Floorplanning-2

Hierarchical Bottom up approach

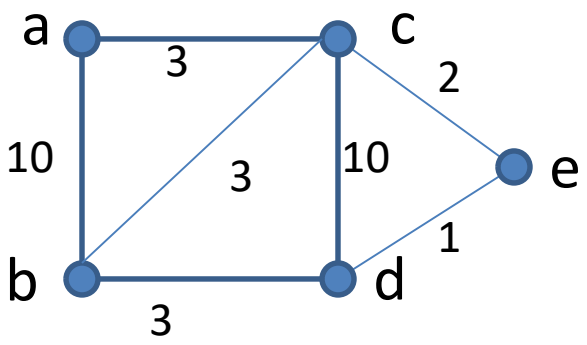
Modules are represented as a graph where edges represent the connectivity of modules

Modules with high connectivity are clustered together
(limiting the number in each cluster to some level)

An optimal floorplan for each cluster is determined by exhaustive enumeration

The cluster is merged into a larger modules for higher level processing

Hierarchical Bottom up approach: Greedy procedure

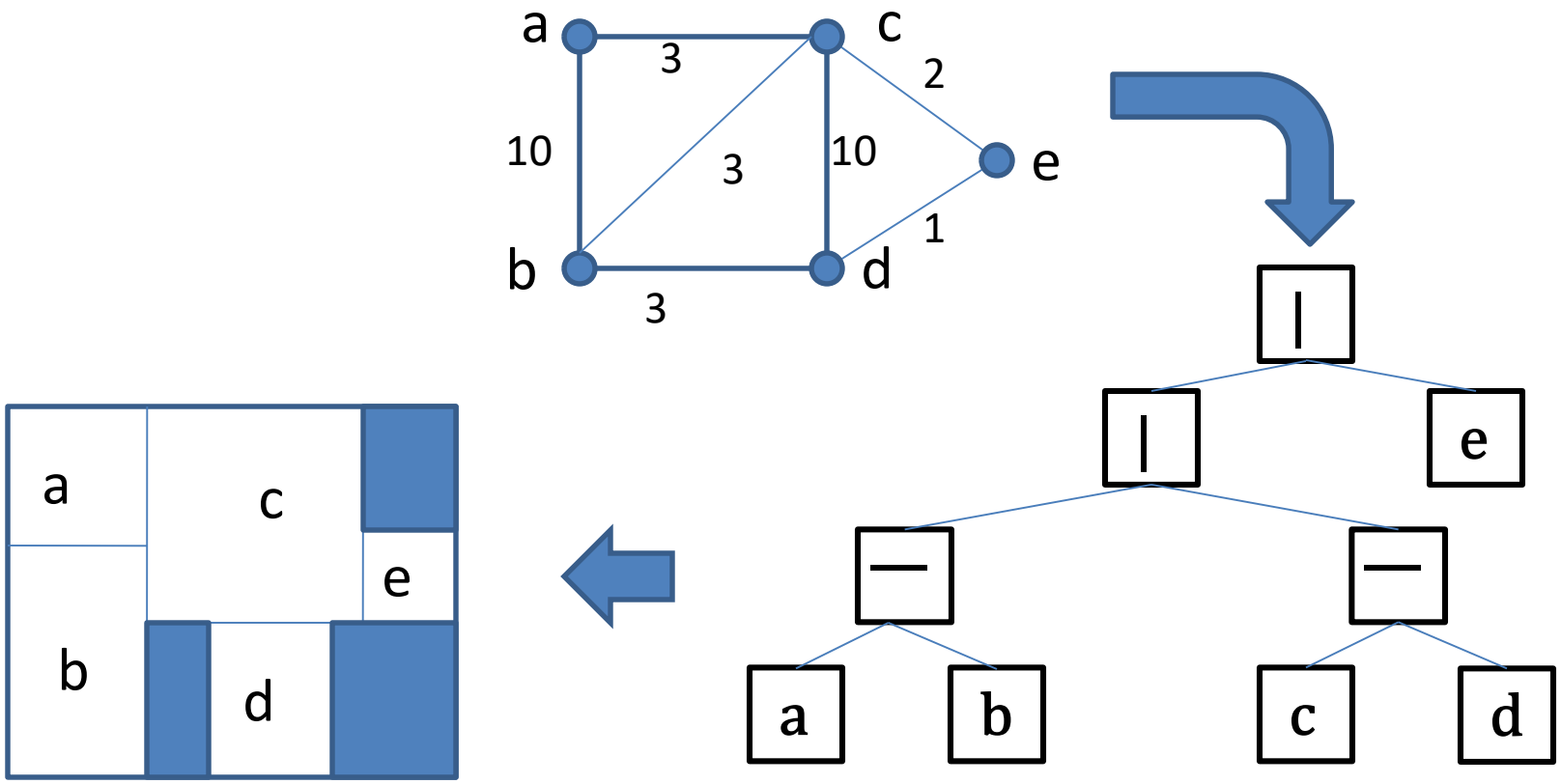


Choose the heaviest edge, two modules of this edge are clustered together (restrict the number of each cluster to some d)

