Chapter 3: Floorplanning

Sadiq M. Sait & Habib Youssef

King Fahd University of Petroleum & Minerals
College of Computer Sciences & Engineering
Department of Computer Engineering

September 2003

Introduction

- Problem Statement
- Formal Model
- Objective Function
- Solution Approaches
- Cluster Growth
- Simulated Annealing
- Mathematical Formulation
- Dual Graph
- Discussion

Problem Statement

- Given n rectangular blocks and connectivity information, find dimensions and positions of each block.
- Constraints: Geometric, Routability, Performance.
- Floorplanning is a generalization of Placement.
- Where for placement shape and pin positions of circuit components are fixed, in floorplanning these have some specified flexibility.
- This added flexibility must be captured by the floorplan model.
- Aspects that need to be modelled consist of: The components, the interconnections, the flexible interfaces (blocks and chip), the chip carrier (layout surface), any designer stated constraints, and the objective to optimize.

Example

Module	Width	Height
1	1	1
2	1	1
3	2	1
4	1	2
5	1	3

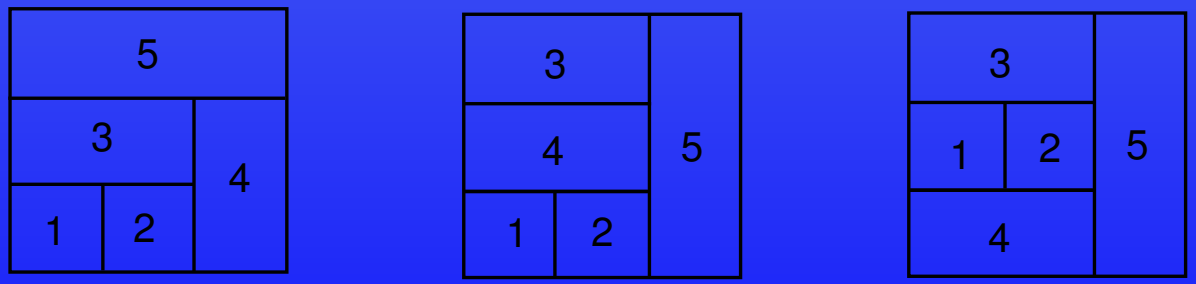


Figure 1: Some feasible floorplans for modules in the table

Floorplanning Model

(1) $S = \{1, 2, \cdots, i, \cdots, n\}$, a set S of n rectangular modules;

(2) S_1 and S_2, a partition of S, where S_1 and S_2 are the sets of the modules with fixed and free orientation respectively;

(3) an interconnection matrix $C_{n \times n} = [c_{ij}]$, $1 \leq i, j \leq n$, where c_{ij} indicates the connectivity between modules i and j;

(4) a list of n triplets $(A_1, r_1, s_1), \cdots (A_i, r_i, s_i), \cdots (A_n, r_n, s_n)$ where, $A_i = w_i \times h_i$, is the area of block i, r_i and s_i are lower and upper bound constraints on the shape of block i ($r_i \neq s_i$ if the block is flexible, and $r_i = s_i$ if the block is rigid);

(5) two additional positive numbers p and q ($p \leq q$), which are lower and upper bound constraints on the shape of the rectangle enveloping the n blocks.

Required Output

A feasible floorplan solution, i.e., an enveloping rectangle R subdivided by horizontal and vertical line segments into n non-overlapping rectangles labelled $1, 2, \cdots, i, \cdots, n$, such that the following constraints are satisfied,

(1) $w_i \times h_i = A_i$, $1 \leq i \leq n$;

(2) $r_i \leq \frac{h_i}{w_i} \leq s_i$ for modules with fixed orientation;

(3) $r_i \leq \frac{h_i}{w_i} \leq s_i$ or $\frac{1}{s_i} \leq \frac{h_i}{w_i} \leq \frac{1}{r_i}$ for modules with free orientation;

(4) $x_i \geq w_i$ and $y_i \geq h_i$, $1 \leq i \leq n$, where x_i and y_i are the dimensions of basic rectangle i,

(5) $p \leq \frac{H}{W} \leq q$, where H and W are the height and width of the enveloping rectangle R.

Required Output - contd

- A feasible floorplan optimizing the desired cost function is an optimum floorplan.

Cost Functions:

- There are no universally accepted criteria for measuring the quality of floorplans. Possible criteria can be,

(1) minimize area,

(2) minimize wirelength,

(3) maximize routability,

(4) minimize delays, or

(5) a combination of some of above criteria.

Floorplanning Approaches

- Constructive: Start from a seed module; then other modules are selected one (or a group) at a time and added to the partial floorplan; continue until all modules have been selected.
- Examples: Cluster growth, force-directed, partitioning and slicing, connectivity clustering, mathematical programming, and rectangular dualization.
- Iterative: Start from an initial floorplan; this floorplan undergoes a series of perturbations until a feasible floorplan is obtained or no more improvements can be achieved.
- Examples: Simulated annealing, simulated evolution, force directed interchange/relaxation, and genetic algorithm.

Floorplanning Approaches

- Knowledge-based: A knowledge expert system is implemented which consists of three basic elements:
  - A knowledge base that contains data describing the floorplan problem and its current state,
  - Rules stating how to manipulate the data in the knowledge base in order to progress toward a solution, and
  - An inference engine controlling the application of the rules to the knowledge base.

Cluster Growth Approach

- The floorplan is constructed in a greedy fashion one module at a time until each module is assigned to a location of the floorplan.
- A seed module is selected and placed into a corner of the floorplan (lower left corner).
- The remaining modules are selected one at a time and added to the partial floorplan, while trying to grow evenly on upper, diagonal, and right sides simultaneously and maintaining any stated aspect ratio constraint on the chip.

Cluster growth

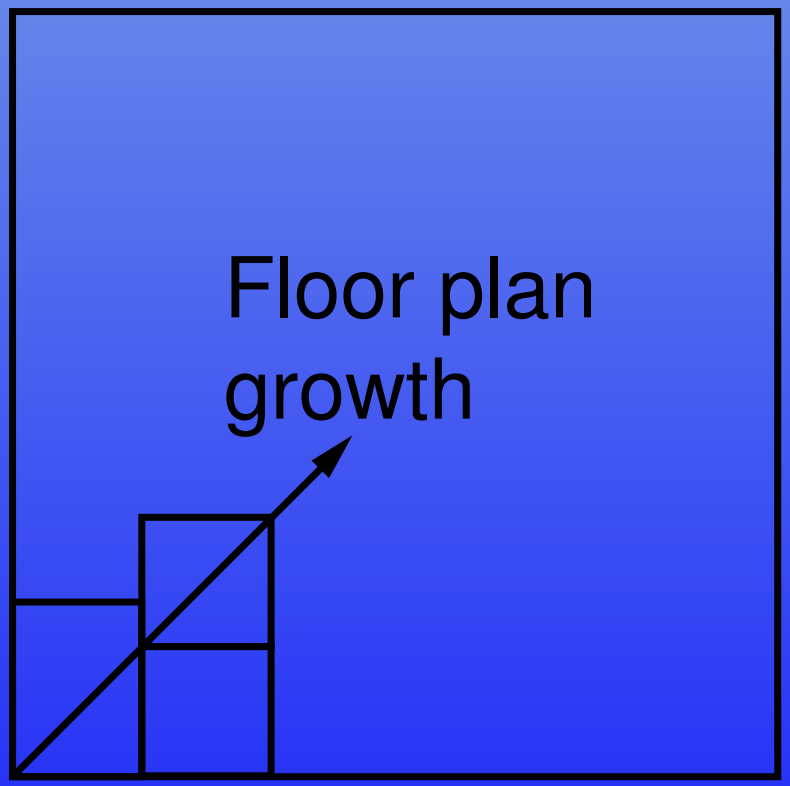


Figure 2: Cluster growth floorplanning

Cluster growth - contd

- To determine the order in which modules should be selected, the modules are initially organized into a linear order.
- Linear ordering algorithms order the given module netlist into a linear list so as to minimize the number of nets that will be cut by any vertical line drawn between any consecutive modules in the linear order (Gota 1977, and Kang 1983).

Linear Ordering

Algorithm Linear_Ordering
S : Set of all modules; Order: Sequence of ordered modules; (*initially empty*)
Begin
Seed:= Select Seed module; Order:=[Seed]; S:=$S-$ {Seed};
Repeat
ForEach module $m \in S$ **Do**
Compute the gain for selecting module m;
$gain_m$:= number of nets terminated by $m-$
number of new nets started by m;
End ForEach ;
Select the module $m*$ with maximum gain;
If there is a tie **Then**
Select the module that terminates the largest number of nets;
ElseIf there is a tie **Then**
Select the module that has the largest number of continuing nets;
ElseIf there is a tie **Then**
Select the module with the least number of connections;
Else break remaining ties as desired; (*append $m*$ to the ordered sequence*)
Order:= [!Order,$m*$]; S:= $S - \{m*\}$
Until $S = \emptyset$
End.

