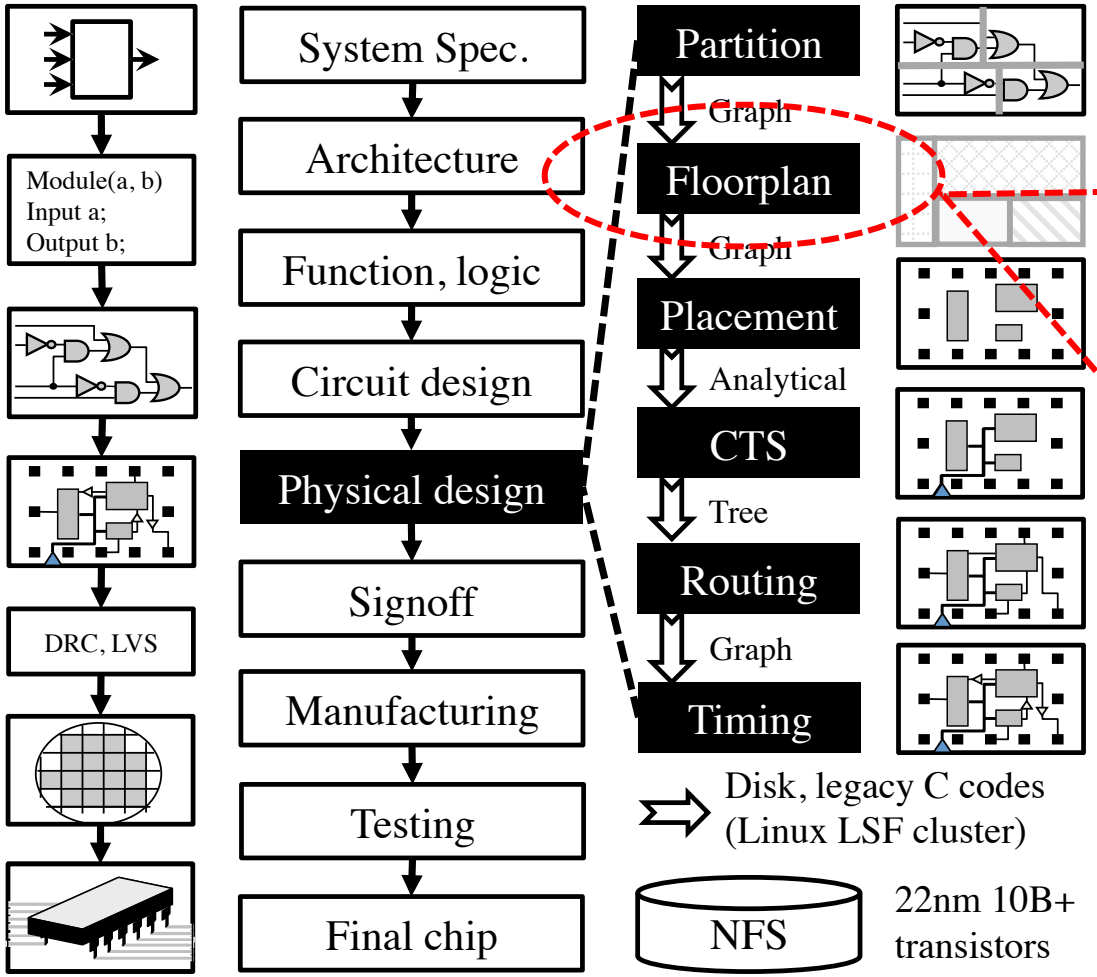
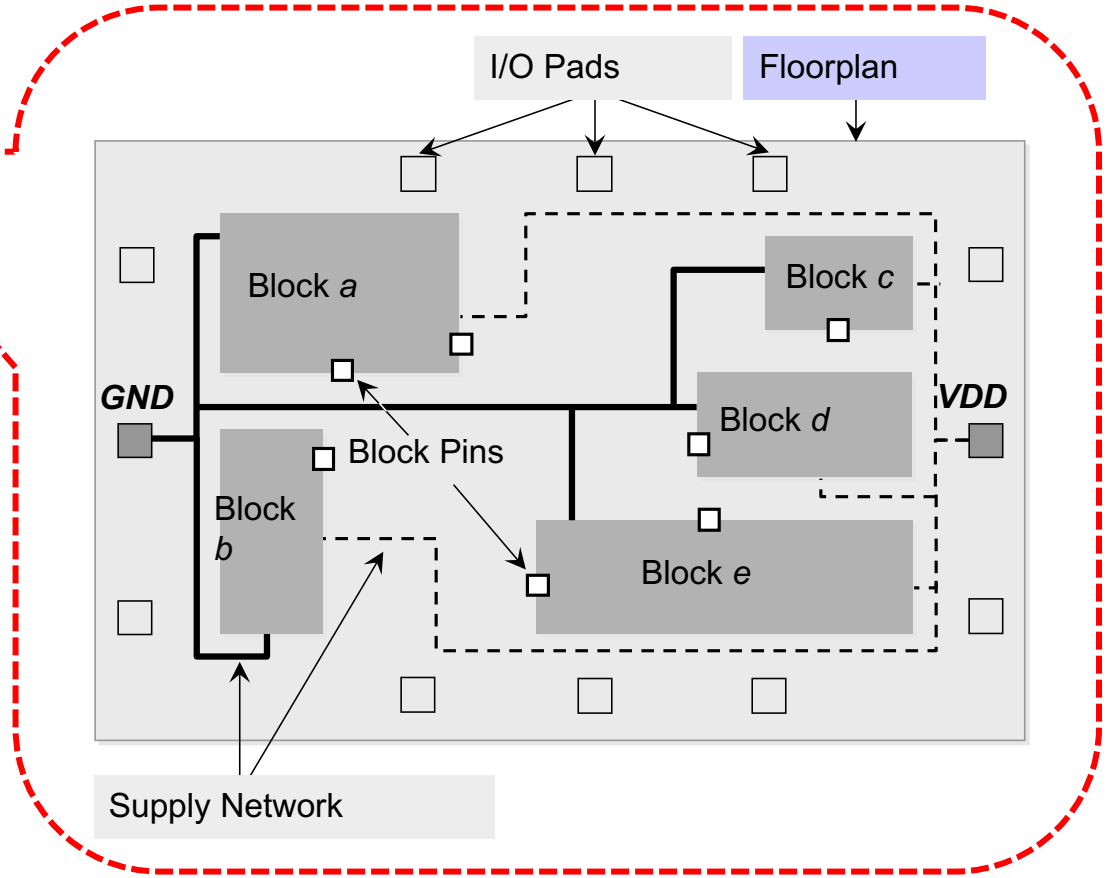
Lecture 9: Floorplan – I

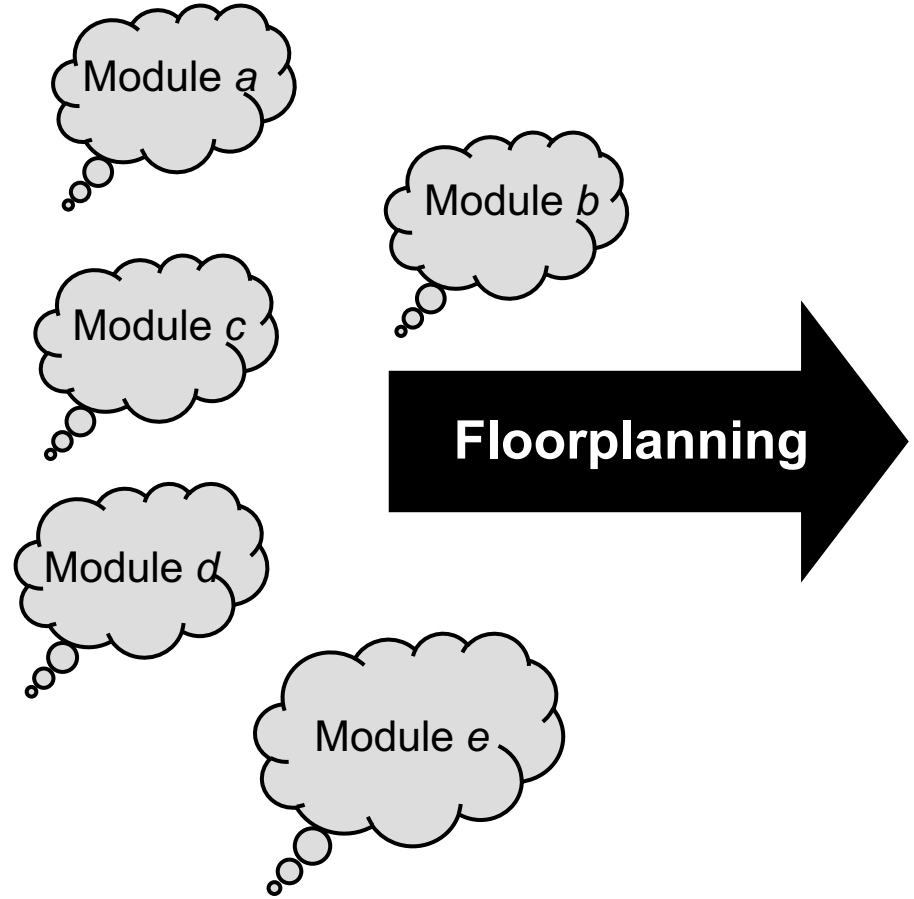
Tsung-Wei (TW) Huang
Department of Electrical and Computer Engineering
University of Utah, Salt Lake City, UT

