EE582
Physical Design Automation of VLSI Circuits and Systems

Prof. Dae Hyun Kim
School of Electrical Engineering and Computer Science
Washington State University

Floorplanning

What We Will Study

- Floorplanning
  - Problem definition
  - Deterministic algorithms
    - Linear-programming
  - Stochastic algorithms
    - Simulated-annealing
      - Polish expression
      - Sequence pair

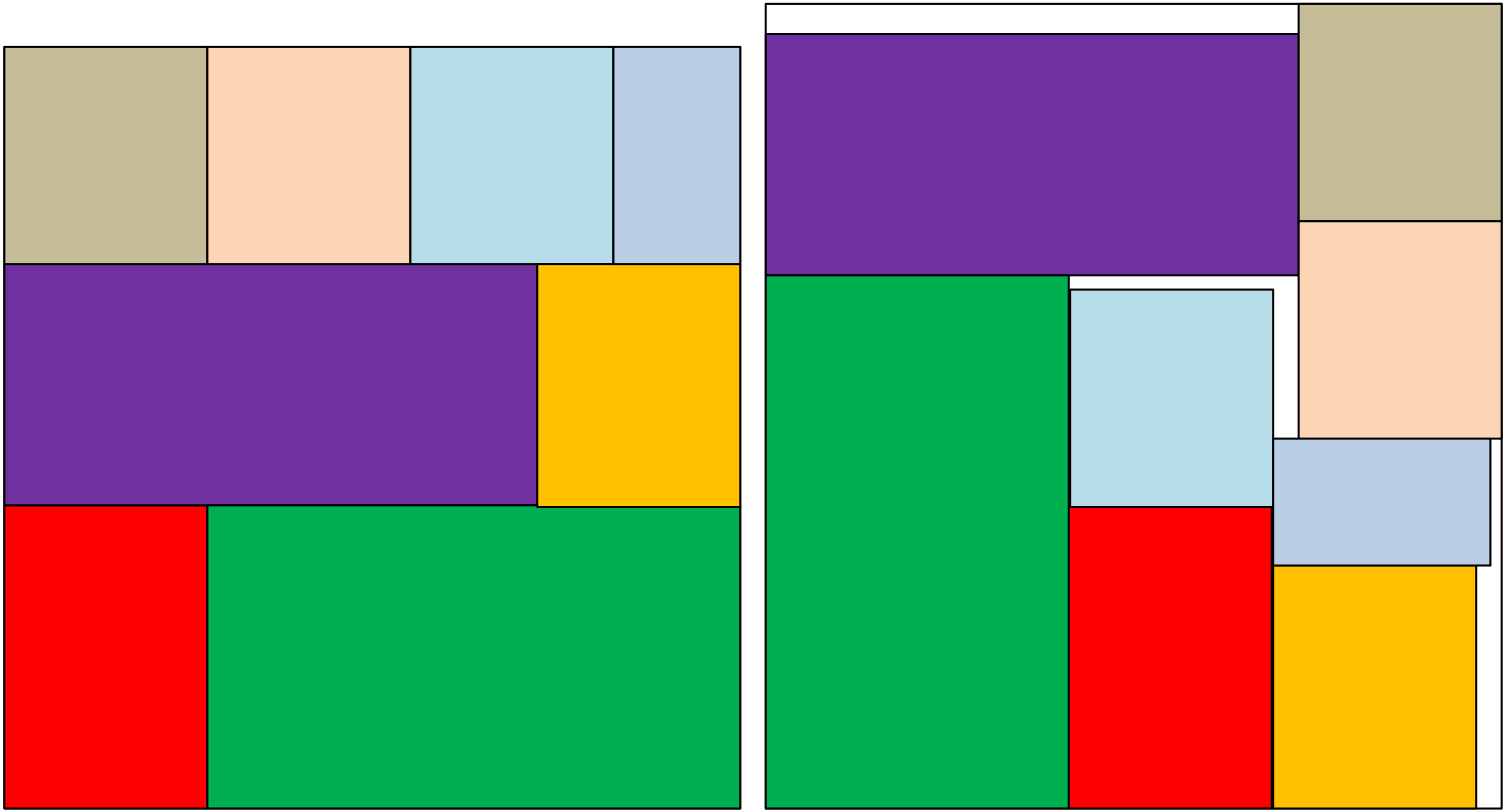
Problem Definition

- Given
  - A set of modules (blocks): $M = \{m_1, m_2, \ldots, m_n\}$
    - (width, height) for each module is also given. (e.g., m_1 = (10um, 20um))
  - A set of nets (netlist): $N = \{n_1, n_2, \ldots, n_m\}$
  - Outline: Chip width and height

- Find a floorplan
  - Minimize
    - Area
    - Wirelength

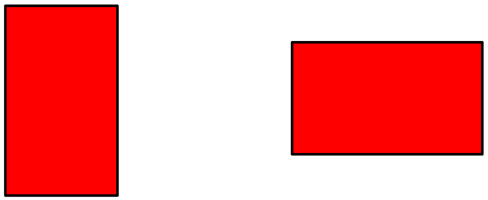
- Constraints
  - No overlap between modules

Example

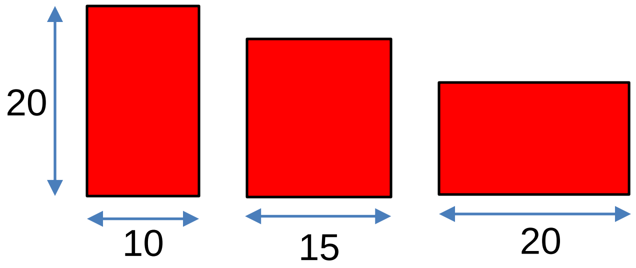


Problem Definition

- Later on, we will solve more complex problems.
  - Rotatable blocks
    - Some blocks are rotatable.



  - Soft blocks
    - Some blocks are soft.
      - Area: fixed. Aspect ratio = [0.5, 2.0]



Floorplanning Algorithms

- Deterministic algorithms
  - Linear-programming
- Stochastic algorithms
  - Simulated-annealing
    - Polish expression
    - Sequence pair

Linear Programming

- Formulation

Minimize $\sum_{j=1}^{n} c_i x_j$
subject to

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \, , i = 1, 2, \ldots, m$$

x_j : variables
c_j, a_{ij}, b_{ij} : constants

Linear Programming

- Extension
  - Integer linear programming

Minimize $\sum_{j=1}^{n} c_i x_j$
subject to
$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \, , i = 1, 2, \dots, m$
$x_j \in Z$

x_j : variables
c_j, a_{ij}, b_{ij} : constants

Linear Programming

- Extension
  - Binary integer linear programming

Minimize $\sum_{j=1}^{n} c_i x_j$
subject to
$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \, , i = 1, 2, \ldots, m$
$x_j \in \{0,1\}$

x_j : variables
c_j, a_{ij}, b_{ij} : constants

Linear Programming

- Example
  - An oil refinery produces two products.
    - Jet fuel
    - Gasoline
  - Profit
    - Jet fuel: $1 per Barrel
    - Gasoline: $2 per Barrel
  - Conditions (constraints)
    - Only 10,000 barrels of crude oil are available per day.
    - The refinery should produce at least 1,000 barrels of jet fuel.
    - The refinery should produce at least 2,000 barrels of gasoline.
    - Both products are shipped in trucks whose delivery capacity is 180,000 barrel-miles.
    - The jet fuel is delivered to an airfield 10 miles away from the refinery.
    - The gasoline is transported a distributor 30 miles away from the refinery.
  - Objective
    - Maximize the profit.
    - How much of each product should be produced?

Linear Programming

- Example
  - An oil refinery produces two products.
    - Jet fuel (variable: **x**)
    - Gasoline (variable: **y**)
  - Profit
    - Jet fuel: $1 per Barrel
    - Gasoline: $2 per Barrel
  - Conditions (constraints)
    - Only 10,000 barrels of crude oil are available per day. (**$x + y \le 10{,}000$**)
    - The refinery should produce at least 1,000 barrels of jet fuel. (**$x \ge 1{,}000$**)
    - The refinery should produce at least 2,000 barrels of gasoline. (**$y \ge 2{,}000$**)
    - Both products are shipped in trucks whose delivery capacity is 180,000 barrel-miles. (**$10x + 30y \le 180{,}000$**)
    - The jet fuel is delivered to an airfield 10 miles away from the refinery.
    - The gasoline is transported a distributor 30 miles away from the refinery.
  - Objective
    - Maximize the profit. (maximize **$x + 2y$**)
    - How much of each product should be produced?