Linear Ordering - contd

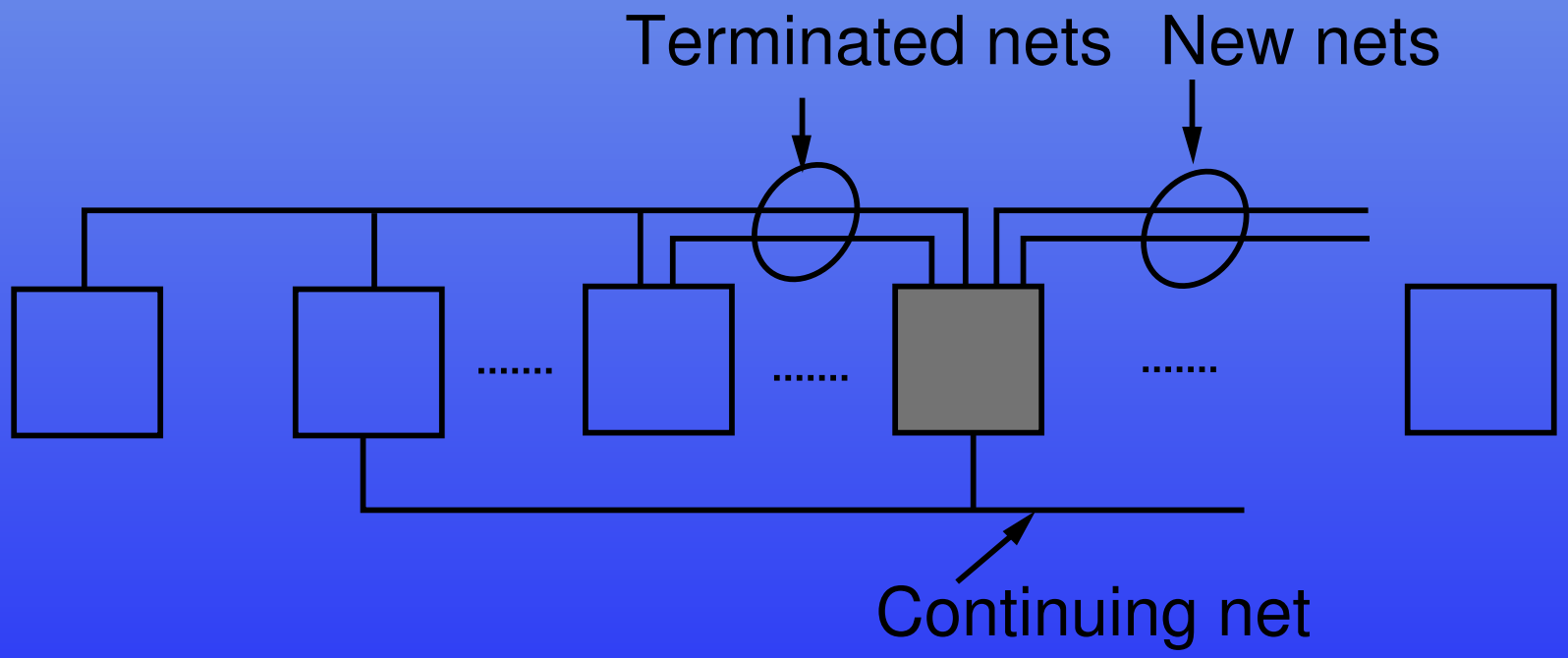


Figure 3: Classification of nets during linear ordering.

Cluster Growth - Algorithm

Algorithm Cluster_Growth
S : Set of all modules;
Begin
Order:= Linear_Ordering(S);
Repeat
nextmodule:= b where Order=[b, !rest]
Order:= rest;
Select a location for b that will result in minimum increase in cost function;
(*cost may be function of the contour of the partial floorplan, size and shape of b, and wiring length*).
Until Order = $\emptyset$
End.

Cluster Growth - contd

- Another approach may consist of folding the linear order in a row structure while satisfying also shape constraints on the chip as well as on all the modules.
- It is usually the case that floorplanning is followed by a global routing step. Global routing is executed in order to appraise the net routes, therefore leading to a fairly accurate measure of the required routing space.
- A common approach to global routing is to build a global routing graph which models the regions of the floorplan, as well as relationships (the routing regions also called routing channels) between these regions. This graph is also called the channel connectivity graph.

Cluster Growth - contd

The vertices in the channel connectivity graph are usually assigned weights specifying the cost of assigning a net to the channels.

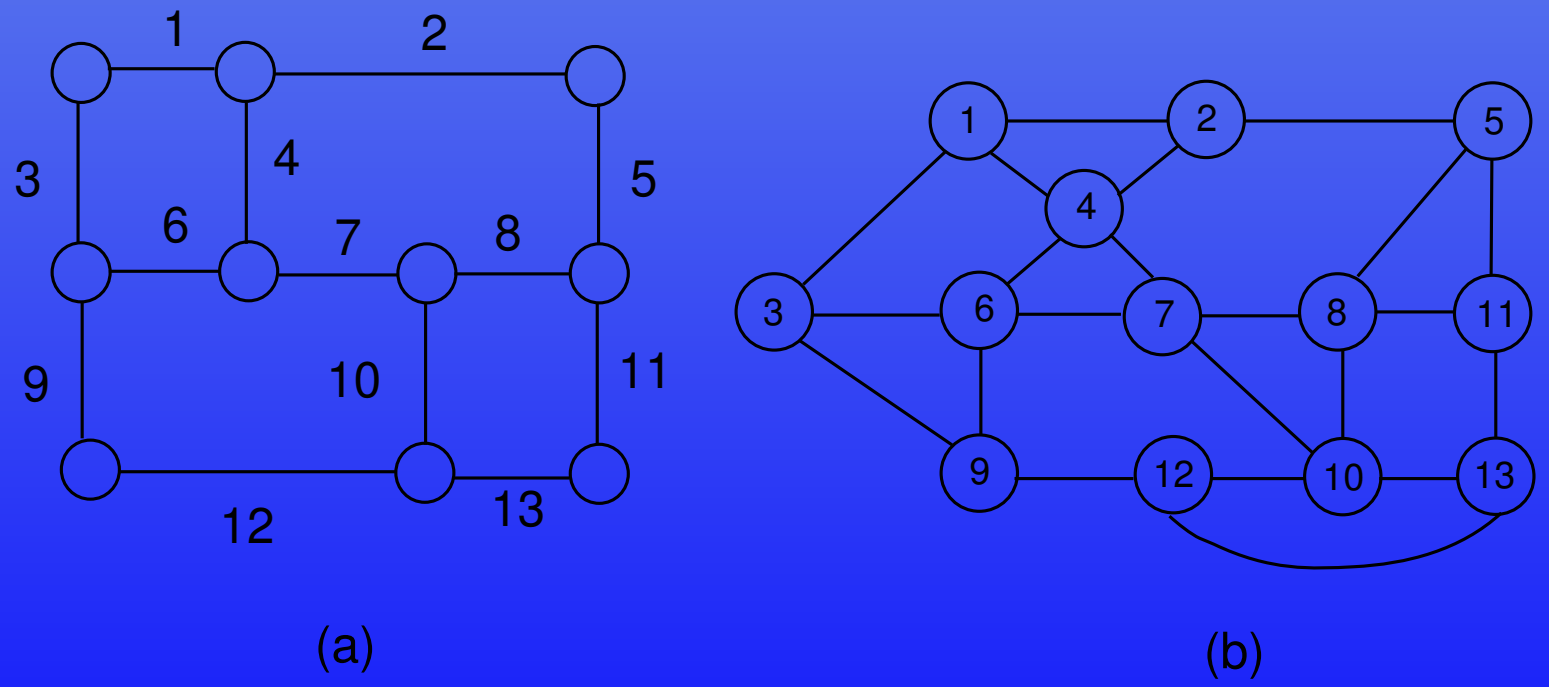


Figure 4: (a) Channel intersection graph. (b) Corresponding channel connectivity graph.

Global Routing

- Global routing consists of performing a routing plan for each net, thus, determining for each net the set of channels through which the net will be routed.
- This amounts to performing the following tasks for each net:

  (1) mark the channel vertices in which the particular net has pins;

  (2) find a minimum cost Steiner tree connecting the marked vertices.

Simulated Annealing

- First application of simulated annealing to placement reported by Jepsen and Gelatt (1983).
- Since then, there have been several successful applications of simulated annealing to the floorplanning problem.
- In SA, first an initial solution is selected; then a controlled walk through the search space is performed until no sizeable improvement can be made or we run out of time.
- Two approaches can be used to perform floorplanning by simulated annealing: (1) direct approach and (2) indirect approach.

Simulated Annealing - contd

- The direct approach manipulates actual physical coordinates, sizes, and shapes of modules.
- The indirect approach works on an abstract representation of the floorplan, consisting usually of a graph representation or a floorplan tree.
- Then a subsequent mapping process is required to generate a real floorplan from its corresponding abstract representation.

SA Algorithm

Algorithm Simulated_annealing($S_0, T_0, \alpha, \beta, M, Maxtime$);
(*S_0 is the initial solution *)
(*T_0 is the initial temperature *)
(*$alpha$ is the cooling rate *)
(*$beta$ a constant *)
begin
$T = T_0$;
$S = S_0$;
$Time = 0$;
repeat
Call $Metropolis(S, T, M)$;
$Time = Time + M$;
$T = \alpha \times T$;
$M = \beta \times M$
until $(Time \geq MaxTime)$;
Output Best solution found
End. (*of Simulated_annealing*)

SA Algorithm - contd

Algorithm Metropolis(S, T, M);
begin
repeat
$NewS$=$neighbor(S)$;
Δh=$(Cost(NewS) - Cost(S))$;
if $((\Delta h < 0)$ **or**
$(random < e^{-\Delta h/T}))$
then $S = NewS$;
{accept the solution}
$M = M - 1$
until $(M = 0)$
End. (**of Metropolis**).

Simulated Annealing - contd

To apply the simulated annealing technique we need to be able to:

(1) Generate an initial solution,

(2) Disturb a feasible solution to create another feasible solution,

(3) Evaluate the objective function for these solutions.

Terminology

Definition (Wong and Liu, DAC, 1986)

- An expression $E = e_1 e_2 \cdots e_{2n-1}$, where each $e_i \in \{1, 2, \cdots, n, H, V\}$, $1 \leq i \leq 2n-1$, is a Polish expression of length $2n-1$ if and only if:

  (1) every operand j, $1 \leq j \leq n$, appears exactly once in the expression;

  (2) the expression E has the balloting property, i.e., for every sub-expression $E_i = e_1 \cdots e_i$, $1 \leq i \leq 2n-1$, the number of operands is greater than the number of operators.

Solution Representation

- The hierarchical structure of a slicing floorplan can be represented by a binary tree with n leaves representing the n basic rectangles, and $n-1$ nodes representing the dissection operators (H for horizontal and V for vertical dissection).
- A postorder traversal of a slicing tree will produce a Polish expression with operators H and V, and with operands the basic rectangles $1, 2, \cdots, n$.
- In a postorder traversal of a binary tree, the tree is traversed by visiting at each node the left subtree, the right subtree, and then the node itself.

