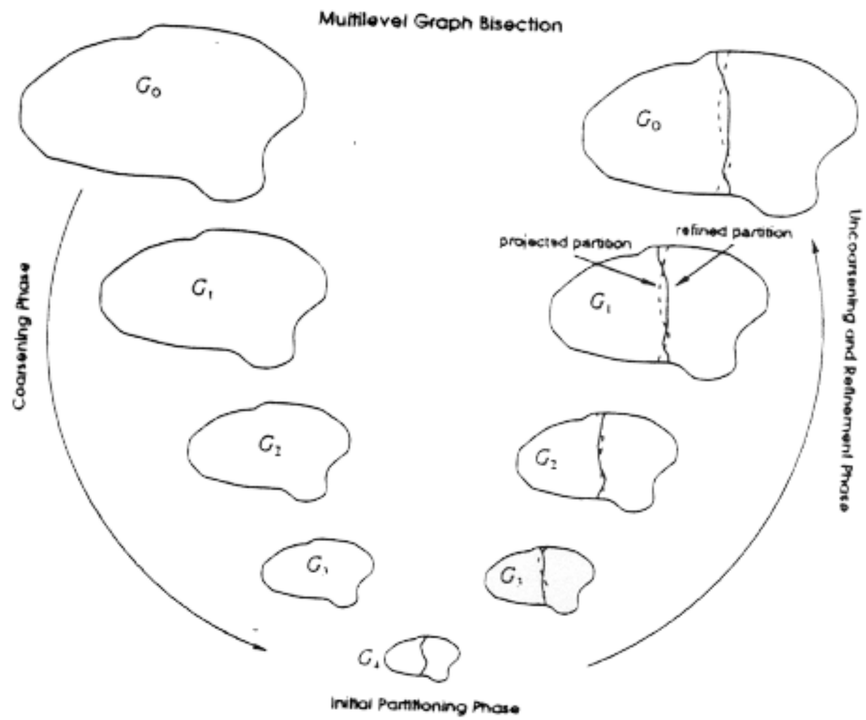
Multilevel Partitioning

- **Three phases** (for bipartitioning)
  - **Coarsening:** construct a sequence of smaller (coarser) graphs.
  - **Initial partitioning:** construct a bipartitioning solution for the coarsest graph.
  - **Uncoarsening & refinement:** the bipartitioning solution is successively projected to the next-level finer graph, and at each level an iterative refinement algorithm (such as KL or FM) is used to further improve the solution.

