Placement

ECE6133

Physical Design Automation of VLSI Systems

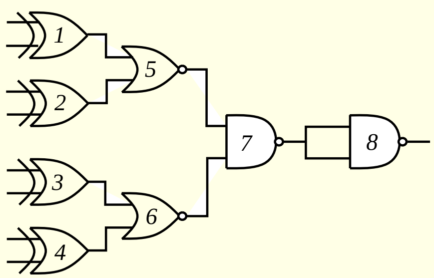
Prof. Sung Kyu Lim

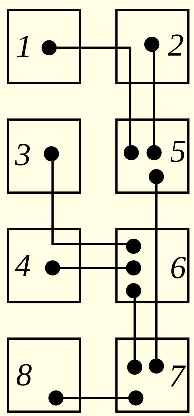
School of Electrical and Computer Engineering

Georgia Institute of Technology

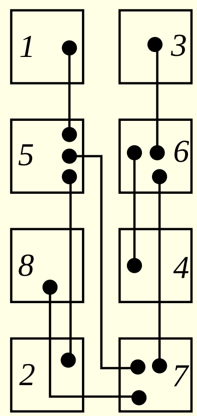
Placement

- The process of arranging the circuit components on a layout surface.
- Inputs: A set of fixed modules, a netlist.
- Goal: Find the best position for each module on the chip according to appropriate cost functions.
  - Considerations: **routability/channel density**, **wirelength**, cut size, performance, thermal issues, I/O pads.

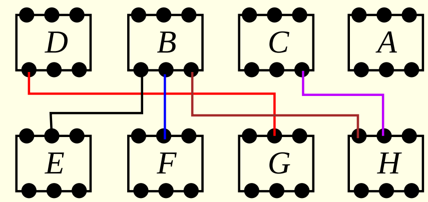




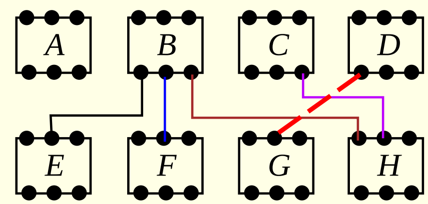
wirelength = 10



wirelength = 12



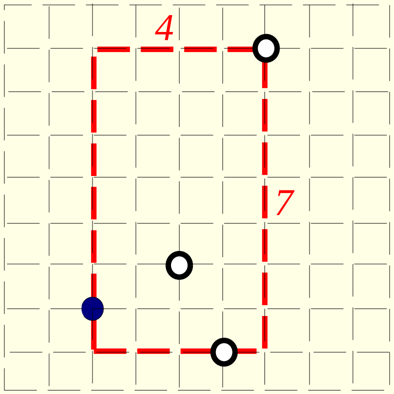
Density = 2 (2 tracks required)



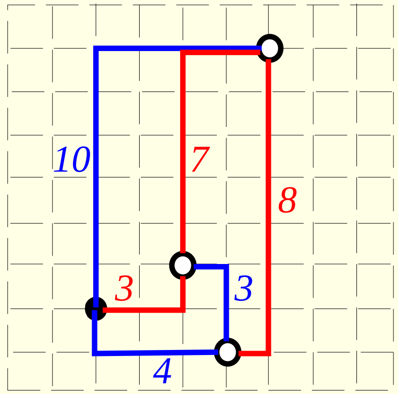
Shorter wirelength, 3 tracks required.

Estimation of Wirelength

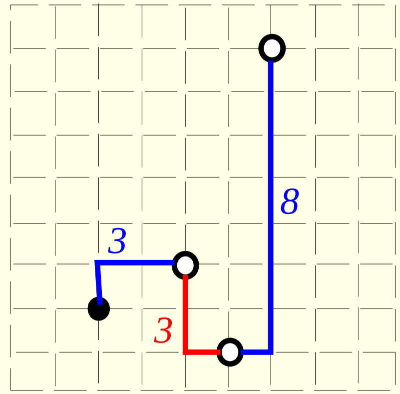
- **Semi-perimeter method:** Half the perimeter of the bounding rectangle that encloses all the pins of the net to be connected. Most widely used approximation!
- **Complete graph:** Since #edges in a complete graph ($\frac{n(n-1)}{2}$) is $\frac{n}{2}\times$ # of tree edges $(n-1)$, $wirelength \approx \frac{2}{n}\sum_{(i,j)\in net} dist(i,j)$.
- **Minimum chain:** Start from one vertex and connect to the closest one, and then to the next closest, etc.
- **Source-to-sink connection:** Connect one pin to all other pins of the net. Not accurate for uncongested chips.
- **Steiner-tree approximation:** Computationally expensive.
- **Minimum spanning tree**



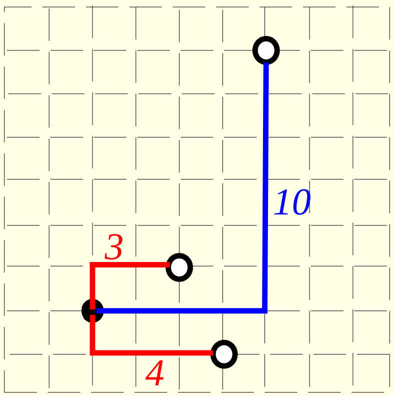
semi−perimeter len = 11



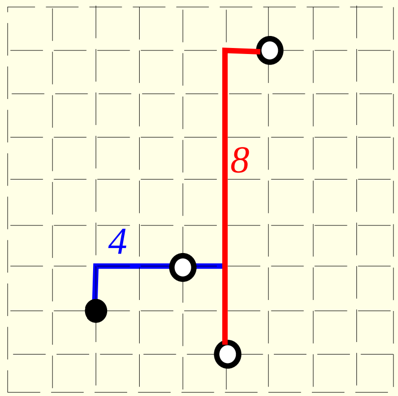
*complete graph len * 2/n = 17.5*



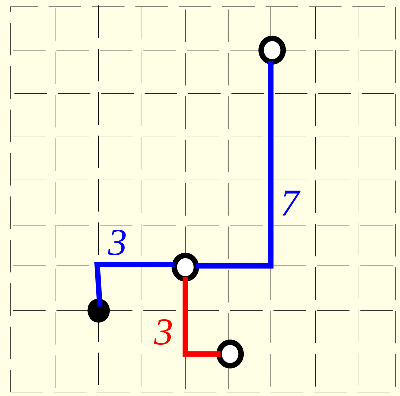
chain len = 14



source−to−sink len = 17



Steiner tree len = 12



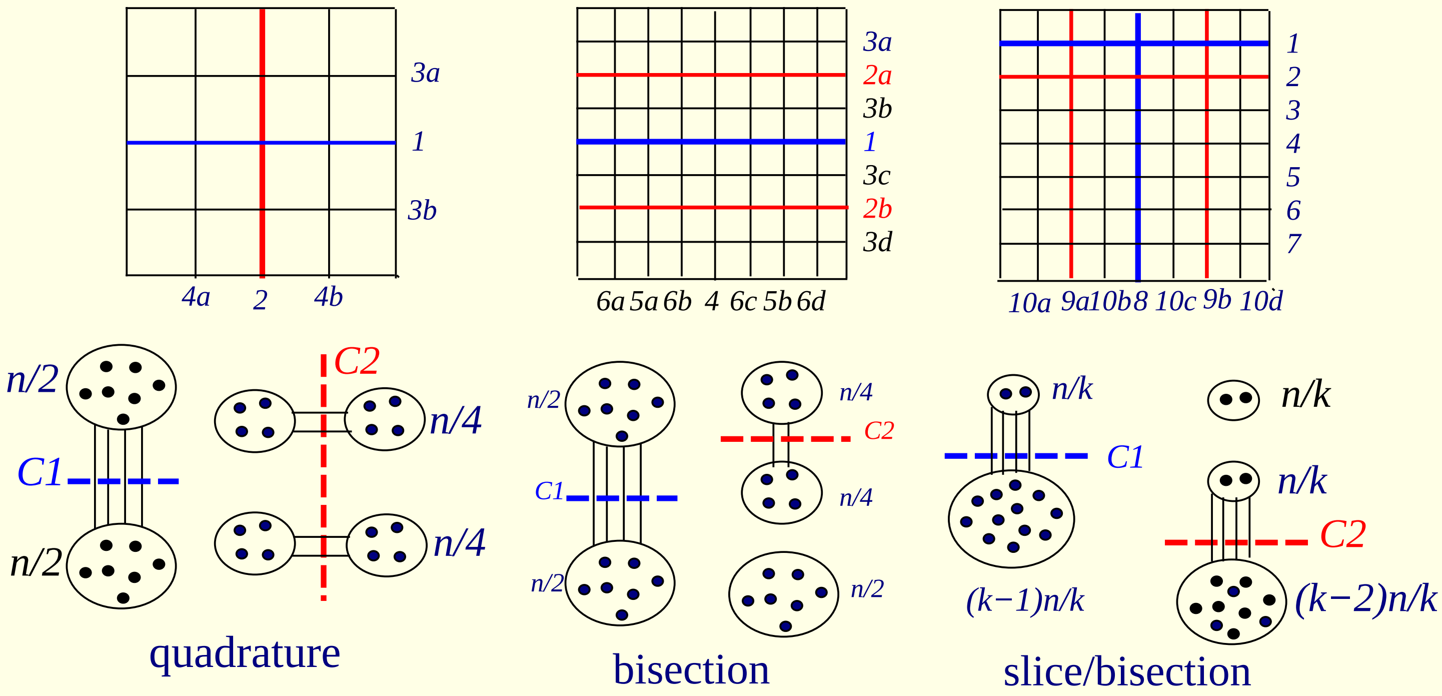
Spanning tree len = 13

Placement Methods

- **Constructive methods**
  - Cluster growth algorithm
  - Force-directed method
  - Algorithm by Goto
  - Min-cut based method
- **Iterative improvement methods**
  - Pairwise exchange
  - Simulated annealing: Timberwolf
  - Genetic algorithm
- **Analytical methods**
  - Gordian, Gordian-L

Min-Cut Placement

- Breuer, "A class of min-cut placement algorithms," DAC-77.
- **Quadrature:** suitable for circuits with high density in the center.
- **Bisection:** good for standard-cell placement.
- **Slice/Bisection:** good for cells with high interconnection on the periphery.



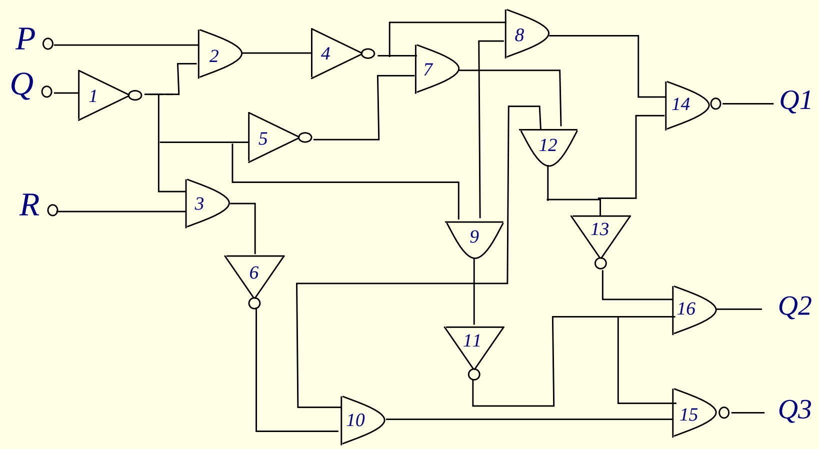
Algorithm for Min-Cut Placement

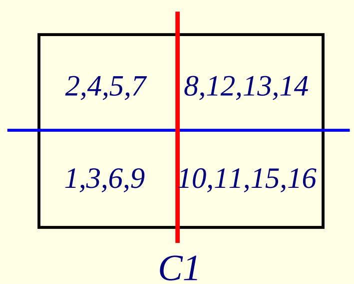
Algorithm: Min_Cut_Placement(N, n, C)
/* N: the layout surface */
/* n: # of cells to be placed */
/* n_0: # of cells in a slot */
/* C: the connectivity matrix */

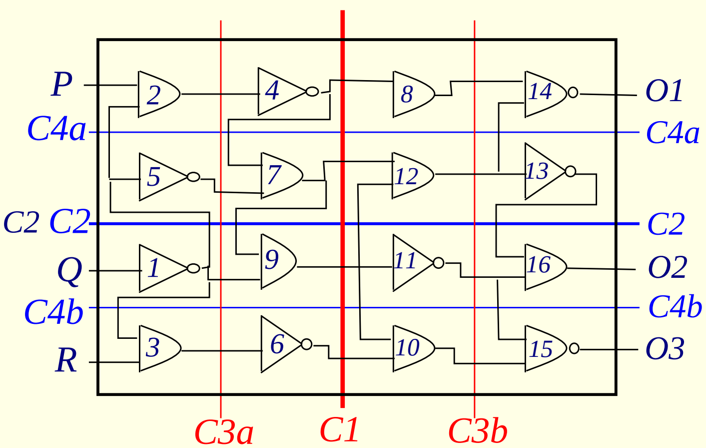
begin
if $(n \leq n_0)$ **then** PlaceCells(N, n, C);
else
 $(N_1, N_2) \leftarrow$ CutSurface(N);
 (n_1, C_1), $(n_2, C_2) \leftarrow$ Partition(n, C);
 Call Min_Cut_Placement(N_1, n_1, C_1);
 Call Min_Cut_Placement(N_2, n_2, C_2);
end

Quadrature Placement Example

- Apply K-L heuristic to partition + Quadrature Placement: Cost $C_1 = 4$, $C_{2L} = C_{2R} = 2$, etc.

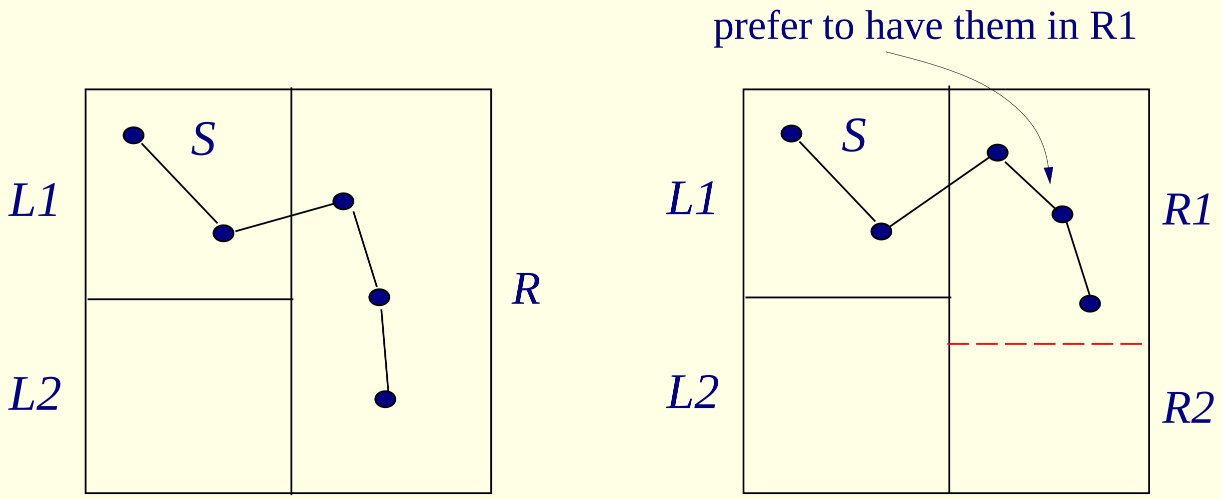






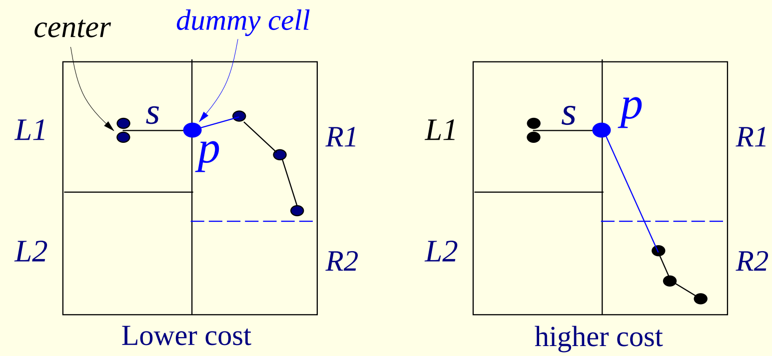
Min-Cut Placement with Terminal Propagation

- Dunlop & Kernighan, "A procedure for placement of standard-cell VLSI circuits," IEEE TCAD, Jan. 1985.
- Drawback of the original min-cut placement: Does not consider the positions of terminal pins that enter a region.
  - What happens if we swap $\{1, 3, 6, 9\}$ and $\{2, 4, 5, 7\}$ in the previous example?