Solution Representation

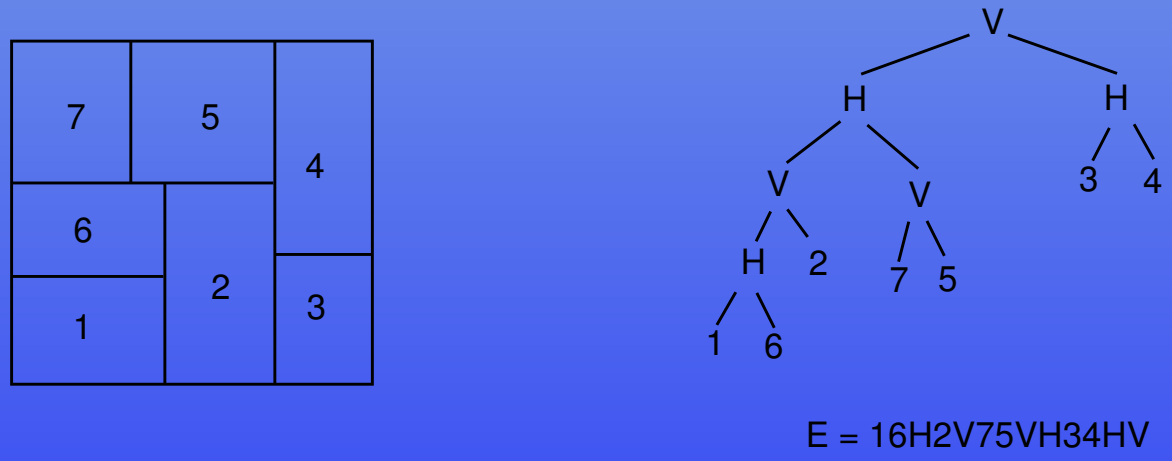


Figure 5: (a) A rectangular dissection. (b) Its corresponding slicing tree.

Operators H and V carry the following meanings:
ijH means rectangle j on-top-of rectangle i;
ijV means rectangle i to-the-left-of rectangle j.

Terminology

Definition

- A Polish expression $E = e_1 e_2 ... e_{2n-1}$ is called normalized if and only if E has no consecutive H's or V's.
- For example, E_1= $12H43VH$ is a normalized Polish expression, while E_2= $12V43HH$ is not normalized.
- The classification of Polish expressions into normalized versus non-normalized Polish expressions is for the purpose of removing redundant solutions from the solution space.

Terminology

- There is a one-to-one correspondence between the set of Polish expressions of length $2n-1$ and the set of slicing trees with n leaves.
- However, in general, there may be several Polish expressions that correspond to the same slicing floorplan.
- This is an undesirable property because:

  (1) the search space will be enlarged with several duplicate solutions, since several Polish expressions may represent the same slicing floorplan;

  (2) the number of Polish expressions corresponding to a given slicing floorplan can vary from structure to structure; this will bias the search for floorplans with a larger number of corresponding slicing trees.

Floorplan example

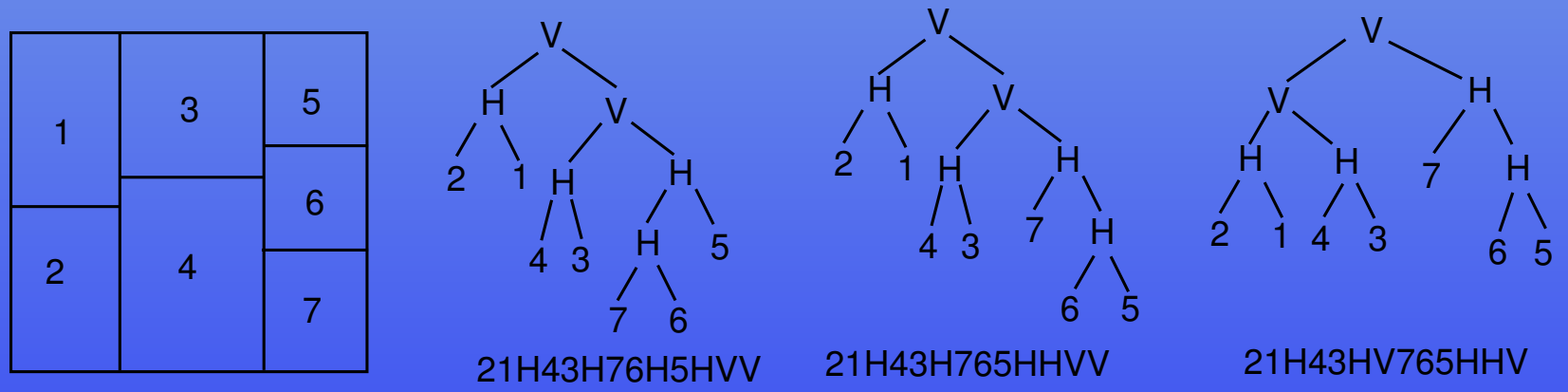


Figure 6: A rectangular dissection with several slicing tree representations.

Terminology

Lemma 1:

- There is a one-to-one correspondence between the set of skewed slicing trees with n leaves and the set of normalized polish expressions of length $2n - 1$.

Lemma 2:

- There is a one-to-one correspondence between the set of normalized Polish expressionsof length $2n - 1$ and the set of slicing structures with n basic rectangles.
- Lemma 2 says, that given a normalized Polish expression, we can construct a unique rectangular slicing structure (i.e., a floorplan).

Terminology

Definition

- A sequence $C = op_1 op_2 ... op_k$ of k *operators* is called a chain of length k if and only if $op_i \neq op_{i+1}$, $1 \leq i \leq k-1$.
- Let $E = e_1 e_2 ... e_{2n-1}$ be a normalized Polish expression that can be expressed as $E = P_1 C_1 P_2 C_2 ... P_n C_n$, where the C_i's are chains (possibly of zero length), and $P_1 P_2 \cdots P_n$ is a permutation of the operands $1, 2, \cdots, n$.
- Two operands in E are called adjacent if and only if they are consecutive elements in $P_1, P_2, ..., P_n$.
- An operand and an operator are adjacent if and only if they are consecutive elements in $e_1, e_2 ... e_{2n-1}$.

Terminology

Example

- E= $123VH54HV$ = $P_1P_2P_3C_3P_4P_5C_5$
- $C_1 = C_2 = C_4$ are empty chains $C_3 = VH$, $C_5 = HV$
- $P_1 = 1$, $P_2 = 2$, $P_3 = 3$, $P_4 = 5$, $P_5 = 4$
- 1 and 2 are adjacent operands;
- 3 and 5 are also adjacent operands;
- 3 and V are adjacent operand and operator.

Solution Perturbation

The Move Set:

- Floorplan solutions are represented by normalized Polish expressions.
- Three types of moves are suggested to perturb a given normalized Polish expression:

M_1: swap two adjacent operands;

M_2: complement some chain of nonzero length; (where $\overline{V} = H$ and $\overline{H} = V$);

M_3: swap two adjacent operand and operator.

Solution Perturbation - contd

- Care must be taken to make sure that neighbors of normalized expressions are also normalized.
- The first two moves always produce a normalized Polish expression from a normalized expression.
- However, the third move may at times result in a non-normalized Polish expression. Therefore, whenever an M_3 move is made, we must check that the resulting expression is a normalized Polish expression, i.e.,
  (a) it does not contain two identical consecutive operators,
  (b) it does not violate the balloting property.
- In case an M_3 move violates either (a) or (b), the move is rejected.

Terminology

- Checking that the new expression E does not contain two identical consecutive operators is straightforward and achievable in $O(1)$ time.
- The following quick test is sufficient to know whether an M_3 move will violate the balloting property or not.

Lemma 3:

- Let N_k be the number of operators in the Polish expression $E = e_1, e_2, ..., e_k, 1 \leq k \leq 2n - 1$.
- Assume that the M_3 move swaps the operand e_i with the operator e_{i+1}, $1 \leq i \leq k - 1$. Then, the swap will not violate the balloting property if and only if $2N_{i+1} < i$.