Physical Design Flow

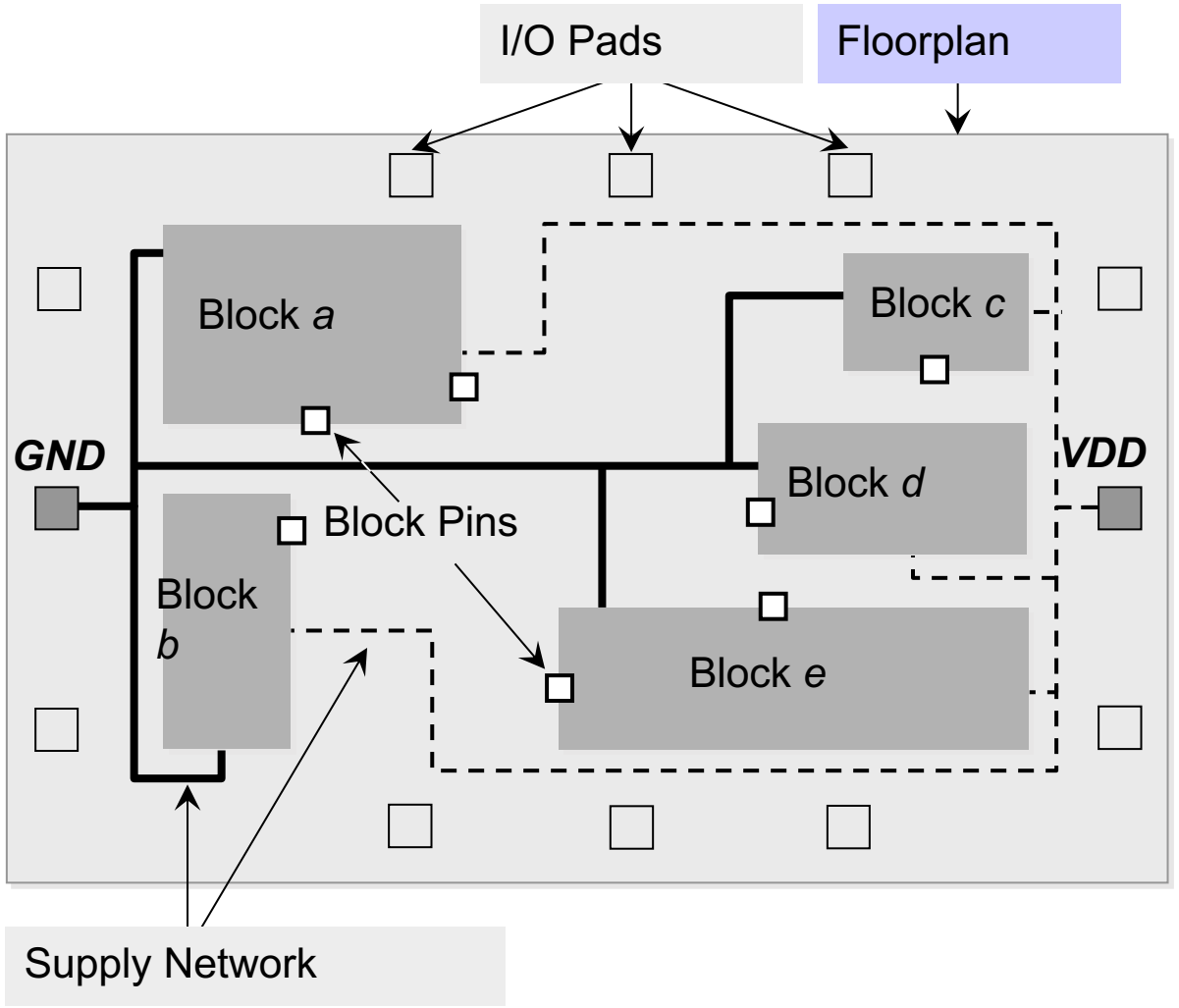




Floorplanning



Partitioned modules (e.g., FM algorithm)

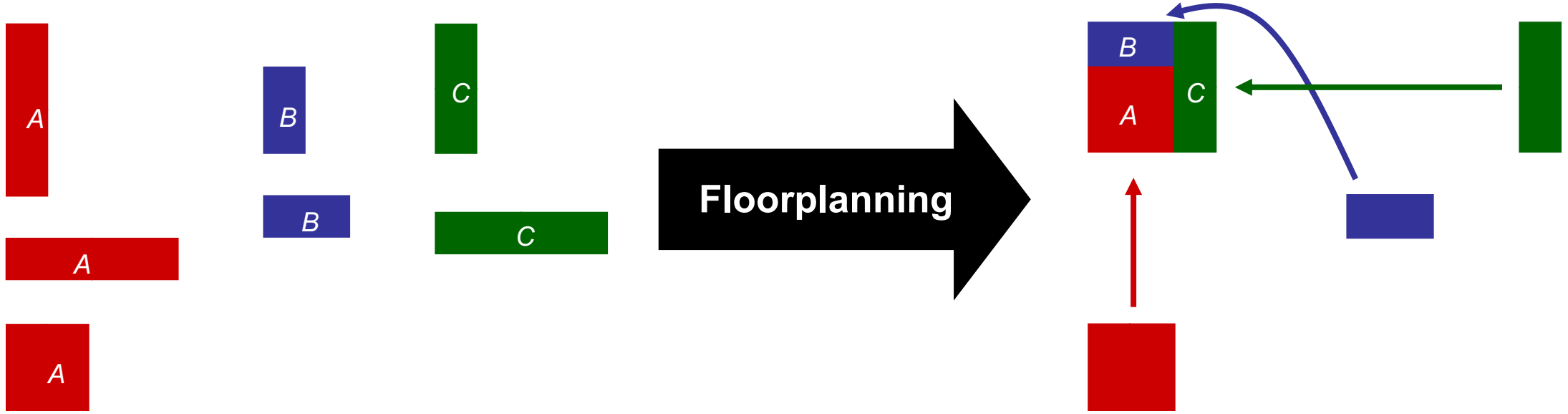


Floorplanning Problem

- The floorplanning problem is to plan the positions and shapes of the modules at the beginning of the design cycle to optimize the circuit performance…
  - chip area
  - total wirelength
  - delay of critical path
  - routability
  - others, ex: noise, heat dissipation, …

Example

- Given three blocks with the following potential widths & heights
  - Block *A*: $w = 1, h = 4$ or $w = 4, h = 1$ or $w = 2, h = 2$
  - Block *B*: $w = 1, h = 2$ or $w = 2, h = 1$
  - Block *C*: $w = 1, h = 3$ or $w = 3, h = 1$
- Goal: Pack a floorplan with minimum total area enclosed

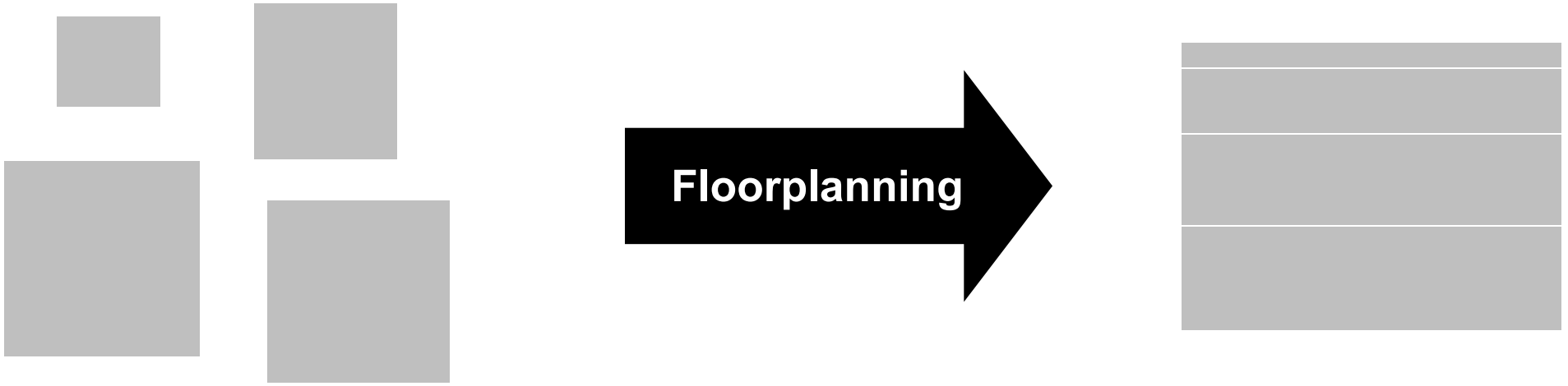


Formal Problem Formulation

- **Input:**
  - n Blocks with areas $A_1, \ldots, A_n$
  - Bounds r_i and s_i on the aspect ratio of block B_i
- **Output:**
  - Coordinates (x_i, y_i), width w_i and height h_i for each block such that $h_i w_i = A_i$ and $r_i \leq h_i/w_i \leq s_i$
- **Objective:**
  - To minimize the packed area and interconnect wirelength

Bounds on Aspect Ratio

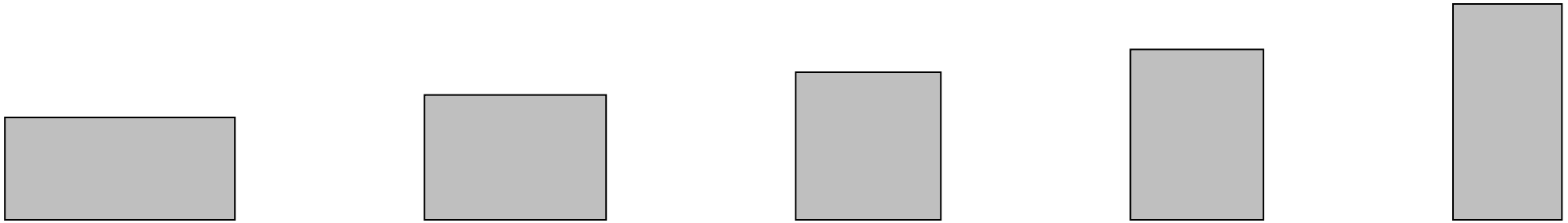
- If there is no bound on the aspect ratios, can we pack everything tightly?