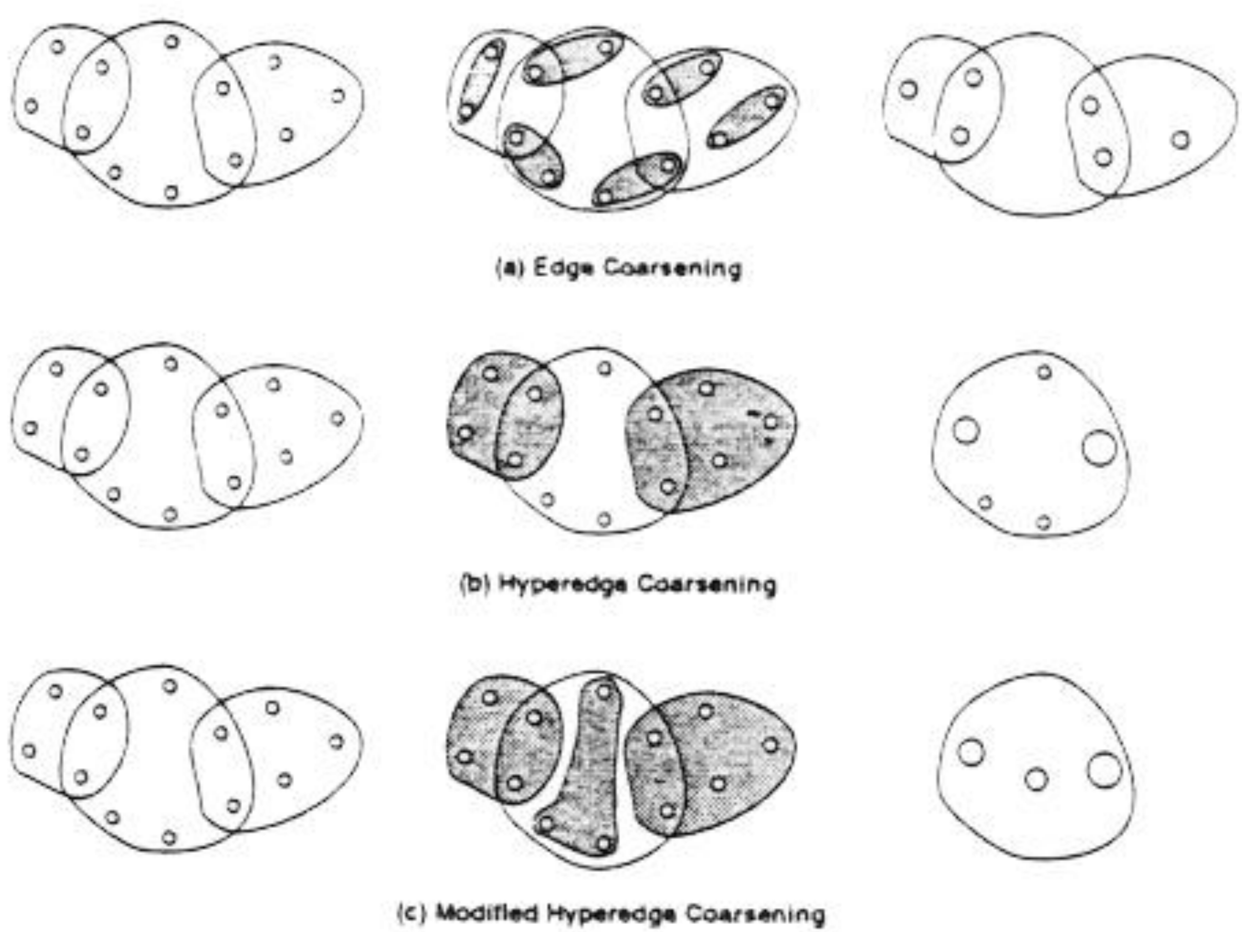


hMETIS

- Kayrpis, Aggarwal, Kumar and Shekhar, "Multilevel hypergraph partitioning: application in VLSI domain," DAC, 1997.
- Three coarsening algorithms:

  - **Edge coarsening:** A maximal matching of the vertices.
  - **Hyperedge coarsening**: a set of hyperedges is selected, and the vertices belonging to a selected hyperedge are merged into a cluster. (Preference: hyperedges with large weights and hyperedges of small size.)
  - **Modified hyperedge coarsening:** hyperedge coarsening + merging the remaining vertices of each hyperedge into a cluster.

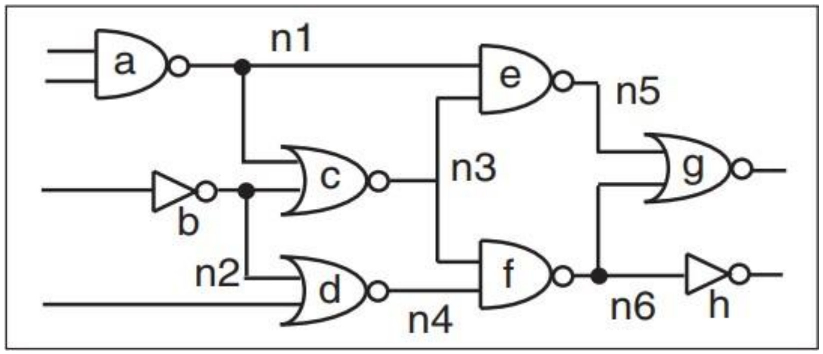
Coarsening Algorithms

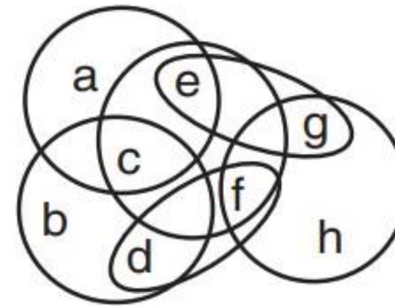


Coarsening Algorithms (Cont'd)

- Edge coarsening (EC)
  1. Unmark all nodes.
  2. Repeat until all nodes are marked.
     - Randomly select an unmarked node v
     - Collect the neighbors of v, which is the set of nodes that are unmarked and are included in the hyperedges that contain v.
     - For each neighbor n of v, compute the weight of edge (v, n) by assigning a value $1/(|h| - 1)$, where h denotes a hyperedge that contains both n and v.
     - Examine all neighbors of v and select the neighbor m with the maximum edge weight.
     - Merge v and m to form a cluster, and mark v and m .