Solution Evaluation

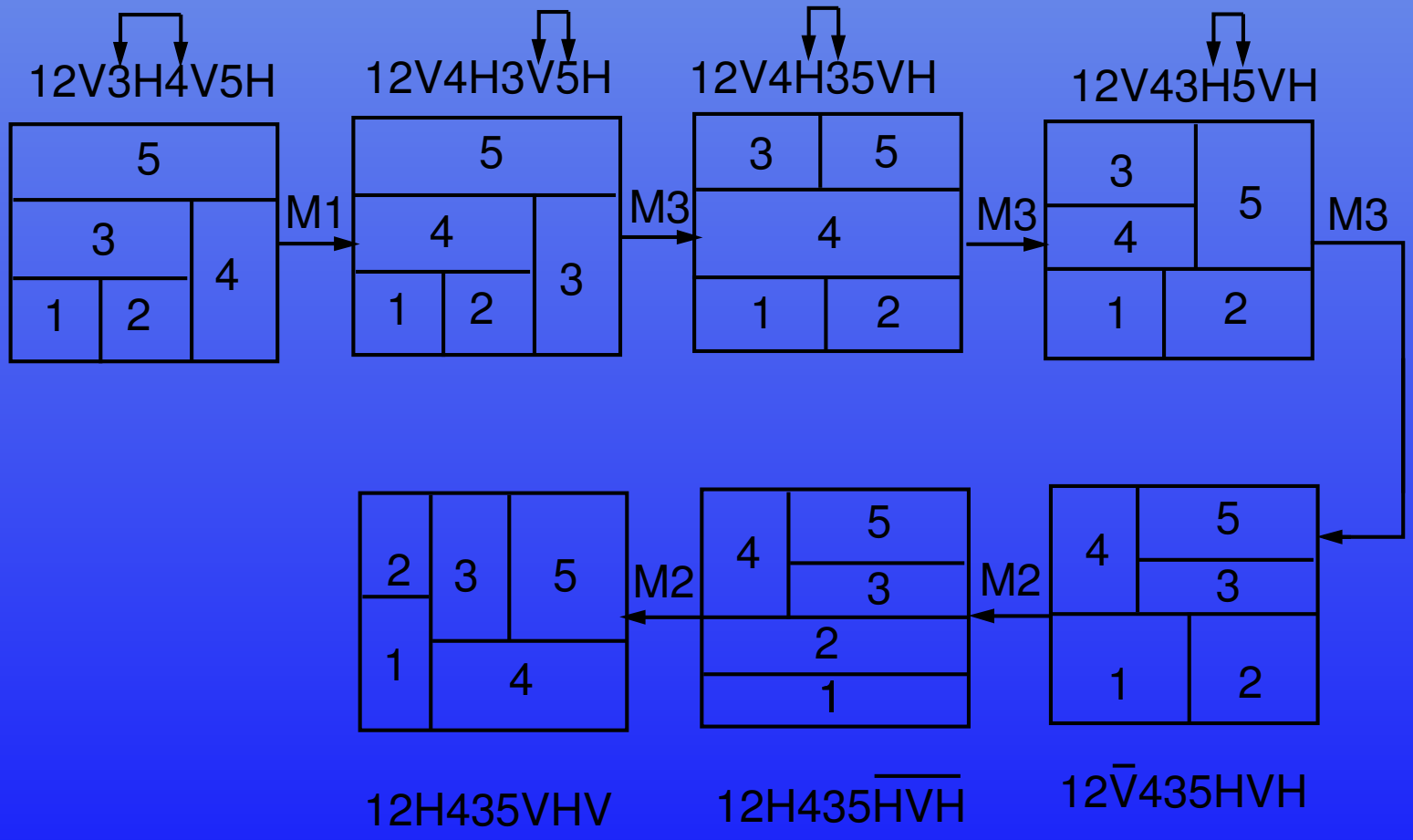


Figure 7: An example of walk through a floorplan solution space with 5 modules

Solution Evaluation

- Usually the principal goal is to achieve the floorplan with minimum area and overall interconnection length.

$$Cost(F) = \alpha A + \lambda W$$

where A is the area of the smallest rectangle enveloping the n basic rectangles, and W is a measure of the overall wiring length.

- A possible estimate of W may be defined as follows,

$$W = \sum_{ij} c_{ij} \cdot d_{ij}$$

where c_{ij} is equal to the number of connections between blocks i and j, and d_{ij} is the center-to-center distance between basic rectangles i and j.

Area Evaluation

Definition

- Let Γ be a continuous curve on the plane. Γ is called a bounding curve if it satisfies the following conditions:

  (1) it is decreasing, i.e., for any two points (x_1, y_1) and (x_2, y_2) on Γ, if $x_1 \leq x_2$ then $y_2 \leq y_1$;

  (2) Γ lies completely in the first quadrant, i.e., $\forall (x, y) \in \Gamma$, $x > 0$ and $y > 0$; and

  (3) it partitions the first quadrant into two connected regions. The connected region containing all the points (x, x) for very large x is called the bounded area with respect to the bounding curve Γ.

Area Evaluation

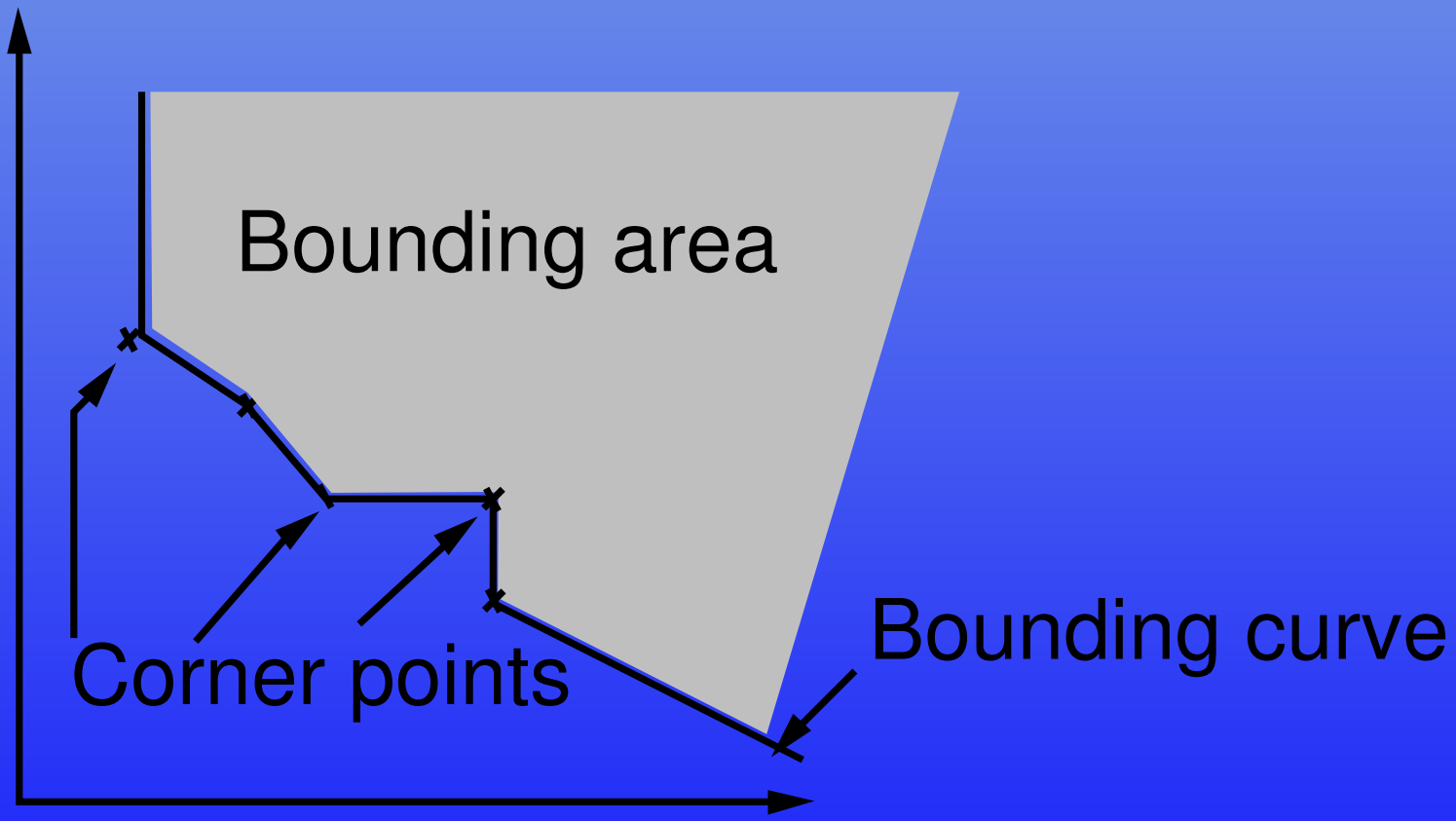


Figure 8: A piecewise linear bounding curve.

Definitions

- Let Γ and Λ be two bounding curves. Two arithmetic operations on bounding curves are defined as follows:

  (1) the bounding curve corresponding to $\Gamma H \Lambda$ is obtained by summing the two curves along the y-axis, i.e., $\Gamma H \Lambda = \{(u, v+w) | (u, v) \in \Gamma$ and $(u, w) \in \Lambda\}$;

  (2) the bounding curve corresponding to $\Gamma V \Lambda$ is obtained by summing the two curves along the x-axis, i.e., $\Gamma V \Lambda = \{(u+v, w) | (u, w) \in \Gamma$ and $(v, w) \in \Lambda\}$.

- A piecewise linear bounding curve is completely characterized by an ordered list of its corner points.
- To add two piecewise linear curves along either direction, it is sufficient to sum up the two curves at their corner points.

Definitions

- For the floorplanning problem, each module i, $1 \leq i \leq n$ is constrained as follows,

  (1) height = h_i, width = w_i, and area $A_i = w_i h_i$;

  (2) $r_i \leq \frac{h_i}{w_i} \leq s_i$, if module i has fixed orientation;

  (3) $r_i \leq \frac{h_i}{w_i} \leq s_i$ or $\frac{1}{s_i} \leq \frac{h_i}{w_i} \leq \frac{1}{r_i}$, if module i has free orientation;

  (4) $r_i = s_i$, if module i is rigid, and $r_i \neq s_i$ if module i is flexible.

- Each basic rectangle i, $1 \leq i \leq n$ must be large enough to accommodate module i. Hence, $x_i \geq w_i$ and $y_i \geq h_i$, where x_i and y_i are the width and height of basic rectangle i.