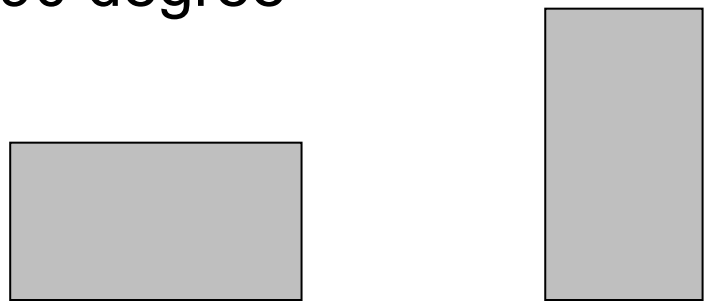
- But we don't want to layout blocks as long strips, so we require $r_i \leq h_i/w_i \leq s_i$ for each module *i*

Bounds on Aspect Ratio (cont'd)

- In practice, we allow several shapes for soft blocks

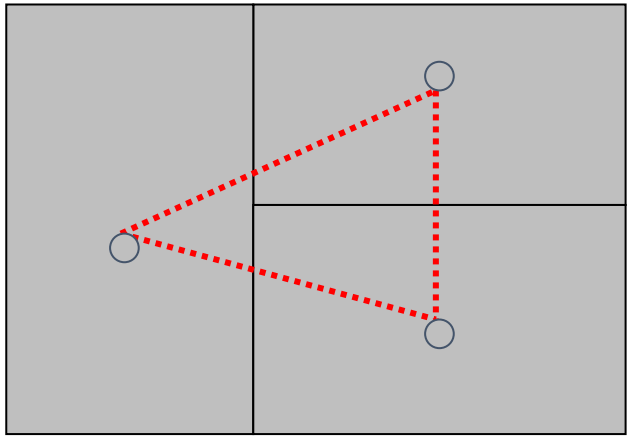


- For hard blocks, the orientations can be changed
  - E.g., rotate by 90 degree

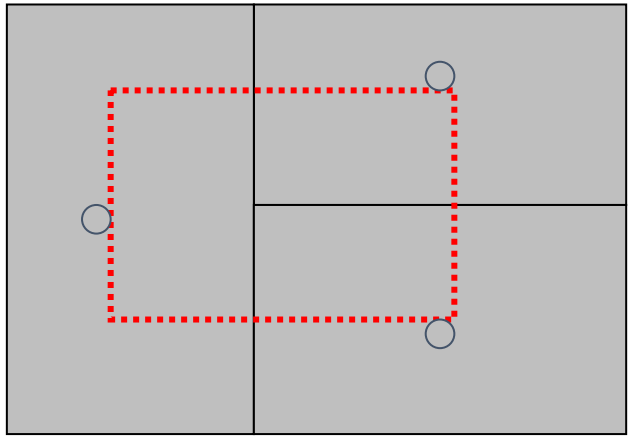


Wirelength Estimation

- **Exact wirelength of each net is unknown until routing**
  - Gate-to-gate connection is routed/established via “wire”
- **Some possible wirelength estimations:**
  - Center-to-center estimation
  - Half-perimeter wirelength (HPWL) estimation



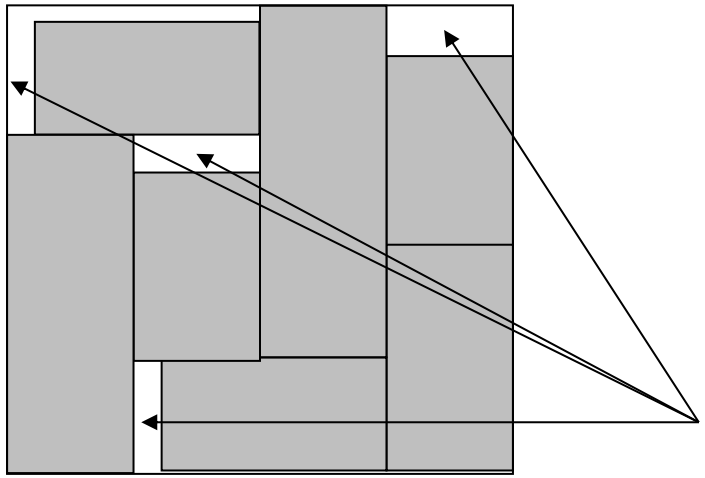
Center-to-center



Half-perimeter wirelength

Dead Space

- Dead space is the space that is wasted



Dead space (white space)

- Minimizing area is the same as minimizing dead space
- Dead space percentage is computed as

$$(A - \Sigma_i A_i) / A \times 100\%$$

Objective Function

- A commonly used objective function is a weighted sum of area (or dead space area) and wirelength

$$\text{cost} = \alpha A + \beta L$$

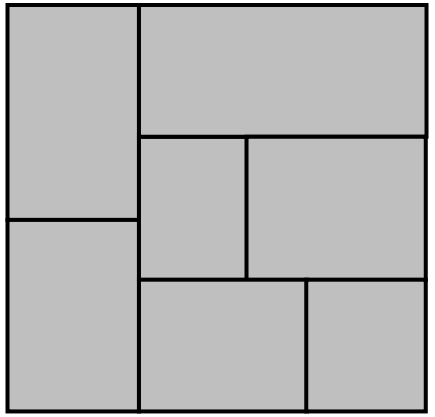
  - A is the total area of the packing
  - L is the total wirelength
  - α and β are tuning parameters

Formal Problem Formulation (revisited)

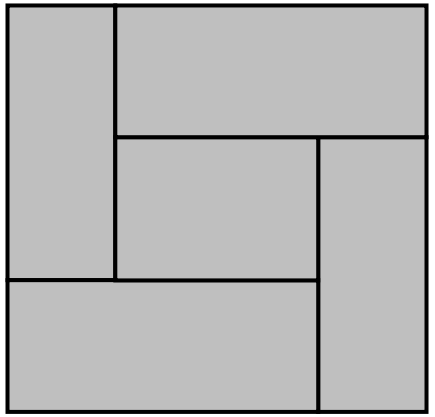
- **Input:**
  - n Blocks with areas $A_1, \ldots, A_n$
  - Bounds r_i and s_i on the aspect ratio of block B_i
- **Output:**
  - Coordinates (x_i, y_i), width w_i and height h_i for each block such that $h_i w_i = A_i$ and $r_i \leq h_i/w_i \leq s_i$
- **Objective:**
  - Minimize = $\alpha A + \beta L$
    - A: total area
    - L: wirelength

Slicing Floorplan

- **Slicing floorplan**
  - One that can be obtained by repetitively subdividing (slicing) rectangles horizontally or vertically
- **Non-slicing floorplan**
  - One that may not be obtained by repetitively subdividing alone
- **Slicing floorplans are much easier to handle**
  - Efficient data structures exist for efficient computational manipulations