Coarsening Algorithms (Cont'd)





- Assume the weight of each net is 1
- EC result (visit unmarked nodes and break ties in alphabetical order)

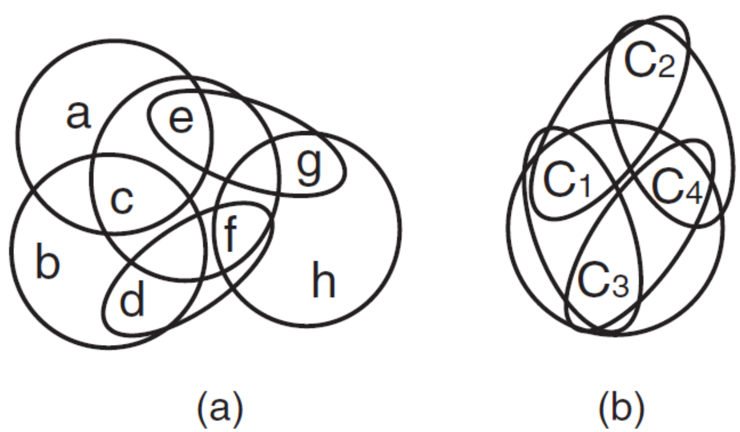
Cluster	Nodes
C_1	$\{a, c\}$
C_2	$\{b, d\}$
C_3	$\{e, g\}$
C_4	$\{f, h\}$

Coarsening Algorithms (Cont'd)

- Netlist transformation based on EC result

Net	Gate-level	Cluster-level	Final
n_1	$\{a, c, e\}$	$\{C_1, C_1, C_3\}$	$\{C_1, C_3\}$
n_2	$\{b, c, d\}$	$\{C_2, C_1, C_2\}$	$\{C_1, C_2\}$
n_3	$\{c, e, f\}$	$\{C_1, C_3, C_4\}$	$\{C_1, C_3, C_4\}$
n_4	$\{d, f\}$	$\{C_2, C_4\}$	$\{C_2, C_4\}$
n_5	$\{e, g\}$	$\{C_3, C_3\}$	$\emptyset$
n_6	$\{f, g, h\}$	$\{C_4, C_3, C_4\}$	$\{C_3, C_4\}$

- Hypergraph before EC (a) and after EC (b)



Coarsening Algorithms (Cont'd)

- Hyperedge coarsening (HEC)
    1. Unmark all nodes.
    2. Sort hyperedges in a decreasing order of their weights and break ties in favor of smaller size.
    3. Visit each hyperedge h in the sorted order, and if all nodes in h are unmarked, merge all nodes in h to form a cluster and mark them.
    4. After visiting all hyperedges, each unmarked node forms a cluster of its own.
- Modified hyperedge coarsening (MHEC)
    1. Apply HEC to the hypergraph.
    2. Visit hyperedges again in the sorted order, and for each hyperedge h that contains one or more unmarked nodes, all the unmarked nodes in h are merged to form a cluster, and they are marked.

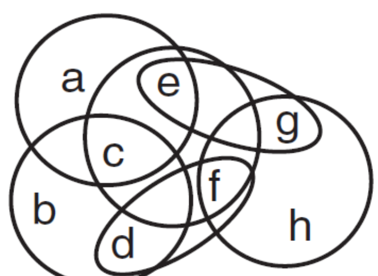
Coarsening Algorithms (Cont'd)

- HEC result (break ties in alphabetical order) and netlist transformation based on HEC result

