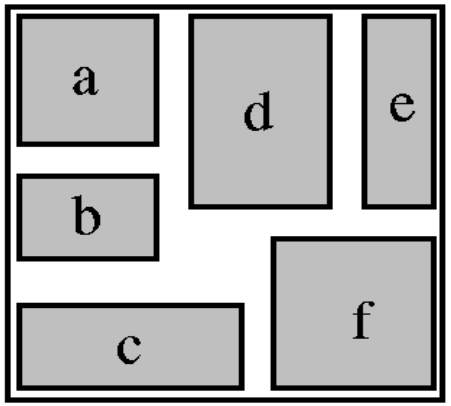
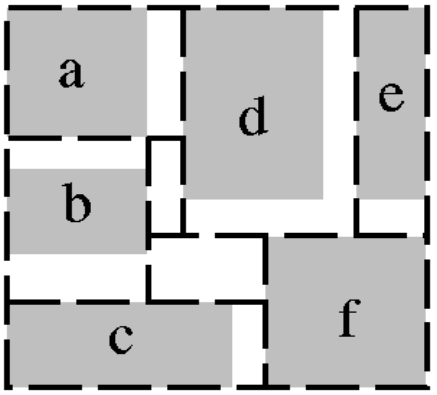
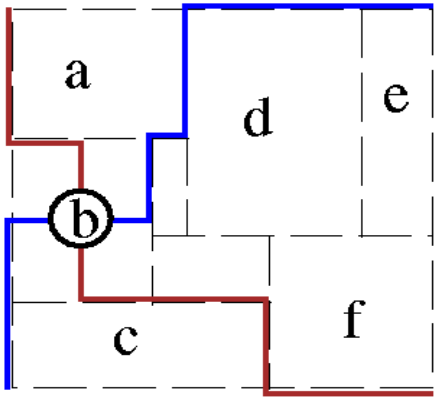
Sequence Pair (SP)

- Murata, Fujiyoshi, Nakatake and Kajitani, "Rectangle-packing-based module placement," ICCAD'95.
- Represent a *packing* by a pair of module permutations called sequence-pair (e.g., (*abdecf*, *cbfade*)).
- The set of all sequence-pairs is a P-admissible solution space whose size is $(n!)^2$.
- Search in the P-admissible solution space by simulated annealing.
  - Swap two modules in the first sequence.
  - Swap two modules in both sequences.
  - Rotate a module.