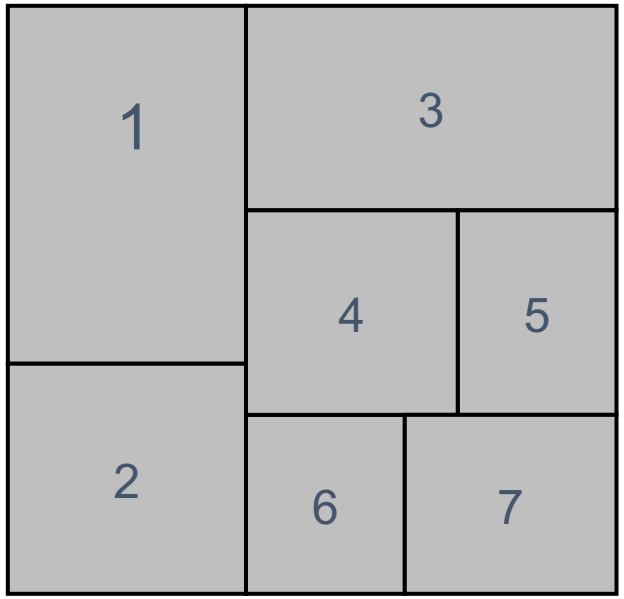


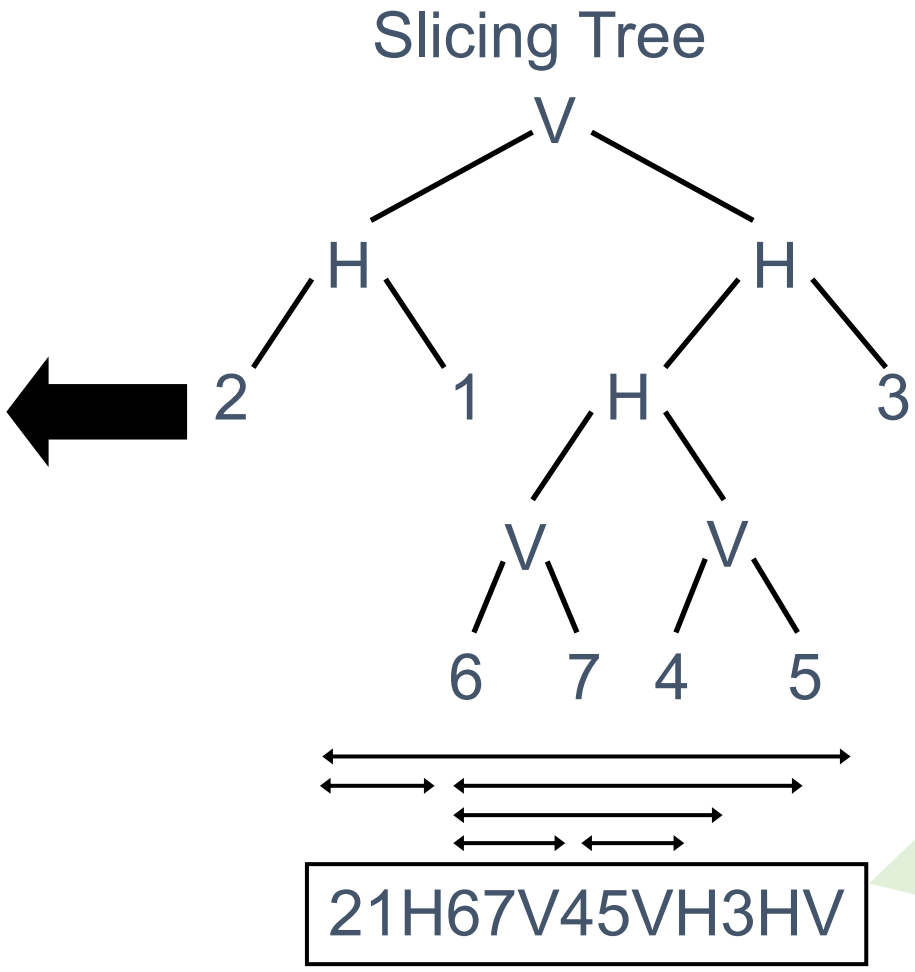
Slicing Floorplan



Non-slicing Floorplan

Representation of Slicing Floorplan



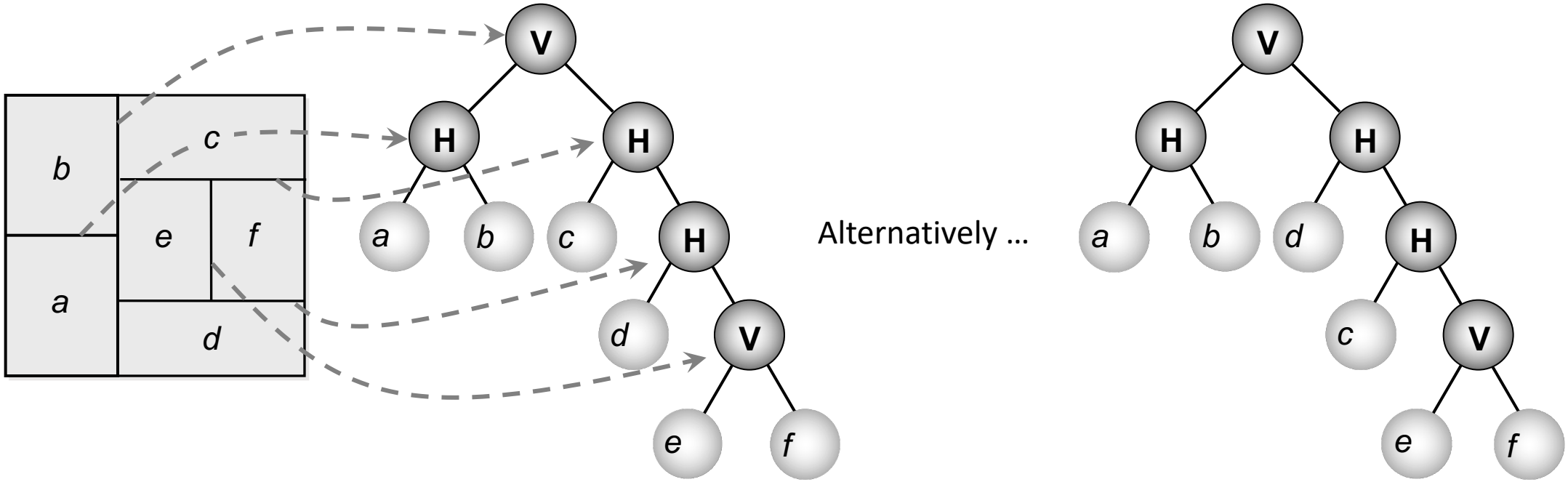


Q: Is slicing tree unique?

Polish Expression: Postorder traversal of the slicing tree

Slicing Tree is NOT Unique

- Two possible slicing trees representing the same floorplan

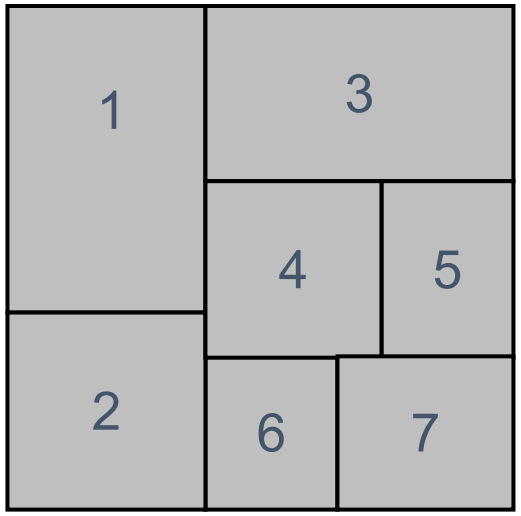


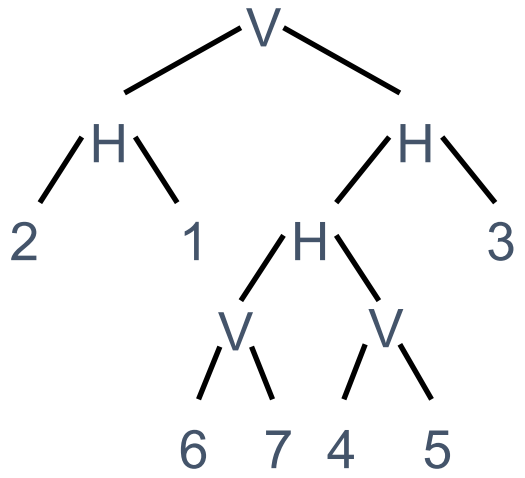
Polish Expression

- Succinct representation of slicing floorplan
  - Roughly specifies relative positions of blocks
- Postorder traversal of slicing tree algorithm (recursive)
  1. Call postorder traversal of left sub-tree
  2. Call postorder traversal of right sub-tree
  3. Print the label of the current root
- For *n* blocks, a Polish Expression contains *n* operands (blocks) and *n-1* operators (H, V)
- However, for a given slicing floorplan, the corresponding slicing tree (and hence polish expression) is not unique. Therefore, there is some redundancy in the representation

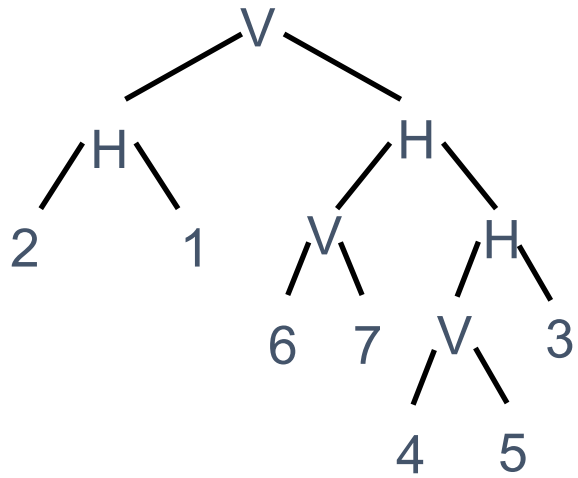
Redundancy of Polish Expression

- Recall that the slicing tree representation is not unique





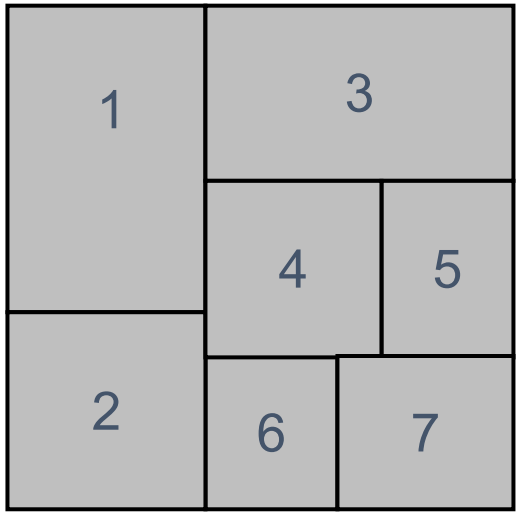
21H67V45VH3HV



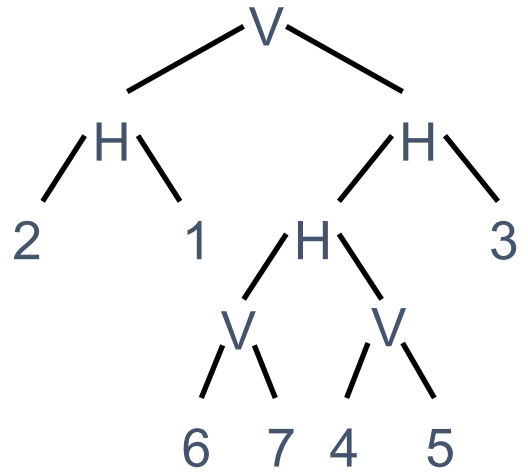
21H67V45V3HHV

Skewed and Normalized Slicing Tree

- **Skewed Slicing Tree:** no node and its right child are the same
- **Normalized Polish Expression:** no consecutive H's or V's