Vertices in a cluster are merged and edge weights are summed up accordingly

Repeat the above procedure unless finished

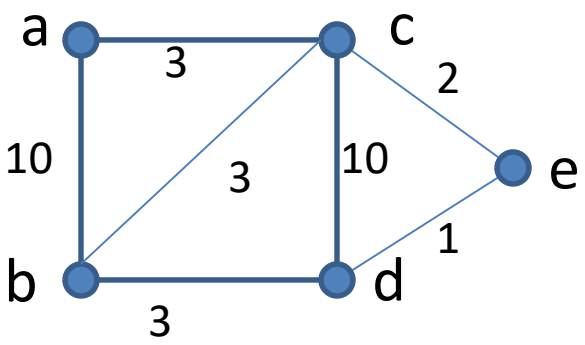
Hierarchical Bottom up approach: Greedy procedure

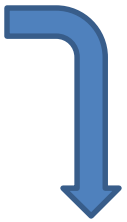


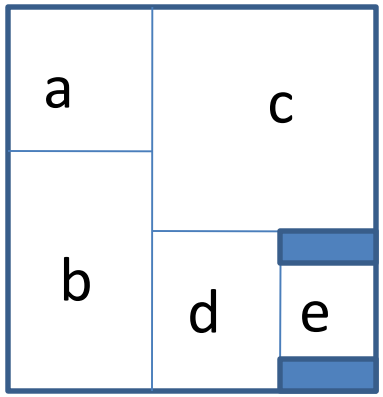
The problem: Some lightweight edges may be chosen at some higher levels of hierarchy

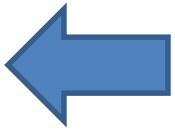
The possible solution: Arbitrarily assign a small cluster to a neighboring cluster when its size is too small

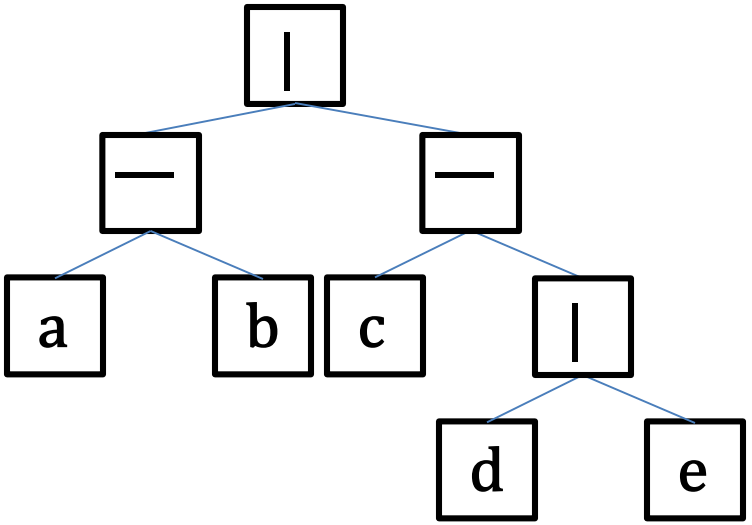
Hierarchical Bottom up approach: Greedy procedure