Linear Programming

- Formulation

Minimize $\sum_{j=1}^{n} c_i x_j$
subject to

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \, , i = 1, 2, \ldots, m$$

x_j : variables
c_j, a_{ij}, b_{ij} : constants

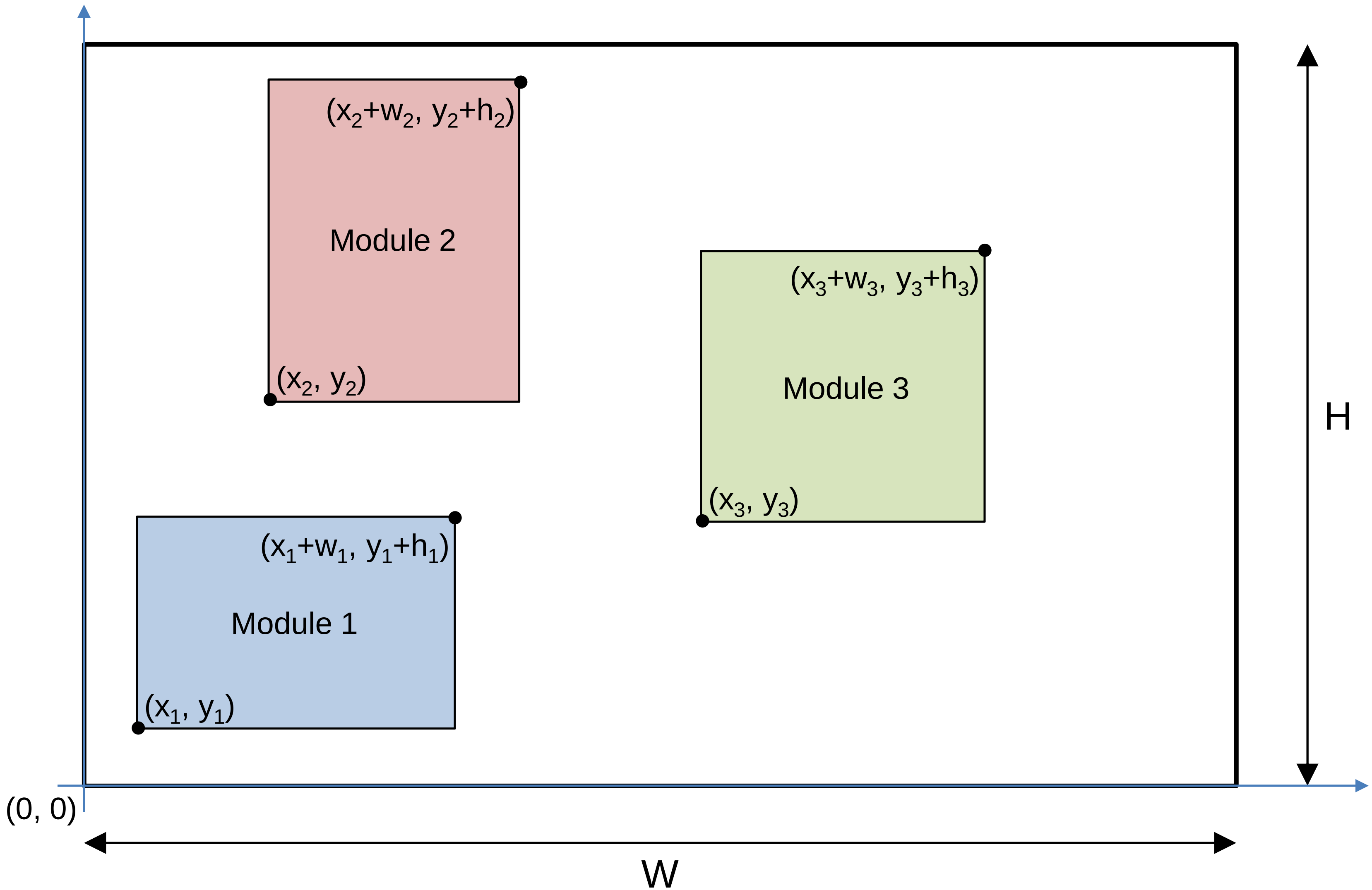
Maximize x + 2y (Minimize –x – 2y)
subject to

$$x + y \leq 10{,}000$$
$$x \geq 1{,}000$$
$$y \geq 2{,}000$$
$$10x + 30y \leq 180{,}000$$

Linear Programming-Based Floorplanning

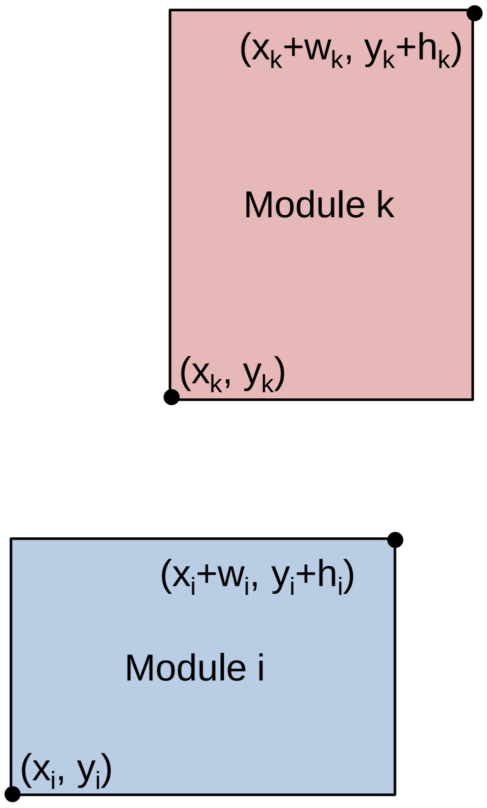
- Analytical approach
  - Case 1) All modules are rigid and not rotatable.
    - Module 1: (width, height) = (w_1, h_1)
    - Module 2: (width, height) = (w_2, h_2)
    - …
- Constraints
  - The width and the height of the floorplan are given (fixed-outline floorplanning)

LP-Based Floorplanning



LP-Based Floorplanning

- Analytical formulation
  - Boundary conditions
    - $x_i \geq 0, x_i + w_i \leq W$
    - $y_i \geq 0, y_i + h_i \leq H$
  - No overlap conditions
    - i is to the left of k: $x_i + w_i \leq x_k$
    - i is to the right of k: $x_k + w_k \leq x_i$
    - i is below k: $y_i + h_i \leq y_k$
    - i is above k: $y_k + h_k \leq y_i$



LP-Based Floorplanning

- Linear programming formulation for no overlaps
  - i is to the left of k: $x_i + w_i \leq x_k$
  - i is to the right of k: $x_k + w_k \leq x_i$
  - i is below k: $y_i + h_i \leq y_k$
  - i is above k: $y_k + h_k \leq y_i$

  - Introduce two binary variables, x_{ik} and y_{ik}.

x_{ik}	y_{ik}	Meaning
0	0	i is to the left of k
0	1	i is below k
1	0	i is to the right of k
1	1	i is above k

LP-Based Floorplanning

- Linear programming formulation for no overlaps

x_{ik}	y_{ik}	Meaning
0	0	i is to the left of k
0	1	i is below k
1	0	i is to the right of k
1	1	i is above k

$$x_i + w_i \le x_k + W(x_{ik} + y_{ik})$$
$$y_i + h_i \le y_k + H(1 + x_{ik} - y_{ik})$$
$$x_k + w_k \le x_i + W(1 - x_{ik} + y_{ik})$$
$$y_k + h_k \le y_i + H(2 - x_{ik} - y_{ik})$$

LP-Based Floorplanning

- Formulation

Minimize Y

Subject to

$$
\begin{aligned}
&x_i \geq 0, && 1 \leq i \leq n \\
&y_i \geq 0, && 1 \leq i \leq n \\
&x_i + w_i \leq W && 1 \leq i \leq n \\
&y_i + h_i \leq Y && 1 \leq i \leq n \\
&x_i + w_i \leq x_k + W(x_{ik} + y_{ik}) && 1 \leq i < j \leq n \\
&y_i + h_i \leq y_k + H(1 + x_{ik} - y_{ik}) && 1 \leq i < j \leq n \\
&x_k + w_k \leq x_i + W(1 - x_{ik} + y_{ik}) && 1 \leq i < j \leq n \\
&y_k + h_k \leq y_i + H(2 - x_{ik} - y_{ik}) && 1 \leq i < j \leq n
\end{aligned}
$$

LP-Based Floorplanning

- Analytical approach
  - Case 2) All modules are rigid and rotatable.
    - Module 1: (width, height) = (w_1, h_1) or (h_1, w_1)
    - Module 2: (width, height) = (w_2, h_2) or (h_2, w_2)
    - …
- Constraints
  - The width and the height of the floorplan are given (fixed-outline floorplanning)

LP-Based Floorplanning

- Formulation
  - Introduce a new binary variable for each module.
    - z_i
      - 0: un-rotated ($w = w_i$, $h = h_i$)
      - 1: rotated ($w = h_i$, $h = w_i$)

$$w_i => z_i h_i + (1 - z_i) w_i$$
$$h_i => z_i w_i + (1 - z_i) h_i$$

LP-Based Floorplanning

- Formulation
  - Introduce a new binary variable, z_i, for each module.
    - 0: un-rotated
    - 1: rotated