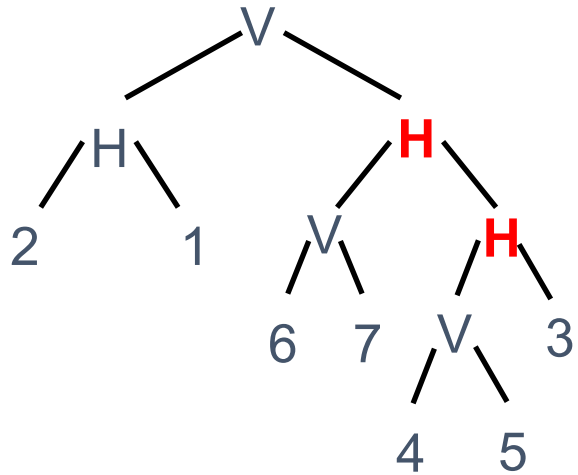


Skewed slicing tree
(also normalized)



21H67V45VH3HV

Not normalized!



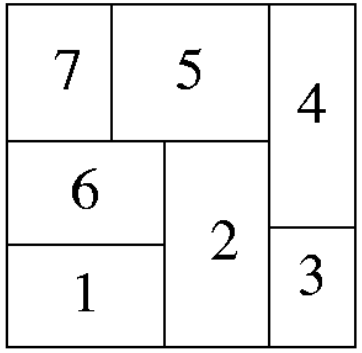
21H67V45V3HHV

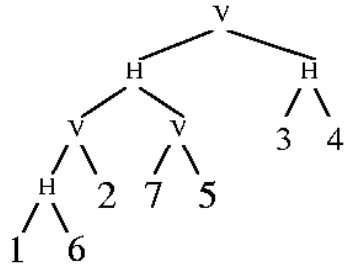
Normalized Polish Expression

- There is a 1-1 correspondence between Slicing Floorplan, Skewed Slicing Tree, and Normalized Polish Expression
- We can use **valid Normalized Polish Expression (NPE)** to represent slicing floorplans
- We formulate as a state space search problem

Solution Representation

- An expression $E = e_1 e_2 \ldots e_{2n-1}$, where $e_i \in \{1, 2, \ldots, n, H, V\}$, $1 \le i \le 2n-1$, is a valid **Polish expression** of length $2n-1$ if
  - every operand j, $1 \le j \le n$, appears exactly once in E;
  - for every sub-expression $E_i = e_1 \ldots e_i$, $1 \le i \le 2n-1$, #operands > #operators.





E = 16H2V75VH34HV

*E = 16+2*75*+34+**

Postorder traversal of a tree!

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

# of operands = 4 ..... = 7

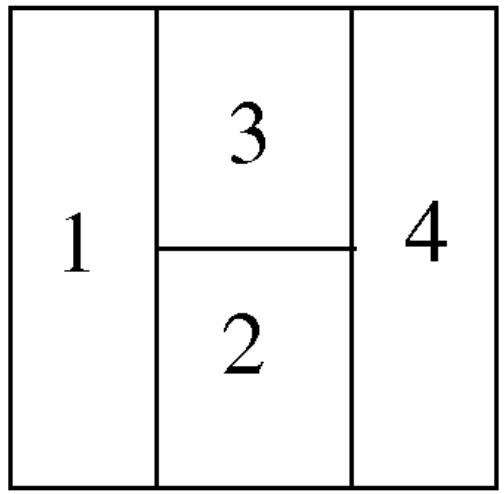
# of operators = 2 ..... = 5

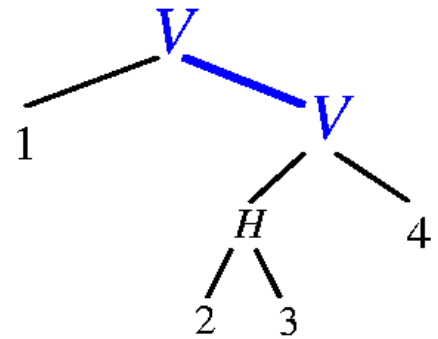
- Polish expression ↔ Postorder traversal.
- *ijH*: *i* below *j*; *ijV*: *i* on the left of *j*.

Example

- 12VH3: invalid
- 123VH: valid
- 1234567HHHHVV: valid
- 1HVVHHV743526: invalid

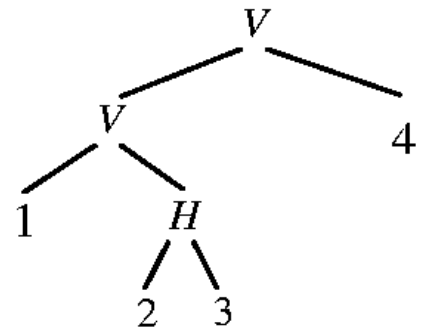
How to Remove Redundant Rep?





$E = 123H4VV$

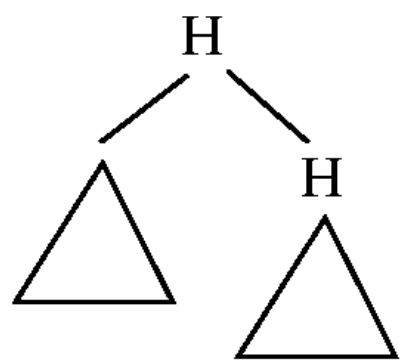
non−skewed!



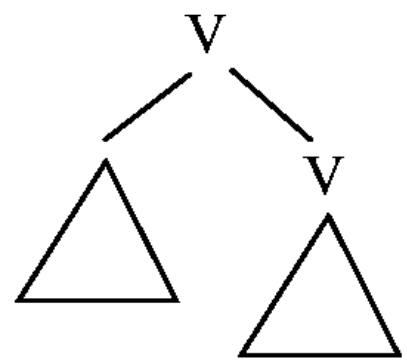
$E = 123HV4V$

skewed!

Non−skewed cases



..... HH .....



..... VV .....