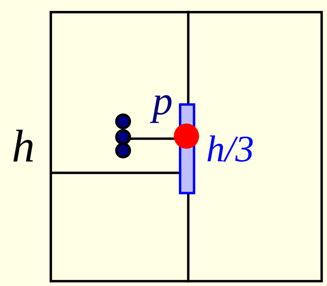
Terminal Propagation

- We should use the fact that s is in L_1!

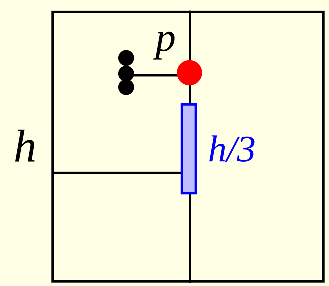


P will stay in R1 for the rest of partitioning!

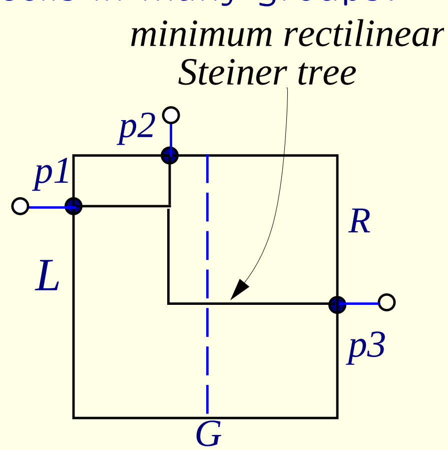
- When not to use p to bias partitioning? Net s has cells in many groups?



Don't use p to bias the solution in either direction!

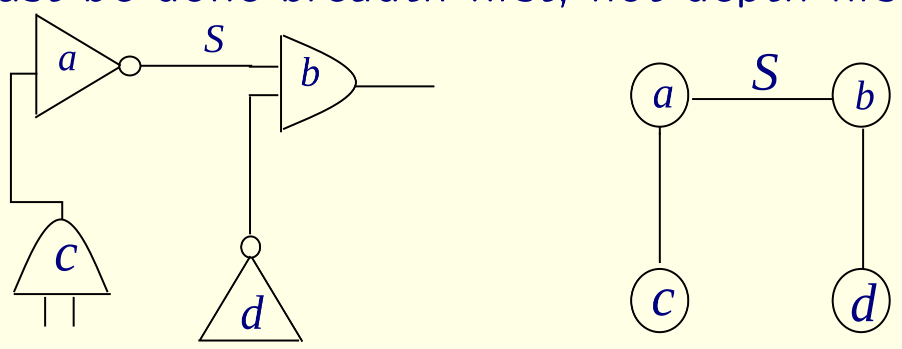


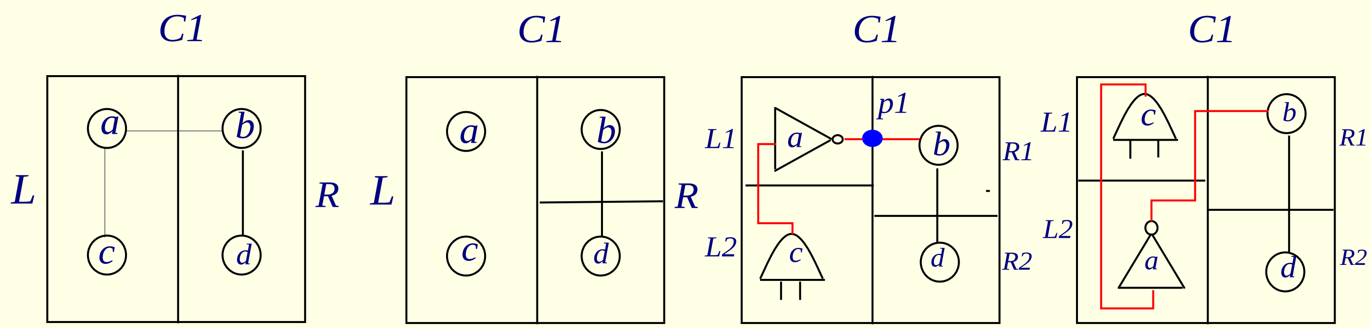
Use p!



Terminal Propagation Example

- Partitioning must be done breadth-first, not depth-first.





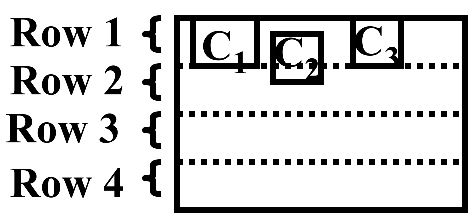
unbiased partition of R

with terminal propagation

without terminal propagation

Creating Rows

- Terminal propagation reduce overall area by ~30%
- Creating rows
  - Choose α and β preferably to balance row to balance row length (during re-arrangement)

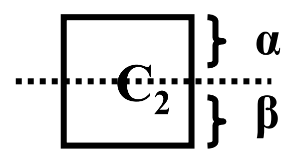


cells in $C_1 \rightarrow$ row1

cells in $C_3 \rightarrow$ row1

cells in C_2

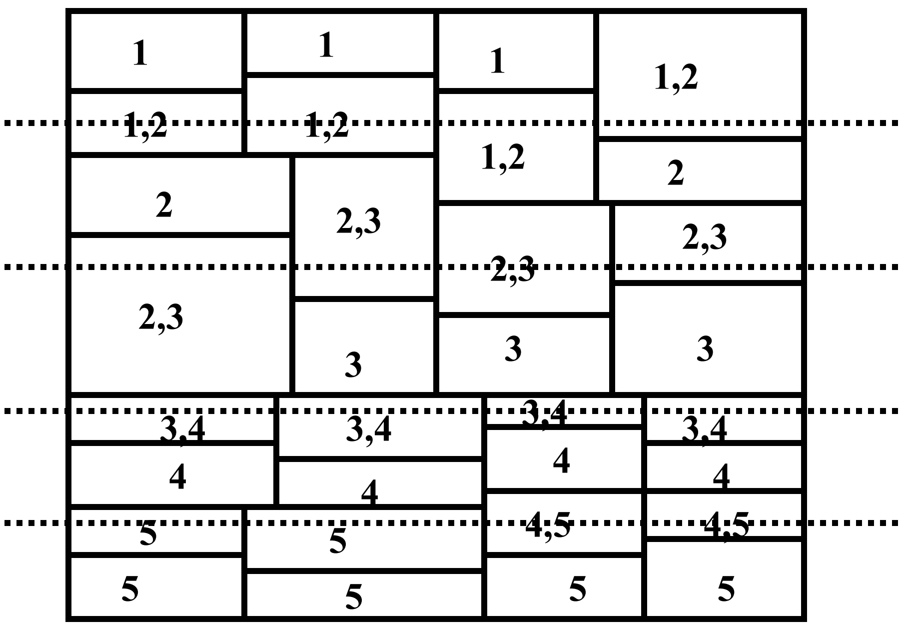
Row 1 Row 2



$\alpha + \beta = 1$

Creating Rows

- **Example**
  - Partitioning of circuit into 32 groups
  - Each group is either assigned to a single row or divided into 2 rows



a four-row standard cell design

Experimental Results

- CMOS Chip with 453 nets and 412 cells
- Manual solution
  - track density=147; feedthroughs=184
- Automated solution
  - without terminal propagation: t.d.=313; f.t.=591
  - (t.d. reduced to 235 by iterative interchanges)
  - with terminal propagation: t.d.=186; f.t.=182
  - (t.d. reduced to 152 by iterative interchanges)
  - Iterative Interchange Refinement is helpful
- The program is in production use as part of an automatic placement system in AT&T Bell Lab.
  - Solutions within 10% of the best hand layout

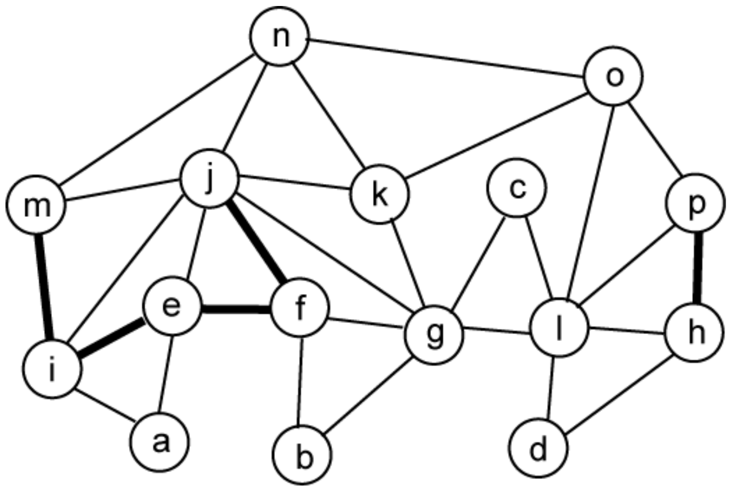
Remarks on Min-cut Placement

- **Also implemented F-M partitioning method**
  - **Much faster but solutions appeared to be not as good as K-L**
- **Use Simulated Annealing to do partitioning**
  - **Much slower. If restricted to a reasonable CPU time, solutions are of similar quality of those by F-M method. Easy to implement**
- **Seeking an elegant way to force some cells to be in particular positions**
- **Investigate other algorithms for terminal propagation**
  - **Terminal propagation is the bottleneck of CPU time**

Mincut Placement

- Perform quadrature mincut onto 4 × 4 grid
  - Start with vertical cut first

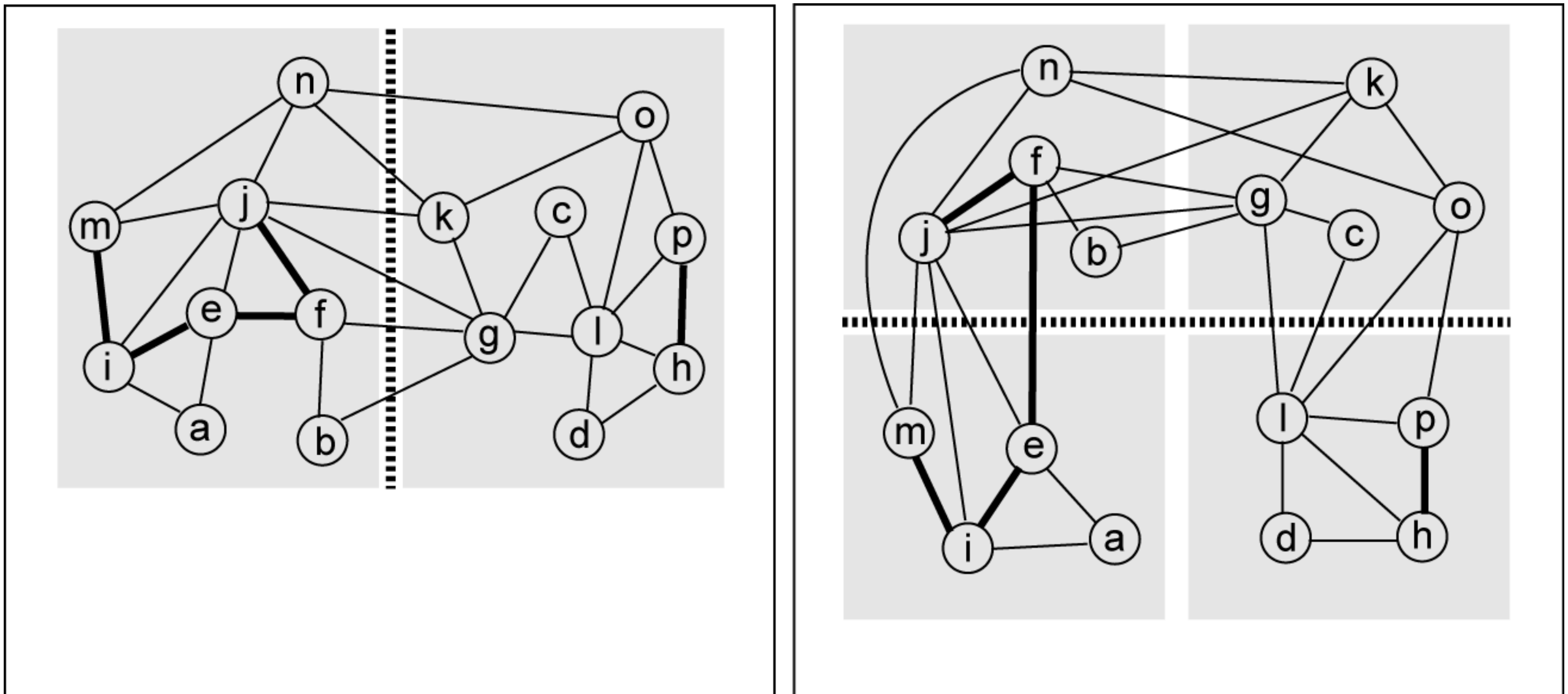
$$
\begin{aligned}
n_1 &= \{e, f\} \\
n_2 &= \{a, e, i\} \\
n_3 &= \{b, f, g\} \\
n_4 &= \{c, g, l\} \\
n_5 &= \{d, l, h\} \\
n_6 &= \{e, i, j\} \\
n_7 &= \{f, j\} \\
n_8 &= \{g, j, k\} \\
n_9 &= \{l, o, p\} \\
n_{10} &= \{h, p\} \\
n_{11} &= \{i, m\} \\
n_{12} &= \{j, m, n\} \\
n_{13} &= \{k, n, o\}
\end{aligned}
$$



undirected graph model w/ k-clique weighting
thin edges = weight 0.5, thick edges = weight 1

Cut 1 and 2

- First cut has min-cutsize of 3 (not unique)
  - Both cuts 1 and 2 divide the entire chip