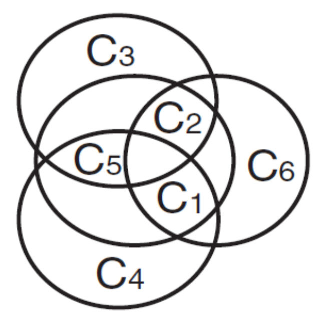
Cluster	Nodes
C_1	$\{d, f\}$
C_2	$\{e, g\}$
C_3	$\{a\}$
C_4	$\{b\}$
C_5	$\{c\}$
C_6	$\{h\}$

Net	Gate-level	Cluster-level	Final
n_1	$\{a, c, e\}$	$\{C_3, C_5, C_2\}$	$\{C_3, C_5, C_2\}$
n_2	$\{b, c, d\}$	$\{C_4, C_5, C_1\}$	$\{C_4, C_5, C_1\}$
n_3	$\{c, e, f\}$	$\{C_5, C_2, C_1\}$	$\{C_5, C_2, C_1\}$
n_4	$\{d, f\}$	$\{C_1, C_1\}$	$\emptyset$
n_5	$\{e, g\}$	$\{C_2, C_2\}$	$\emptyset$
n_6	$\{f, g, h\}$	$\{C_1, C_2, C_6\}$	$\{C_1, C_2, C_6\}$

- Hypergraph before HEC (a) and after HEC (b)



(a)



(b)

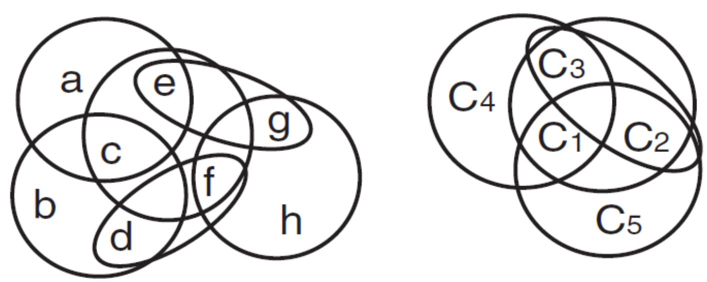
Coarsening Algorithms (Cont'd)

- MHEC result (break ties in alphabetical order) and netlist transformation based on MHEC result

Cluster	Nodes
C_1	$\{d, f\}$
C_2	$\{e, g\}$
C_3	$\{a, c\}$
C_4	$\{b\}$
C_5	$\{h\}$

Net	Gate-level	Cluster-level	Final
n_1	$\{a, c, e\}$	$\{C_3, C_3, C_2\}$	$\{C_3, C_2\}$
n_2	$\{b, c, d\}$	$\{C_4, C_3, C_1\}$	$\{C_4, C_3, C_1\}$
n_3	$\{c, e, f\}$	$\{C_3, C_2, C_1\}$	$\{C_3, C_2, C_1\}$
n_4	$\{d, f\}$	$\{C_1, C_1\}$	$\emptyset$
n_5	$\{e, g\}$	$\{C_2, C_2\}$	$\emptyset$
n_6	$\{f, g, h\}$	$\{C_1, C_2, C_5\}$	$\{C_1, C_2, C_5\}$

- Hypergraph before MHEC (a) and after MHEC (b)



(a) (b)

Uncoarsening & Refinement Algorithms

- Two uncoarsening & refinement algorithms:

  - FM algorithm with modifications:

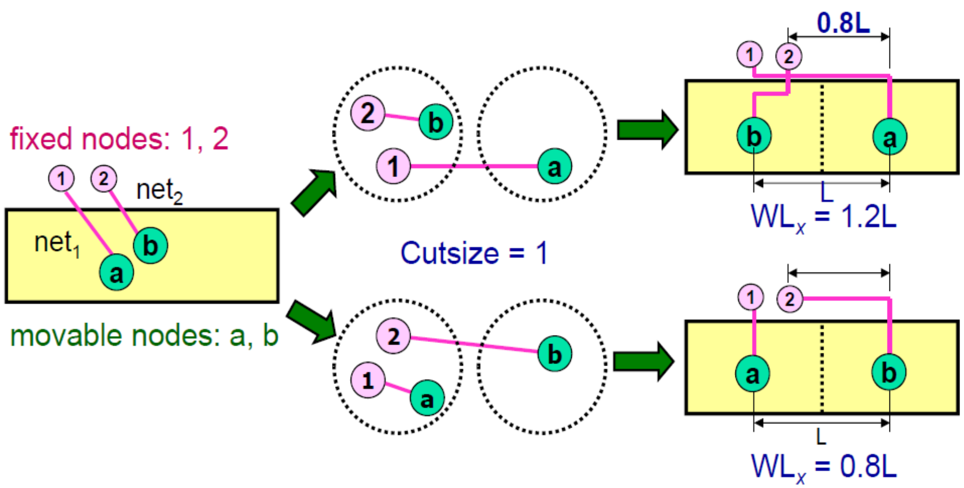
    * Restrict the maximum number of passes to 2.