Assign a small cluster to a neighboring cluster when its size is too small

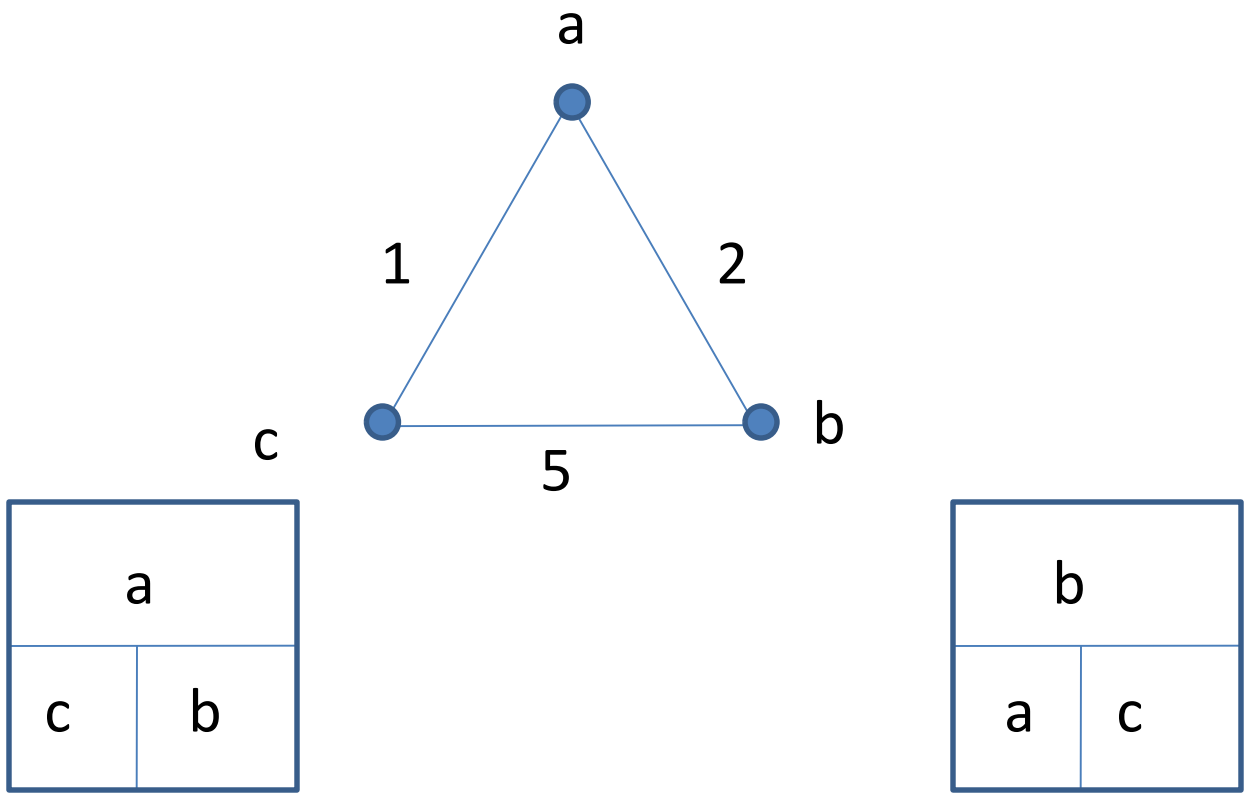
Cost of Floorplan

Cost is dependent on area and connections

Area can be obtained from dimensions of each cluster

Routing cost= the edge weights multiplied by distance between the centres of clusters

Cost of Floorplan:
Estimating routing cost: an example



distance(a,b)= distance (a,c)= 2.24
distance (b,c)=2
Cost= 2.24 x 2 +2.24 x 1 +2.0 x 5 = 16.72

distance(a,b)= distance (b,c)= 2.24
distance (a,c)=2
Cost= 2.24 x 5 +2.24 x 2 +2.0 x 1 = 17.68

Hierarchical Top Down approach

Partition of the module

Each partition is assigned to a child floorplan

Repeat the above procedure recursively

May result two partitions of incompatible sizes

Top Down and Bottom up approaches may be combined

Floorplanning: Simulated annealing approach

The idea originated from observations of crystal formation

When a material is heated, molecules move around randomly
When it is cooled, it forms the crystal

The above process is executed repeatedly

The annealing process moves from one solution to other

Random moves of molecules is simulated by randomly accepting moves during the initial phases of algorithm
As algorithm proceeds, the temperature decreases

The algorithm accepts a move (v_i, v_j) if cost (v_i) $\leq$ cost (v_j)

The best cost among all the solutions are recorded

Floorplanning: Simulated annealing approach

Guiding Factors -
Solution space
movement from one solution to another
cost evaluation function