Minimize Y

Subject to

$$
\begin{aligned}
&x_i \geq 0, && 1 \leq i \leq n \\
&y_i \geq 0, && 1 \leq i \leq n \\
&x_i + z_i h_i + (1 - z_i) w_i \leq W && 1 \leq i \leq n \\
&y_i + z_i w_i + (1 - z_i) h_i \leq Y && 1 \leq i \leq n \\
&x_i + z_i h_i + (1 - z_i) w_i \leq x_k + M(x_{ik} + y_{ik}) && 1 \leq i < j \leq n \\
&y_i + z_i w_i + (1 - z_i) h_i \leq y_k + M(1 + x_{ik} - y_{ik}) && 1 \leq i < j \leq n \\
&x_k + z_k h_k + (1 - z_k) w_k \leq x_i + M(1 - x_{ik} + y_{ik}) && 1 \leq i < j \leq n \\
&y_k + z_k w_k + (1 - z_k) h_k \leq y_i + M(2 - x_{ik} - y_{ik}) && 1 \leq i < j \leq n \\
&M = \max(W, H) \ \ or \ (W + H)
\end{aligned}
$$

LP-Based Floorplanning

- Analytical approach
  - Case 3) Some modules are flexible (soft).
    - Module 1: area = $A_1 = w_1*h_1$. $w_1 = [w_{1\_min}, w_{1\_max}]$
    - Module 2: area = $A_2 = w_2*h_2$. $w_2 = [w_{2\_min}, w_{2\_max}]$
    - …

- Constraints
  - The width and the height of the floorplan are given (fixed-outline floorplanning)

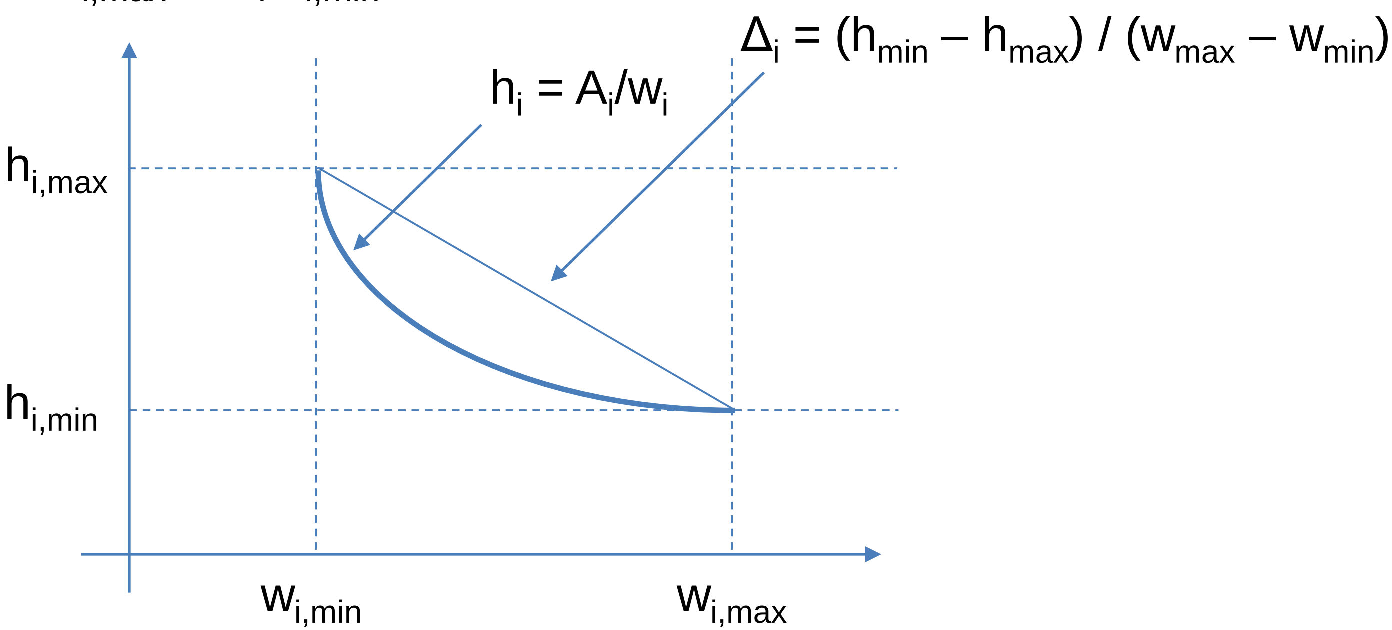
LP-Based Floorplanning

- Formulation
  - w_i and h_i are variables.
  - $w_i * h_i \geq A_i$ is not linear.

- Linear formulation (Linearization)
  - First-order approximation
    - $h_i = \Delta_i w_i + c_i$ (y = mx+c)
    - $\Delta_i = (h_{i,min} - h_{i,max}) / (w_{i,max} - w_{i,min})$
    - $c_i = h_{i,max} - \Delta_i w_{i,min}$

LP-Based Floorplanning

- Linear formulation (Linearization)
  - First-order approximation
    - $h_i = \Delta_i w_i + c_i$ $(y = mx+c)$
    - $\Delta_i = (h_{i,min} - h_{i,max}) / (w_{i,max} - w_{i,min})$
    - $c_i = h_{i,max} - \Delta_i w_{i,min}$



LP-Based Floorplanning

- Formulation

Minimize Y

Subject to

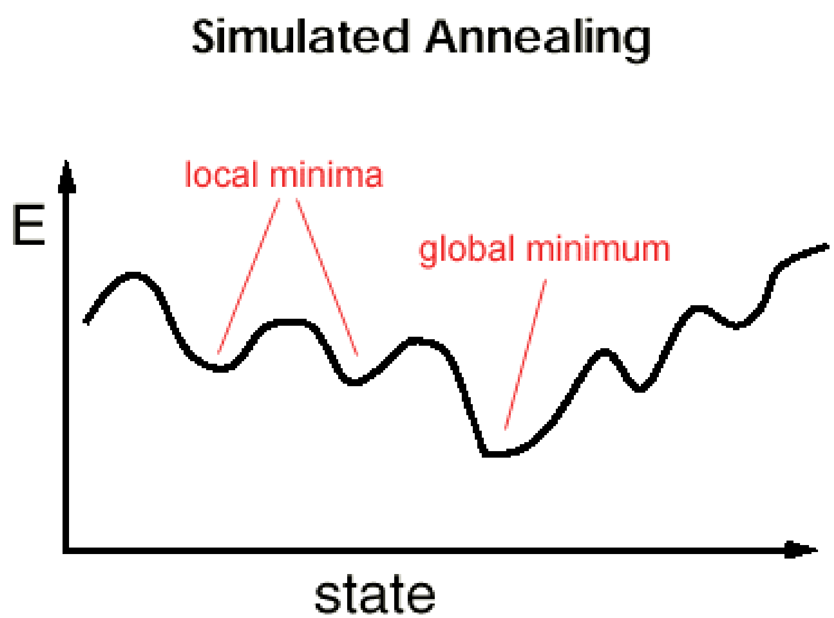
$$x_i \geq 0, \quad 1 \leq i \leq n$$
$$y_i \geq 0, \quad 1 \leq i \leq n$$
$$x_i + w_i \leq W \quad 1 \leq i \leq n$$
$$y_i + (\Delta_i w_i + c_i) \leq Y \quad 1 \leq i \leq n$$
$$w_i \geq w_{i,min} \quad 1 \leq i \leq n$$
$$w_i \leq w_{i,max} \quad 1 \leq i \leq n$$
$$x_i + w_i \leq x_k + W(x_{ik} + y_{ik}) \quad 1 \leq i < j \leq n$$
$$y_i + (\Delta_i w_i + c_i) \leq y_k + H(1 + x_{ik} - y_{ik}) \quad 1 \leq i < j \leq n$$
$$x_k + w_k \leq x_i + W(1 - x_{ik} + y_{ik}) \quad 1 \leq i < j \leq n$$
$$y_k + (\Delta_k w_k + c_k) \leq y_i + H(2 - x_{ik} - y_{ik}) \quad 1 \leq i < j \leq n$$

Floorplanning Algorithms

- Deterministic algorithms
  - Linear-programming
- Stochastic algorithms
  - Simulated-annealing
    - Polish expression
    - Sequence pair

Simulated Annealing

- Similar to the simulated annealing algorithm used for partitioning.



