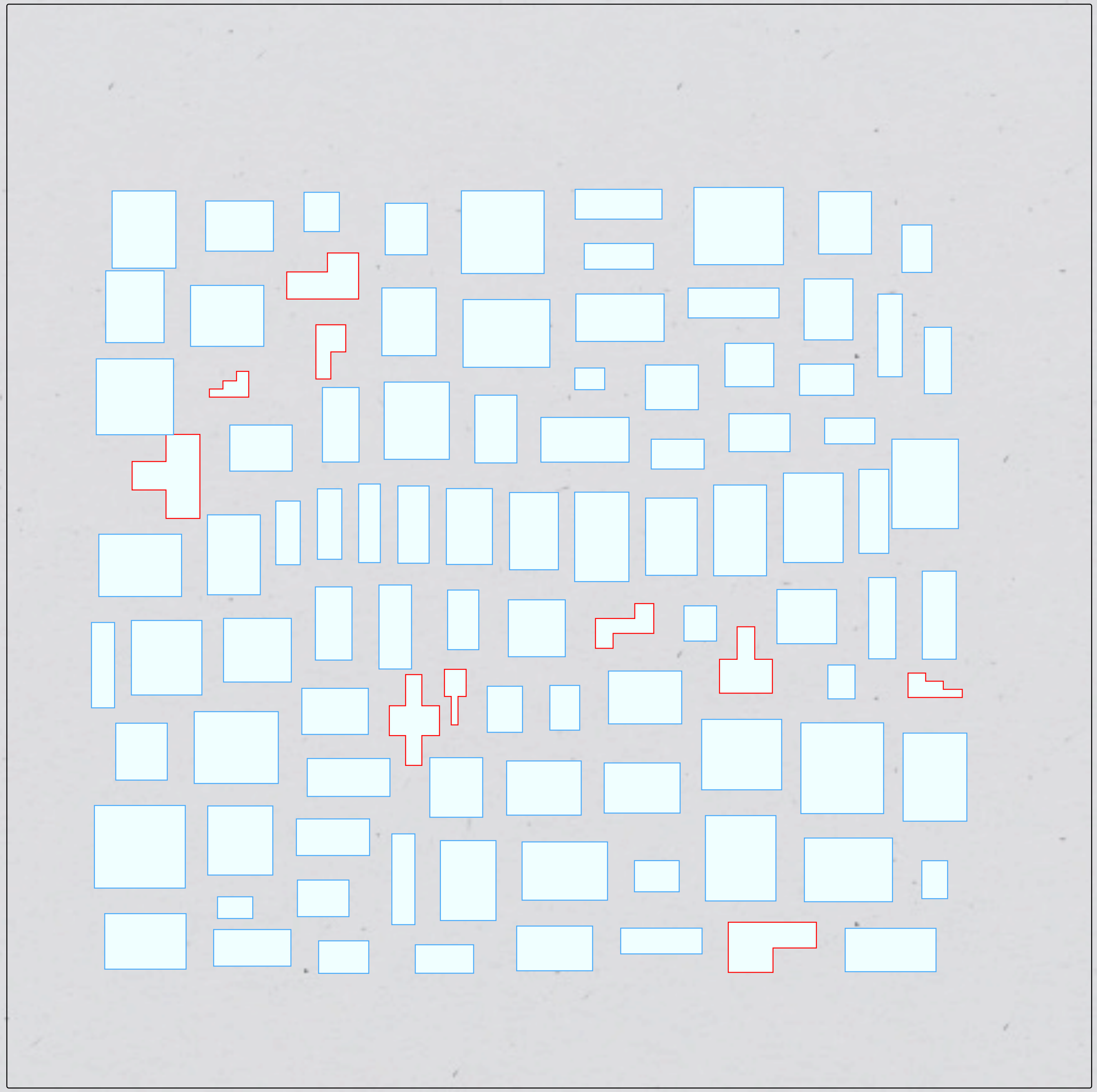
Experimental results

* Average of 100 trials
  * WL Comparison: Rectangle placement with B*-tree
    * n100: 32.06, n200: 58.33, n300 71.00