A binary tree representation of a floorplan is converted into a normalized Polish expression

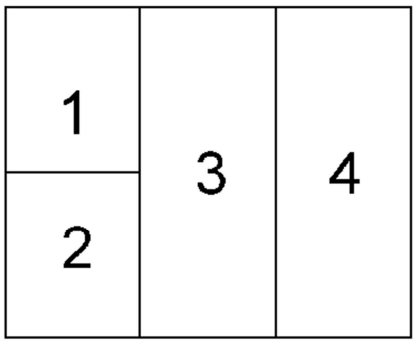
Movements
Exchange two operands when there are no other operands in between
Complement a series of operators between two operands
Exchange adj operand and operator if the resulting expression is a normalized Polish expression

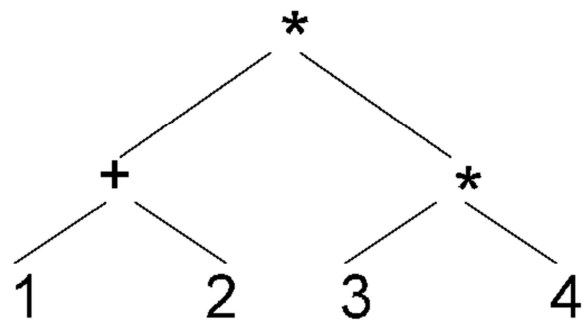
Floorplanning: Simulated annealing approach

- solution is restricted to slicing floorplans only
- for easy representation of slicing floorplans, *Polish* notation is used
- Polish expression is a string of symbols obtained by traversing the binary floorplan tree in post-order

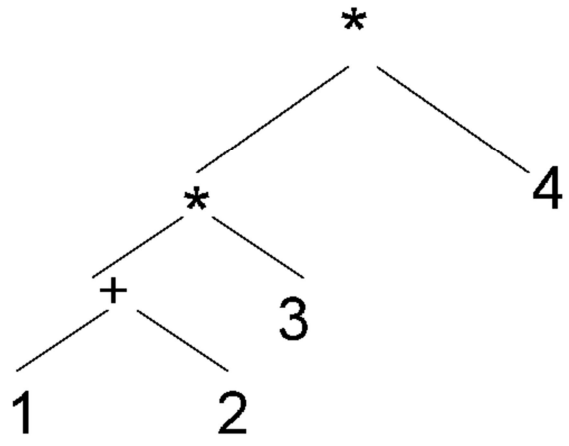
Floorplanning: Simulated annealing approach

Generating polish expressions from floorplan trees -





1 2 + 3 4 * *



12+3*4*

Normalized Polish expression -

*if there is no consecutive *'s and +'s in the expression*

Floorplanning: Simulated annealing approach

Theorem. There is a 1-1 correspondence between the set of normalized Polish expressions of length 2n - 1, and the set of slicing structures with n leaves

Three types of movements used to move from one floorplan to another -

1. Exchange two operands when there are no other operands in between
2. Complement a series of operators between two operands
3. Exchange adjacent operand and operator if the resulting expression is in normalized Polish form

Floorplanning: Simulated annealing approach

Annealing Schedule

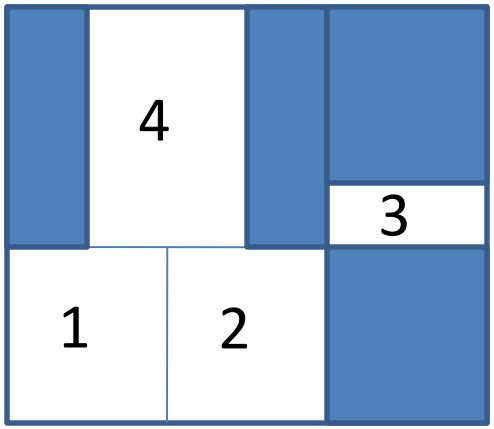
Temperature changes in the manner $T_i = r. T_{i-1}$
Typical value of r is 0.85
Algorithm starts with an initial normalized Polish expression

Default initial Polish expression is
12 * 3 * 4 * …. * n *, which corresponds to placing n modules horizontally next to each other

At each temperature, enough moves are attempted until there are N downhill moves or the total number of moves exceeds 2N, where $N = \gamma.n$, where γ is user specified symbol