Simulated Annealing

- Algorithm

  T = T_0 (initial temperature)

  S = S_0 (initial solution)

  Time = 0

```
repeat
    Call Metropolis (S, T, M);
    Time = Time + M;
    T = α · T;  // α: cooling rate (α < 1)
    M = β · M;
until (Time ≥ maxTime);
```

Simulated Annealing

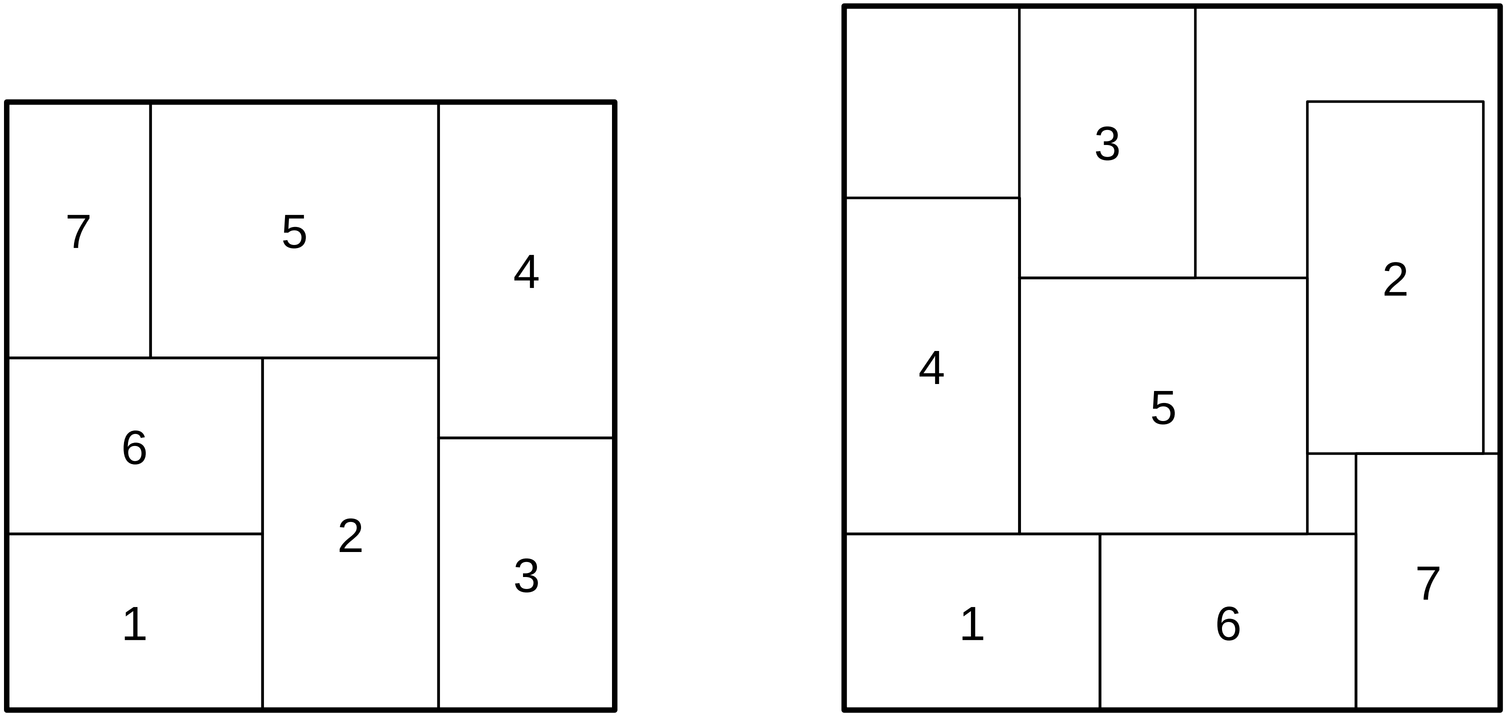
- Algorithm

Metropolis (S, T, M) // M: # iterations

```
repeat
    NewS = neighbor(S);  // get a new solution by perturbation
    Δh = cost(NewS) – cost(S);
    If ((Δh < 0) or (random < e^(-Δh/T)))
        S = NewS;  // accept the new solution
    M = M – 1;
until (M==0)
```

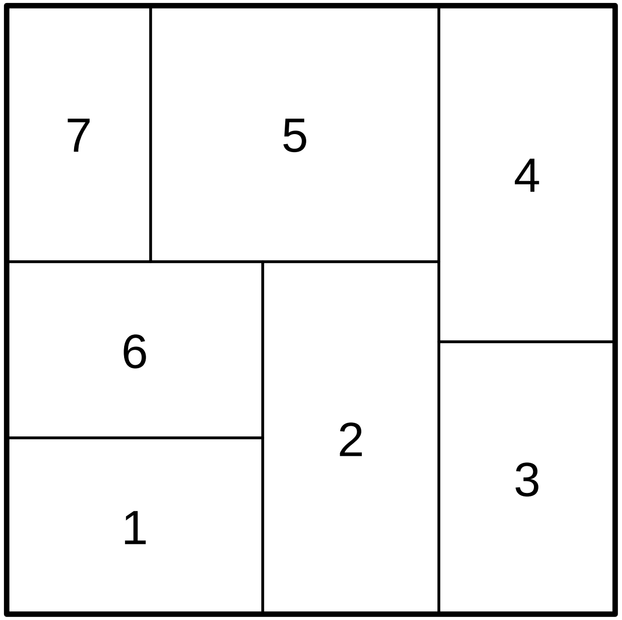
Simulated Annealing

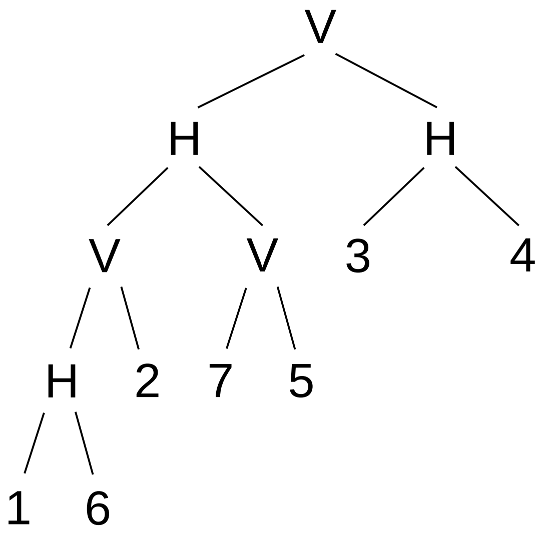
- How can we represent floorplans?
  - Polish expression (slicing floorplan)
  - Sequence pair (non-slicing floorplan)



Polish Expression

- Polish expression ↔ post-order traversal





E = 1 6 H 2 V 7 5 V H 3 4 H V

Polish Expression

- Polish expression
  - $E = e_1\ e_2 \ldots e_{2n-1}$ where $e_i \in \{1, 2, \ldots, n, H, V\}$ is a Polish expression of length (2n-1) if and only if
    - Every operand j ($1 \le j \le n$) appears exactly once
    - (balloting property) for every subexpression $E_k = e_1 \ldots e_k$, #operands > #operators.

operators

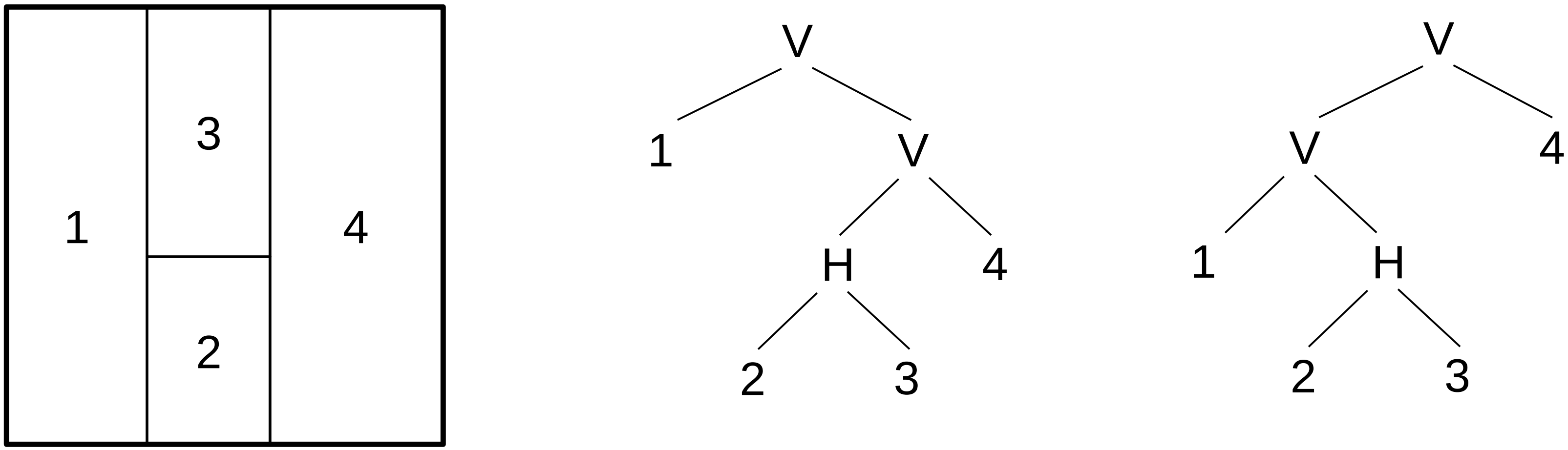
E = 1 6 H 2 V 7 5 V H 3 4 H V

operands: 3
operators: 1

operands: 7
operators: 4

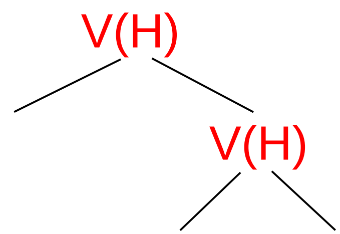
Polish Expression

- Redundancy in the solution representation



Polish Expression

- Non-skewed vs. Skewed



Non-skewed
(… VV …)
or
(… HH …)

Polish Expression

- Normalized polish expression
  - E = e_1 e_2 … e_{2n-1} is called normalized if and only if
    - E has no consecutive operators of the same type (H or V).
    - In other words, it's skewed.
  - Using the normalized polish expression, we remove the redundancy and construct a unique representation.