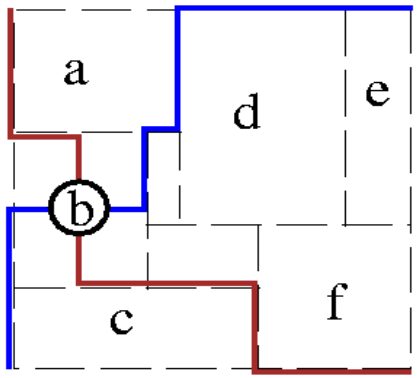
A floorplan



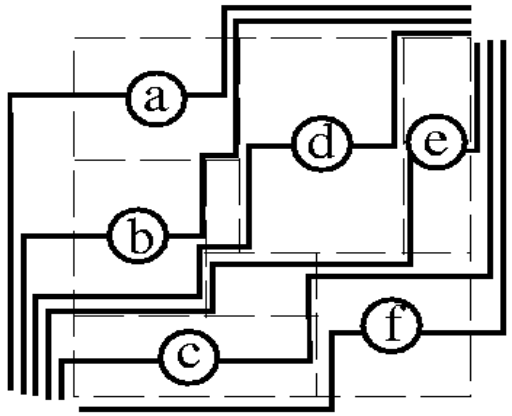
Loci of module b

Relative Module Positions

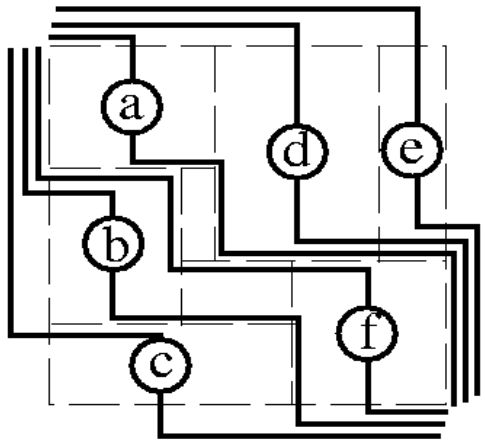
- A floorplan is a partition of a chip into rooms, each containing at most one module.
- **Locus** (right-up, left-down, up-left, down-right)
  1. Take a non-empty room.
  2. Start at the center of the room, walk in two alternating directions to hit the sides of rooms.
  3. Continue until to reach a corner of the chip.
- **Positive locus**: Union of right-up locus and left-down locus.
- **Negative locus**: union of up-left locus and down-right locus.



Loci of module b



Positive loci: abdecf

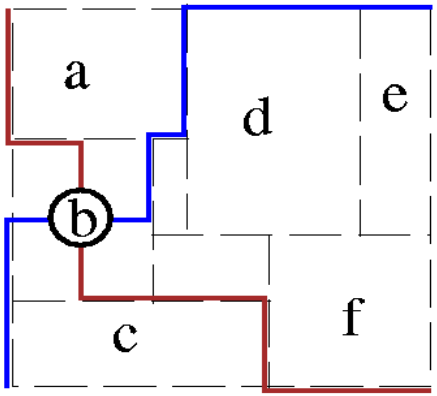


Negative loci: cbfade

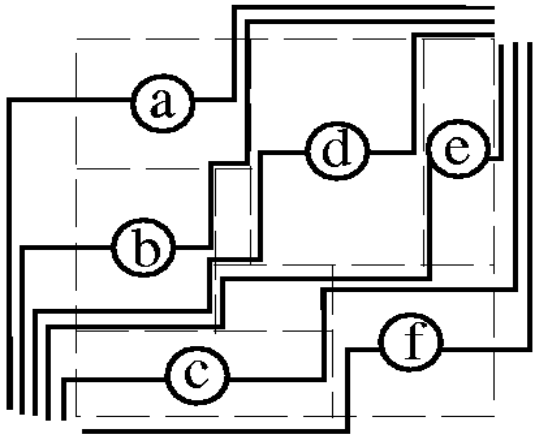
Geometrical Information

- No pair of positive (negative) loci cross each other, i.e., **loci are linearly ordered**.
- **Sequence-Pair** (Γ_+, Γ_-): Γ_+ (Γ_-) is the module permutation representing the order of positive (negative) loci.

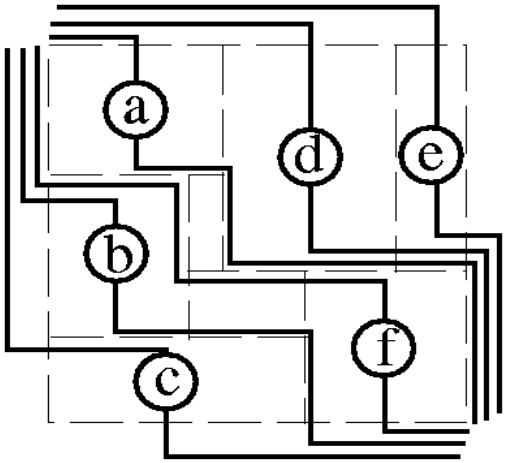
  E.g., (Γ_+, Γ_-) = (*abdecf*, *cbfade*).
- *x*' is **after** (**before**) *x* in both Γ_+ and $\Gamma_- \Rightarrow$ *x*' is **right** (**left**) to *x*.
- *x*' is **after** (**before**) *x* in Γ_+ and before (**after**) *x* in $\Gamma_- \Rightarrow$ *x*'is **below** (**above**) *x*.