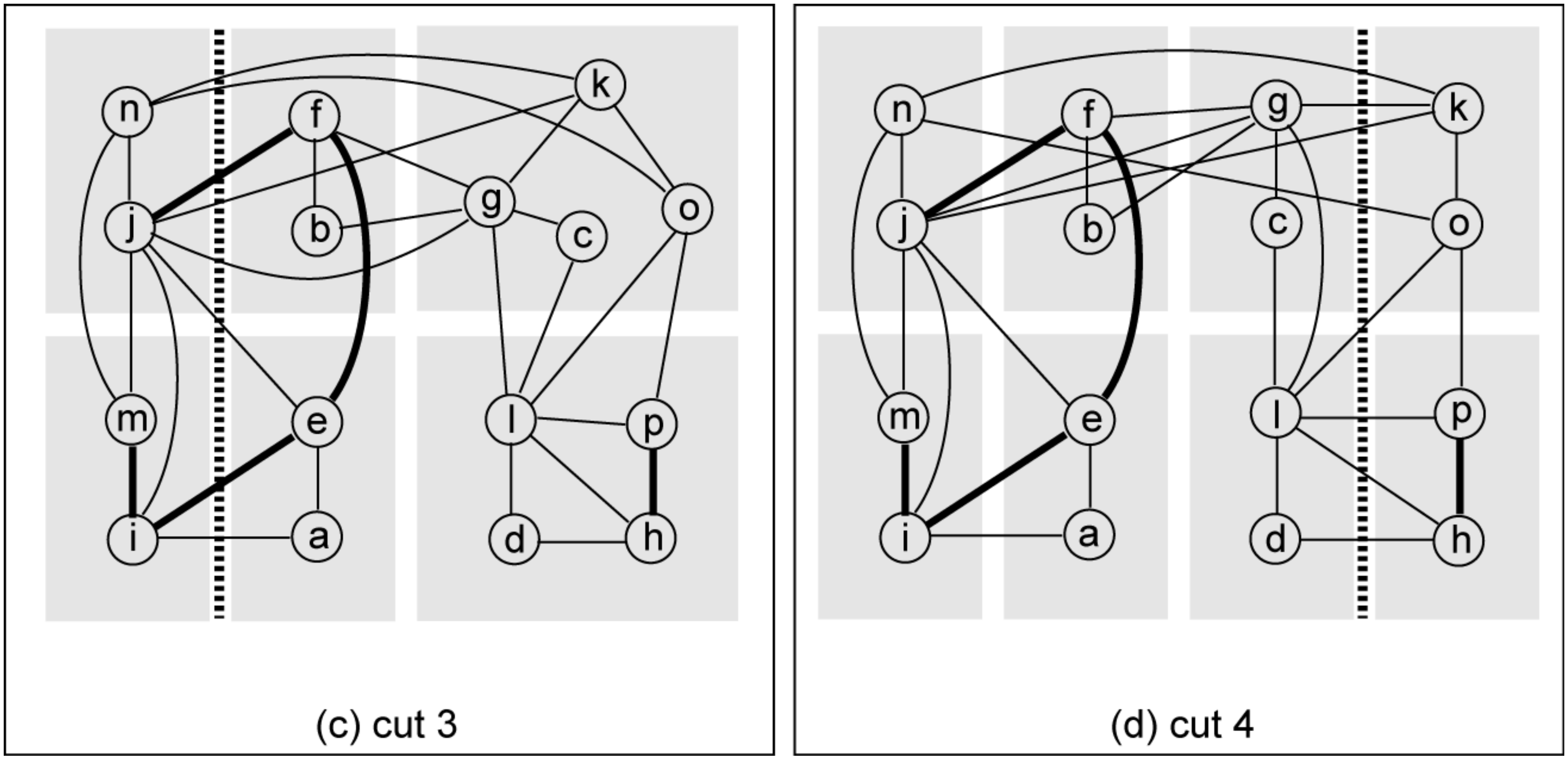


(a) cut 1

(b) cut 2, 1st-level quadrants formed

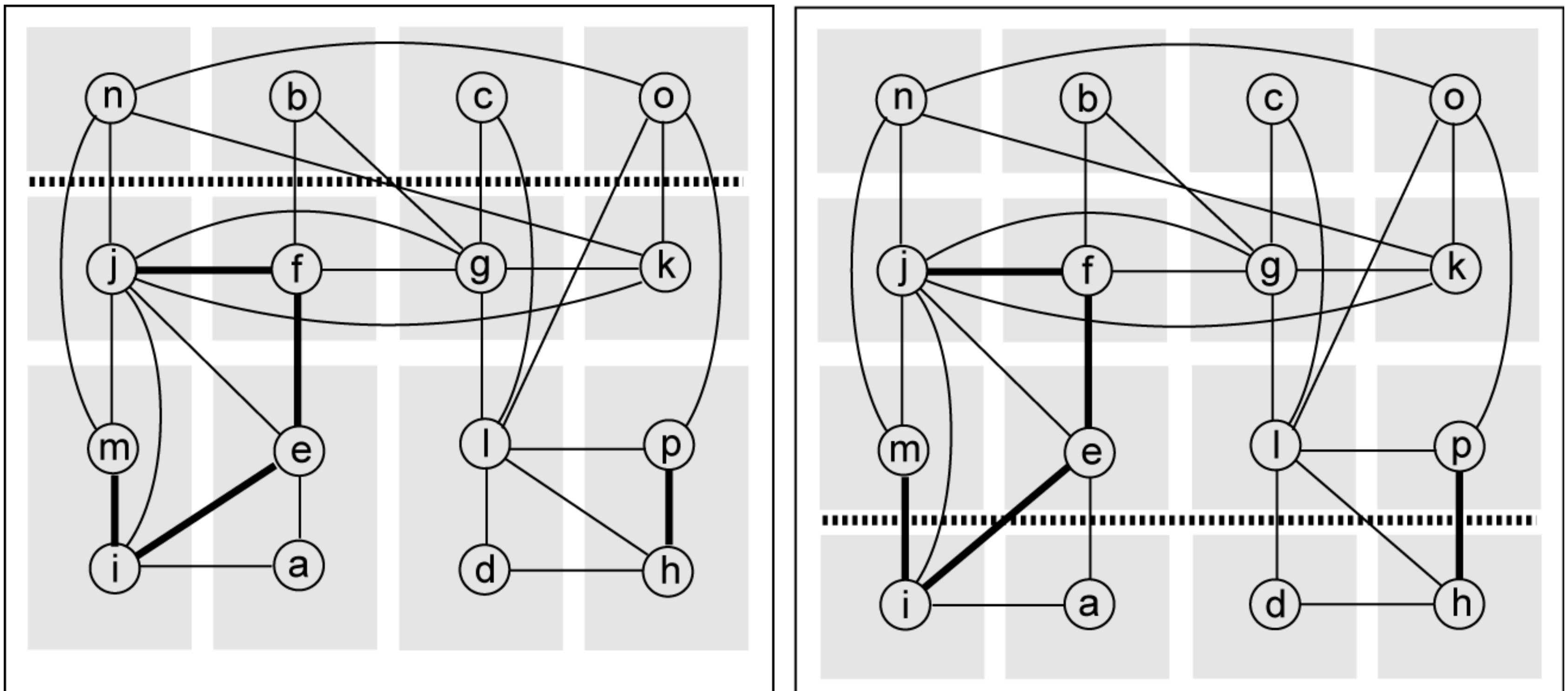
Cut 3 and 4

- Each cut minimizes cutsize
  - Helps reduce overall wirelength



Cut 5 and 6

- 16 partitions generated by 6 cuts
  - HPBB wirelength = 27

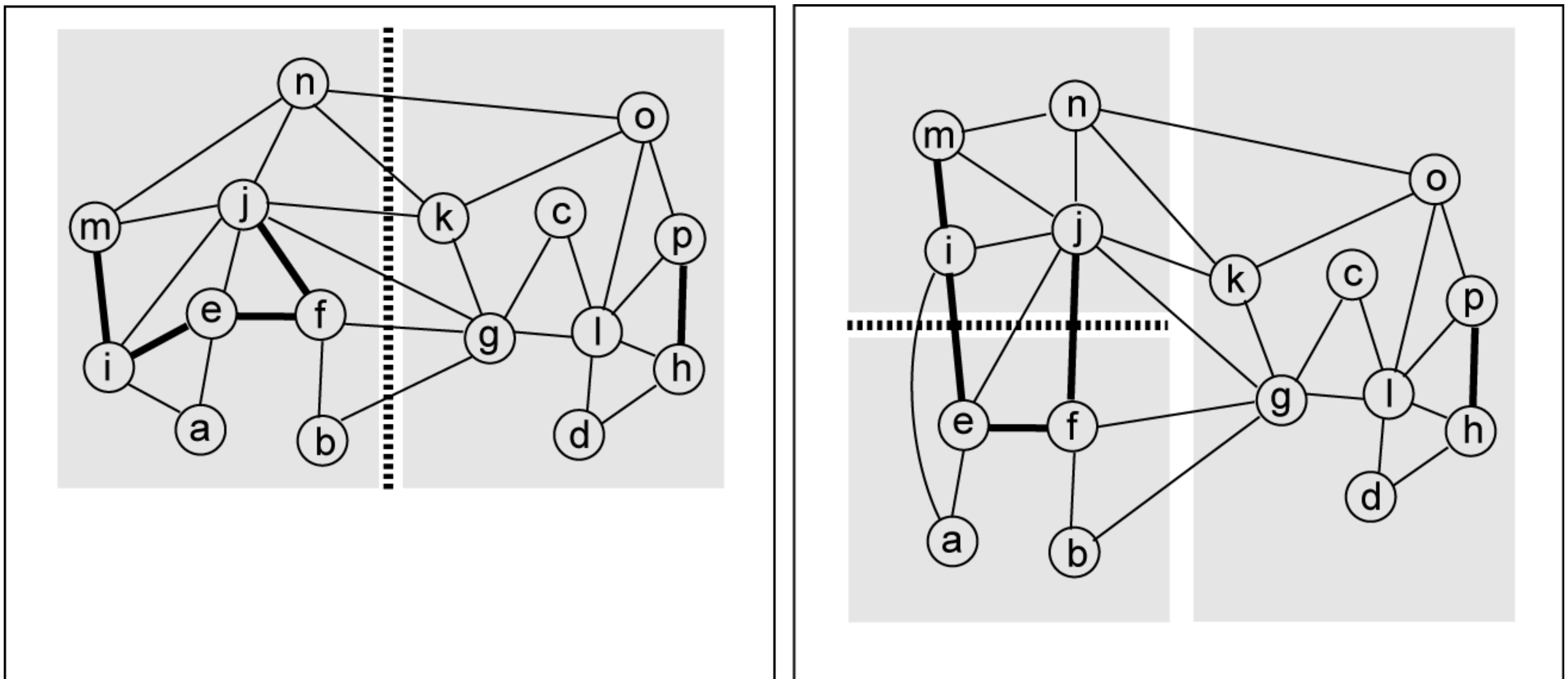


(e) cut 5

(f) cut 6, 2nd-level quadrants formed

Recursive Bisection

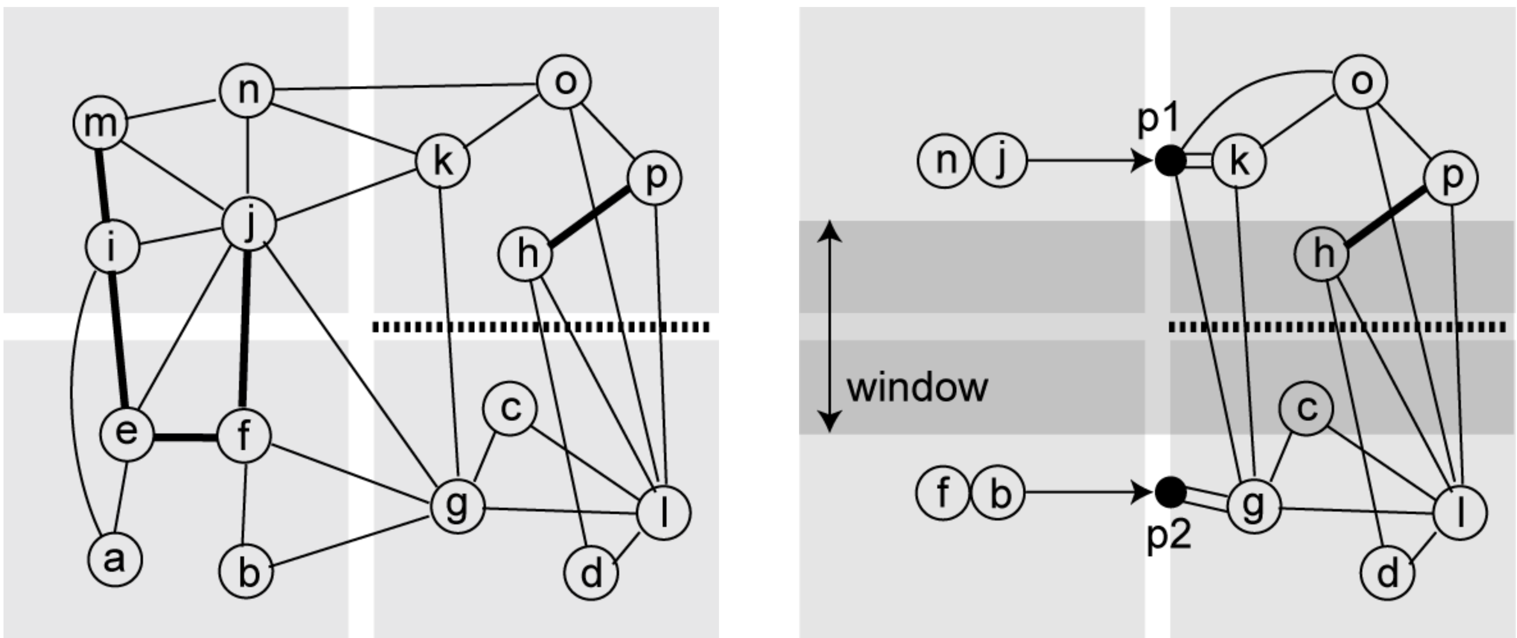
- Start with vertical cut
  - Perform terminal propagation with middle third window



(a) cut 1 (b) cut 2

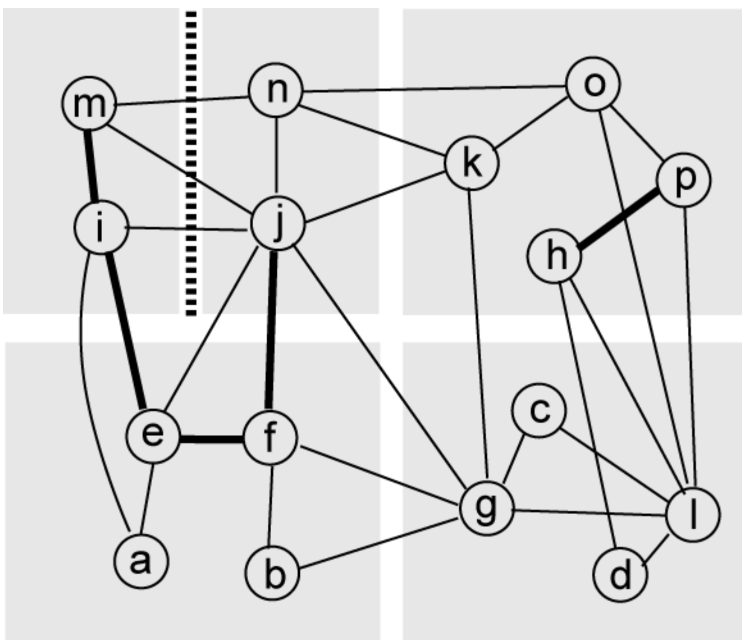
Cut 3: Terminal Propagation

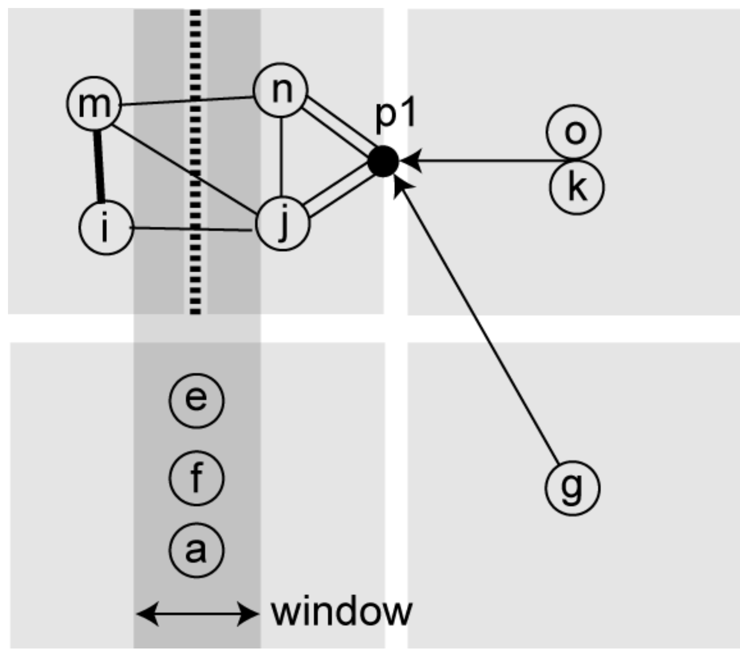
- Two terminals are propagated and are "pulling" nodes
  - Node *k* and *o* connect to *n* and *j*: p_1 propagated (outside window)
  - Node *g* connect to *j*, *f* and *b*: p_2 propagated (outside window)
  - Terminal p_1 pulls *k*/*o*/*g* to top partition, and p_2 pulls *g* to bottom



Cut 4: Terminal Propagation

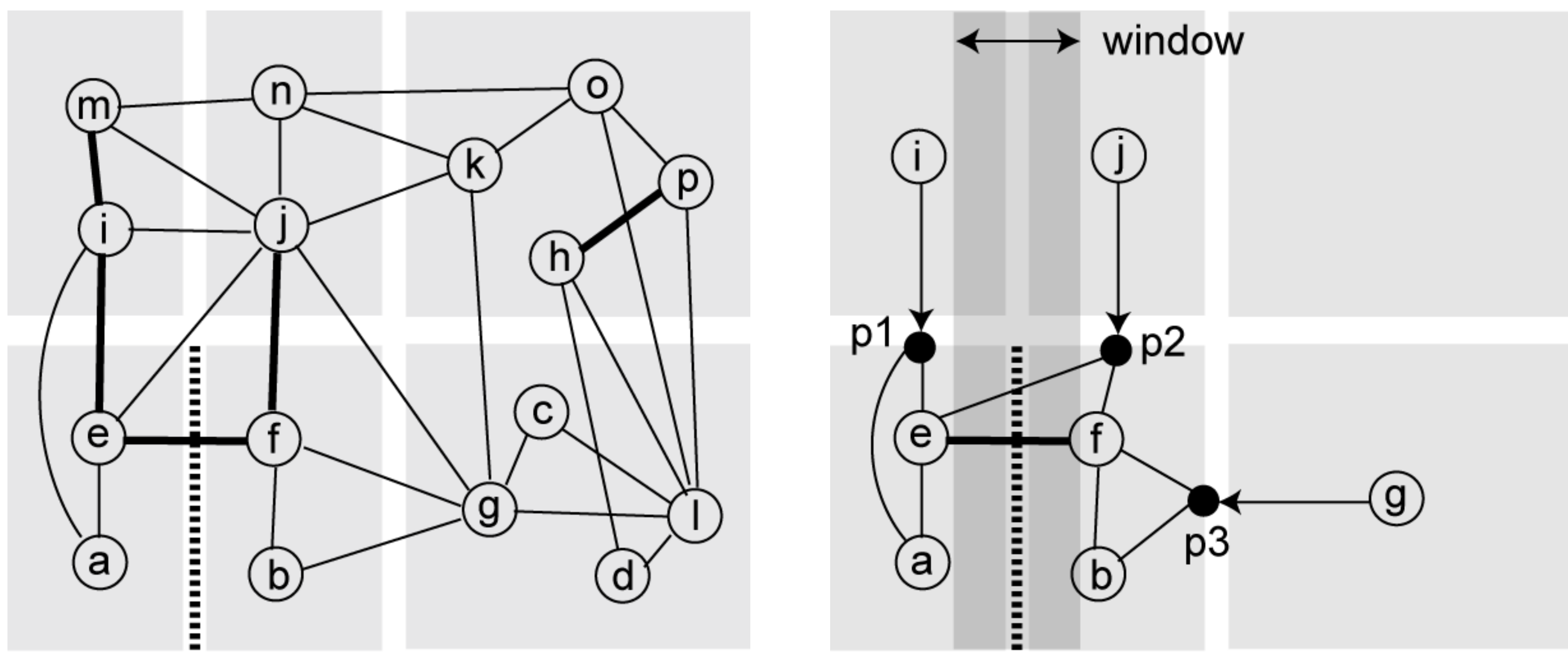
- One terminal propagated
  - Node *n* and *j* connect to *o*/*k*/*g*: p_1 propagated
  - Node *i* and *j* connect to *e*/*f*/*a*: no propagation (inside window)
  - Terminal p_1 pulls *n* and *j* to right partition