Polish Expression

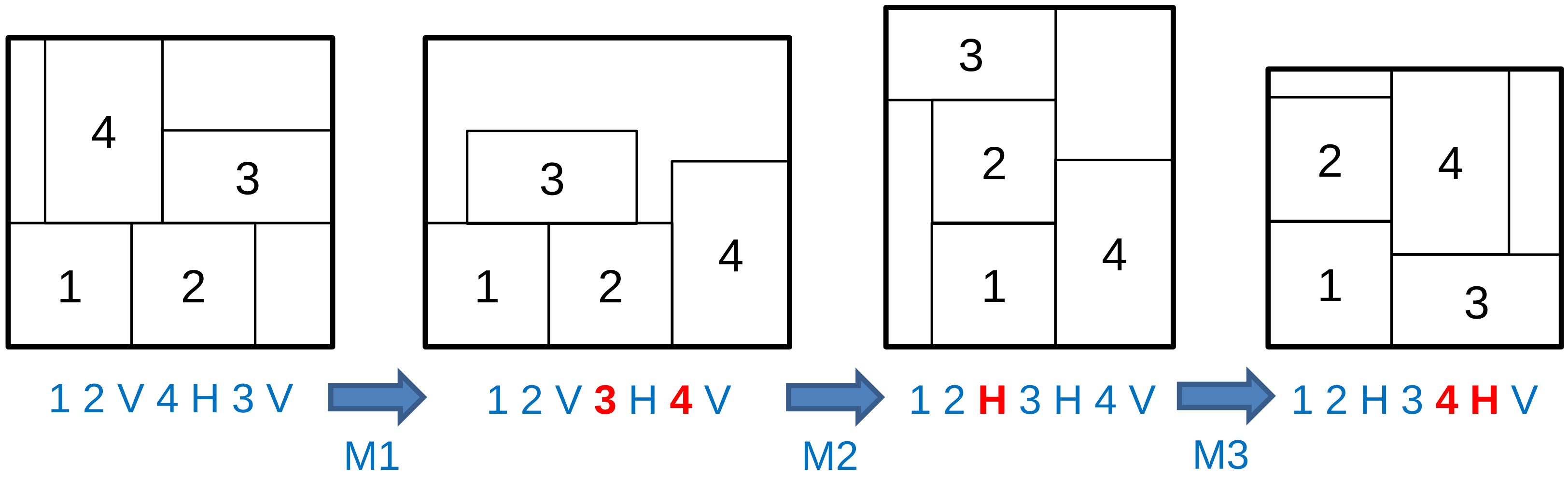
- Solution perturbation
  - Chain: HVHVH … or VHVHV …
  - Adjacent

    1 6 H 3 5 V 2 H V 7 4 H V

    - 1 6: adjacent operands
    - 2 7: adjacent operands
    - 5 V: adjacent operand and operator
  - Moves
    - Move 1 (Operand swap): Swap two adjacent operands.
    - Move 2 (Chain invert): Complement a chain (V→H, H→V)
    - Move 3 (Operator/operand swap): Swap two adjacent operand and operator.

Polish Expression

- Effects of perturbation



Polish Expression

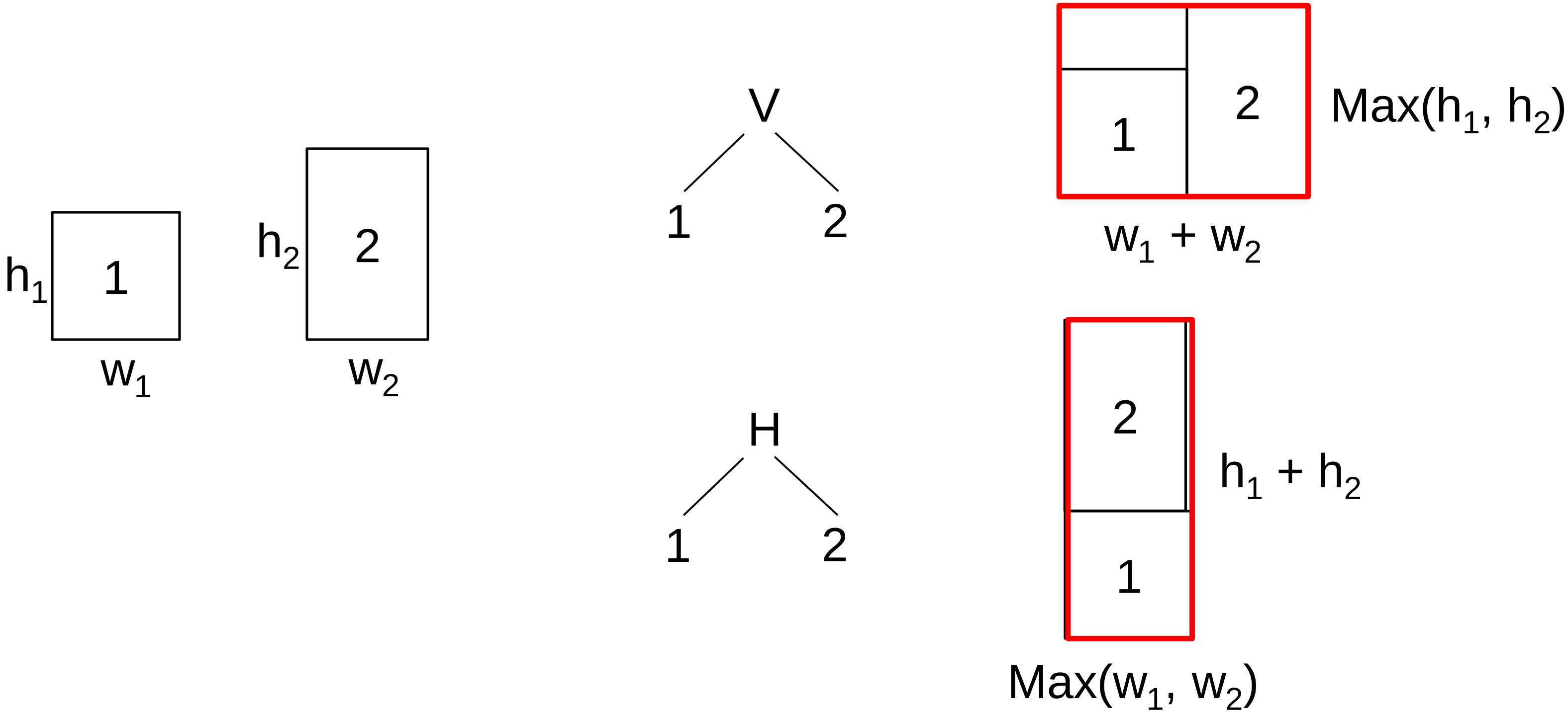
- Does the balloting property hold during moves?
  - (balloting property) for every subexpression $E_k = e_1 \dots e_k$, #operands > #operators.
  - Moves
    - Move 1 (Operand swap): Swap two adjacent operands. (Yes)
    - Move 2 (Chain invert): Complement a chain (H↔V) (Yes)
    - Move 3 (Operator/operand swap): Swap two adjacent operand and operator.
      - Reject “illegal” moves.
  - How can we find “illegal” moves?

Polish Expression

- Operator/operand swap
  - Assume that the type-3 move swaps operand e_i with operator e_{i+1}, ($1 \leq i \leq k-1$). Then, the swap will not violate the balloting property iff $2N_{i+1} < i$.
    - N_k: # operators in the Polish expression $E=e_1e_2\ldots e_k$.