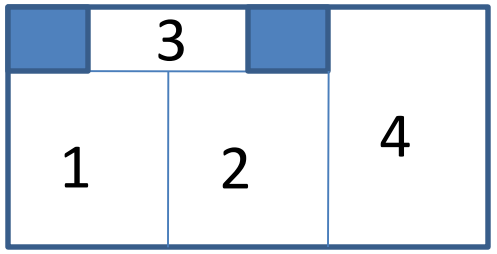
Simulated annealing approach

12*4+3*



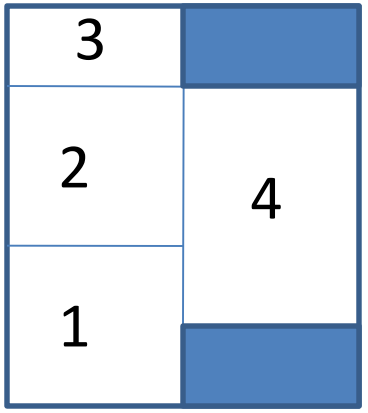
Exchange two operands

12*3+4*



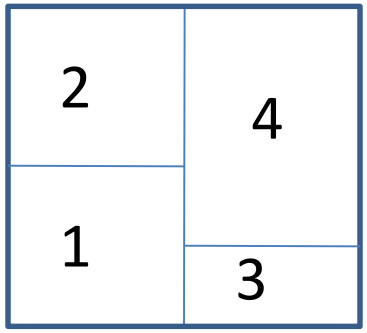
Complement a series of operators between two operands

12+3+4*



Exchange adjacent operator and operand

12+34+*



Sizing

- selecting a particular (length, width) pair for each block
- objective is to minimize space wastage
- complex for non-slicible floorplans

Floorplan sizing problem

The circuit models can usually be implemented in different sizes

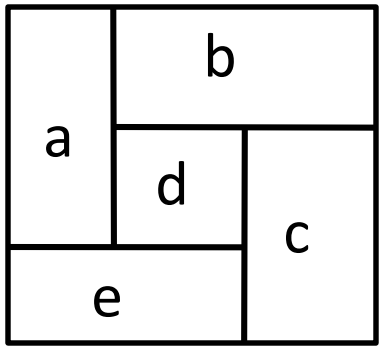
Each module has many implementations with different size, speed, and power consumption trade off

The floorplan sizing problem is actual module implementation

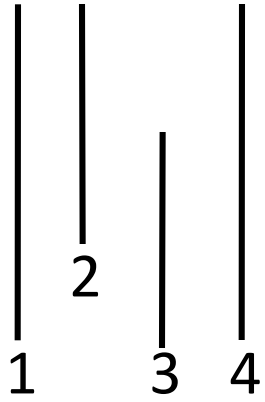
If a cell has only one implementation, it is called a fixed cell

Floorplan sizing problem

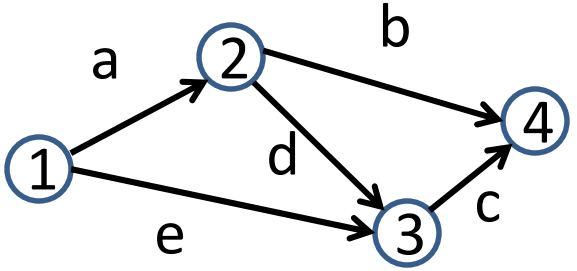
Assume a fixed cell environment



Floorplan



Maximal vertical segments



Horizontal dependency graph

Horizontal dependency graph:

- A directed acyclic graph
- The length of each edge is the width of the corresponding module in the floorplan
- The longest path from leftmost to rightmost segment is the minimum width of the chip

The height of the chip can be determined by vertical dependency graph

Hierarchical Floorplan Sizing

Consider variable cells on a sliceable floorplan

Given:

A sliceable floorplan F, and for each module M_i, a set of possible implementations of the module, where each implementation is described by a pair of real numbers (w_k, h_k) representing width and height of module

If a module has the realizations (w_1, h_1), (w_2, h_2), (w_3, h_3), ……., the realizations of are sorted in a manner such that $w_p < w_q$ and $h_p > h_q$ for all $p < q$