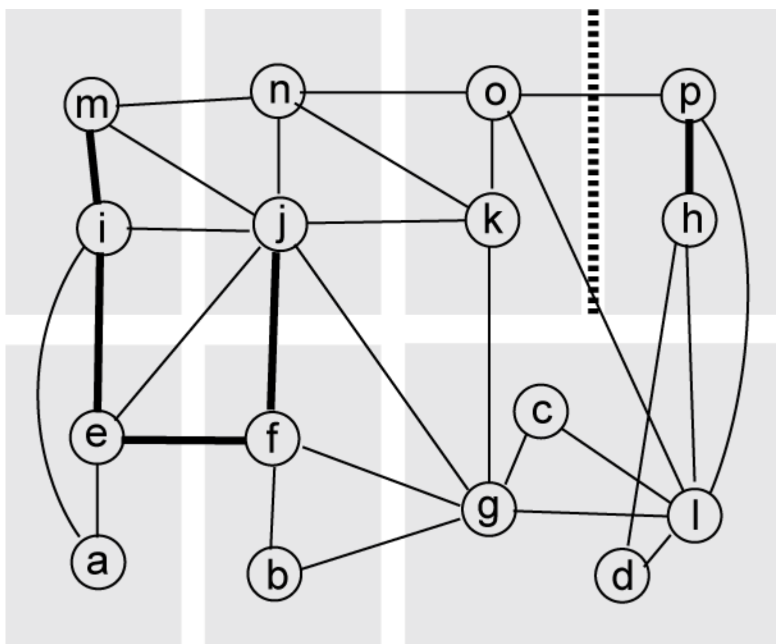
Cut 5: Terminal Propagation

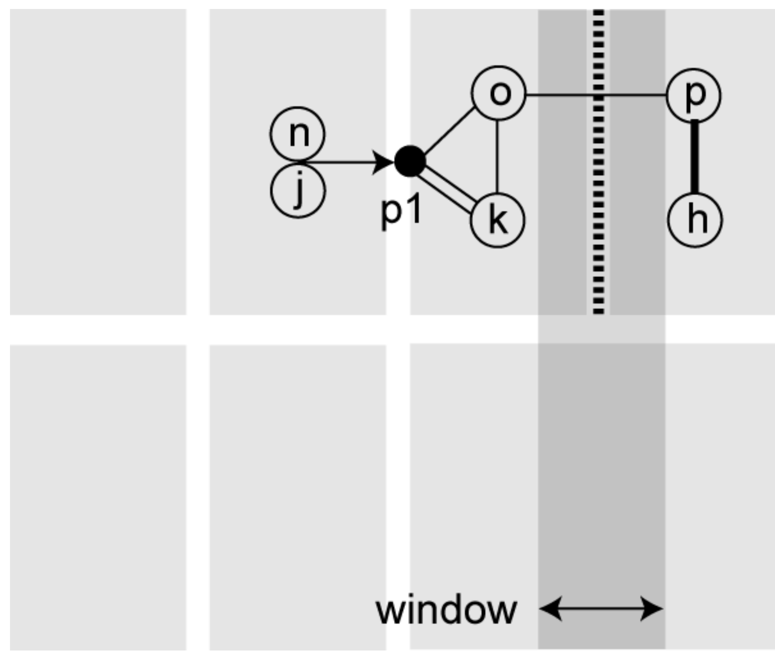
- Three terminals propagated
  - Node i propagated to p_1, j to p_2, and g to p_3
  - Terminal p_1 pulls e and a to left partition
  - Terminal p_2 and p_3 pull $f/b/e$ to right partition



Cut 6: Terminal Propagation

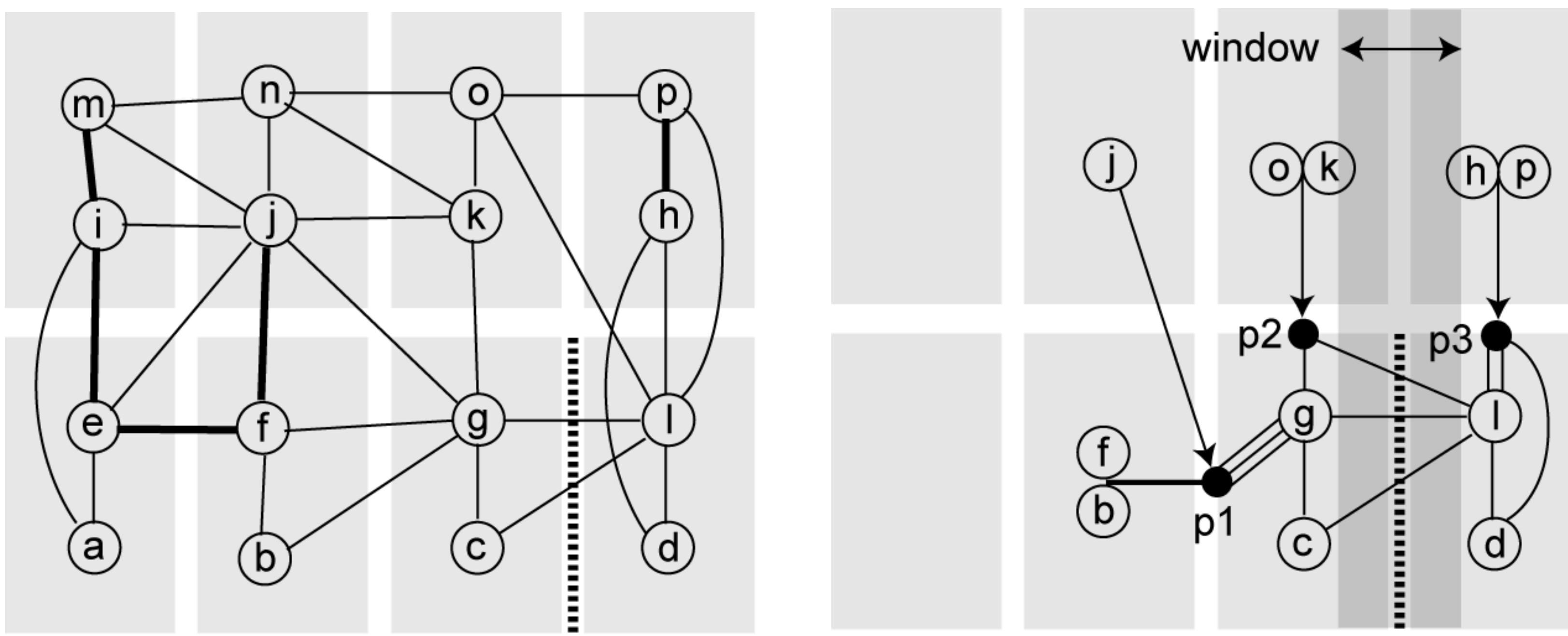
- One terminal propagated
  - Node n and j are propagated to p_1
  - Terminal p_1 pulls o and k to left partition





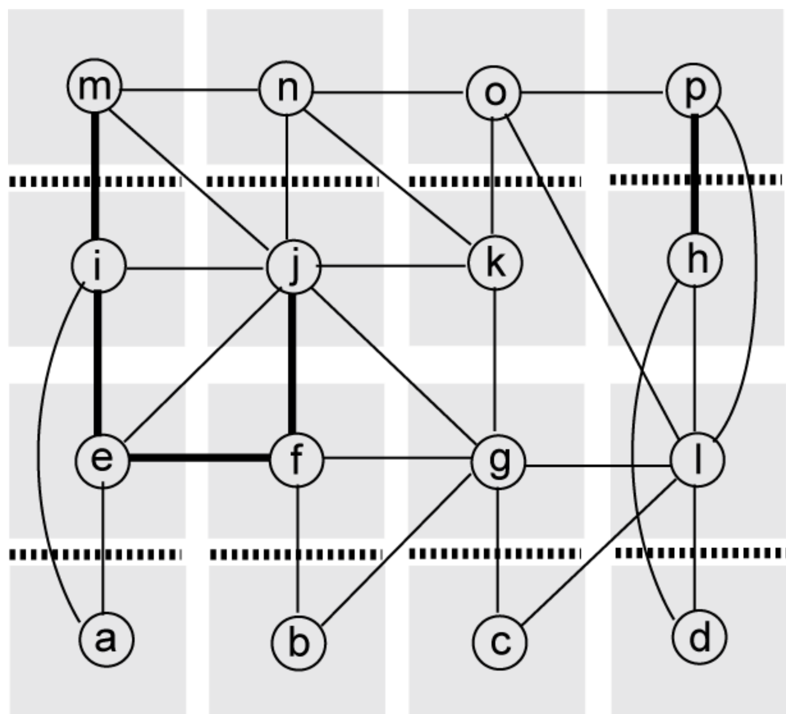
Cut 7: Terminal Propagation

- Three terminals propagated
  - Node $j/f/b$ propagated to p_1, o/k to p_2, and h/p to p_3
  - Terminal p_1 and p_2 pull g and l to left partition
  - Terminal p_3 pull l and d to right partition



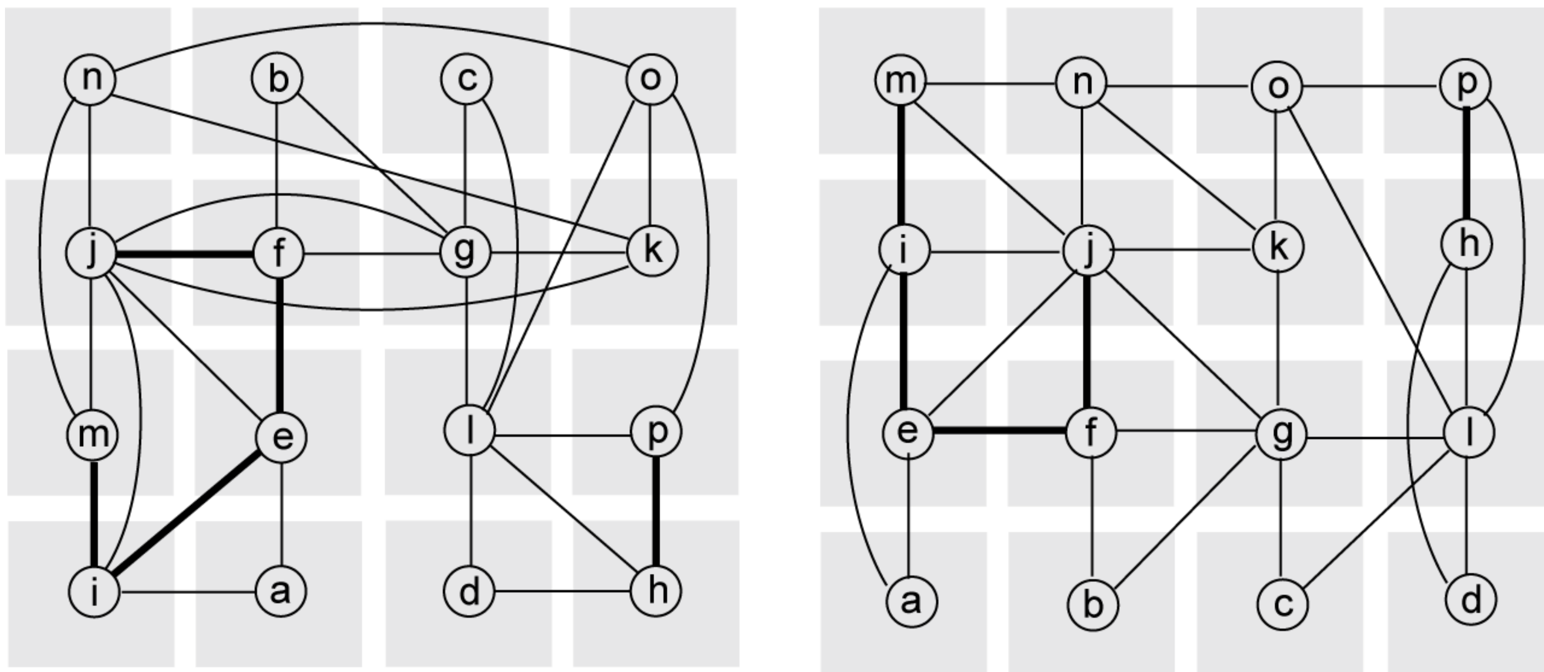
Cut 8 to 15

- 16 partitions generated by 15 cuts
  - HPBB wirelength = 23



Comparison

- Quadrature vs recursive bisection + terminal propagation
  - Number of cuts: 6 vs 15
  - Wirelength: 27 vs 23



quadrature   bisection

Quadratic Programming (QP)

- **Definition**
  - Process of solving optimization problems involving quadratic functions
  - One seeks to optimize (minimize or maximize) a multivariate quadratic function subject to linear constraints on the variables

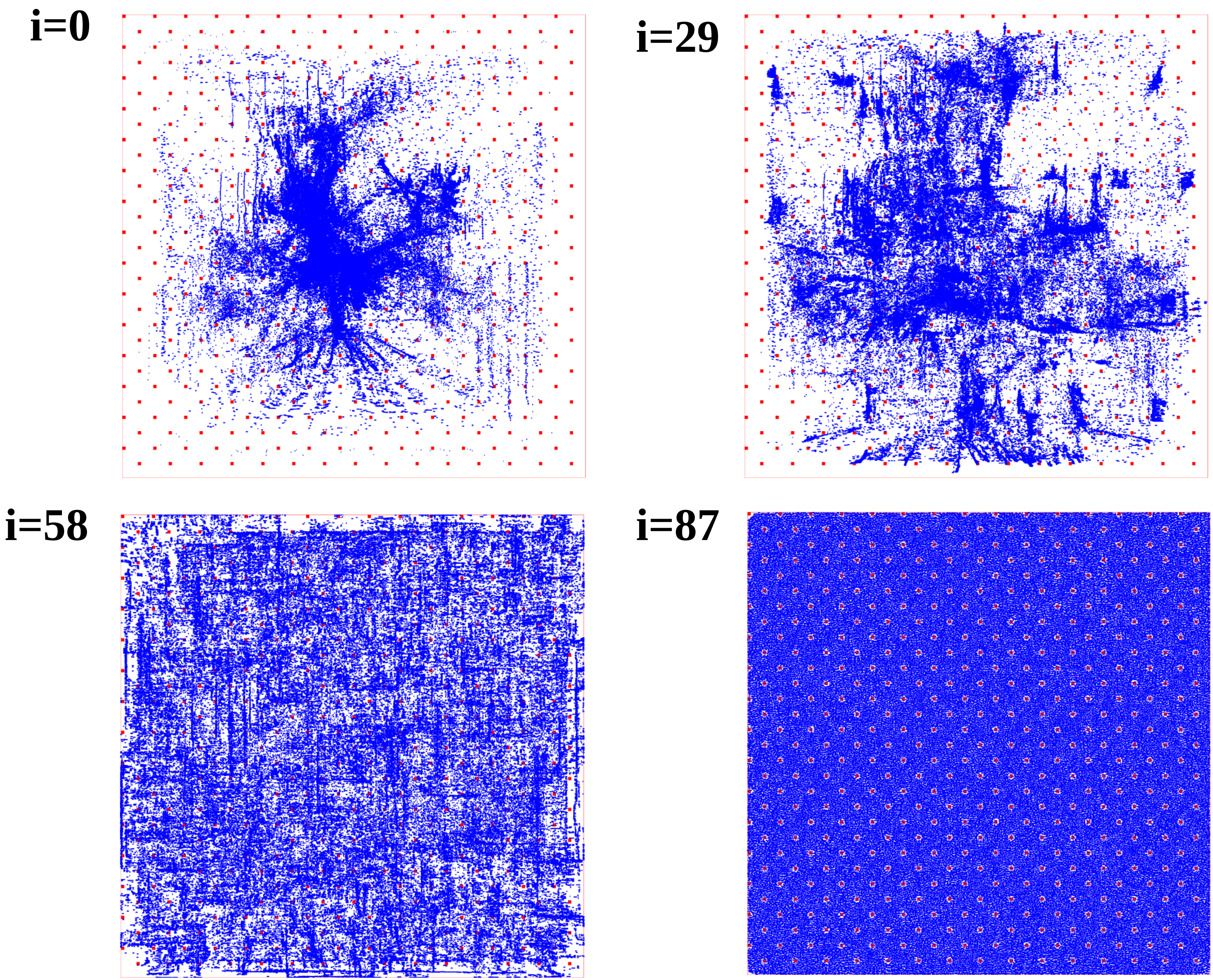
- **QP with n variables and m constraints**