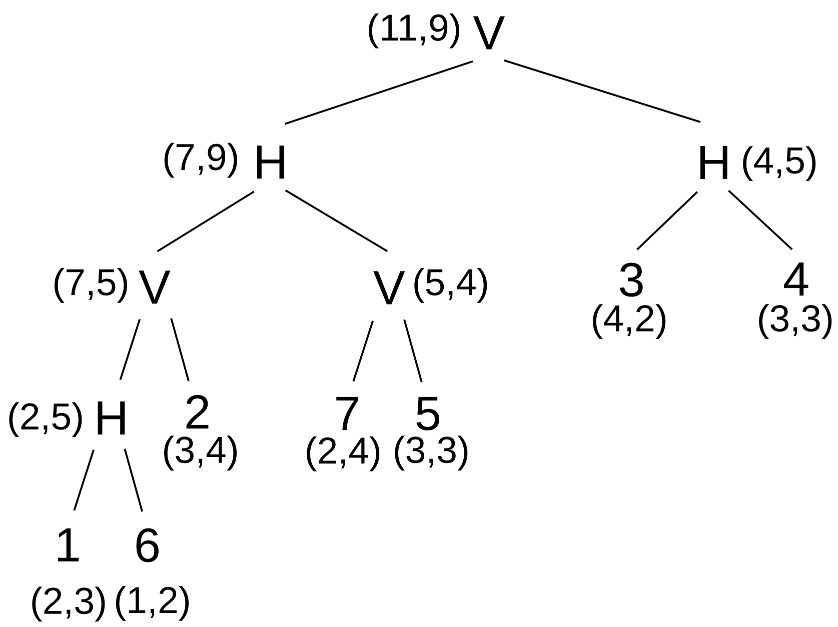
Polish Expression

- Cost function
  - Cost = Area + λ·W
- Area computation



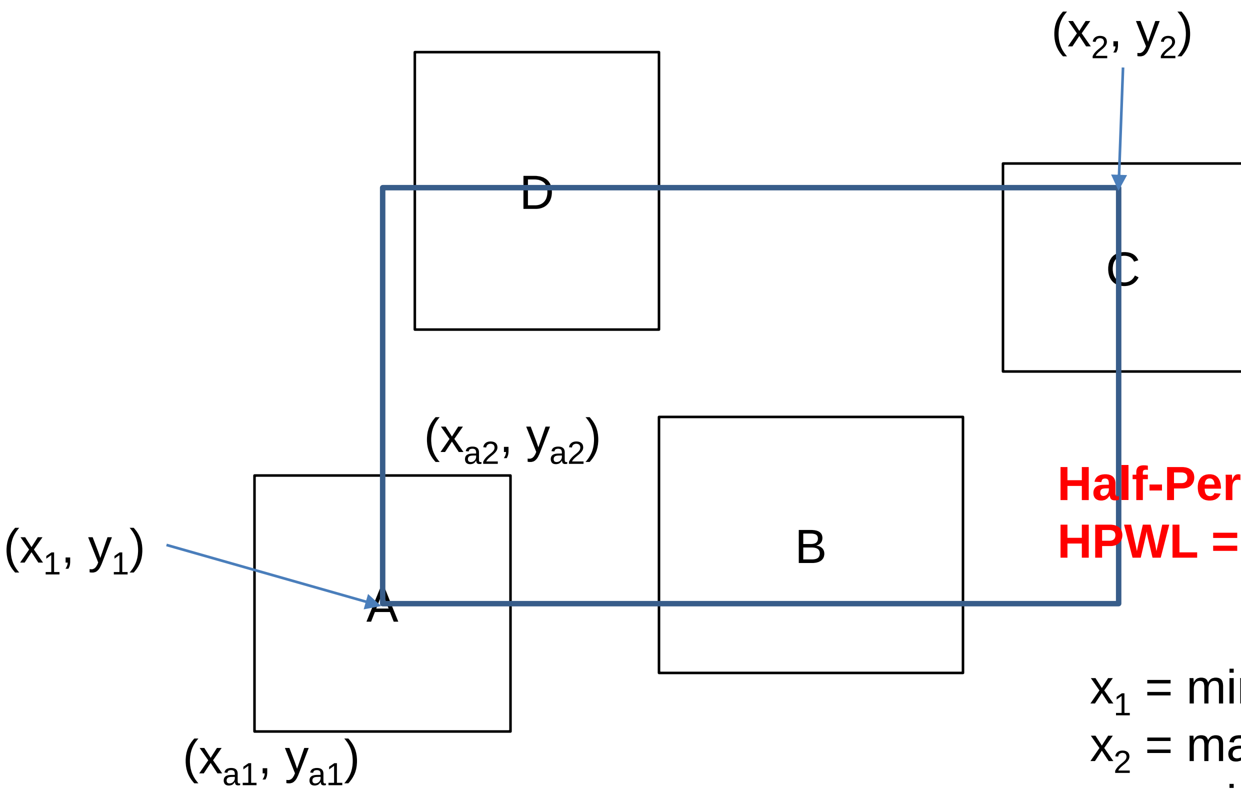
Polish Expression

- Example



Polish Expression

- Wirelength estimation

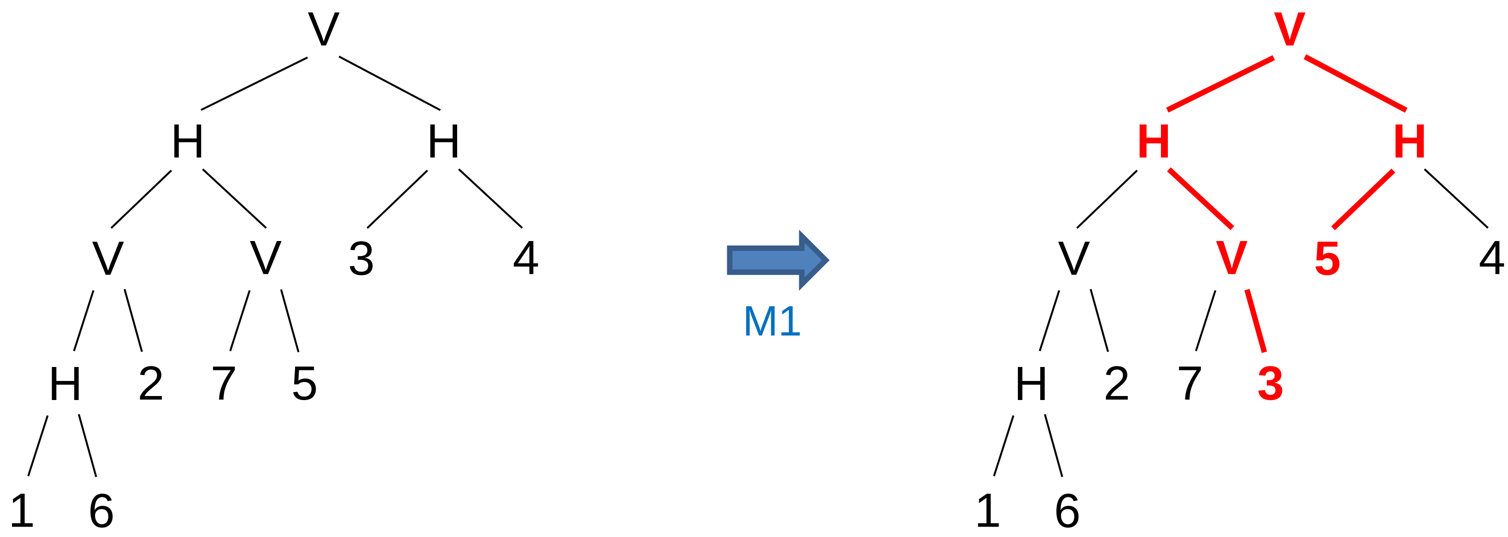


Half-Perimeter WireLength
HPWL = $(x_2 – x_1) + (y_2 – y_1)$

$x_1 = \min(x_{a1}, x_{b1}, x_{c1}, x_{d1})$
$x_2 = \max(x_{a2}, x_{b2}, x_{c2}, x_{d2})$
$y_1 = \min(y_{a1}, y_{b1}, y_{c1}, y_{d1})$
$y_2 = \max(y_{a2}, y_{b2}, y_{c2}, y_{d2})$