    * Stop each pass when no improvement is made from the first *k* moves.

  - Hyperedge refinement: move groups of vertices between subsets so that an entire hyperedge is removed from the cut set.

Partitioning for Wirelength Minimization

- Chen, Chang, Lin, "IMF: Interconnection-driven floorplanning for large-scale building-module designs," ICCAD-05
- Minimizing cut size is *not* equivalent to minimizing wirelength (WL)

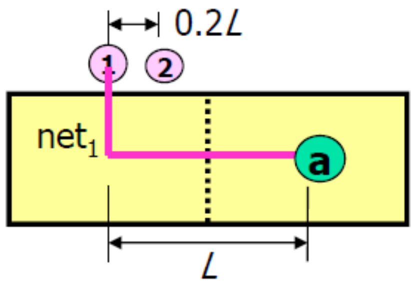


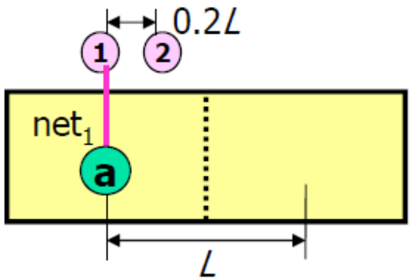
- Problem: **hyperedge weight is a constant value!**
  - Shall map the min-cut cost to wirelength (WL) change
  - Shall assign the hyperedge weight as the value of wirelength contribution if the hyperedge is cut

Net Weight Assignment

- net_1 connects a movable node *a* and a fixed node 1.

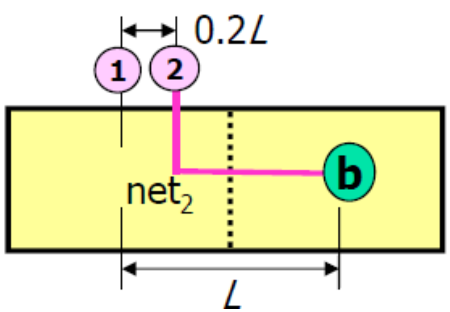
$$\text{Weight}(\text{net}_1) = \text{WL}(\text{net}_1 \text{ is cut}) - \text{WL}(\text{net}_1 \text{ is not cut})$$
$$= L - 0L = L$$