Neighborhood Structure

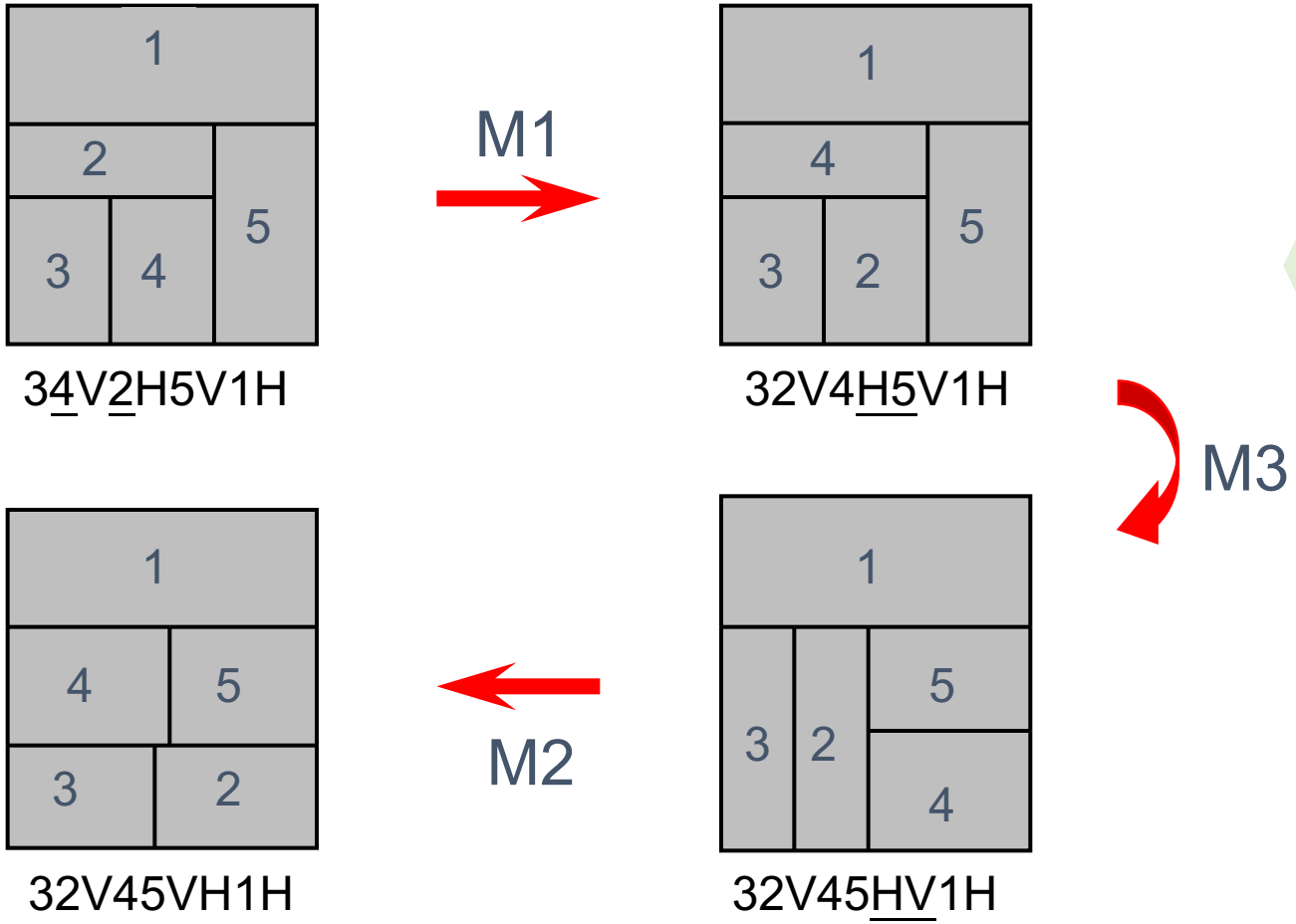
- Chain: HVHVH… or VHVHV…

Chains

- The moves:
  - M1: Swap adjacent operands (ignoring chains)
  - M2: Complement some chain
  - M3: Swap 2 adjacent operand and operator
    - M3 can give you some invalid NPE. Checking for validity after M3 is needed

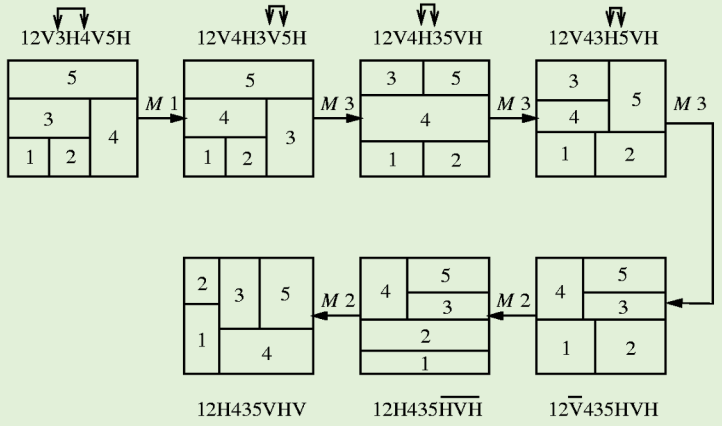
- It can be proved that every pair of valid NPE are connected

Example of Moves

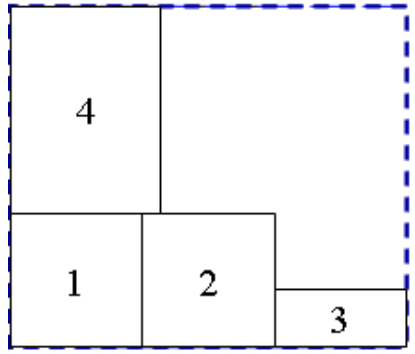


Each move gives a new floorplan result

Key idea: try out many moves and acquire the best solution?

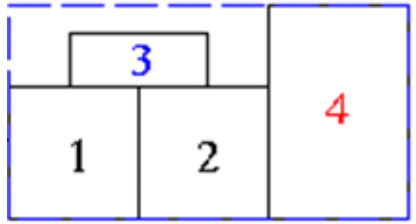


Example of Moves (cont'd)



12V4H3V

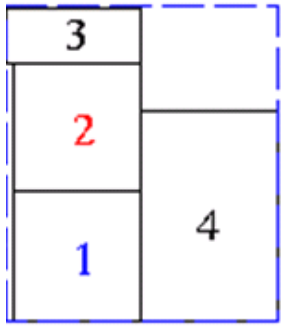
M1



12V3H4V

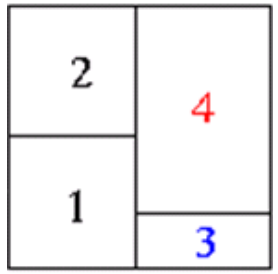
Optimal result is acquired after three moves, M1, M2, and M3

M2



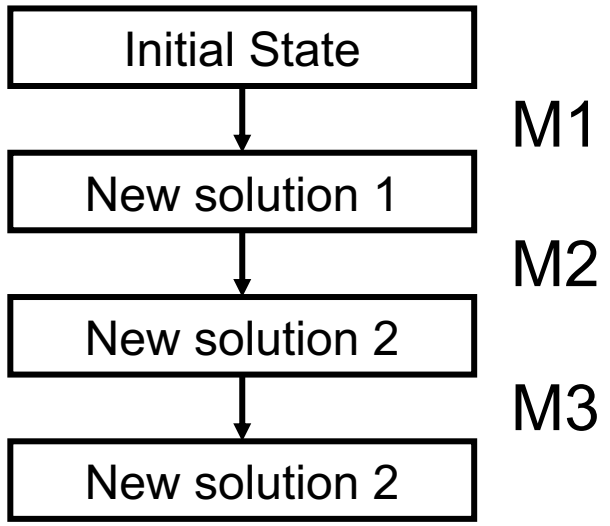
12H3H4V

M3



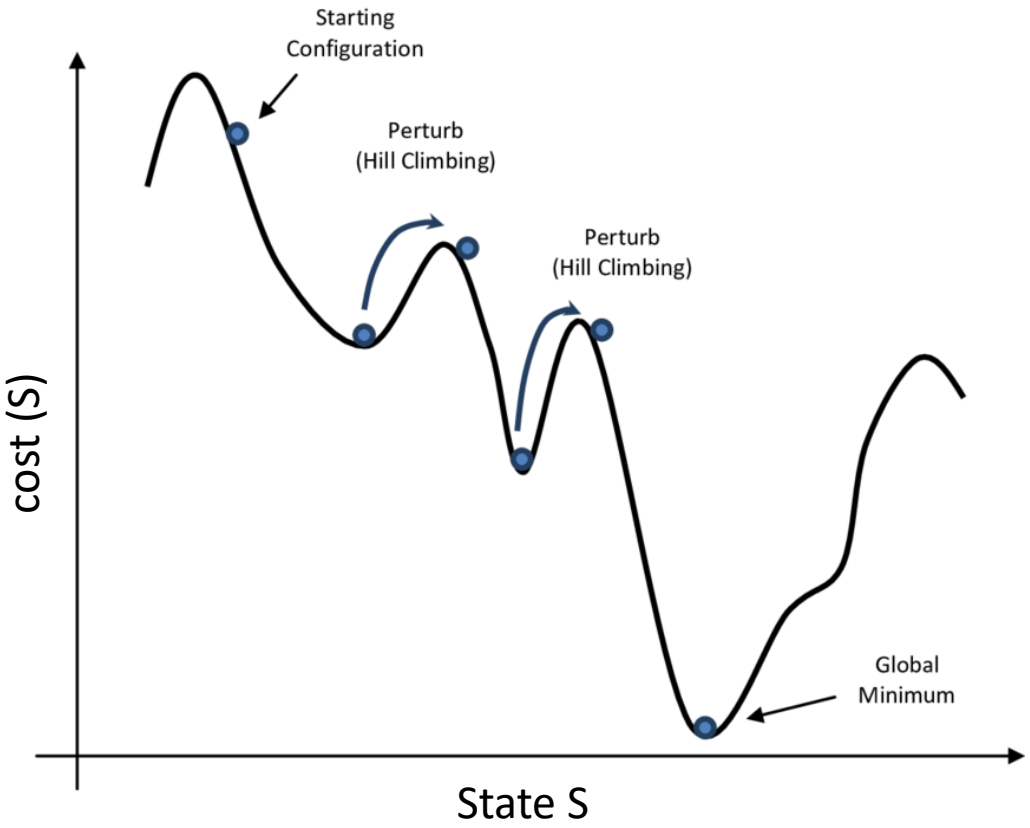
12H34HV

Can we solve this systematically?