Polish Expression

- Incremental cost computation

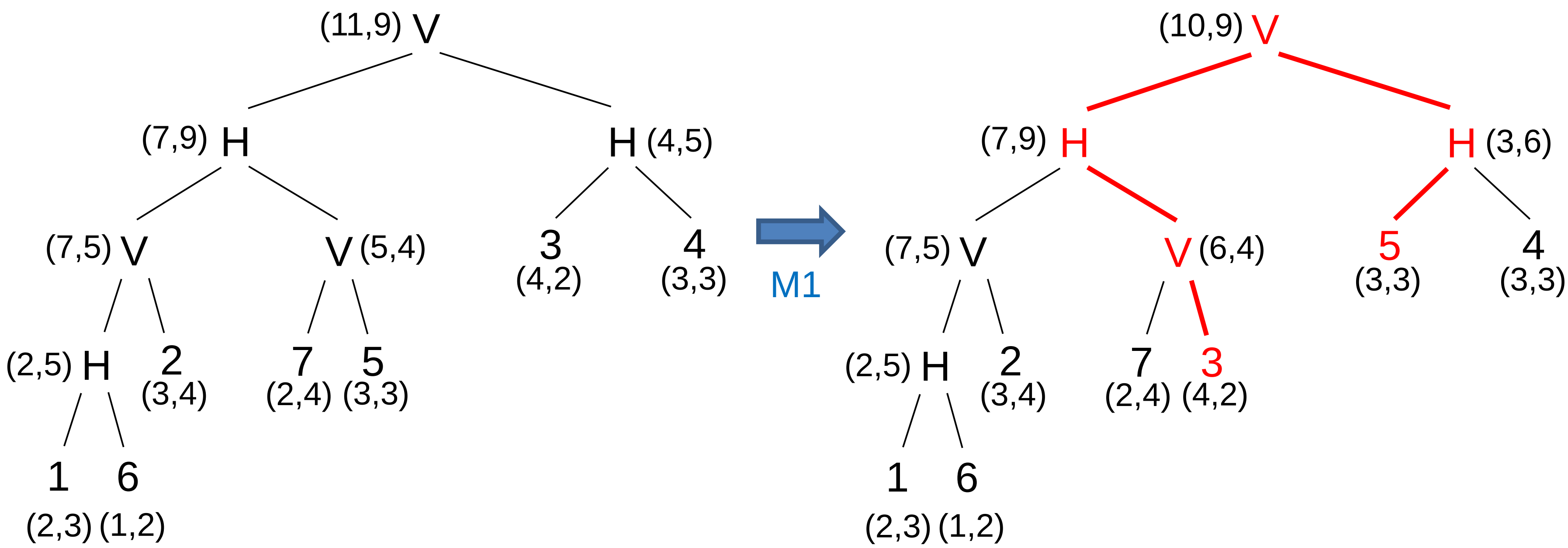


E = 1 6 H 2 V 7 **5** V H **3** 4 H V

E = 1 6 H 2 V 7 **3** V H **5** 4 H V

Polish Expression

- Example



Floorplanning Algorithms

- Deterministic algorithms
  - Linear-programming
- Stochastic algorithms
  - Simulated-annealing
    - Polish expression
    - Sequence pair

Sequence Pair

- P-admissible solution space for a problem
  - The solution space is finite.
  - Every solution is feasible.
  - Implementation and evaluation of each configuration are possible in polynomial time.
  - The configuration corresponding to the best evaluated solution in the space coincides with an optimal solution of the problem.

- Slicing floorplan is not P-admissible.

- Sequence pair is P-admissible.

Sequence Pair

- Represent a solution by a pair of module-name sequences.
  - (1 2 3 4 5), (3 5 1 4 2)

    Positive seq. (Γ_+) Negative seq. (Γ_-)

- Conversion of a sequence pair into its corresponding floorplan.
  - x is after y in both Γ_+ and Γ_- $\Leftrightarrow$ x is right to y.
  - x is before y in both Γ_+ and Γ_- $\Leftrightarrow$ x is left to y.
  - x is after y in Γ_+ and before y in Γ_- $\Leftrightarrow$ x is below to y.
  - x is before y in Γ_+ and after y in Γ_- $\Leftrightarrow$ x is above to y.

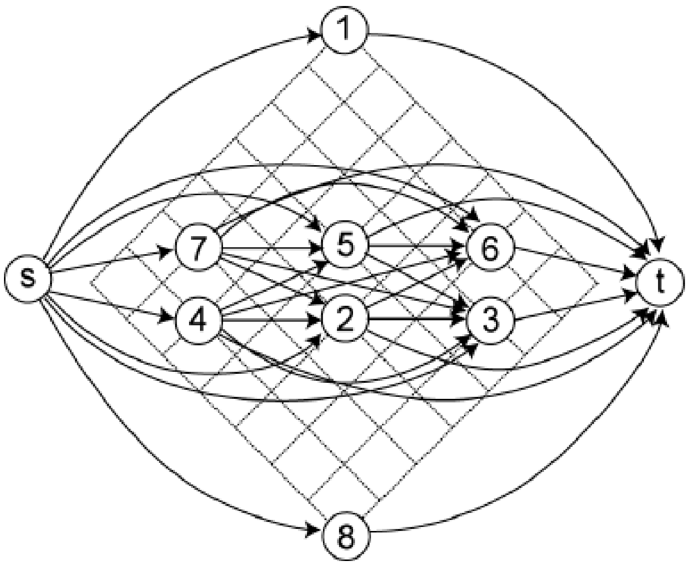
Sequence Pair

- (Γ_+, Γ_-)-Packing
  - Constraint graphs
    - Horizontal constraint graph (HCG)
    - Vertical constraint graph (VCG)

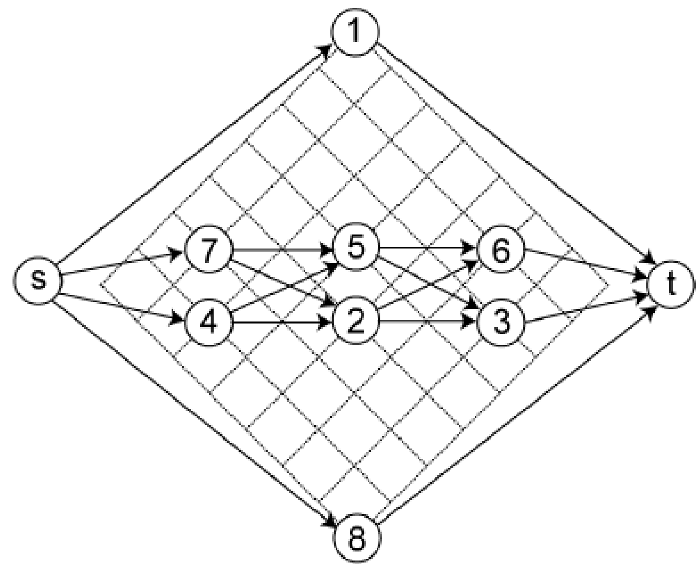
Sequence Pair

(..y..x..) (..y..x..) ⇔ x is right to y.
(..x..y..) (..x..y..) ⇔ x is left to y.
(..y..x..) (..x..y..) ⇔ x is below to y.
(..x..y..) (.y..x..) ⇔ x is above to y.

- Horizontal constraint graph
  - (1 7 4 5 2 6 3 8) (8 4 7 2 5 3 6 1)



(a)



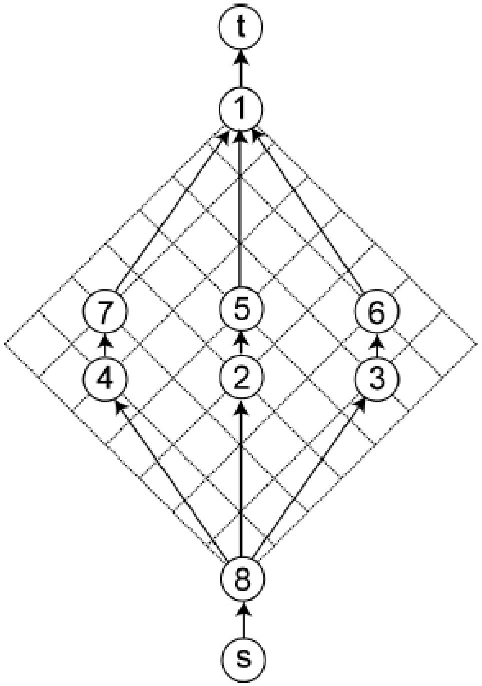
(b)

Sequence Pair

- Vertical constraint graph

(..y..**x**..) (..y..**x**..) ⇔ x is right to y.
(..**x**..y..) (..**x**..y..) ⇔ x is left to y.
(..y..**x**..) (..**x**..y..) ⇔ x is below to y.
(..**x**..y..) (.y..**x**..) ⇔ x is above to y.