Bounding curves

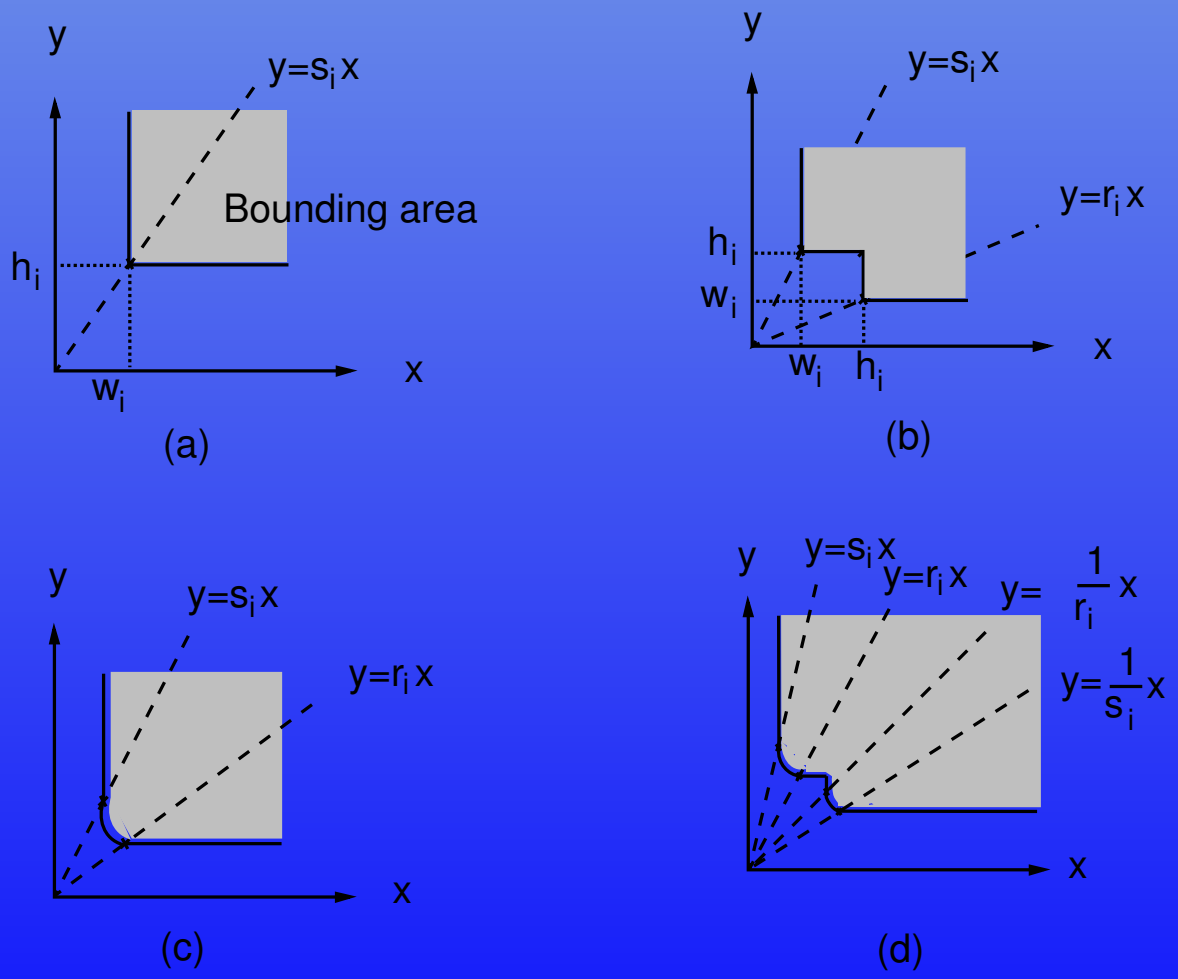


Figure 9: Bounding curves for various classes of modules.

Definitions

- Let T_E be the floorplan tree corresponding to the normalized Polish expression E.
- Let R_E be the rectangular slicing structure corresponding to T_E, and D_E be the set of all possible dimensions of R_E.
- The set of points in D_E constitute a bounding curve Γ_E corresponding to the rectangular dissection R_E.

Computation of Γ_E:

- Every leaf node i, $1 \leq i \leq n$ of T_E has associated with it a bounding curve Γ_i consistent with the shape, size, flexibility, and orientation of the corresponding module i.
- The slicing tree is traversed from the leaves upwards, towards the root, computing on the way the bounding curves Γ_v corresponding to each internal node v.
- $\Gamma_v = \Gamma_l H \Gamma_r$ or $\Gamma_v = \Gamma_l V \Gamma_r$, where l and r are left & right sons of v.

Definitions

- For efficiency reasons, all bounding curves are approximated by piecewise staircase linear curves. The accuracy of area estimation is a function of this staircase approximation. Once all Γ's are computed, the bounding curve Γ_E of R_E is as follows:

  (1) let (a_1, b_1) and (a_{k+1}, b_{k+1}) be the points of intersection between Γ_E and the lines $y = px$ and $y = qx$ respectively (consequence of the shape constraint on R_E, which states that $p \leq \frac{H}{W} \leq q$);

  (2) let $(a_1, b_1), (a_2, b_2), ..., (a_k, b_k)$ be all the corner points of the bounding curve Γ_E which lie between the lines $y = px$ and $y = qx$.

Definitions

- The dimensions of a minimum area realization of the floorplan tree T_E are given by the corner point (a_i, b_i) such that $a_i b_i = \min_j(a_j b_j)$.
- Hence, the minimum area enveloping rectangle has width a_i, height b_i, and area $A = a_i b_i$.
- The final step is to trace back the shapes and orientations of the rectangles (composite or basic) that were selected in the upward traversal of the tree.

Definitions

- When dealing with rigid blocks, we might have width or height mismatch. In that case, the summation of the corresponding two bounding curves along the x or y directions should be changed to the following:
- Let Γ and Λ be two bounding curves.

(1) The bounding curve corresponding to $\Gamma H \Lambda$ is obtained by summing the two curves along the y-axis, i.e., $\Gamma H \Lambda = \{(u, v+w) | (u_1, v)\} \in \Gamma$ and $(u_2, w) \in \Lambda$ and $\{u = \max(u_1, u_2)\}$;

(2) the bounding curve corresponding to $\Gamma V \Lambda$ is obtained by summing the two curves along the x-axis, i.e., $\Gamma V \Lambda = \{(u+v, w) | (u, w_1)\} \in \Gamma$ and $(v, w_2) \in \Lambda$ and $\{w = \max(w_1, w_2)\}$;

Definitions

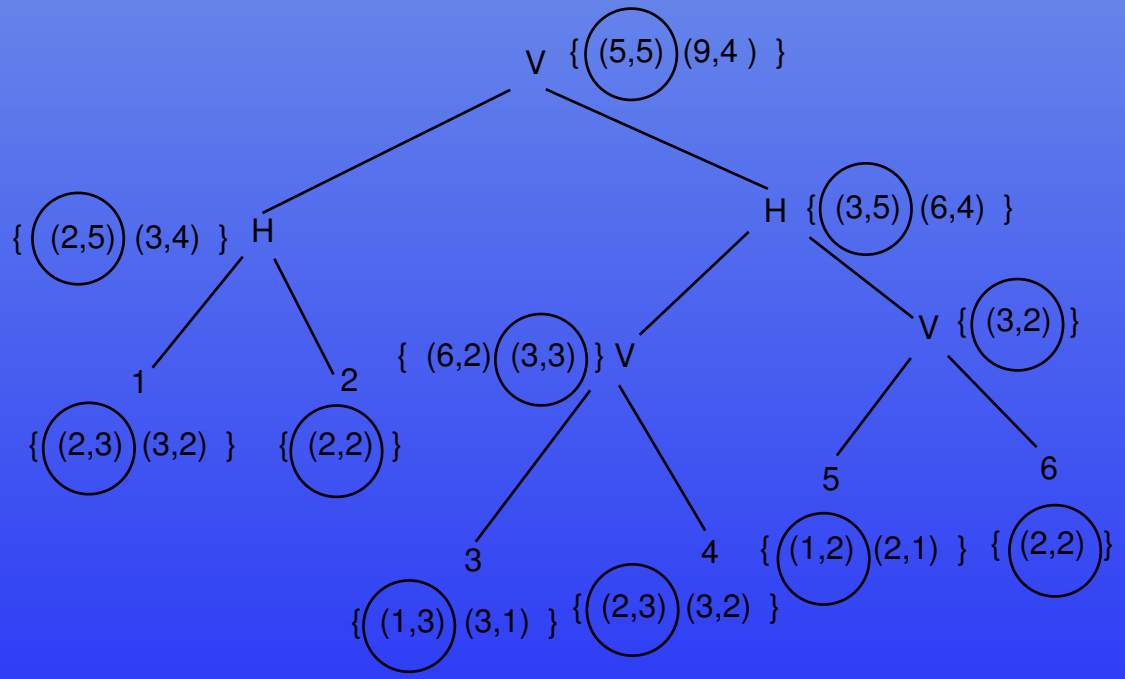
- Let (x_1, y_1) and (x_2, y_2) be two possible implementations of a given rectangle. (x_2, y_2) is a redundant implementation of (x_1, y_1) if and only if $x_2 \geq x_1$ and $y_2 > y_1$, or $x_2 > x_1$ and $y_2 \geq y_1$.
- Redundant implementations should be identified during the summation of the bounding curves and eliminated, since a minimum area enveloping rectangle cannot possibly include such redundant rectangles.
- Only corner points are non-redundant implementations, therefore we should only consider corner point implementations.

Example

Module	Width	Height
1	2	3
2	2	2
3	1	3
4	2	3
5	1	2
6	2	2

- For simplicity reason, we will assume that all modules are rigid and can be rotated 90° with respect to their original orientation.
- $E = 21H34V56VHV$.

Example



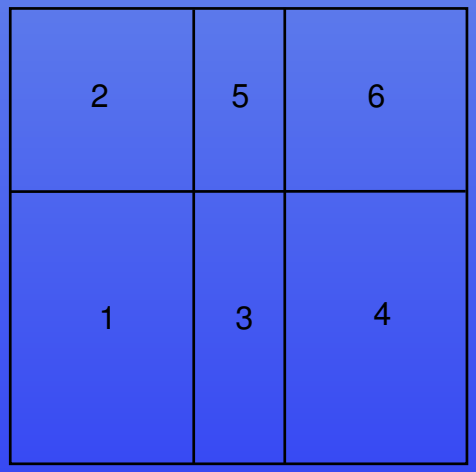


Figure 10: Example of floorplan area computation.

The Algorithm

- When using the simulated annealing technique, there are several important decisions that must be made, which consist of the following:

  (1) a choice of the initial solution;

  (2) a choice of a cooling schedule, that is, (a) choice of the initial temperature, (b) how long before we reduce the temperature, and, (c) the temperature reduction rate;

  (3) a perturbation function;

  (4) a termination condition of the algorithm.