Simulated Annealing (SA) Algorithm

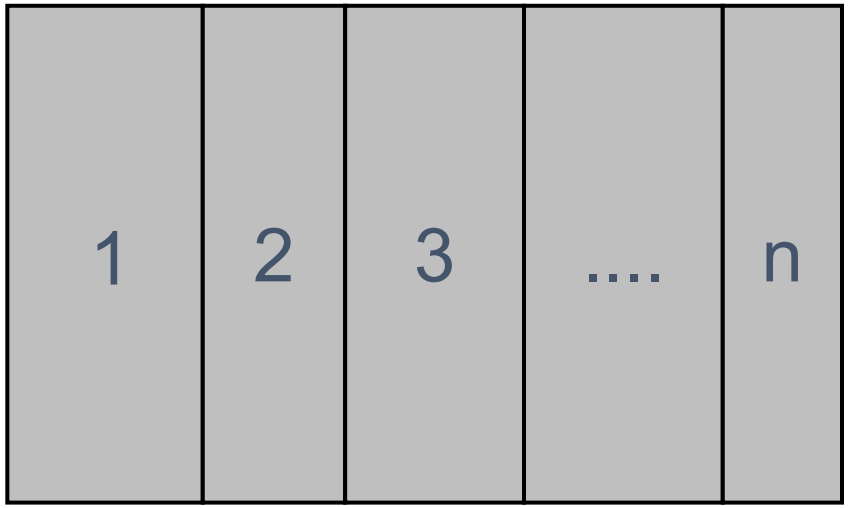
```
1 begin
2 Get an initial solution S;
3 Get an initial temperature T > 0;
4 while not yet "frozen" do
5     for 1  i  P do
6         Pick a random neighbor S' of S;
7         △← cost(S') – cost(S);
          /* down hill move */
8         if △  0 then S←S'
          /* uphill move */
9         if △ > 0 then S←S' with probability;
10    T←rT ; /* reduce temperature */
11 return S
12 end
```



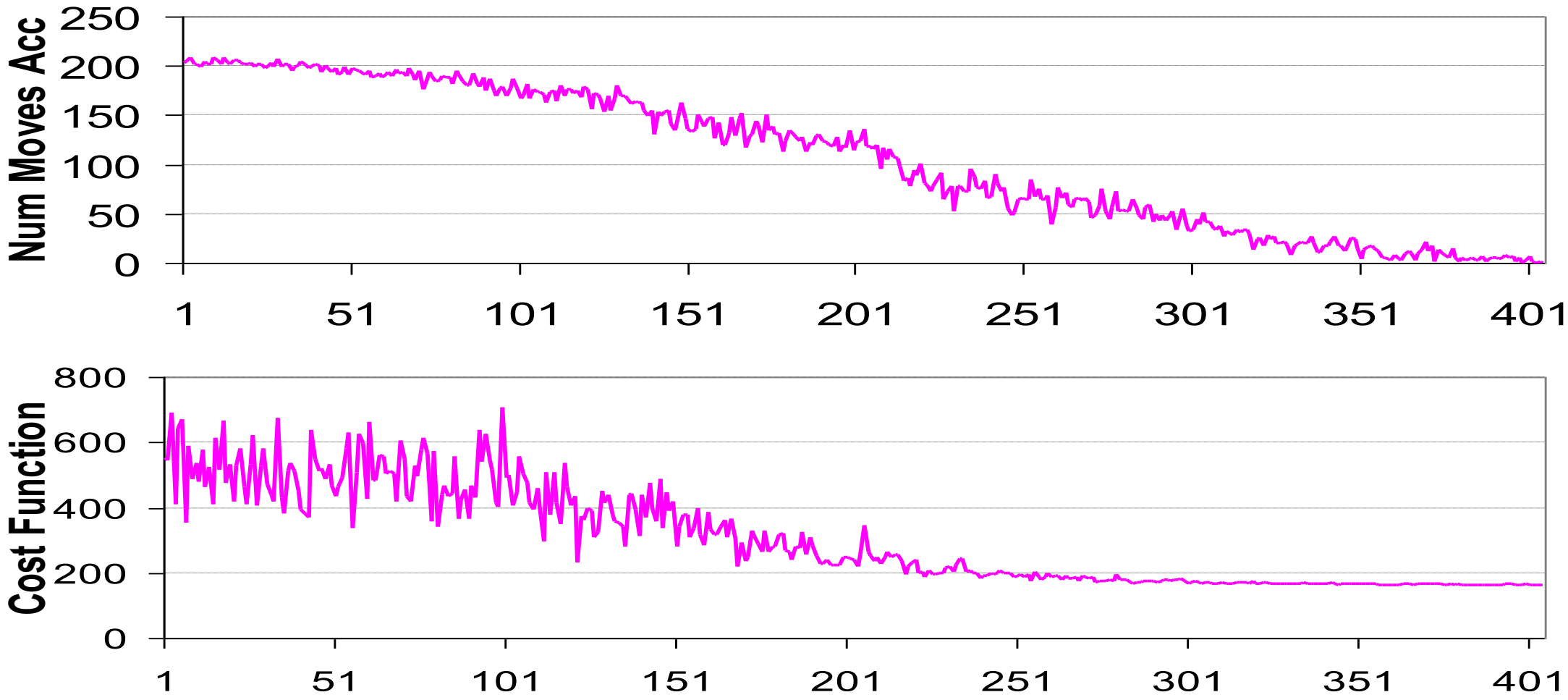
SA-based Floorplan Optimization

- $T_i = \alpha T_{i-1}$ where α=0.85
- At each temperature, try k $\times$ n moves
  - k is around 5 to 10
- Terminate the annealing process if
  - either # of accepted moves < 5%
  - or the temperature is low enough
- Cost metric: $\alpha A + \beta L$
  - Area + Wirelength

Initial State: 1V2V3V…nV

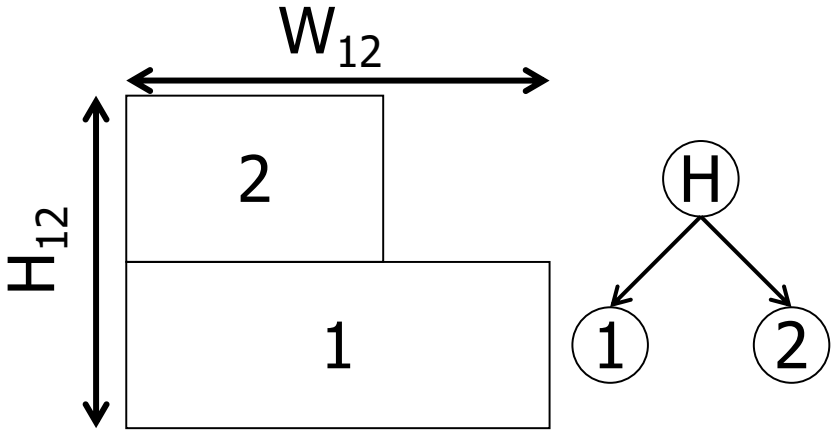


Result of SA-based Floorplan



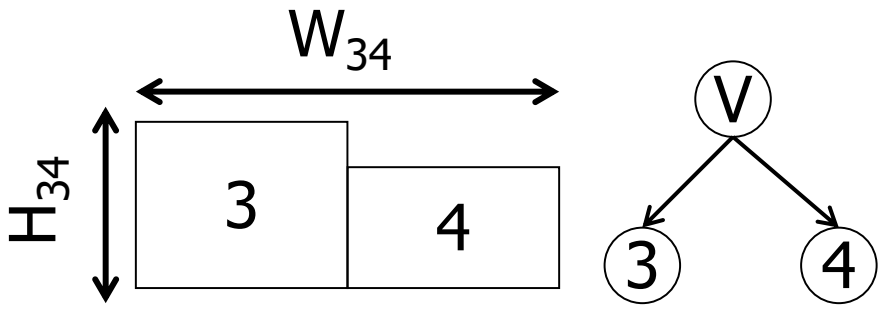
How Do We Know the Area from a PE?

- **Binary operator: V and H**
  - V: maximum on width and summation on height
  - H: maximum on height and summation on width



$W_{12} = \max(W_1 , W_2)$
$H_{12} = H_1 + H_2$

(a) Postfix expression: 12H

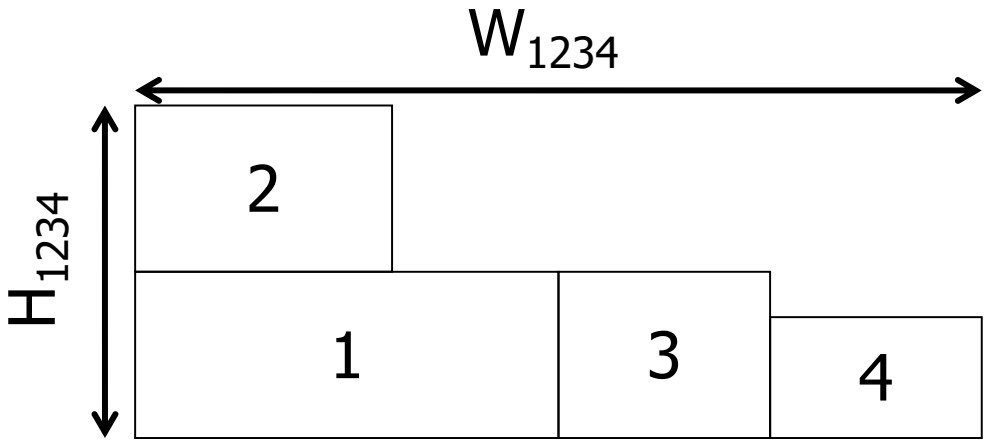


$W_{34} = W_3 + W_4$
$H_{34} = \max(H_3 , H_4)$

(b) Postfix expression: 34V

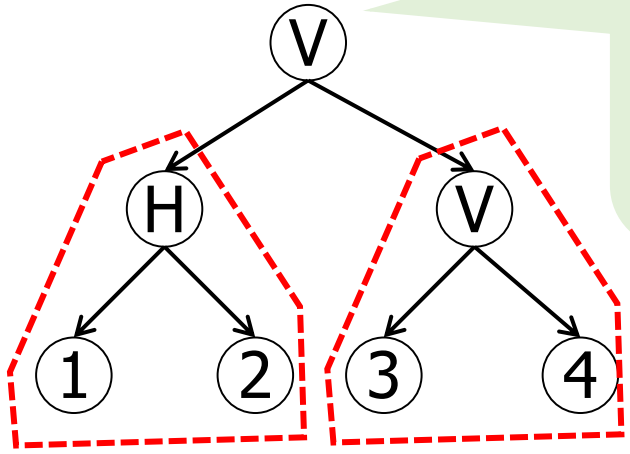
Recover Area Recursively

- **Binary operator: V and H**
  - V: maximum on width and summation on height
  - H: maximum on height and summation on width