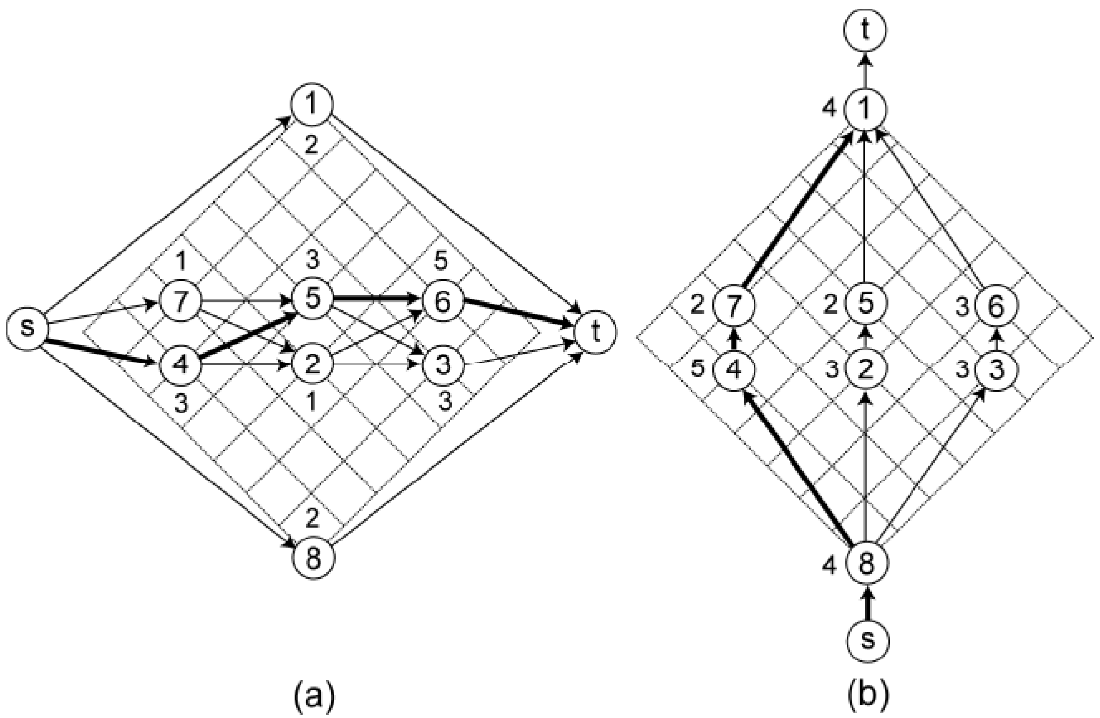
  - (1 7 4 5 2 6 3 8) (8 4 7 2 5 3 6 1)



Sequence Pair

- Computation of the location of each block
  - HCG: determines the x-coordinates.
  - VCG: determines the y-coordinates.

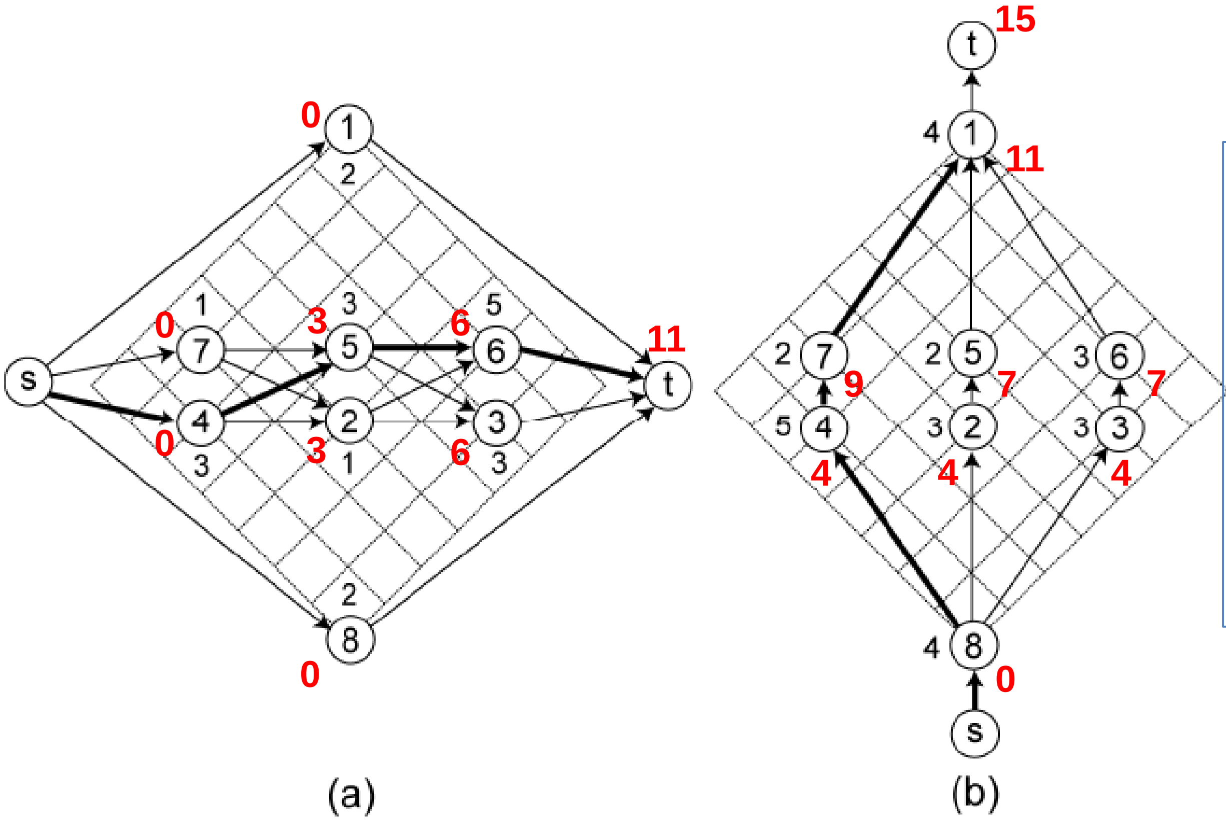
Modules (w, h)
$m_1 = (2, 4)$
$m_2 = (1, 3)$
$m_3 = (3, 3)$
$m_4 = (3, 5)$
$m_5 = (3, 2)$
$m_6 = (5, 3)$
$m_7 = (1, 2)$
$m_8 = (2, 4)$



Sequence Pair

- Use longest source-to-module path length

Modules (w, h)
$m_1 = (2, 4)$
$m_2 = (1, 3)$
$m_3 = (3, 3)$
$m_4 = (3, 5)$
$m_5 = (3, 2)$
$m_6 = (5, 3)$
$m_7 = (1, 2)$
$m_8 = (2, 4)$

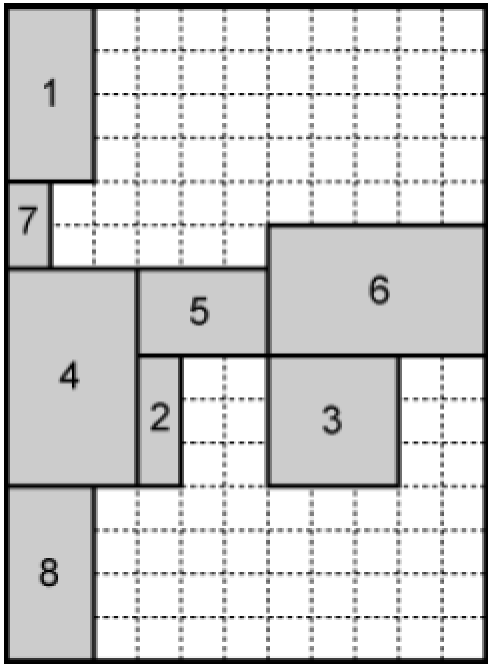


$m_1 = (0, 11)$
$m_2 = (3, 4)$
$m_3 = (6, 4)$
$m_4 = (0, 4)$
$m_5 = (3, 7)$
$m_6 = (6, 7)$
$m_7 = (0, 9)$
$m_8 = (0, 0)$

Sequence Pair

- Floorplan

$m_1 = (0, 11)$
$m_2 = (3, 4)$
$m_3 = (6, 4)$
$m_4 = (0, 4)$
$m_5 = (3, 7)$
$m_6 = (6, 7)$
$m_7 = (0, 9)$
$m_8 = (0, 0)$



Sequence Pair

- Solution perturbation
  - Move 1: Swap two cells in the positive sequence
    - Γ_+ : (..x..y..) → (..y..x..)
  - Move 2: Swap two cells in the negative sequence
    - Γ_- : (..x..y..) → (..y..x..)
  - Move 3: Swap two cells both in the pos/neg sequence
    - Γ_+ : (..x..y..) → (..y..x..)
    - Γ_- : (..y..x..) → (..x..y..)

Sequence Pair + Simulated Annealing

- Algorithm

  $T = T_0$ (initial temperature)

  $S = S_0$ (initial solution)

  Time = 0

```
repeat
    Call Metropolis (S, T, M);
    Time = Time + M;
    T = α · T;  // α: cooling rate (α < 1)
    M = β · M;
until (Time ≥ maxTime);
```

Sequence Pair + Simulated Annealing

- Algorithm

  Metropolis (S, T, M) // M: # iterations

```
repeat
    NewS = neighbor(S);  // get a new solution by perturbation
    Δh = cost(NewS) – cost(S);
    If ((Δh < 0) or (random < e^(-Δh/T)))
        S = NewS;  // accept the new solution
    M = M – 1;
until (M==0)
```