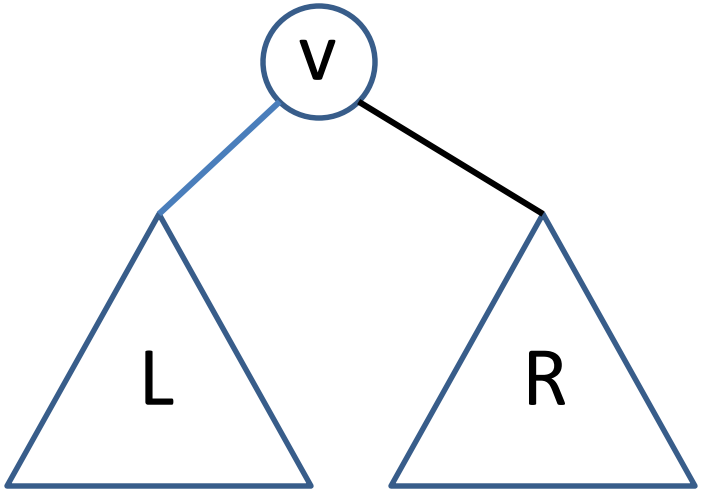
For $w_p < w_q$ if $h_p \not< h_q$, exclude (w_q, h_q) from the list

If a module M_i has si realizations, then

$w_1 < w_2 < w_3, \ldots\ldots < w_{si}$, and

$h_1 > h_2 > h_3, \ldots\ldots > h_{si}$,

Hierarchical Floorplan Sizing



Lemma: Given two subfloorplans corresponding to two subtrees of a node v, one with t and other with s nonredundant implementations, then v has at most s+t-1 nonredundant realizations

Procedure vertical node sizing

Input: Two sorted lists L= {(a_1,b_1), (a_2,b_2), ……,(a_s,b_s)}, and R= {(x_1,y_1), (x_2,y_2), ……., (x_t,y_t)}, where $a_i<a_j$, $b_i>b_j$, $x_i,<x_j$, $y_i>y_j$ for all i < j

Output: A sorted list H= {(c_1,d_1), (c_2,d_2), ……,(c_u,d_u)}, where $c_i<c_j$, $d_i>d_j$, for all i < j

H= Ø, i=1, j=1, k=1

while (i ≤s) and (j≤ t) do

$(c_k, d_k)= (a_i+ x_j, \max(b_i, y_j)$; H= H ∪$(c_k, d_k)$; k=k+1

if $\max(b_i, y_j)= b_i$, then i=i+1;

if $\max(b_i, y_j)= y_j$, then j=j+1;

Hierarchical Floorplan Sizing: Algorithm

Input: A binary tree corresponding to floorplan with leave nodes representing the modules

Output: Complete floorplan

From each leaf node traverse the tree up to root

by vertical node sizing procedure

The sizing algorithm runs in O(dn) time, where n is the number of modules and d is the height of the the slicing tree

For a balanced binary tree, d= logn

In the worst case, d=n

Linear Programming Technique

Nonhierarchical floorplan sizing method

No restriction on the organization of modules

Notation

(w_i,h_i) : width and height of module M_i

(x_i,y_i) : coordinates of the lower left corner of module M_i

(x,y) : width and height of the final floorplan

(a_i,b_i) : min and max values of the aspect ratio w_i/h_i of module M_i

For any module M_i, $a_i \leq w_i/h_i \leq b_i$

Usually, $a_i = 1/b_i$

The solution is to find (x_i, y_i, w_i, h_i) such that all constraints are satisfied and xy is minimized

Linear Programming Technique: Constraints

Nonoverlap constraint

$x_i+w_i \leq x_j$ (M_j is to the right of M_i), or
$x_i-w_j \geq x_j$ (M_j is to the left of M_i), or
$y_i+h_i \leq y_j$ (M_j is to the above of M_i), or
$y_i-h_j \geq y_j$ (M_j is to the below of M_i)

At least one of the above constraints must be satisfied

For each pair (M_i , M_j), p_{ij} (0 or 1) and q_{ij} (0 or 1) are introduced
W and H = upper bounds of the width and height of feasible solution region

$$x_i+w_i \leq x_j + W(p_{ij} + q_{ij})$$
$$x_i-w_j \geq x_j + W(1- p_{ij} + q_{ij})$$
$$y_i+h_i \leq y_j + H(1+ p_{ij} - q_{ij})$$
$$y_i-h_j \geq y_j + H(2- p_{ij} - q_{ij})$$