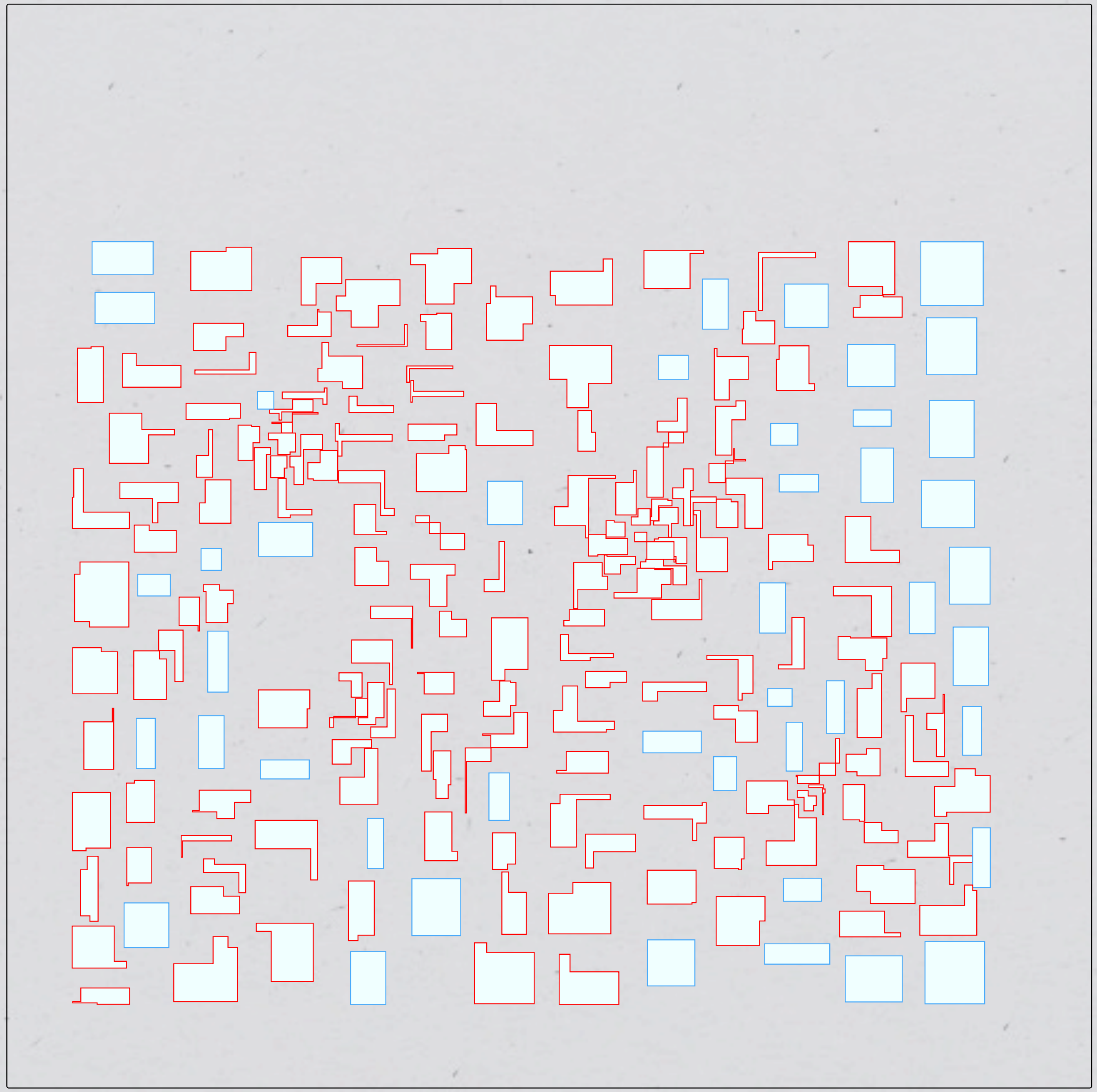
name	ORL	WL	Runtime [sec]
n100	0.204	32.44	5.98
n200	0.248	64.05	56.95
n300	0.770	73.25	132.41