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\mathbf{x}^{\mathrm{T}}Q\mathbf{x} + \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ \text{subject to} & A\mathbf{x} \preceq \mathbf{b}, \end{array}$$

  - n-dimensional vector c
  - n × n-dimensional real symmetric matrix Q
  - m × n-dimensional real matrix A
  - m-dimensional real vector b

Analytical Placement

- **Gordian package:**
  - **GORDIAN: Gordian: VLSI Placement by Quadratic Programming and slicing Optimization: J. M. Kleinhans, G.Sigl, F.M. Johannes, K.J. Antreich, IEEE TCAD, 1991**
  - **GORDIAN-L: Analytical Placement: A Linear or a Quadratic Objective Function?: G. Sigl, K. Doll, F.M. Johannes, DAC91**
- **Gordian: A Quadratic Placement Approach**
  - **Global optimization: solves a sequence of quadratic programming problems**
  - **Partitioning: enforces the non-overlap constraints**


i=0
i=29
i=58
i=87

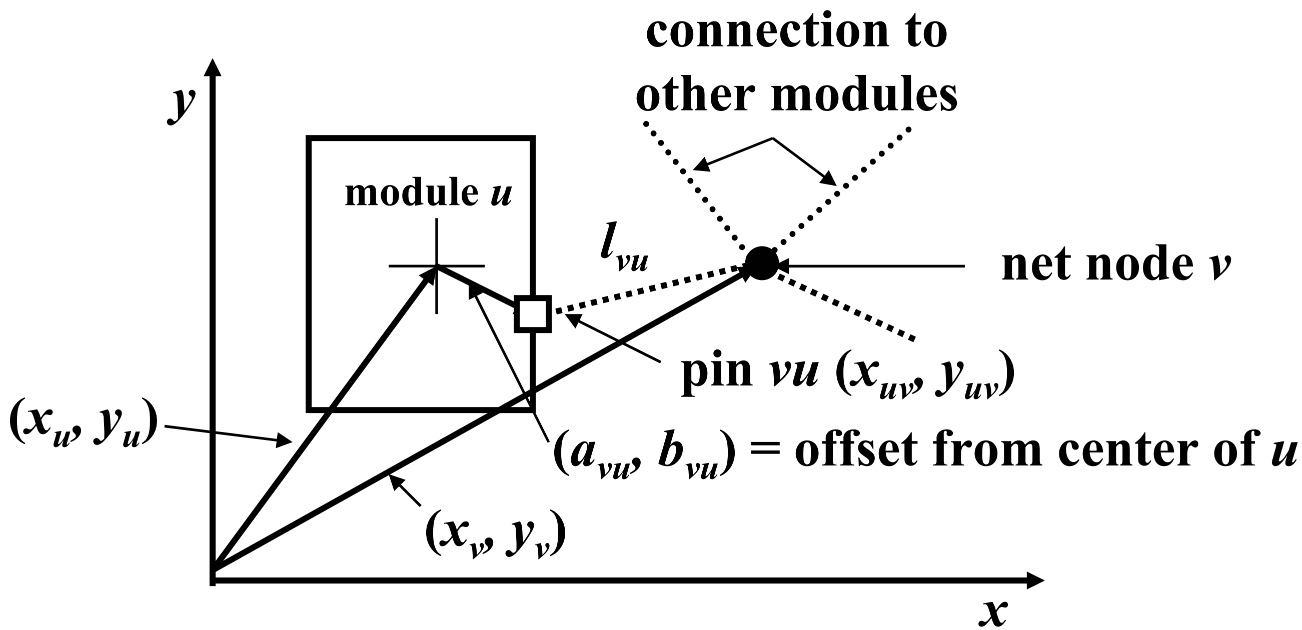
Adaptec1 Stats

- **Circuit stats**
  - **# cells/nets/pins** **210,863/219,687/19,205**
  - **chip size** **6000um × 6000um**
  - **bin size** **50um × 50um**
  - **# placement bins** **120 × 120**
  - **Average bin occupancy** **210K/120^2 =14.6 gates/bin**
- **Wirelength result (HPBB)**
  - **iteration 0** **34,069,060**
  - **iteration 29** **46,352,680**
  - **iteration 58** **80,783,336**
  - **iteration 87** **98,111,904**

Overview of Gordian Package

```
Procedure Gordian
    l:=1;
    global-optimize(l);
    while (there exists |M_l|>k)
        for each r ϵ R(l)
            partition(r, r', r");
        l++;
        setup-constraints(l);
        global-optimize(l);
        repartition(l);
    final-placement(l);
endprocedure
```

Problem Definition



Squared wire length of net v

$$L_v = \sum_{u \in M_v} [(x_{uv} - x_v)^2 + (y_{uv} - y_v)^2]$$

$$x_{uv} = x_u + a_{vu},\ y_{uv} = y_u + b_{vu}$$

Cost Function

- **Minimize the following:**

$$\phi = \frac{1}{2}\sum_{v \in N} L_v w_v$$

$$\phi(x, y) = X^T CX + d_x^T X + Y^T CY + d_y^T Y$$

$$\phi(x) = X^T CX + d^T X$$

Constraints

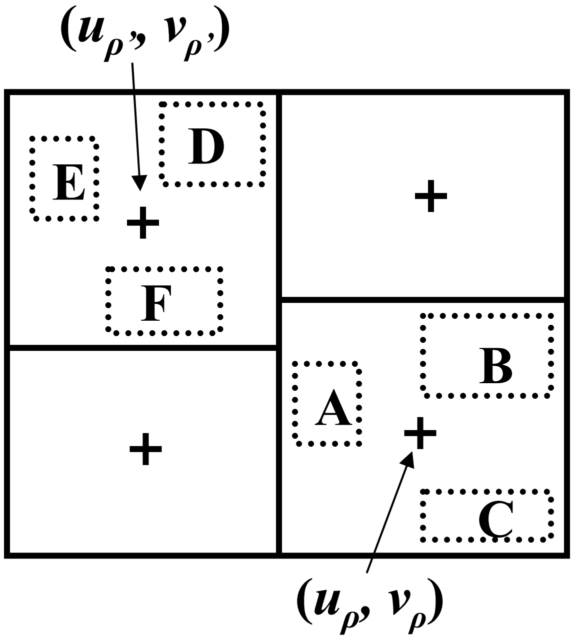
- **The center of gravity constraints**
  - **At level *l*, chip is divided into *q* ($\leq 2^l$) regions**
  - **For region *p*, the center coordinates: (u_p, v_p)**
  - **M_p: set of modules in region *p***
  - **Matrix from for all regions**

$$\sum_{m \in M_p} F_m \cdot x_m = u_p \times \sum_{m \in M_p} F_m$$

  - **Lastly, we have**

$$A^l X = u^l, \text{where } a_{pm} = \begin{cases} F_m \,/ \sum_{m \in M_p} F_m, & \text{if } m \in M_p \\ 0 & \text{otherwise} \end{cases}$$

Problem Formulation



$$
A^{(l)} = \begin{array}{c} \\ \vdots \\ \rho \\ \rho' \\ \vdots \end{array}
\begin{array}{c}
\begin{array}{ccccccc} A & B & C & D & E & F & G \end{array} \\
\left[\begin{array}{ccccccc}
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
* & * & * & 0 & 0 & 0 & \cdots \\
0 & 0 & 0 & * & * & * & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{array}\right]
\end{array}
$$

Linearly constrained Quadratic Programming problem

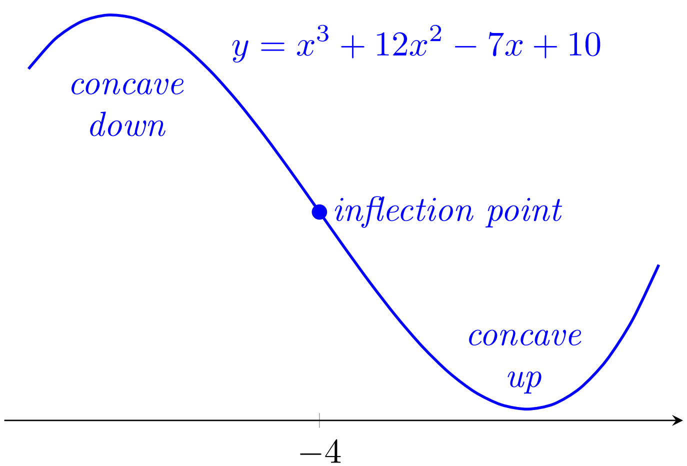
$$\textbf{LQP} : \min_{x \in R^m} \{\Phi(x) = X^T C X + d^T X \text{ such that } A^l X = u^l\}$$

Hessian Matrix

- **Second order partial derivatives of f**
  - **Determine the concavity of the graph of f**
  - **Useful to find local optimal solutions**
  - **Our WL function is quadratic**
    - **Hessian will have constants only**
  - **Laplacian is Hessian!**

$$\mathbf{H} = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1\,\partial x_n} \\[2.2ex] \dfrac{\partial^2 f}{\partial x_2\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2\,\partial x_n} \\[2.2ex] \vdots & \vdots & \ddots & \vdots \\[2.2ex] \dfrac{\partial^2 f}{\partial x_n\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

Hessian matrix



concavity

$$\begin{pmatrix} \frac{25}{6} & -\frac{2}{3} & 0 & 0 & -\frac{7}{6} & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ -\frac{2}{3} & \frac{23}{6} & -\frac{1}{2} & -\frac{1}{2} & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{2} & \frac{25}{6} & -\frac{7}{6} & 0 & -\frac{2}{3} & -\frac{2}{3} & 0 & 0 & 0 \\ 0 & -\frac{1}{2} & -\frac{7}{6} & \frac{23}{6} & 0 & 0 & 0 & 0 & 0 & -1 \\ -\frac{7}{6} & 0 & 0 & 0 & \frac{23}{6} & -\frac{1}{2} & 0 & -1 & 0 & 0 \\ -\frac{1}{2} & -1 & -\frac{2}{3} & 0 & -\frac{1}{2} & \frac{31}{6} & -\frac{2}{3} & -\frac{2}{3} & -\frac{2}{3} & 0 \\ 0 & 0 & -\frac{2}{3} & 0 & 0 & -\frac{2}{3} & \frac{8}{3} & 0 & -\frac{2}{3} & -\frac{2}{3} \\ 0 & 0 & 0 & 0 & -1 & -\frac{2}{3} & 0 & \frac{10}{3} & -\frac{2}{3} & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{2}{3} & -\frac{2}{3} & -\frac{2}{3} & \frac{11}{3} & -\frac{2}{3} \\ 0 & 0 & 0 & -1 & 0 & 0 & -\frac{2}{3} & 0 & -\frac{2}{3} & \frac{10}{3} \end{pmatrix}$$

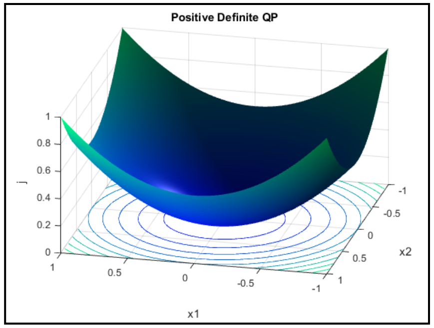
Laplacian

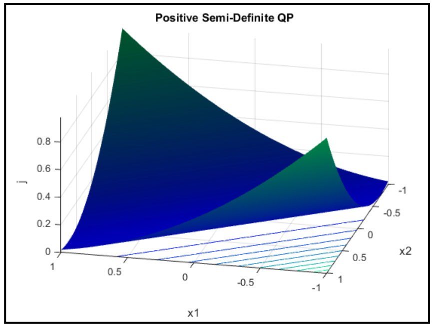
3 Types of Quadratic Programming

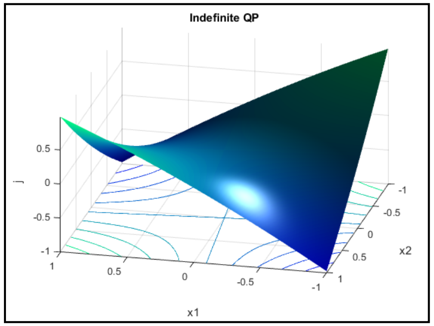
- **Our Gordian QP**

$$\text{LQP:}\ \min_{x \in \Re^m} \left\{ \phi(x) = \tfrac{1}{2} x^T C x + d^T x \,\middle|\, A^{(l)} x = u^{(l)} \right\}.$$

- **3 Types of QP: Depends on C**
  - Positive Definite Hessian Matrix (Bowl)
    - All its eigenvalues are positive
    - One optimal value: Convex
  - Semi-definite Hessian Matrix (Trough)
    - All its eigenvalues are non-negative
    - Line of optimal values: Convex
  - Indefinite Hessian Matrix (Saddle)
    - Optimal is on the boundaries: Non-Convex
    - NP Hard







Gordian Laplacian

- **Our Laplacian C**
  - C is positive definite if C's eigenvalues are nonnegative
  - C is positive definite if $x^T C x$ is positive
  - C is positive definite if C is diagonal and the entries are positive
  - So, C is positive definite

- **So, Gordian QP:**

$$\text{LQP:} \quad \min_{x \in \Re^m} \left\{ \phi(x) = \frac{1}{2} \boldsymbol{x}^T \boldsymbol{C} \boldsymbol{x} + \boldsymbol{d}^T \boldsymbol{x} \,\middle|\, \boldsymbol{A}^{(l)} \boldsymbol{x} = \boldsymbol{u}^{(l)} \right\}. \qquad (7)$$

Since $\phi(\boldsymbol{x})$ is a convex function ($\boldsymbol{C}$ is positive definite) and the linear equality constraints (5) define a convex subspace of $\Re^m$, (7) has a unique global minimum $\phi(\boldsymbol{x}^*)$.