The above system of inequalities solves the overlap constraint

Linear Programming Technique: Constraints

Module size constraint

Consider that the blocks are not of fixed sizes

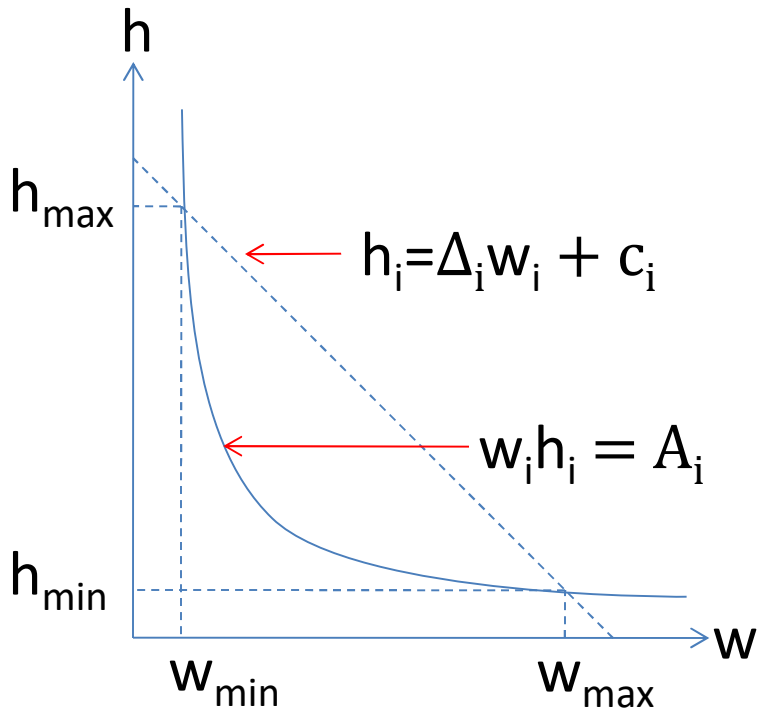
For each module M_i, a lower bound on area A_i and aspect ratios a_i, b_i are given, s.t., $w_i h_i \geq A_i$, and $a_i \leq w_i / h_i \leq b_i$

$w_{min} = \sqrt{A_i a_i}$, $w_{max} = \sqrt{A_i b_i}$, $h_{min} = \sqrt{A_i / b_i}$, $h_{max} = \sqrt{A_i / a_i}$

$h_i = \Delta_i w_i + c_i$

$\Delta_i = (h_{max} - h_{min}) / (w_{min} - w_{max})$

$c_i = h_{max} - \Delta_i w_{min}$



The ineuqualities to be solved

$x_i + w_i \leq x_j + W(p_{ij} + q_{ij})$

$x_i - w_j \geq x_j + W(1 - p_{ij} + q_{ij})$

$y_i + \Delta_i w_i + c_i \leq y_j + H(1 + p_{ij} - q_{ij})$

$y_i - \Delta_j w_j - c_j \geq y_j + H(2 - p_{ij} - q_{ij})$

The Cost function

The function to be minimized = the total area of the floorplan A= xy

Let us fix the width W and minimize the height of floorplan

$$\min y$$

$$x_i + w_i \leq W$$

$$y \geq y_i + h_i$$

Complete LP formulation

The function to be minimized = the total area of the floorplan A= xy

Let us fix the width W and minimize the height of floorplan

min y

$x_i, w_i, y_i \geq 0$

$p_{ij}, q_{ij} = 0 \text{ or } 1$

$x_i + w_i \leq W$

$y \geq y_i + h_i$

$x_i + w_i \leq x_j + W(p_{ij} + q_{ij})$

$x_i - w_j \geq x_j + W(1 - p_{ij} + q_{ij})$

$y_i + \Delta_i w_i + c_i \leq y_j + H(1 + p_{ij} - q_{ij})$

$y_i - \Delta_j w_j - c_j \geq y_j + H(2 - p_{ij} - q_{ij})$

Sizing Methods - Linear Programming

The non-linear objective function can be replaced by fixing the width of the floorplan to W, and attempting to minimize y *only*

The inequalities can be fed into an LP solver LINDO, and a solution obtained.

The procedure may be repeated for several values of *W*, and the desirable one picked up

Wire length and routing area can also be estimated and incorporated to form a set of complex *LP* equations