Efficient

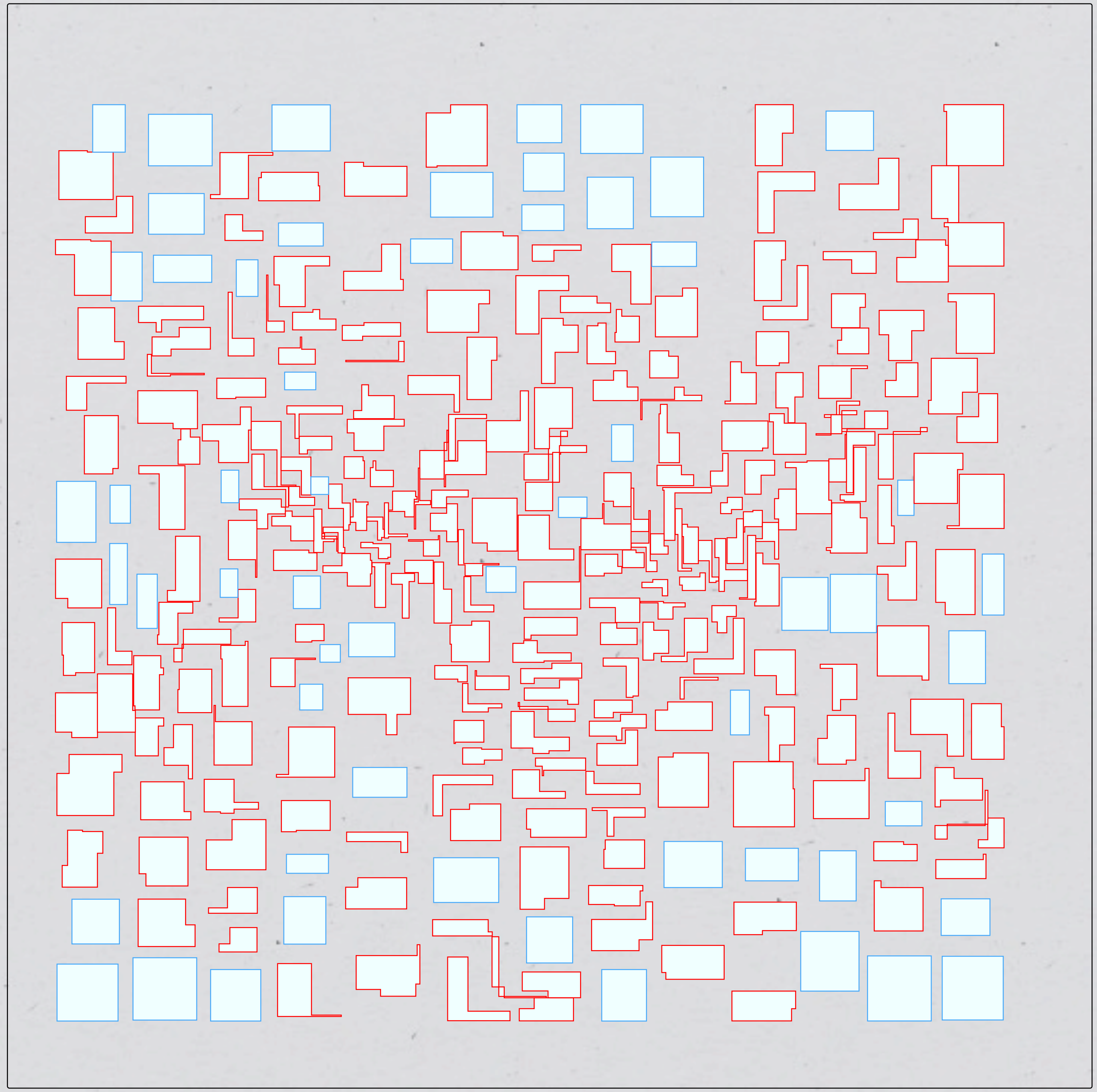
n100



n200



n300



Conclusions

- Proposition of the analytical placement for the rectilinear blocks.
  - to remove overlap: ORL
  - max and min approximation: Stable-LSE
- Confirmation of the proposed method, empirically

Future works

* Refinement of the speed
* Consideration of the routability
* Application to other problems
  * the proposed method: not limited rectilinear blocks
    * applicable to the problem representing the non-overlap conditions with the differentiable functions