Partitioning

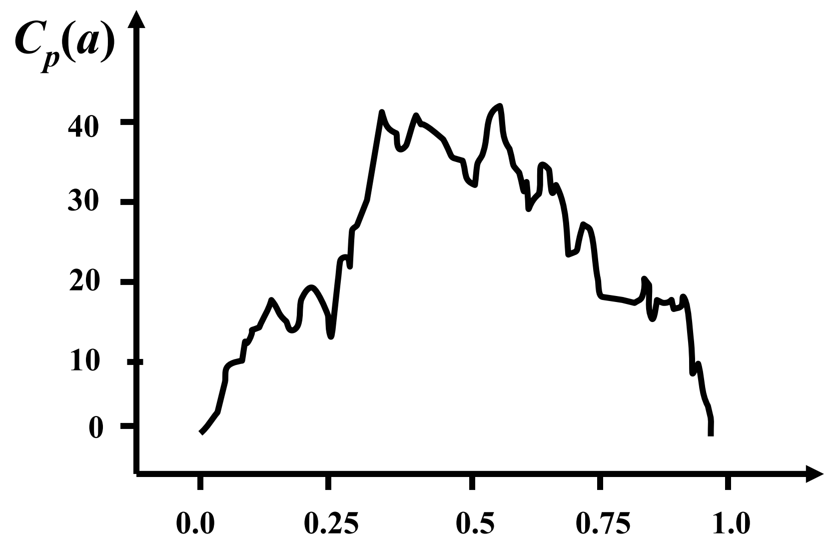
- **Recursive partitioning is needed**
  - to resolve module overlap in global placement
  - global placement problem will be solved again with two additional center_of_gravity constraints

$$M_p \rightarrow (M_{p'}, M_{p''})$$

$$x_{u'} \leq x_{u''} \; u' \in M_{p'} \text{ and } u'' \in M_{p''}$$

$$\alpha = \sum_{u' \in M_{p'}} F_u \Big/ \sum_{u \in M_p} F_u \approx 0.5$$

$$\text{cut value}: C_p(\alpha) = \sum_{v \in N_C} w_v$$

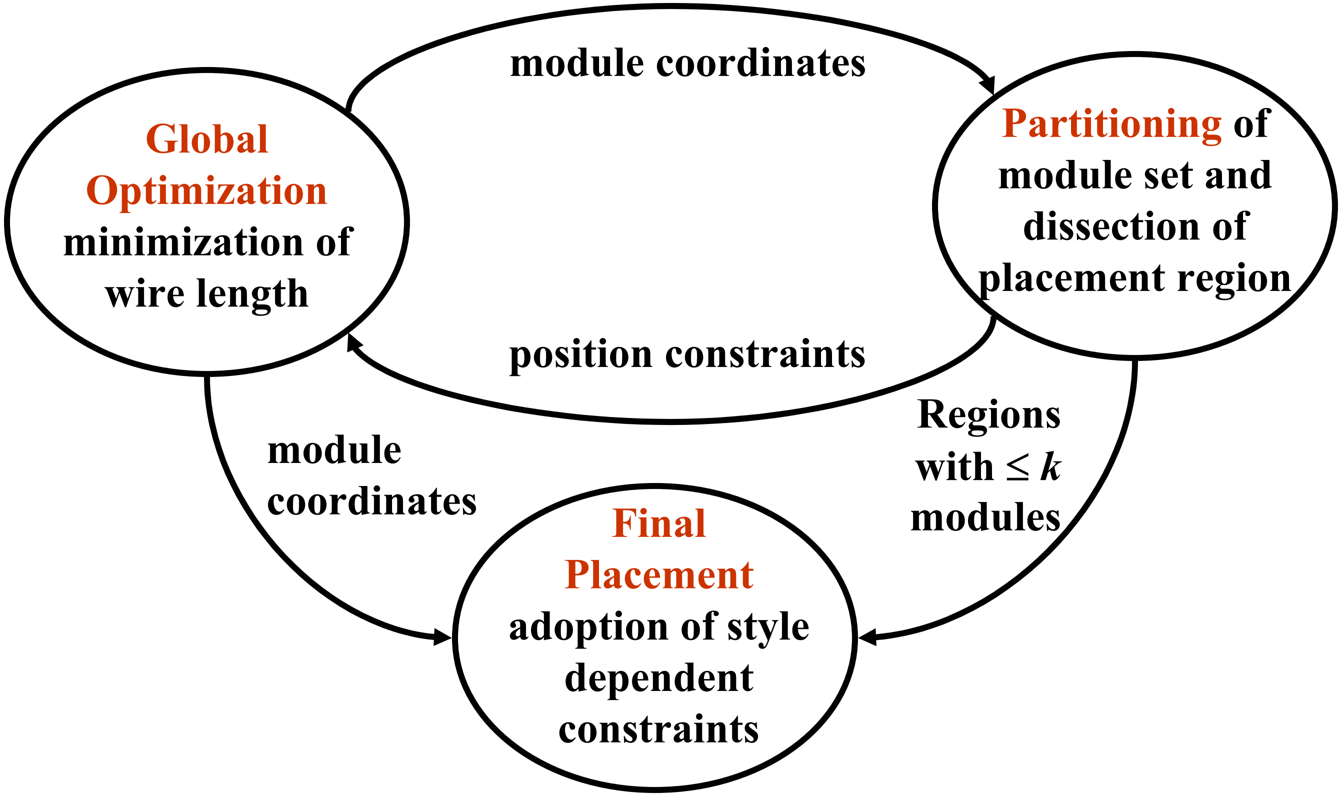


Repartitioning

- **Module exchange after each cut to improve cut size**
  - **terminal propagation using global placement positions**
- **Repartitioning**
  - **to 'undo' the mistake made at the previous level:**

```
Procedure repartition(l)
    if overlap exists
        for each r∈R(l-1)
            merge-regions(r, r', r'');
            partition(r, r', r'');
        setup-constraints(l);
        global-optimize(l);
    endif
```

Summary of Gordian



Complexity: space = O(m), time = O($m^{1.5}\log^2 m$)

Final placement: standard cell, macro-cell & SOG

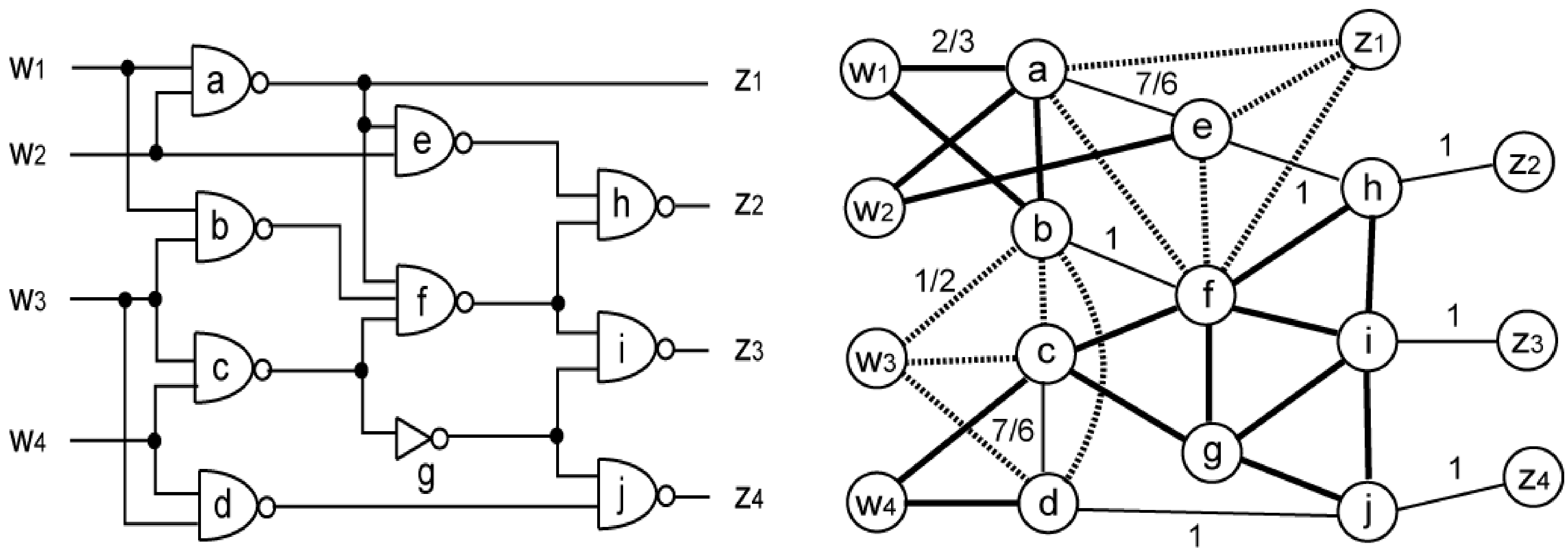
Experimental Results

Comparison of Results for Standard Cell Blocks

	Area After Routing/mm^2		
Circuit	**GORDIAN**	**Min-Cut**	**Annealing**
scb1	2.7	3.1	2.6
scb2	5.8	5.3	5.0
scb3	15.7	25.6	9.1
scb4	14.0	16.9	13.2
scb5	10.6	11.3	10.9
scb6	11.3	12.7	12.8
scb7	16.4	20.2	19.8
scb8	51.7	89.2	59.5
scb9	54.0	98.6	80.0
CPU-time scb8	120s	366s	39851s
CPU-time scb9	135s	440s	34709s
ratio	1	:3	:300

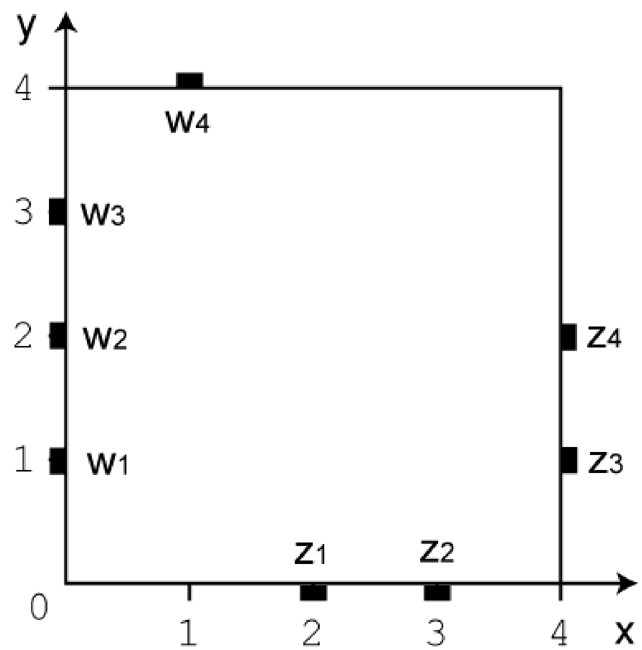
GORDIAN Placement

- Perform GORDIAN placement
  - Uniform area and net weight, area balance factor = 0.5
  - Undirected graph model: each edge in *k*-clique gets weight 2/*k*



IO Placement

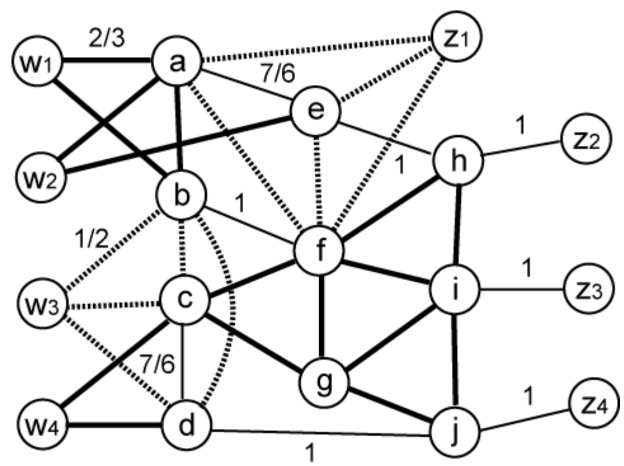
- Necessary for GORDIAN to work



Adjacency Matrix

- Shows connections among movable nodes
  - Among nodes *a* to *j*

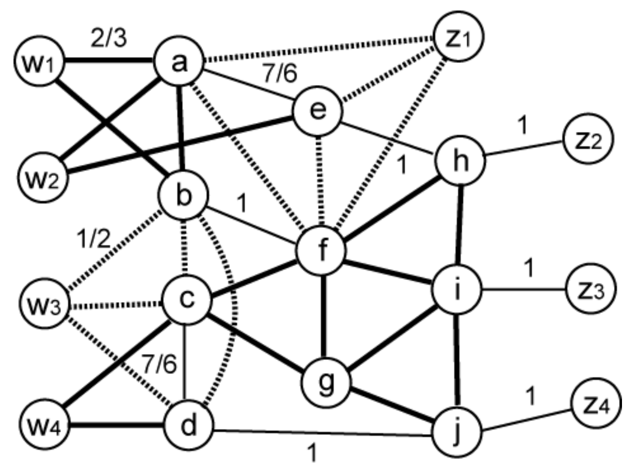
$$
\begin{pmatrix}
0 & \frac{2}{3} & 0 & 0 & \frac{7}{6} & \frac{1}{2} & 0 & 0 & 0 & 0 \\
\frac{2}{3} & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{2} & 0 & \frac{7}{6} & 0 & \frac{2}{3} & \frac{2}{3} & 0 & 0 & 0 \\
0 & \frac{1}{2} & \frac{7}{6} & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\frac{7}{6} & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 1 & 0 & 0 \\
\frac{1}{2} & 1 & \frac{2}{3} & 0 & \frac{1}{2} & 0 & \frac{2}{3} & \frac{2}{3} & \frac{2}{3} & 0 \\
0 & 0 & \frac{2}{3} & 0 & 0 & \frac{2}{3} & 0 & 0 & \frac{2}{3} & \frac{2}{3} \\
0 & 0 & 0 & 0 & 1 & \frac{2}{3} & 0 & 0 & \frac{2}{3} & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{2}{3} & \frac{2}{3} & \frac{2}{3} & 0 & \frac{2}{3} \\
0 & 0 & 0 & 1 & 0 & 0 & \frac{2}{3} & 0 & \frac{2}{3} & 0
\end{pmatrix}
$$



Pin Connection Matrix

- Shows connections between movable nodes and IO
  - Rows = movable nodes, columns = IO (fixed)

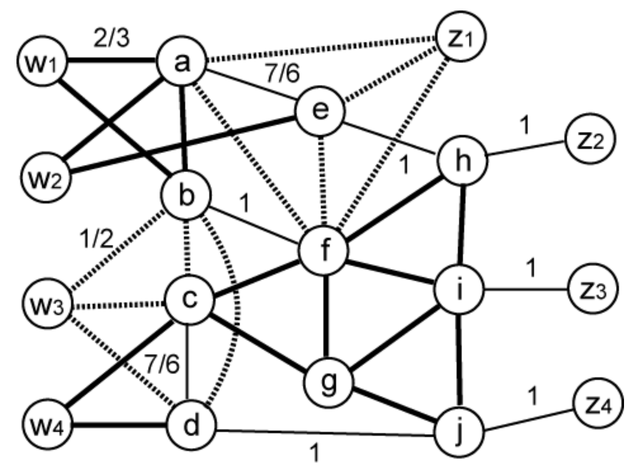
$$
\begin{pmatrix}
\frac{2}{3} & \frac{2}{3} & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\
\frac{2}{3} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{2} & \frac{2}{3} & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{2} & \frac{2}{3} & 0 & 0 & 0 & 0 \\
0 & \frac{2}{3} & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}
$$



Degree Matrix

- Based on both adjacency and pin connection matrices
  - Sum of entries in the same row (= node degree)

$$
\begin{pmatrix}
\frac{25}{6} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{23}{6} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{25}{6} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{23}{6} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{23}{6} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{31}{6} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{8}{3} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{10}{3} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{11}{3} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{10}{3}
\end{pmatrix}
$$



Laplacian Matrix

- Degree matrix minus adjacency matrix

$$\begin{pmatrix}
\frac{25}{6} & -\frac{2}{3} & 0 & 0 & -\frac{7}{6} & -\frac{1}{2} & 0 & 0 & 0 & 0 \\
-\frac{2}{3} & \frac{23}{6} & -\frac{1}{2} & -\frac{1}{2} & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & -\frac{1}{2} & \frac{25}{6} & -\frac{7}{6} & 0 & -\frac{2}{3} & -\frac{2}{3} & 0 & 0 & 0 \\
0 & -\frac{1}{2} & -\frac{7}{6} & \frac{23}{6} & 0 & 0 & 0 & 0 & 0 & -1 \\
-\frac{7}{6} & 0 & 0 & 0 & \frac{23}{6} & -\frac{1}{2} & 0 & -1 & 0 & 0 \\
-\frac{1}{2} & -1 & -\frac{2}{3} & 0 & -\frac{1}{2} & \frac{31}{6} & -\frac{2}{3} & -\frac{2}{3} & -\frac{2}{3} & 0 \\
0 & 0 & -\frac{2}{3} & 0 & 0 & -\frac{2}{3} & \frac{8}{3} & 0 & -\frac{2}{3} & -\frac{2}{3} \\
0 & 0 & 0 & 0 & -1 & -\frac{2}{3} & 0 & \frac{10}{3} & -\frac{2}{3} & 0 \\
0 & 0 & 0 & 0 & 0 & -\frac{2}{3} & -\frac{2}{3} & -\frac{2}{3} & \frac{11}{3} & -\frac{2}{3} \\
0 & 0 & 0 & -1 & 0 & 0 & -\frac{2}{3} & 0 & -\frac{2}{3} & \frac{10}{3}
\end{pmatrix}$$

Fixed Pin Vectors

- Based on pin connection matrix and IO location

  Each entry i in d_x, denoted $d_{x,i}$, is computed as follows:

  $$d_{x,i} = -\sum_j p_{ij} \cdot x(p_j)$$

  where p_{ij} denotes the entry of the pin connection matrix, and $x(p_j)$ is the x-coordinate of the corresponding IO pin j.