Loci of module b



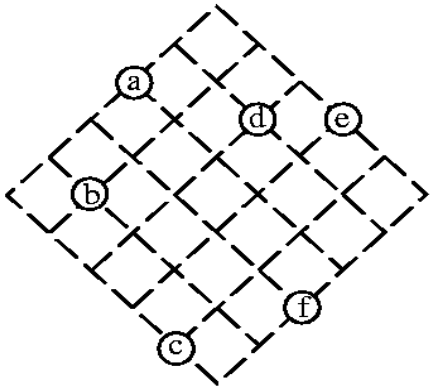
Positive loci: abdecf



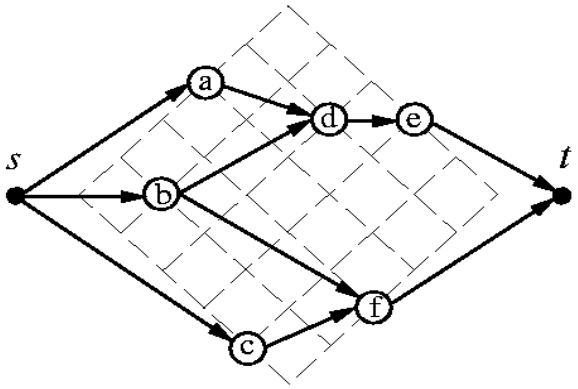
Negative loci: cbfade

Optimal (Γ_+, Γ_-)-Packing

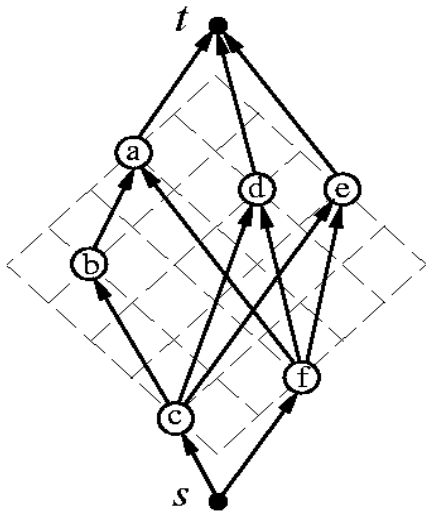
- For every sequence-pair (Γ_+, Γ_-), there is an optimal (Γ_+, Γ_-)-packing.
- **Horizontal constraint graph** $G_H(V, E)$ (similarly for $G_V(V, E)$):
  - V: source s, sink t, n vertices for modules.
  - E: (s, x) and (x, t) for each module x, and (x, x') iff x must be left-to x'.
  - **Vertex weight**: 0 for s and t, **width** of each module for the corresponding vertex.



Packing for sequence pair: (abdecf, cbfade)



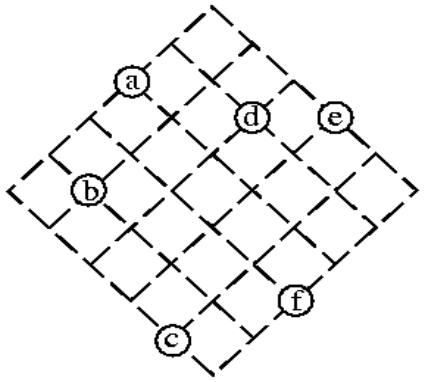
Horizontal constraint graph (Transitive edges are not shown)



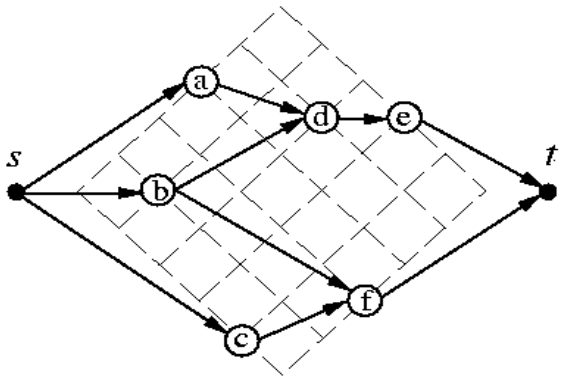
Vertical constraint graph (Transitive edges are not shown)