$W_{1234} = W_{12} + W_{34}$

$H_{1234} = \max(H_{12}, H_{34})$

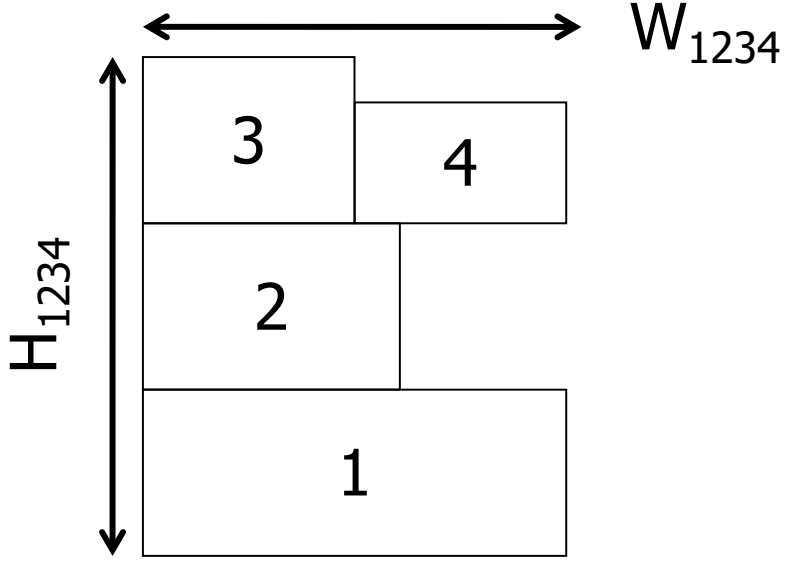


Bottom-up merging left and right subtrees

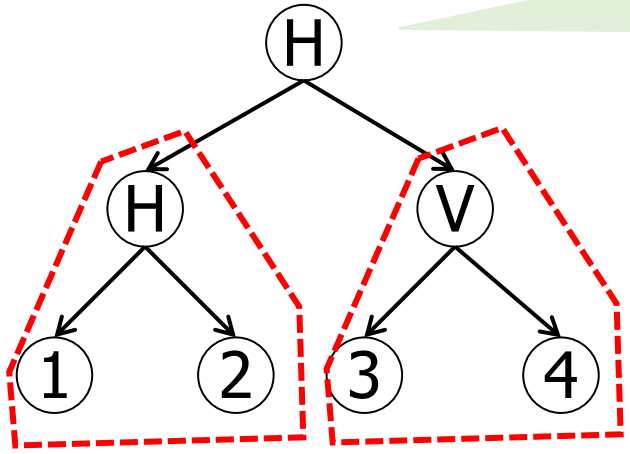
(c) Postfix expression: 12H34VV

Recover Area Recursively (cont'd)

- **Binary operator: V and H**
  - V: maximum on width and summation on height
  - H: maximum on height and summation on width



$W_{1234} = \max(W_{12} + W_{34})$
$H_{1234} = H_{12} + H_{34}$

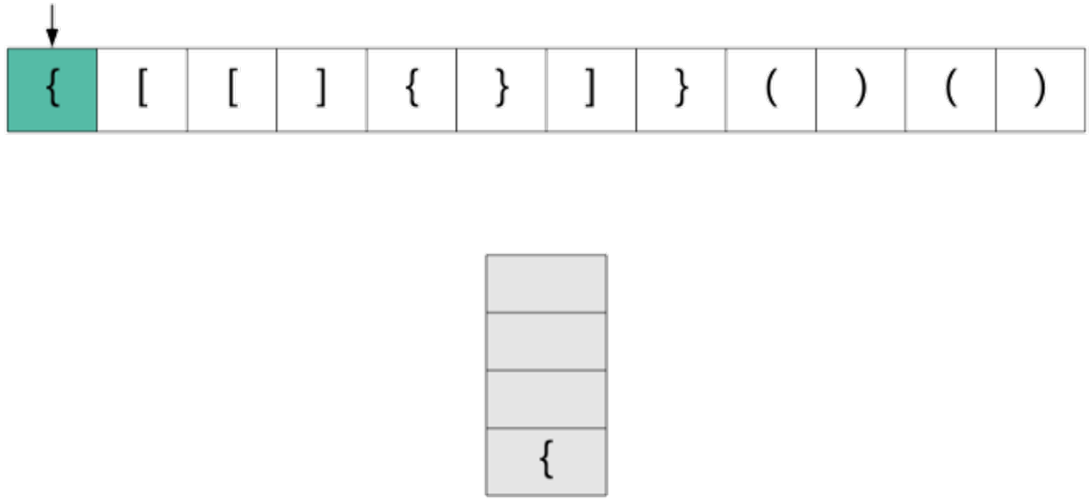


(d) Postfix expression: 12H34VH

Bottom-up merging left and right subtrees

Implementation using Stack

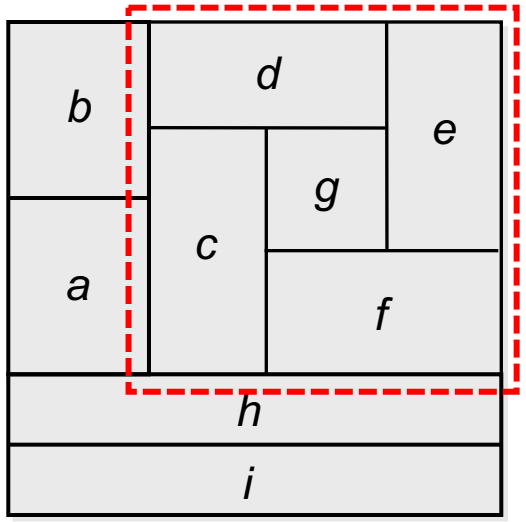
- Very similar to the parathesis checking problem

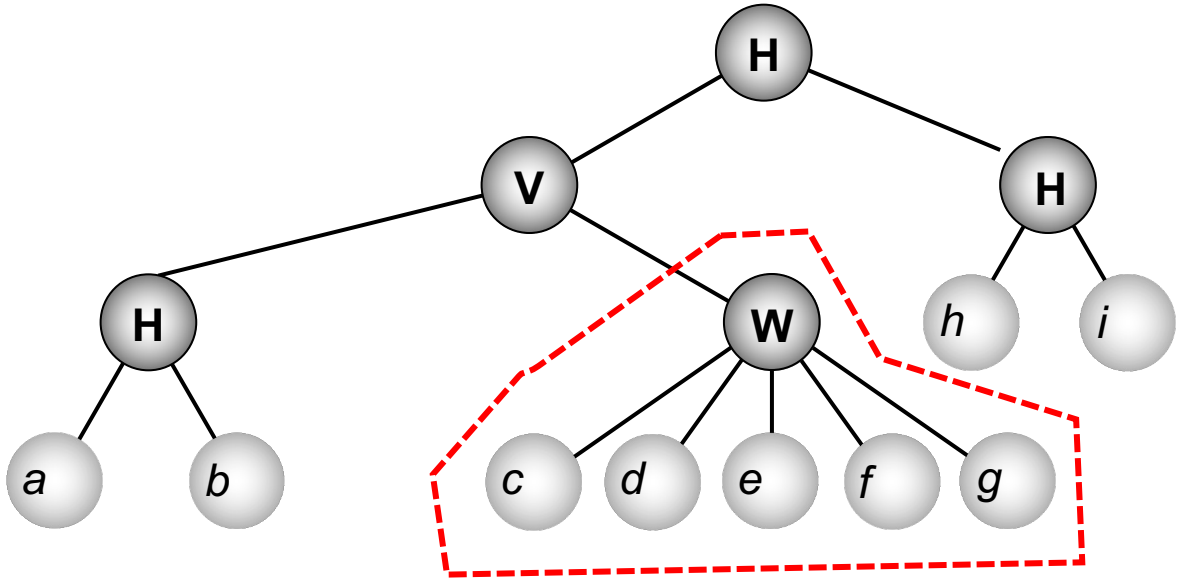


- Modification
  - Number: push the module into the stack
  - V/H: pack the top-two blocks (subtrees) vertically/horizontally

How do We Handle Non-Slicing Floorplan?

- Extend NEP to include pre-defined non-slicing block





H — Horizontal division (objects to the top and bottom)

V — Vertical division (objects to the left and right)

W — Wheel (4 objects cycled around a center object)

Summary

- **Floorplan problem is NP hard (difficult to solve efficiently)**
- **How to represent it compactly is a big deal**
  - Slicing is easier to deal with, so let's start with it
  - Next lectures will cover other representations
- **Polish expression is an elegant slicing-tree representation**
  - Need to be unique (NPE) to remove redundant space
  - Can be implemented efficient using linear-time data structure
- **Simulated annealing algorithm fits well to floorplan**
  - Optimize the floorplan area and wirelength simultaneously