  - Y-direction is defined similarly

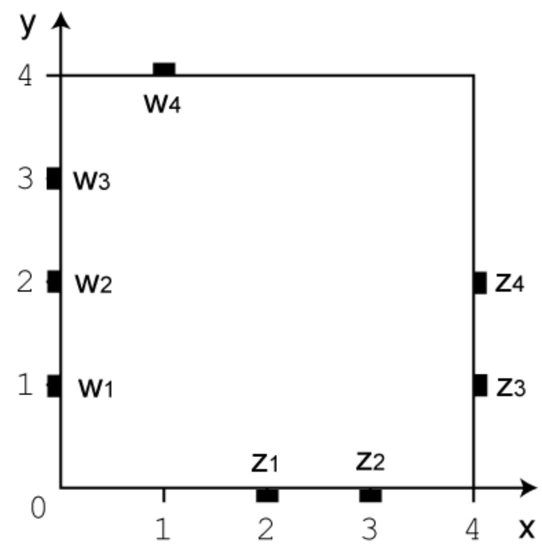
Fixed Pin Vectors (cont)

$$d_{x,1} = -(\frac{2}{3} \cdot 0 + \frac{2}{3} \cdot 0 + 0 \cdot 0 + 0 \cdot 1 + \frac{1}{2} \cdot 2 + 0 \cdot 3 + 0 \cdot 4 + 0 \cdot 4) = -1$$

By examining the remaining 9 movable cells, we get

$$d_x^T = \begin{pmatrix} -1 & 0 & -\frac{2}{3} & -\frac{2}{3} & -1 & -1 & 0 & -3 & -4 & -4 \end{pmatrix}$$

$$\begin{pmatrix} \frac{2}{3} & \frac{2}{3} & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ \frac{2}{3} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{2}{3} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{2}{3} & 0 & 0 & 0 & 0 \\ 0 & \frac{2}{3} & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$



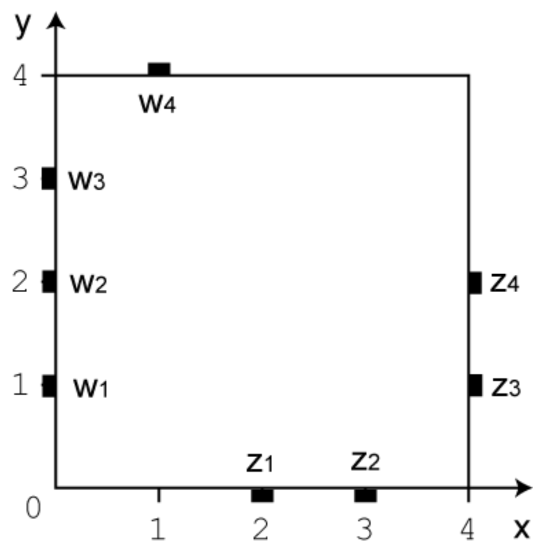
Fixed Pin Vectors (cont)

$$d_{y,1} = -(\frac{2}{3} \cdot 1 + \frac{2}{3} \cdot 2 + 0 \cdot 3 + 0 \cdot 4 + \frac{1}{2} \cdot 0 + 0 \cdot 0 + 0 \cdot 1 + 0 \cdot 2) = -2$$

By examining the remaining 9 movable cells, we get

$$d_y^T = \begin{pmatrix} -2 & -\frac{13}{6} & -\frac{25}{6} & -\frac{25}{6} & -\frac{4}{3} & 0 & 0 & 0 & -1 & -2 \end{pmatrix}$$

$$\begin{pmatrix} \frac{2}{3} & \frac{2}{3} & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ \frac{2}{3} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{2}{3} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{2}{3} & 0 & 0 & 0 & 0 \\ 0 & \frac{2}{3} & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$



Level 0 QP Formulation

- No constraint necessary

Minimize

$$\phi(x) = \frac{1}{2} x^T C x + {d_x}^T x$$

and

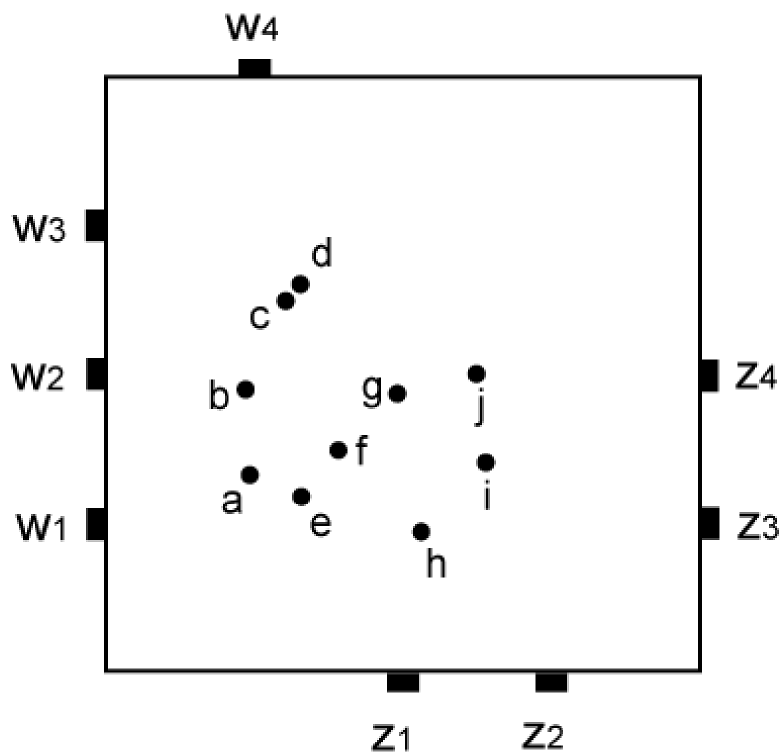
$$\phi(y) = \frac{1}{2} y^T C y + {d_y}^T y$$

We use MOSEK and obtain the following solution:

$$x^T = \begin{pmatrix} 0.95 & 0.92 & 1.21 & 1.32 & 1.32 & 1.61 & 1.98 & 2.13 & 2.59 & 2.51 \end{pmatrix}$$
$$y^T = \begin{pmatrix} 1.27 & 1.83 & 2.48 & 2.61 & 1.16 & 1.45 & 1.84 & 0.92 & 1.41 & 2.03 \end{pmatrix}$$

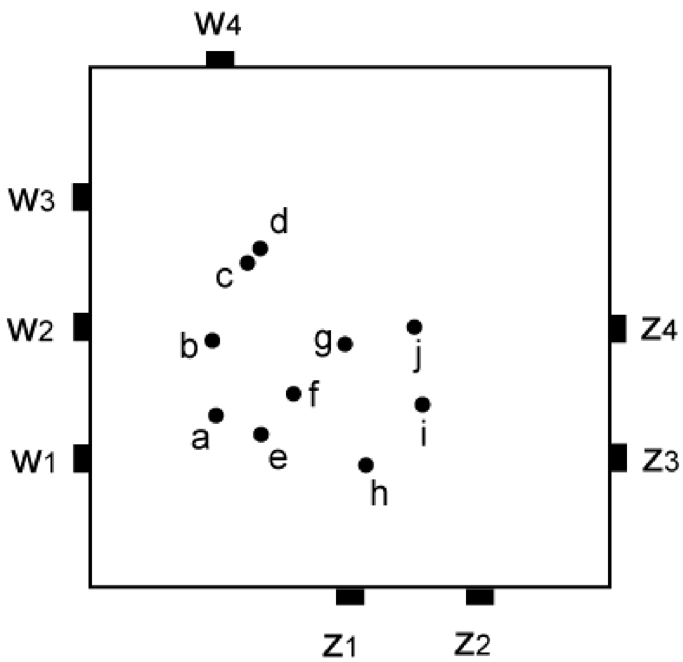
Level 0 Placement

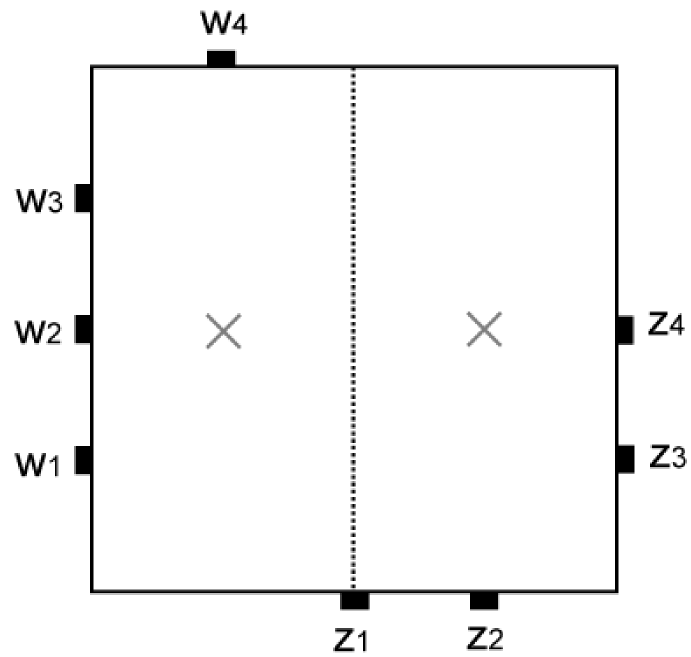
- Cells with real dimension will overlap



Level 1 Partitioning

- Perform level 1 partitioning
  - Obtain center locations for center-of-gravity constraints





Level 1 Constraint

We first sort the nodes based on their x-coordinates:

$$\{b, a, c, e, d, f, g, h, j, i\}$$

We perform partitioning under $\alpha = 0.5$:

$$S_{\rho'} = \{b, a, c, e, d\},\ S_{\rho''} = \{f, g, h, j, i\}$$

The center location vectors are:

$$u_x^{(1)} = \begin{pmatrix} 1 \\ 3 \end{pmatrix},\ u_y^{(1)} = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$$

We build the matrix $A^{(1)}$ for the center-of-gravity constraint at level $l = 1$:

$$A^{(1)} = \begin{pmatrix} \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \end{pmatrix}$$

Level 1 LQP Formulation

We now solve the following Linearly constrained QP (LQP) to obtain the new placement for the movable nodes:

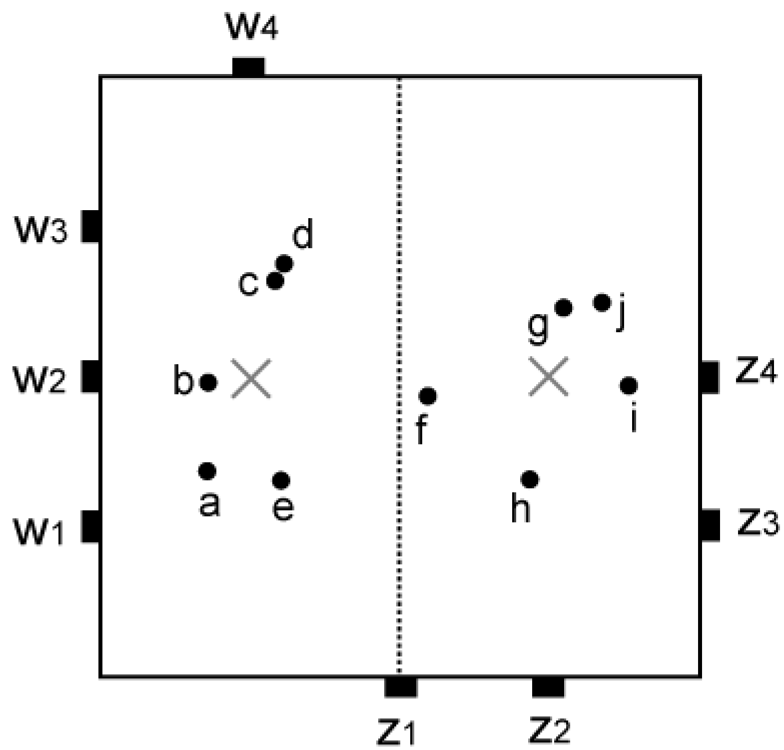
$$\text{Minimize } \phi(x) = \frac{1}{2}x^T C x + {d_x}^T x, \text{ subject to } A^{(1)} \cdot x = u_x^{(1)}$$

$$\text{Minimize } \phi(y) = \frac{1}{2}y^T C y + {d_y}^T y, \text{ subject to } A^{(1)} \cdot y = u_y^{(1)}$$

The solutions are as follows:

$$x^T = \begin{pmatrix} 0.70 & 0.71 & 1.17 & 1.21 & 1.22 & 2.17 & 3.10 & 2.84 & 3.56 & 3.33 \end{pmatrix}$$
$$y^T = \begin{pmatrix} 1.34 & 1.94 & 2.66 & 2.76 & 1.30 & 1.83 & 2.45 & 1.32 & 1.91 & 2.49 \end{pmatrix}$$

Level 1 Placement



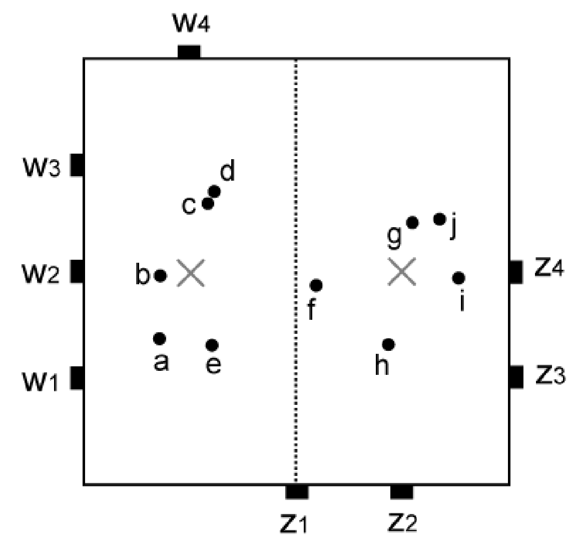
Verification

- Verify that the constraints are satisfied in the left partition

The following cells are partitioned to the left: $a(0.70, 1.34)$, $b(0.71, 1.94)$, $c(1.17, 2.66)$, $d(1.21, 2.76)$, and $e(1.22, 1.30)$. Thus, the center of gravity is located at:

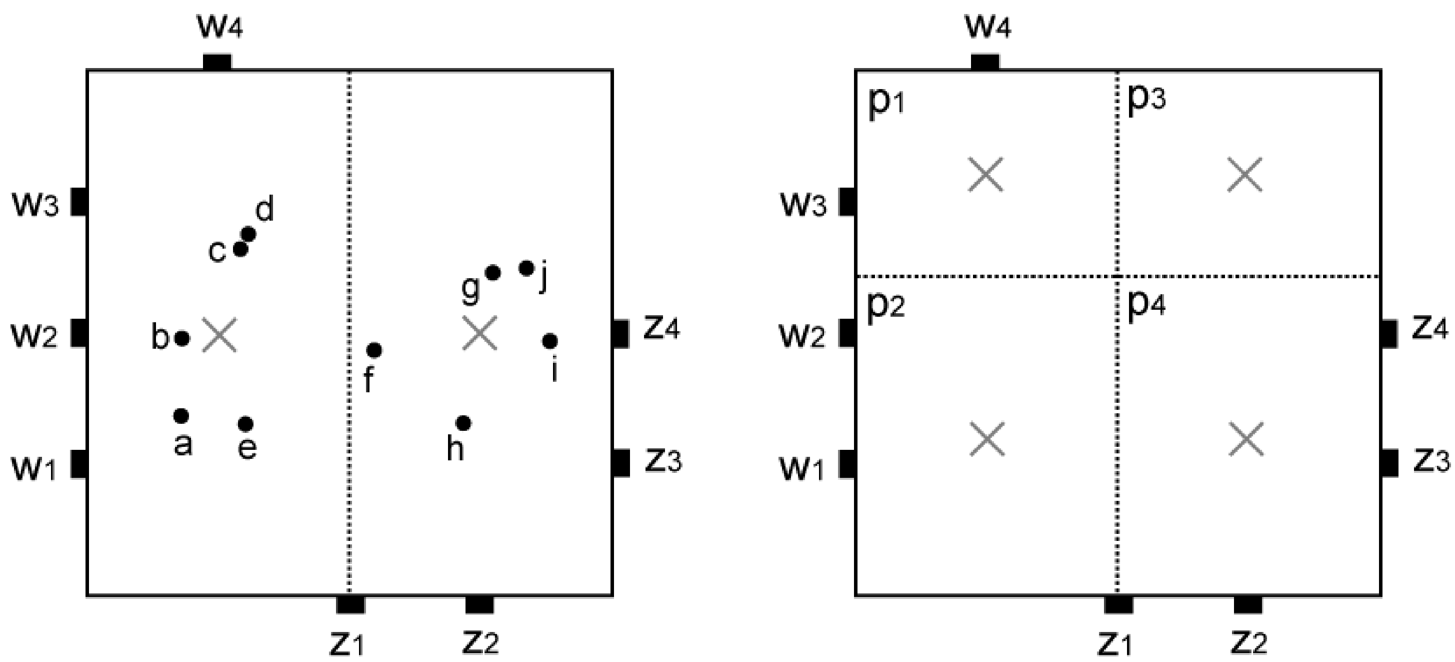
$$\frac{0.70 + 0.71 + 1.17 + 1.21 + 1.22}{5} = 1.00$$
$$\frac{1.34 + 1.94 + 2.66 + 2.76 + 1.30}{5} = 2.00$$

This agrees with the center location $(1, 2)$.



Level 2 Partitioning

- Add two more cut-lines
  - This results in p_1={c,d}, p_2={a,b,e}, p_3={g,j}, p_4={f,h,i}



chip height is 4
we split 5 cells into 2:3 ratio

Level 2 Constraint

The center location vectors are:

$$u_x^{(2)} = \begin{pmatrix} 1 \\ 1 \\ 3 \\ 3 \end{pmatrix}, \; u_y^{(2)} = \begin{pmatrix} 3.2 \\ 1.2 \\ 3.2 \\ 1.2 \end{pmatrix}$$

Next, we build the matrix $A^{(2)}$ for the center-of-gravity constraint at level $l = 2$. Recall that $p_1 = \{c, d\}$, $p_2 = \{a, b, e\}$, $p_3 = \{g, j\}$, $p_4 = \{f, h, i\}$. Thus,

$$A^{(2)} = \begin{pmatrix} 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} & 0 \end{pmatrix}$$

where the rows denote the partitions p_1 through p_4, and the columns denote the cells a through j.