Optimal (Γ_+, Γ_-)-Packing (cont'd)

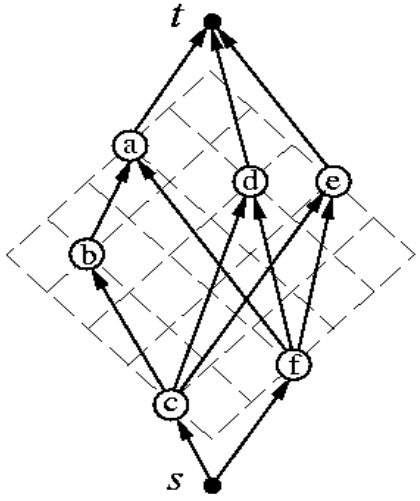
- **Optimal (Γ_+, Γ_-)-packing** can be obtained in $O(n^2)$ time by applying a longest path algorithm on a vertex-weighted directed acyclic graph.
  - G_H and G_V are independent.
  - The x and y coordinates of each module are determined by assigning the longest path length between s and the vertex of the module in G_H and G_V, respectively.
- More efficient algorithms for obtaining optimal (Γ_+, Γ_-)-packing: $O(n\log n)$ by Takahashi, IEICE'96; $O(n\log n)$ by Tang, Tian & Wong, DATE'00; $O(n\log\log n)$ by Tang & Wong, ASP-DAC'01.



Packing for sequence pair: (abdecf, cbfade)



Horizontal constraint graph (Transitive edges are not shown)



Vertical constraint graph (Transitive edges are not shown)

O-Tree

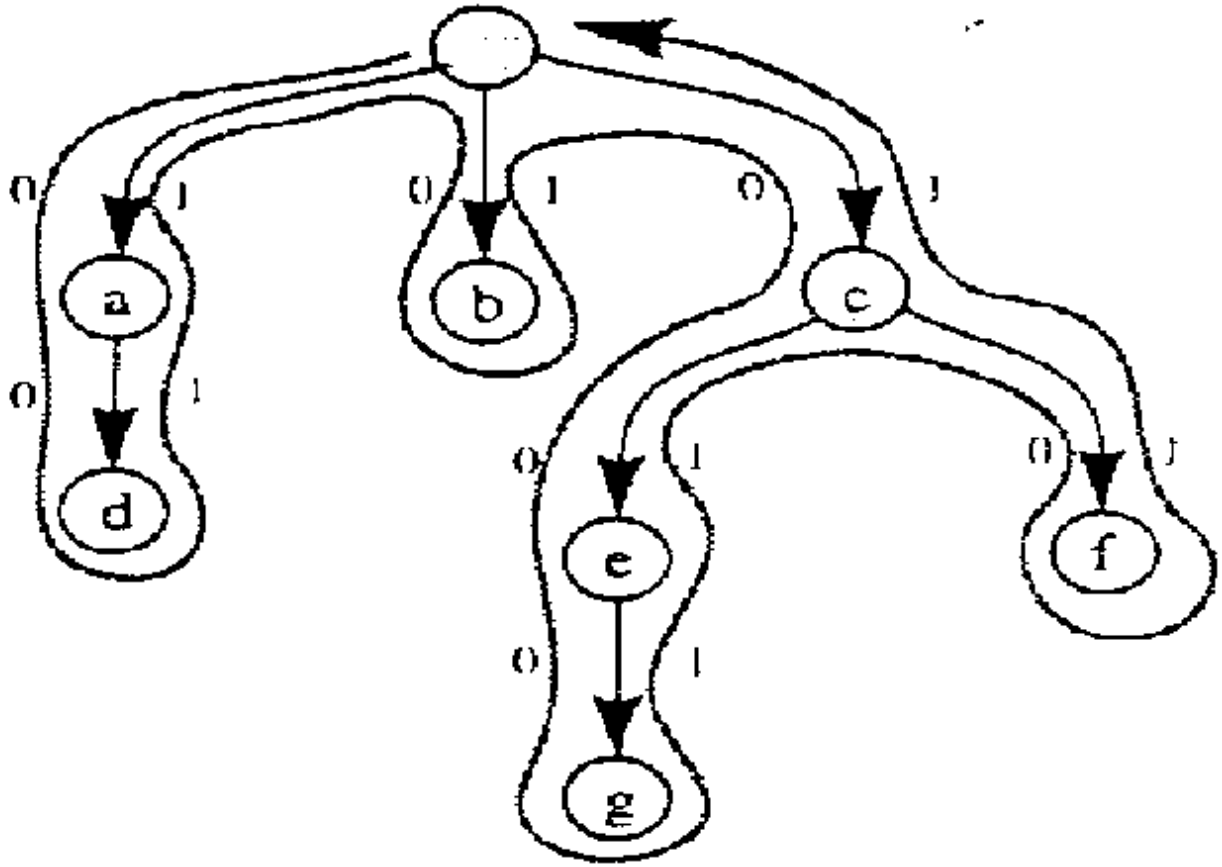
- Guo, Cheng, and Yoshimura, "An o-tree representation of non-slicing floorplan and its applications," DAC'99.
- Definitions:

  - A placement is **L-compact** (**B-compact**) if and only if no module can be moved left (down) from its original position with other modules' positions fixed.

  - A placement is **LB-compact** (or **admissible**) if and only if it is both L-compact and B-compact.
- Given any placement P_1, a corresponding admissible placement P_2 can be obtained by a sequence of x-direction and y-direction compactions. The overall area of P_2 is no larger than the overall area of P_1.

O-Tree Encoding

(00110100011011,*adbcegf*) with Depth-First-Search



O-Tree Encoding (cont'd)

- **Space needed to store (T,π):** Given a tree with n nodes in addition to its root, the label of each node can be encoded into a $[\lg n]$ bit string, and hence $n(2+[\lg n])$ bits are needed to store (T,π) where $2n$ bits for T, and $n[\lg n]$ bits for π.

- **Number of possible (T,π)'s:** $O(n!2^{2n-2}/n^{1.5})$

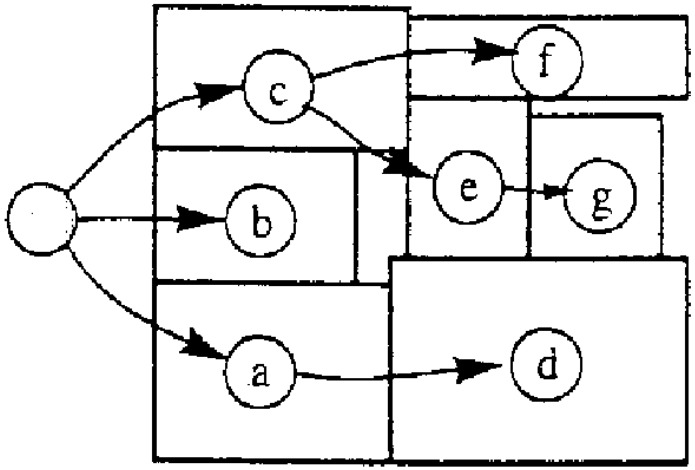
O-Tree and Placement

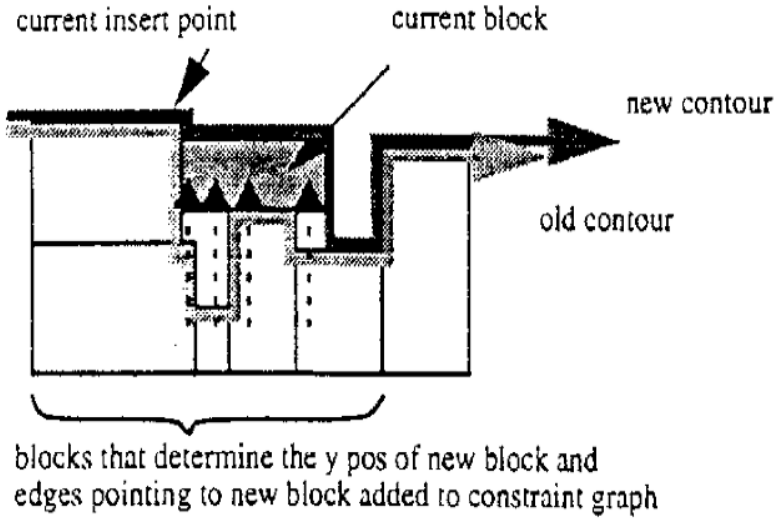
- **Horizontal O-Tree:**

  - Suppose i is the parent of j, $\Psi(j)$ is the set of blocks each of which appears before j in π and overlaps with j in the x-coordinate projections.

(00110100011011,*adbcegf*)

$$\begin{cases} x_j = x_i + w_i \\ y_j = 0 \,|\, (\max_{k \in \psi(j)} y_k + h_k) \end{cases}$$

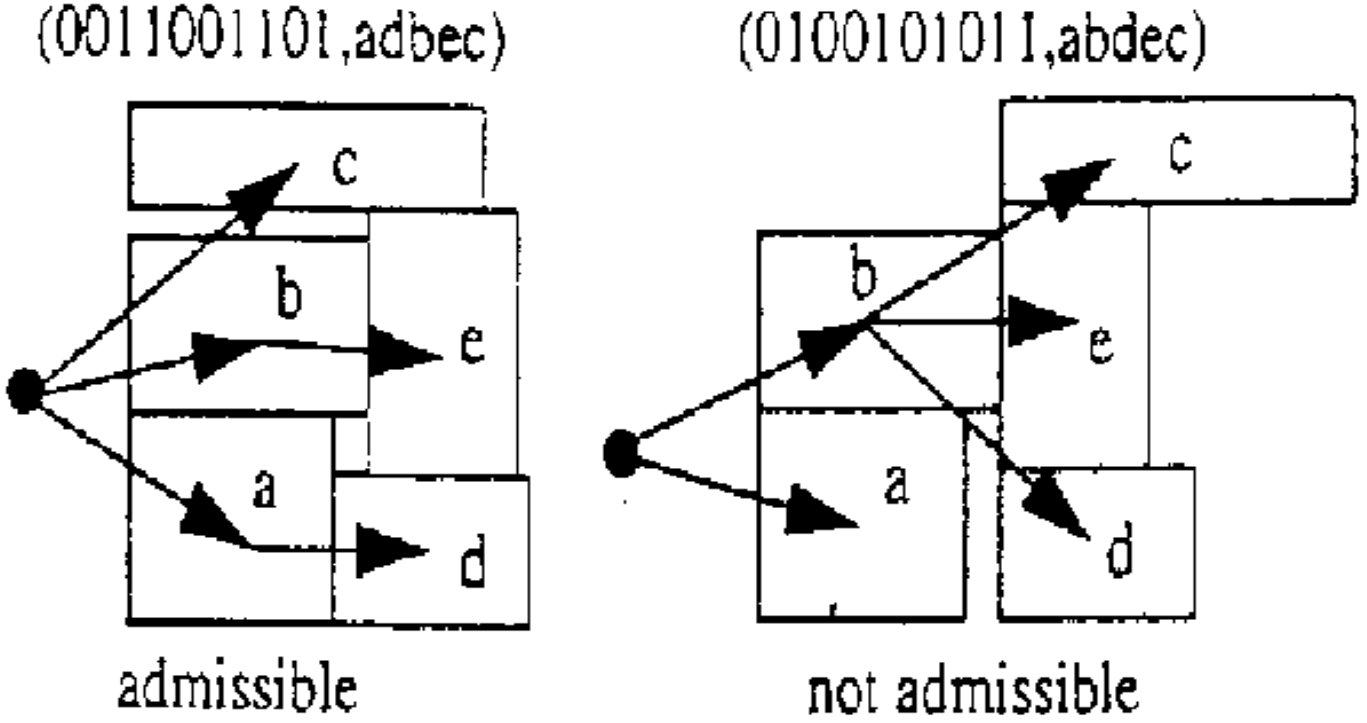




- **Vertical O-Tree:** can be defined similarly

O-Tree and Placement (cont'd)

An O-tree is **admissible** if its corresponding placement is admissible.