Algorithm

Algorithm Simulated_Annealing_Floorplanning
$Best = E = E_0$ = 12V3V4V....nV;
$T_0 = \frac{\Delta_{avg}}{\ln(P)}$
uphill = 0; MT = 0; M = 0;
Repeat
MT = uphill = Reject = 0;
Repeat
Select_Move(M);
Case M of
M_1 : Select two adjacent operands e_i and e_j; $NewE$ = Swap(E, e_i, e_j);
M_2 : Select a nonzero length chain C of operators; $NewE$ = Complement(E, C);
M_3 : Done = False
While NOT(Done) **Do**
Select two adjacent opd e_i and opr e_{i+1};
If ($e_{i-1} \neq e_{i+1}$) and ($2N_{i+1} < i$) **Then** Done = TRUE;
EndWhile;
$NewE$ = Swap(E, e_i, e_{i+1});
EndCase

Algorithm - contd

MT = MT + 1;
$\Delta Cost$ = $Cost(NewE) - Cost(E)$;
If ($\Delta Cost < 0$) OR (RANDOM< $e^{-\Delta Cost/T}$) **Then**
  Begin
    If ($\Delta Cost > 0$) **Then** uphill = uphill+1;
    E = $NewE$; (*Accept $NewE$*);
    If $Cost(E) < Cost(Best)$ **Then** Best = E;
  End
  Else Reject = Reject + 1; (*reject the move*)
  EndIf
Until (uphill $> N$) OR ($MT > 2N$)
T = λT
Until (Reject/MT < .05) OR ($T \leq \epsilon$) OR Out_of_Time;
End.

Initial temperature T_0:

- A sequence of random moves are performed and the average cost change for all uphill moves Δ_{avg} is computed.
- Then T_0 is chosen such that $e^{\frac{\Delta_{avg}}{T_0}} = P$, where P is the initial probability of accepting uphill moves. P is initially set very close to 1.

Perturbation function:

- First, the type of move is randomly selected.
- Then a pair of adjacent elements are chosen.
- In case the move is of type M_3, we should make sure that the perturbation does not lead to a non-normalized Polish expression. In case it does, another pair of elements is selected.
- This is repeated until the swapping of the two elements does not violate the normality property of the Polish expression.
- Each generated normalized Polish expression is evaluated with respect to its cost (i.e., area of enveloping rectangle and overall wiring length).

Perturbation function:

- If the new expression has an improved cost, then it is accepted. Otherwise, if it has a higher cost (worse solution) then it is accepted with a probability that is a decreasing function of the annealing temperature.
- At each temperature, a number of trials are attempted until either we make N uphill moves (bad moves), or the total number of moves exceeds $2N$, where N is an increasing function of n, the number of basic rectangles.
- When we exit from the inner **Repeat** loop, the temperature is reduced by a fixed ratio λ. A recommended value for λ is $\lambda = 0.85$.
- The algorithm terminates when the number of good moves becomes too small ($\leq 5\%$ of all moves made), or when the temperature becomes too low.

Mathematical Formulation

(Shragowitz, Sutanthavibul, and Rosen 1990)

- The constraints specifying a feasible floorplan are described by a set of mathematical equations, and solved using mathematical programming techniques.
- Two major difficulties:
  (1) Nonlinearity of the floorplanning problem.
  (2) Size of the problem.

Notation & Problem Definition

$S = \{1, 2, \cdots, n\}$ set of n modules;
S_1 subset of modules with fixed orientation;
S_2 subset of modules with free orientation;

- Each module i has width w_i and height h_i; $x_i$and y_i are coordinates of lower left corner of module i, $1 \leq i \leq n$.
- Then, for two modules i and j not to overlap, $1 \leq i < j \leq n$, at least one of the following linear constraints must be satisfied:

if i is to the left of j:	$x_i + w_i \leq x_j$
if i is below j:	$y_i + h_i \leq y_j$
if i is to the right of j:	$x_i - w_j \geq x_j$
if i is above j:	$y_i - h_j \geq y_j$

Notation & Problem Definition

- For two modules i and j not to overlap in either the x-direction or the y-direction, it is sufficient to enforce one and only one equation.
- In order to state that in equations form, two additional 0-1 integer variables, x_{ij} and y_{ij}, are introduced for each (i, j) pair. These 0-1 variables have the following interpretation:

  $x_{ij} = 0$ and $y_{ij} = 0 \leftarrow i$ left of j -enforced

  $x_{ij} = 0$ and $y_{ij} = 1 \leftarrow i$ below j -enforced

  $x_{ij} = 1$ and $y_{ij} = 0 \leftarrow i$ right of j -enforced

  $x_{ij} = 1$ and $y_{ij} = 1 \leftarrow i$ above j -enforced

Notation & Problem Definition

- Let W and H be upper bounds on the floorplan width and height. Hence, $|x_i - x_j| \leq W$ and $|y_i - y_j| \leq H$
- If W and H are not given, then possible estimates of these quantities could be $W = \sum_i w_i$ and $H = \sum_i h_i$.
- Therefore, to enforce that no two modules overlap,

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

Linear Prog. Formulation:

Assumption: one dimension of the chip (W), is fixed.

- CASE 1: All modules are rigid and have fixed orientation.
  Constraints:
  (1) no two modules overlap
  (2) each module is enclosed within the floorplan enveloping rectangle of width W and height Y, i.e., $x_i + w_i \leq W$ and $y_i + h_i \leq Y, 1 \leq i \leq n$;
  (3) all module coordinates are positive, $x_i \geq 0$ and $y_i \geq 0, 1 \leq i \leq n$.
  Objective:
- Since the width W is fixed, a possible objective to minimize would be Y, the height of the floorplan.

Linear Prog. Formulation:

To summarize, we end up with the following 0-1 integer linear program:

$$
\begin{cases}
Y \leftarrow minimize & \\
\text{Subject to :} & \\
x_i + w_i \leq W, & 1 \leq i \leq n \\
y_i + h_i \leq Y, & 1 \leq i \leq n \\
x_i + w_i \leq x_j + W(x_{ij} + y_{ij}), & 1 \leq i < j \leq n \\
x_i - w_j \geq x_j - W(1 - x_{ij} + y_{ij}), & 1 \leq i < j \leq n \\
y_i + h_i \leq y_j + H(1 + x_{ij} - y_{ij}), & 1 \leq i < j \leq n \\
y_i - h_j \leq y_j - H(2 - x_{ij} - y_{ij}), & 1 \leq i < j \leq n \\
x_i \geq 0, y_i \geq 0, & 1 \leq i \leq n
\end{cases}
$$

Size of the linear program:

$2 \times n$ continuous variables, $n(n-1)$ integer variables, and $2n^2$ linear constraints. For large n, this will lead to unacceptably large programs.

Linear Prog. Formulation:

- **CASE 2:** All modules rigid and rotation allowed.
  For each free-orientation module i, one 0-1 integer variable is introduced z_i.
  $z_i = 0 \rightarrow$ module i is not rotated;
  $z_i = 1 \rightarrow$ module i is rotated.

$$
\begin{cases}
Y \leftarrow minimize & \\
\text{Subject to :} & \\
x_i + z_i h_i + (1 - z_i) w_i \leq W, & 1 \leq i \leq n \\
y_i + z_i w_i + (1 - z_i) h_i \leq Y, & 1 \leq i \leq n \\
x_i + z_i h_i + (1 - z_i) w_i \leq x_j + M(x_{ij} + y_{ij}), & 1 \leq i < j \leq n \\
x_i - z_j h_j + (1 - z_j) w_j \geq x_j - M(1 - x_{ij} + y_{ij}), & 1 \leq i < j \leq n \\
y_i + z_i w_i - (1 - z_i) h_i \leq y_j + M(1 + x_{ij} - y_{ij}), & 1 \leq i < j \leq n \\
y_i - z_j w_j - (1 - z_j) h_j \leq y_j - M(2 - x_{ij} - y_{ij}), & 1 \leq i < j \leq n \\
x_i \geq 0, y_i \geq 0, & 1 \leq i \leq n
\end{cases}
$$

- M could be set equal to $\max(W, H)$ or $W + H$.
- Size of the linear program: The number of equations did not change from the first formulation. However, the number of 0-1 integer variables have increased by n, which is equal to the number of modules.

Linear Prog. Formulation:

- **CASE 3:** Some of the modules are flexible:
- Some of the modules are allowed to vary in shape as long as they keep a fixed area $A_i = w_i h_i$.
- This complicates the matter a bit as the equality $A_i = w_i h_i$ is a nonlinear relationship.
- To maintain a linear program, we must linearize this relationship.
- Let $w_{i,\max}$ and $h_{i,\max}$ be the maximum width and height of module i, $1 \leq i \leq n$.
- A possible linearization approach is to make a Taylor's series expansion of A_i about the point $w_{i,\max}$, and use the first two terms of the series as an approximation of A_i.

Linear Prog. Formulation:

- The Taylor's series expansion of a function $f(x)$ about the point x_0 is:

$$f(x) = \sum_{k=0}^{\infty} \frac{(x - x_0)^k}{k!} \times f^{(k)}(x_0)$$

- By evaluating the above Taylor's series expansion for $h_i = \frac{A_i}{w_i} = f(w_i)$ and $x_0 = w_{i,\max}$, and taking the first two terms, we get the following:

$$f(w_i) = h_i = \frac{A_i}{w_{i,\max}} + A_i \frac{(w_{i,\max} - w_i)}{w^2{}_{i,\max}} + O(w_i - w_{i,\max})$$

Linear Prog. Formulation:

- Let $h_{i,0} = \frac{A_i}{w_{i,\max}}, \Delta_i = w_{i,\max} - w_i$
- and $\lambda_i = \frac{A_i}{w^2{}_{i.\max}}$.
- If we drop the error term, then the above equation can be written as follows:

$$h_i = h_{i,o} + \lambda_i \Delta_i$$

- The linear approximation of the area of the module is illustrated in the figure.