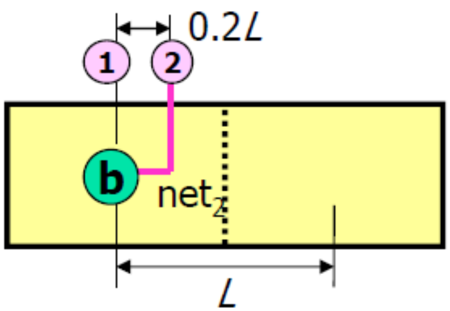




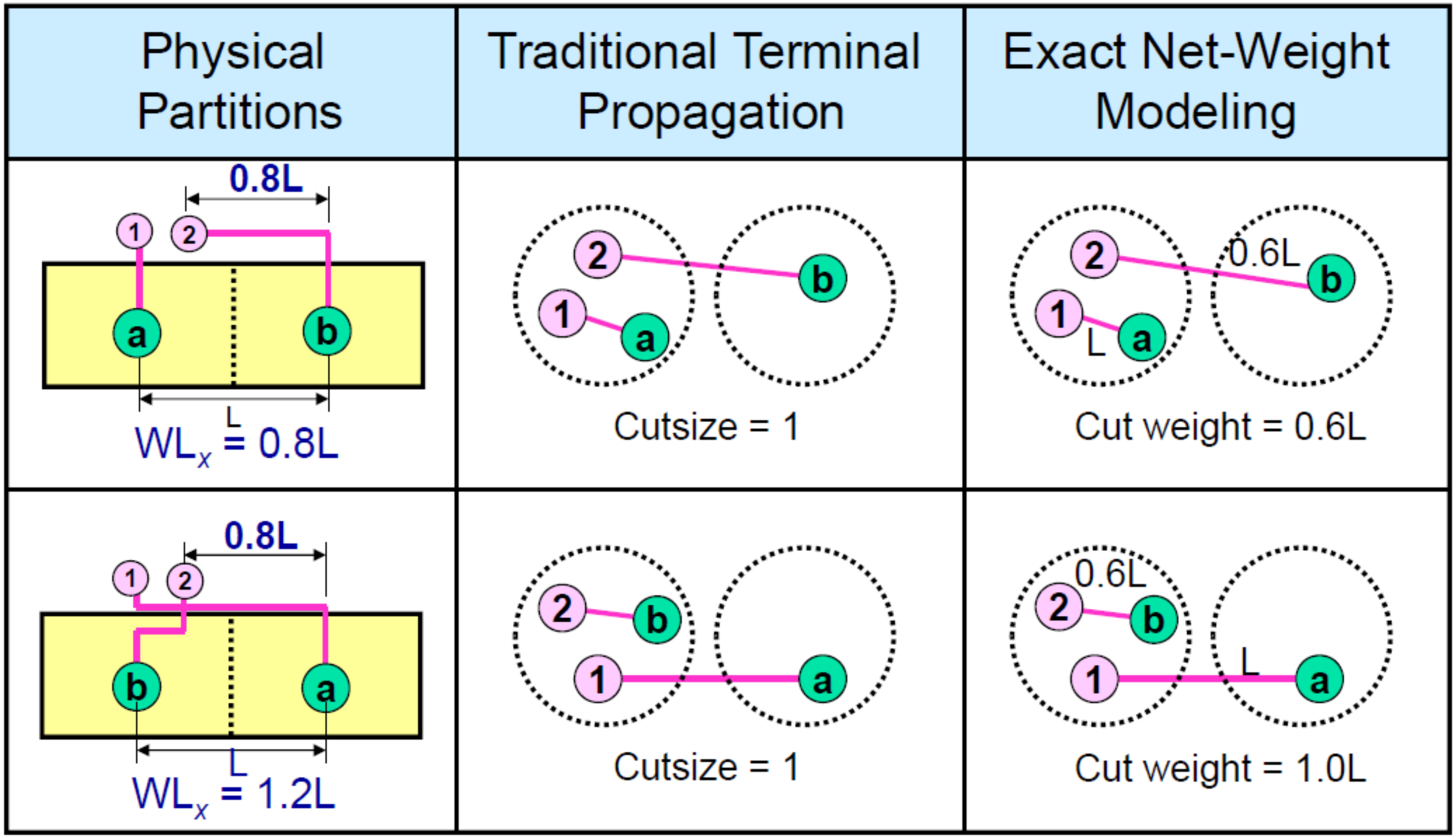
- net_2 connects a movable node *b* and a fixed node 2.

$$\text{Weight}(\text{net}_2) = \text{WL}(\text{net}_2 \text{ is cut}) - \text{WL}(\text{net}_2 \text{ is not cut})$$
$$= 0.8L - 0.2L = 0.6L$$





Examples



Cut weight is proportitional to the wirelength (WL)

WL = Cut weight + 0.2L

(0.2L is the WL lower bound: placing a & b in the left side)

Relationship Between WL and Cut Weight

- Theorem: $WL_i = w_{1,i} + n_{cut,i}$
  - $n_{cut,i}$: cut weight for net i
  - $w_{1,j}$: the wirelength lower bound for net i

- Then, we have

Constant

$$min(\sum WL_i) = min\left(\sum\left(w_{1,i} + n_{cut,i}\right)\right) = \sum w_{1,i} + min\left(\sum n_{cut,i}\right)$$

Finding the minimum wirelength is equivalent to finding the cut weight