Admissible O-Tree Transformation (AOT)

- Given a horizontal O-tree T, we can first get a vertical constraint graph G_y by applying "OT2OCG" to T in linear time, and then get a vertical O-tree T_y by applying "CG2OT" to G_y in linear time. After applying the same procedures OT2OCG and CG2OT again, we can get another horizontal O-tree. The OT2OCG and CG2OT are iterated until an admissible O-tree is found.

H-O-tree $\xrightarrow{\text{OT2OCG}}$ **G_v(B-compact)** $\xrightarrow{\text{CG2OT}}$ **V-O-tree** $\xrightarrow{\text{OT2OCG}}$ **G_h(L-compact)** $\xrightarrow{\text{CG2OT}}$ **H-O-tree...**

- All compactions are **monotone** because modules are either moved down or left. Therefore, convergence of the above iteration is assured and we can get an admissible O-tree.

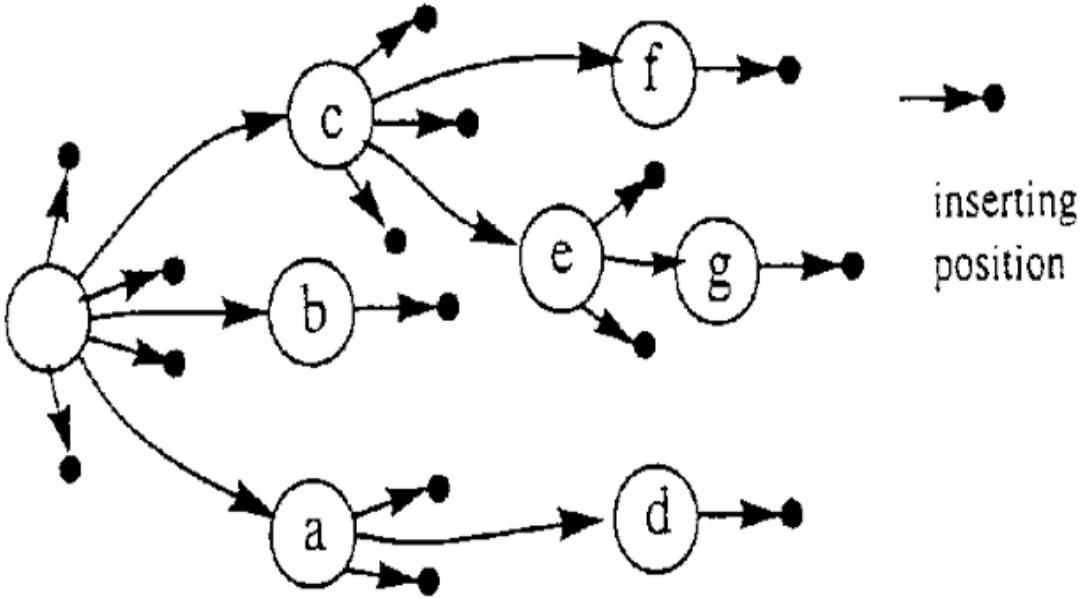
Solution Perturbation

a Select a module B_i in the O-tree (T,π).

b Delete B_i from the O-tree (T,π).

c Insert B_i in the position with the best cost value among all possible inserting positions in (T,π) as an external node.

d Perform a-c on it orthogonal O-tree.



•Given any O-tree with n nodes, the number of possible inserting position as external nodes is $2n$-1.

A Deterministic Placement Algorithm

- Perturb O-trees in sequence.

  - Select nodes in sequence and find the best perturb position for each of them.

  - A perturbed O-tree can be made admissible using AOT.
- Implementation is straightforward.