Linear Prog. Formulation:

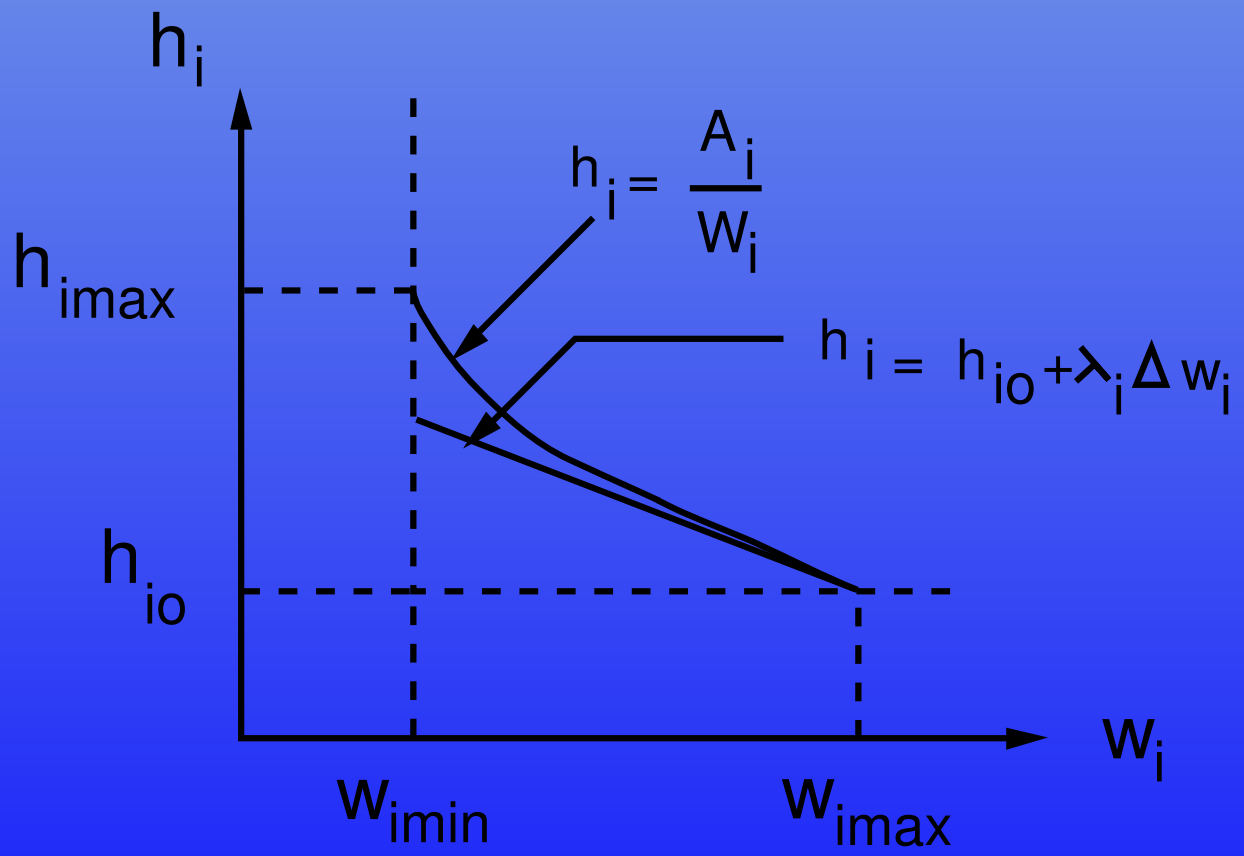


Figure 11: Linear approx. of the relationship $A_i = w_i \times h_i$

Linear Prog. Formulation:

- In above Equation, $h_{i,0}$ and λ_i are known constant parameters. Hence, this approximation will require the addition of only one continuous variable Δ_i for each module $i, 1 \leq i \leq n$.
- Equations which state the conditions of no overlapping between modules i and j must be rewritten to take into account the flexibility of some of the modules. Three cases can be distinguished:
- (1) Both modules are rigid: (same equations)

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

Linear Prog. Formulation:

- (2) Module i is flexible and module j is rigid:
- In this case, the height of module i should be replaced with its linear approximation in terms of w_i, i.e., $h_i = h_{i,o} + \lambda_i \Delta_i$.
- The constraints for no overlapping between flexible module i and rigid module j become,

$$x_i + w_{i,\max} - \Delta_i \leq x_j + W(x_{ij} + y_{ij})$$
$$y_i + h_{i,0} + \lambda_i \Delta_i \leq y_j + H(1 + x_{ij} - y_{ij})$$
$$x_i - w_j \geq x_j - W(1 - x_{ij} + y_{ij})$$
$$y_i - h_j \geq y_j - H(2 - x_{ij} - y_{ij})$$

- Note that in previous equations w_i is replaced by $w_{i,\max} - \Delta_i$.

Linear Prog. Formulation:

- (3) Both modules i and j are flexible:
- In this case, both h_i as well as h_j must be replaced with their linear approximations.
- Here again we express $w_i = w_{i,\max} - \Delta_i$. Also, w_j is expressed the same way.

$$x_i + w_{i,\max} - \Delta_i \leq x_j + W(x_{ij} + y_{ij})$$
$$y_i + h_{i,0} + \lambda_i \Delta_i \leq y_j + H(1 + x_{ij} - y_{ij})$$
$$x_i - w_{j,\max} + \Delta_j \geq x_j - W(1 - x_{ij} + y_{ij})$$
$$y_i - h_{j,0} - \lambda_j \Delta_j \geq y_j - H(2 - x_{ij} - y_{ij})$$

Successive Augmentation

Example

- The major problem is the program size.
- Smallest program (when all modules are rigid and have fixed orientation) will have $2 \times n$ continuous variables, $n(n-1)$ integer variables, and $2n^2$ linear constraints.
- For a value of $n = 100$ modules (medium size problem), the linear program will have 200 continuous variables, 990 integer variables, and 20000 linear constraints.
- Approach is realistic when the number of modules is very small (around 10).

Successive Augmentation

- A linear program is formulated using a subset S_1 of n_1 modules.
- Then a second subset S_2 of n_2 modules is selected and the corresponding linear program is formulated, with the additional constraints that the previously selected n_1 modules have fixed locations, shapes, and orientations.
- The floorplanning problem is solved when we solve problems corresponding to remaining subsets S_2, ..., S_k such that, $\sum_{i=1}^{k} n_i = n$.

Successive Augmentation

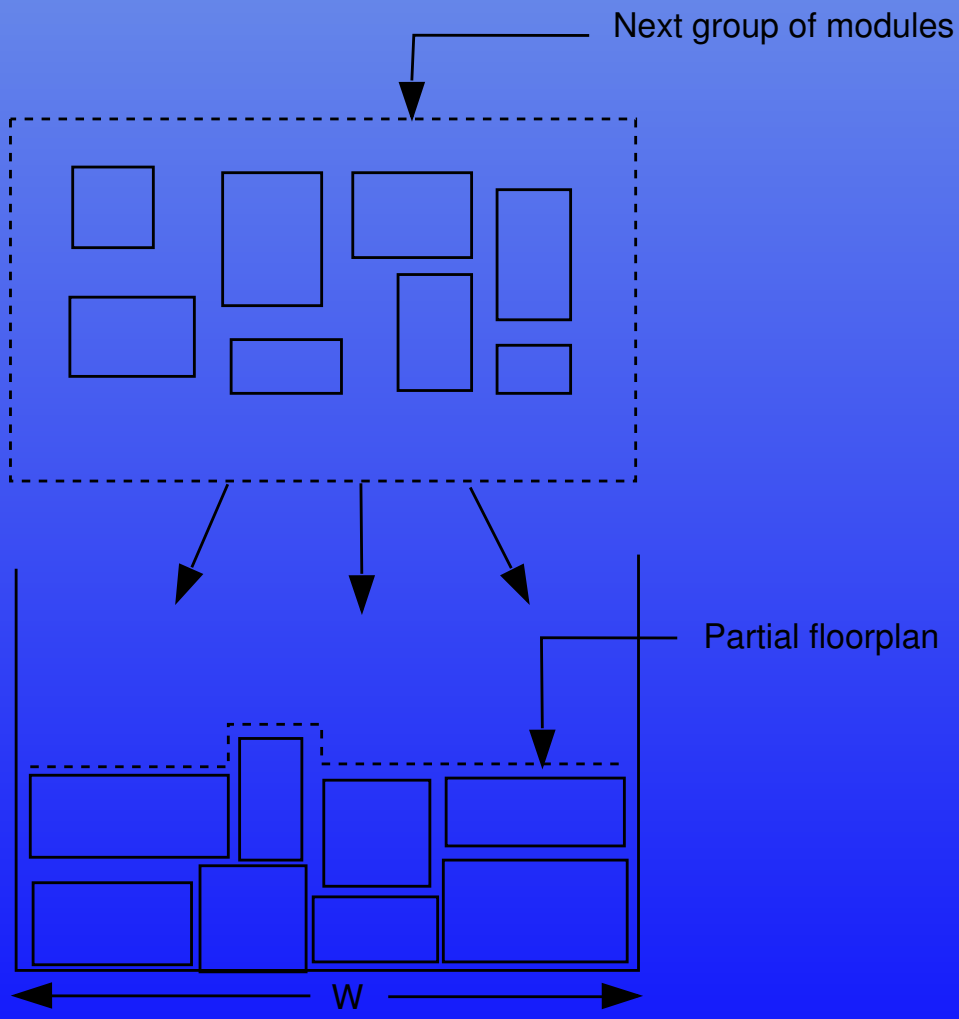


Figure 12: Successive augmentation approach

Successive Augmentation

- Successive augmentation raises two new problems:
- (1) How to select the next subgroup of modules; and
- (2) how to formulate the successive mixed integer programs while minimizing the number of required integer variables.
- For (1), a possible strategy is to use linear ordering to order the modules into a linear list based on their connectivity.
- For (2), the size of each successive mixed integer program depends on the cardinality of the next group of modules as well as the partially constructed floorplan. We must describe the partial floorplan using the smallest possible number of constraints and variables.
- The main idea consists of replacing the already placed modules by a set of covering rectangles. The number of covering rectangles is guaranteed to be always less than the number of original modules (usually much less).

Successive Augmentation

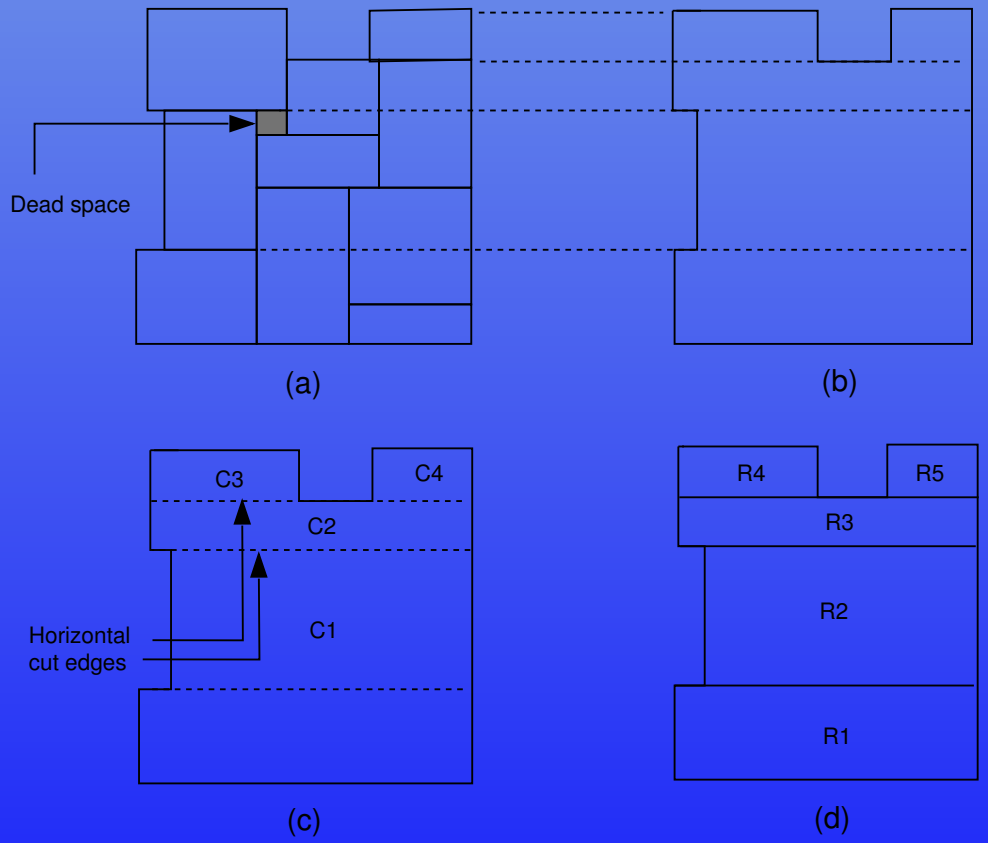


Figure 13: Steps for determining a set of covering rectangles of the partial floorplan

Algorithm

ALGORITHM Greedy_Floorplanning;
BEGIN
  Order the n modules;
  Select a first subset S_1 of n_1 modules;
  Formulate a first mixed integer linear program;
  Solve this first problem;
  $k = n_1$;
  $i = 1$;
  WHILE $k < n$ DO
    BEGIN
      Select the next subset S_i of n_i modules;
      Find a set of d_i covering rectangles of partial floorplan;
      Formulate a mixed integer program with n_i free
      modules and d_i fixed basic rectangles;
      Solve this i^{th} problem;
      $k = k + n_i$;
    END;
  Perform Global routing and adjust floorplan accordingly;
END.

Dual Graph Technique

- The graph dualization technique seeks to find a topological layout of the modules which is consistent with the overall topological relations of the blocks, as well as the sizes and shapes of these blocks.
- This approach consists of the following:
  1. The original circuit s modelled by a graph $G = (V, E)$. The set of vertices V model the modules and the set of edges E model module interconnections.
  2. The graph G is then planarized.
  3. Next, a rectangular dual of this planar graph is found, where faces of the dual correspond to modules, and edges correspond to interfaces between the modules (module adjacencies). The edges of the dual model the routing channels through which signal nets will be routed.
  4. Finally a drawing of the dual graph is sought such that the rectangular area assigned to each module is large enough to accommodate the module.

Dual Graph Technique

A final adjustment step is usually necessary to provide sufficient routing space for the interconnections.

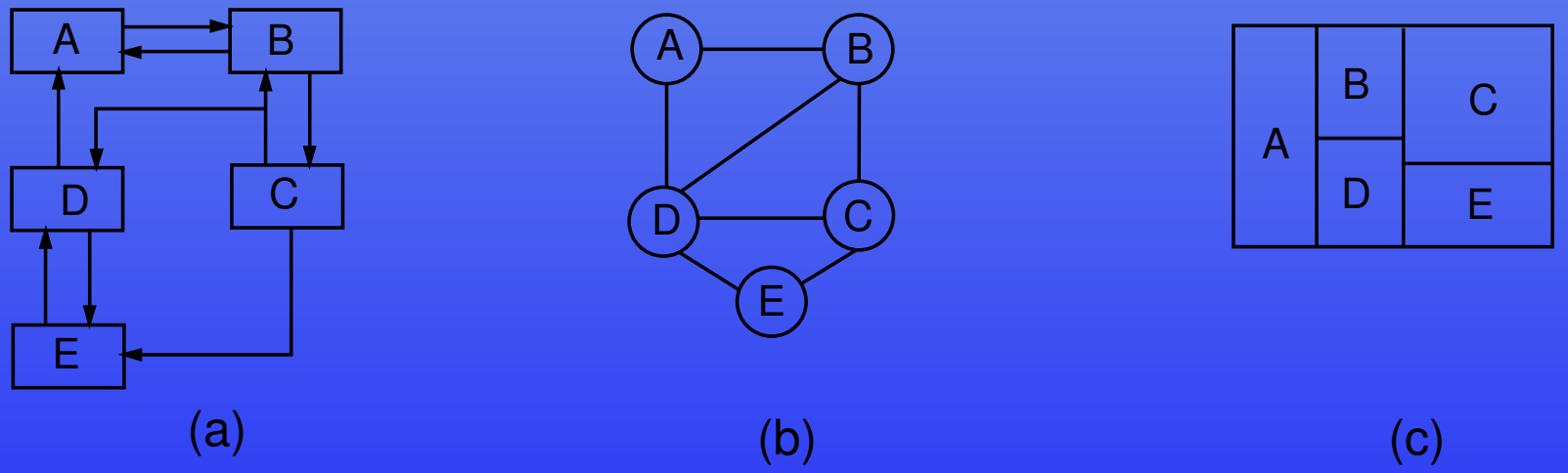


Figure 14: Steps of floorplan design by rectangular dualization. (a) A circuit. (b) Graph model for (a). (c) Rectangular dual for (b).

Terminology

- A plane graph is a graph that can be embedded in the plane with no two edges crossing each other.
- A rectangular floorplan R can be represented by a channel intersection graph. $G = (V, E)$.
- The graph G is a planar graph. Each vertex in $V(G)$ represents a line intersection point of R. There is an edge $(u, v) \in E(G)$ if and only if the intersection points modelled by u and v are adjacent. $V(G)$ and $E(G)$ are the vertex set and edge set of graph G.
- The inner faces of G are called rooms.

Dual Graph

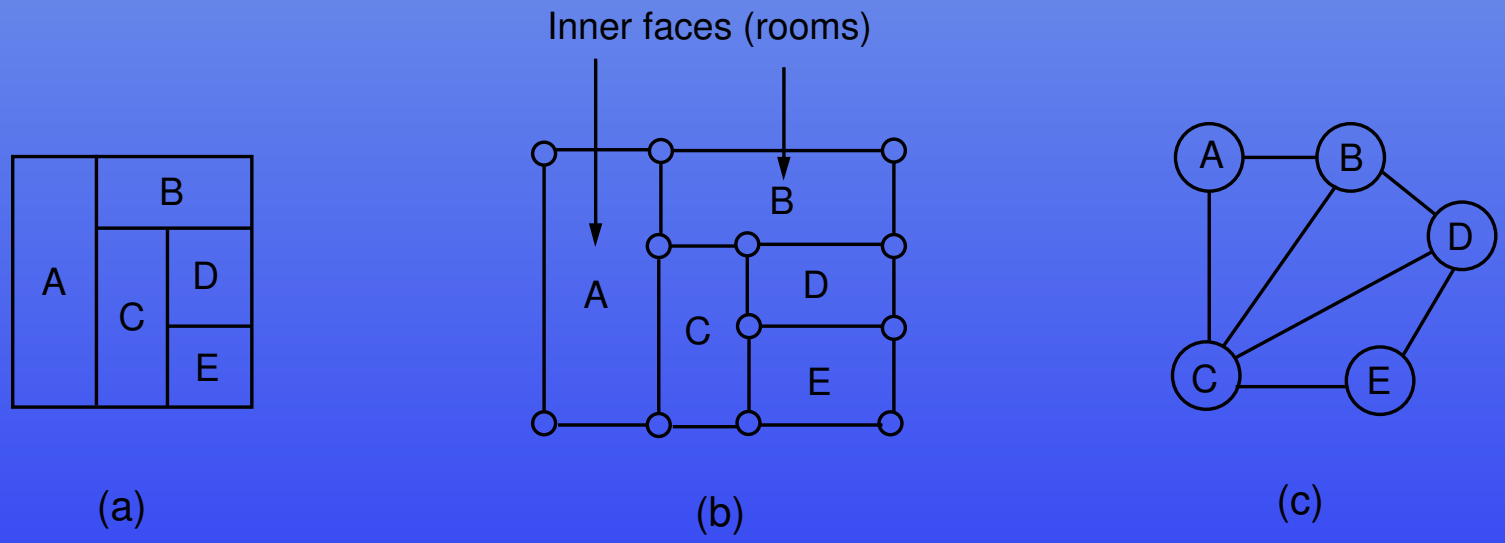


Figure 15: (a) Rectangular floorplan. (b) Its channel intersection planar graph. (c) Its inner dual graph.

Discussion

- Inclusion of performance issues in constraints and/or objective.
- Problem when neither W nor H are fixed?
- Routability.