Level 2 LQP Formulation

We now solve the following LQP to obtain the placement of the movable nodes:

$$\text{Minimize } \phi(x) = \frac{1}{2} x^T C x + {d_x}^T x, \text{ subject to } A^{(2)} \cdot x = u_x^{(2)}$$

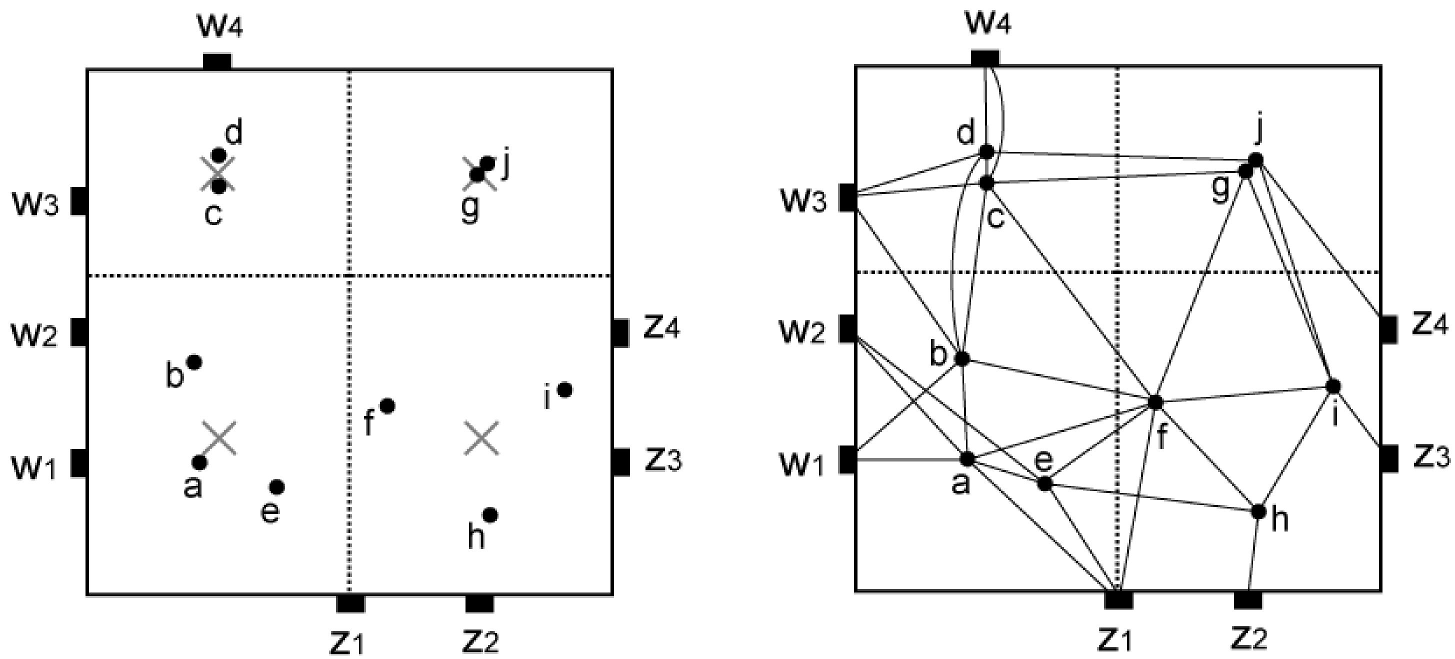
$$\text{Minimize } \phi(y) = \frac{1}{2} y^T C y + {d_y}^T y, \text{ subject to } A^{(2)} \cdot y = u_y^{(2)}$$

The solutions are as follows:

$$x^T = \begin{pmatrix} 0.83 & 0.78 & 1.00 & 1.00 & 1.39 & 2.28 & 2.89 & 3.06 & 3.66 & 3.11 \end{pmatrix}$$
$$y^T = \begin{pmatrix} 1.01 & 1.78 & 3.08 & 3.32 & 0.82 & 1.44 & 3.18 & 0.59 & 1.57 & 3.22 \end{pmatrix}$$

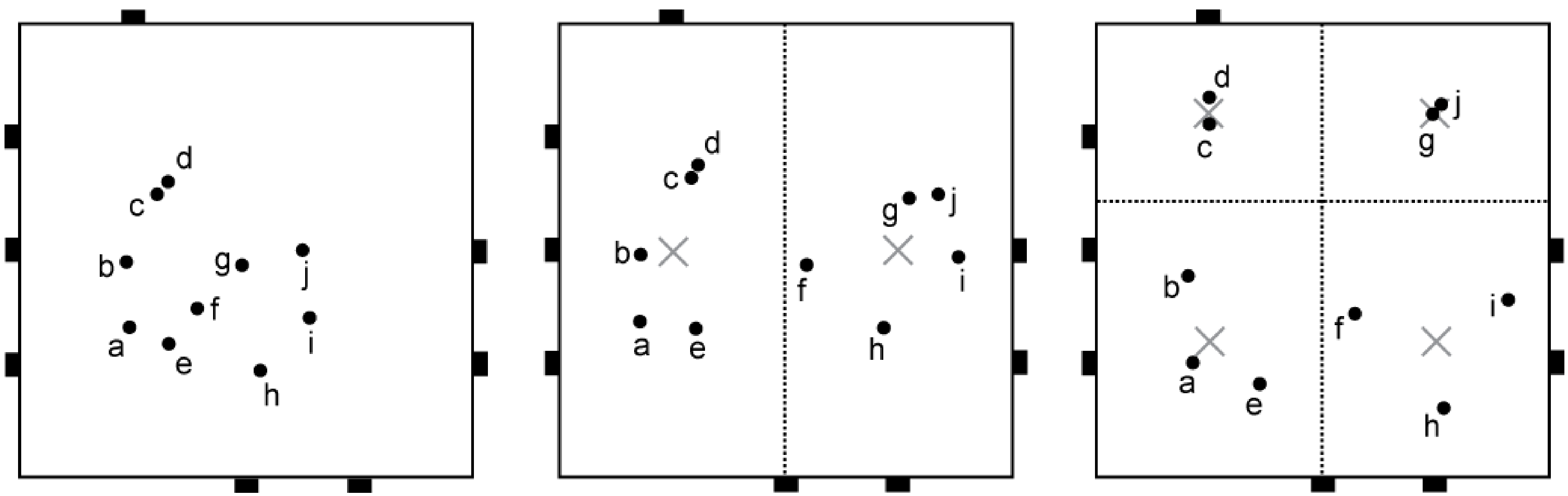
Level 2 Placement

- Clique-based wiring is shown

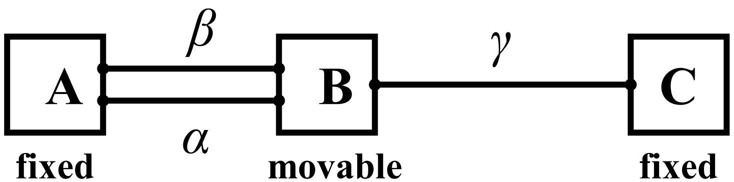


Summary

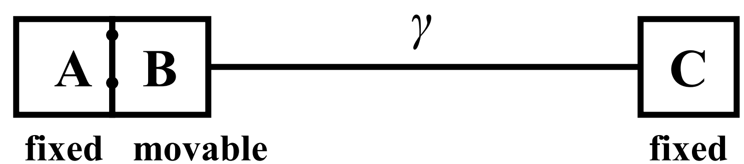
- Center-of-gravity constraint
  - Helps spread the cells evenly while monitoring wirelength
  - Removes overlaps among the cells (with real dimension)



Linear vs. Quadratic Objective



Quadratic objective function



Linear objective function

Quadratic:

$$\varphi_q = l_\alpha^2 + l_\beta^2 + l_\gamma^2 = 2(l - l_\gamma)^2 + l_\gamma^2$$

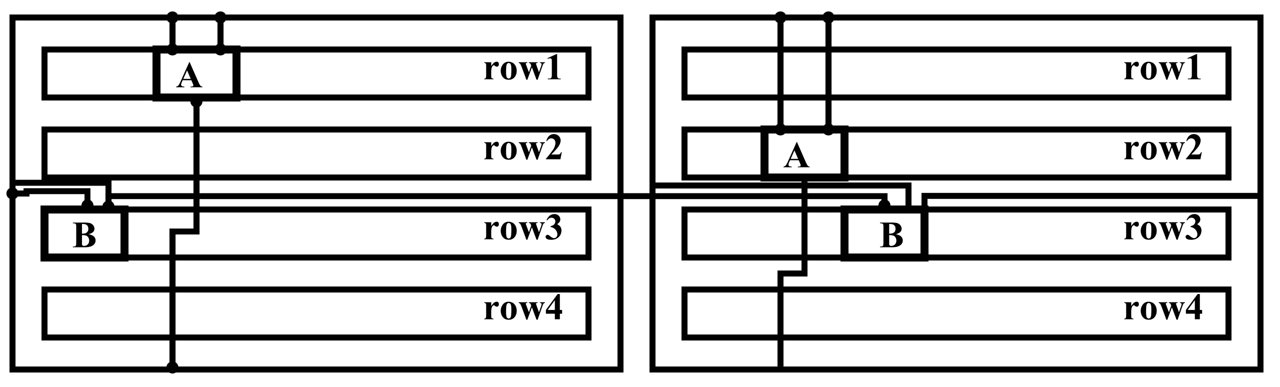
$$\varphi_q' = -4\left(l - l_\gamma\right) + 2l_\gamma = 0, \text{So the optimal } l_\gamma = \frac{2}{3}l$$

Linear:

$$\varphi_l = l_\alpha + l_\beta + l_\gamma, \text{So the optimal } l_\gamma = l$$

Linear vs. Quadratic Objective

- **Quadratic objective function**
  - tends to make very long net shorter than linear objective function
  - lets short nets become slightly longer



Linear objective function　　Quadratic objective function

Optimizing Linear Objective

- **Global Placement with linear objective function**

$$\phi_q = \sum_{v \in N} \sum_{u \in M_v} (x_{uv} - x_v)^2 \rightarrow \textbf{quadratic objective function}$$

$$\phi_l = \sum_{v \in N} \sum_{u \in M_v} \left| x_{uv} - x_v \right| \quad \rightarrow \textbf{linear objective function}$$

- **Trick**
  - **use quadratic programming to minimize linear objective function**

$$\phi_l = \sum_{v \in N} \sum_{u \in M_v} \frac{(x_{uv} - x_v)^2}{\left| x_{uv} - x_v \right|} = \sum_{v \in N} \sum_{u \in M_v} \frac{(x_{uv} - x_v)^2}{g_{uv}}$$

$$g_{uv} = \left| x_{uv} - x_v \right|, \; g_v = \sum_{u \in M_v} \left| x_{uv} - x_v \right|$$

Analytical Placement Results

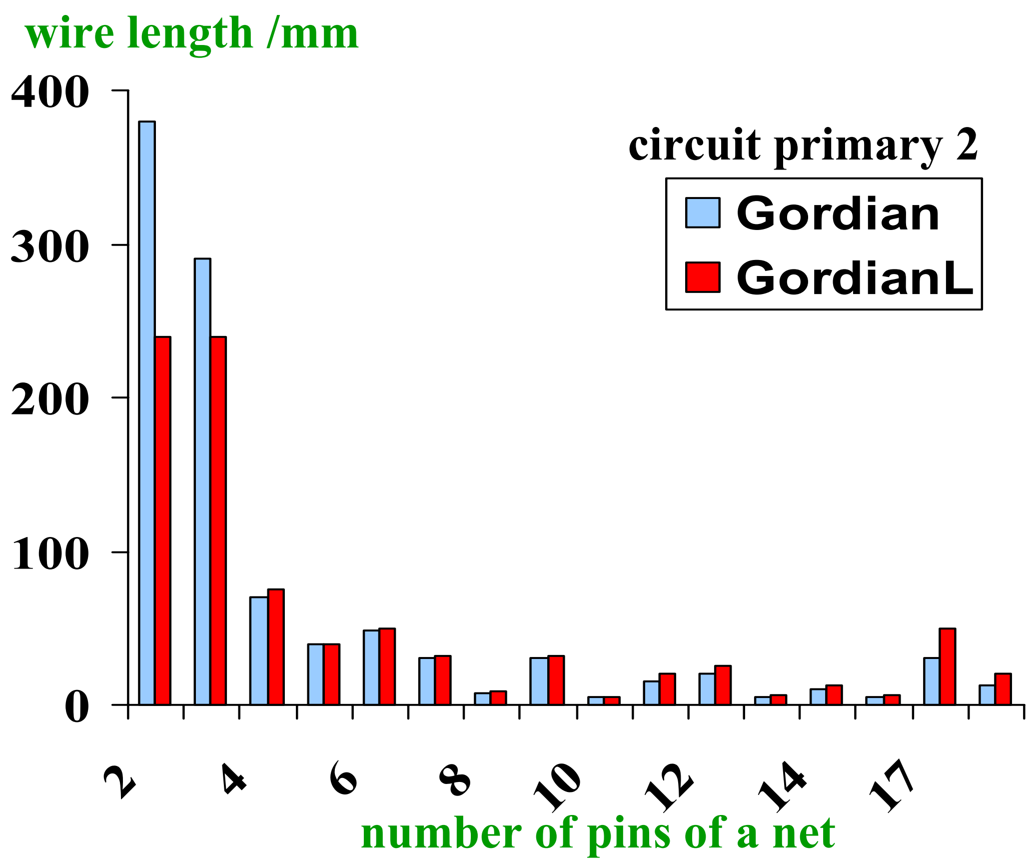
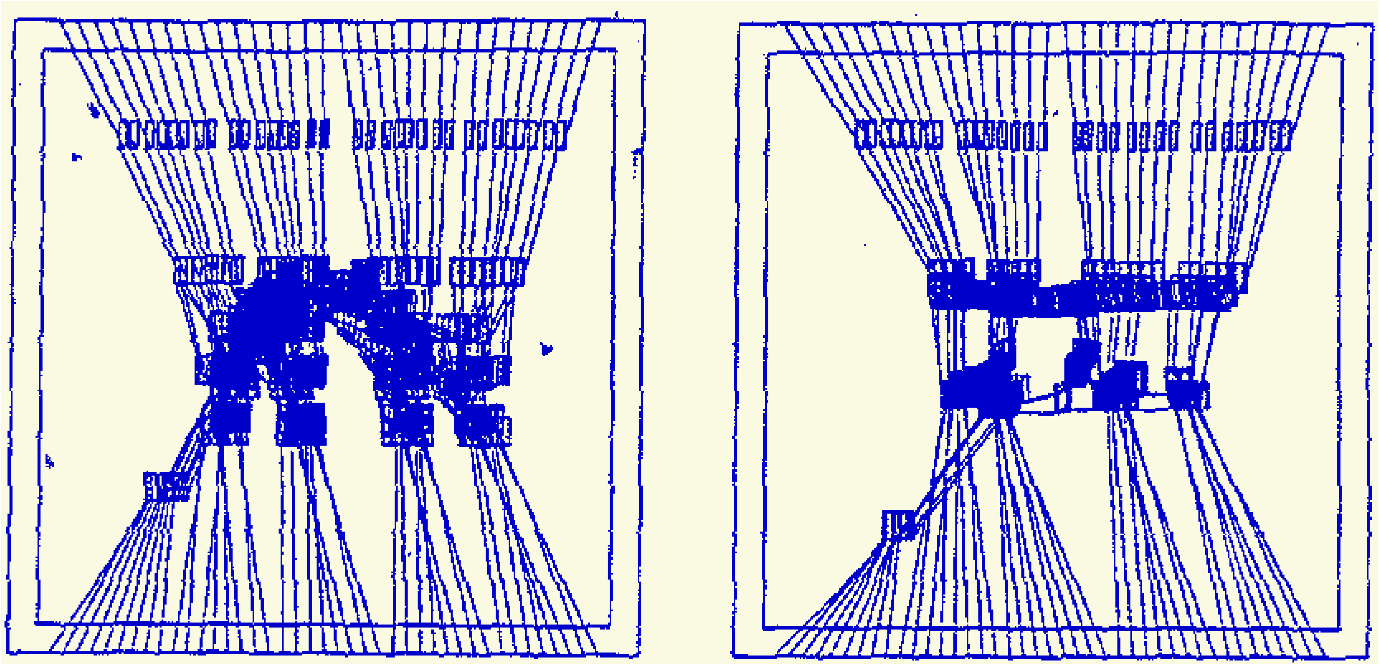


Figure: Sum of wire lengths versus #pins

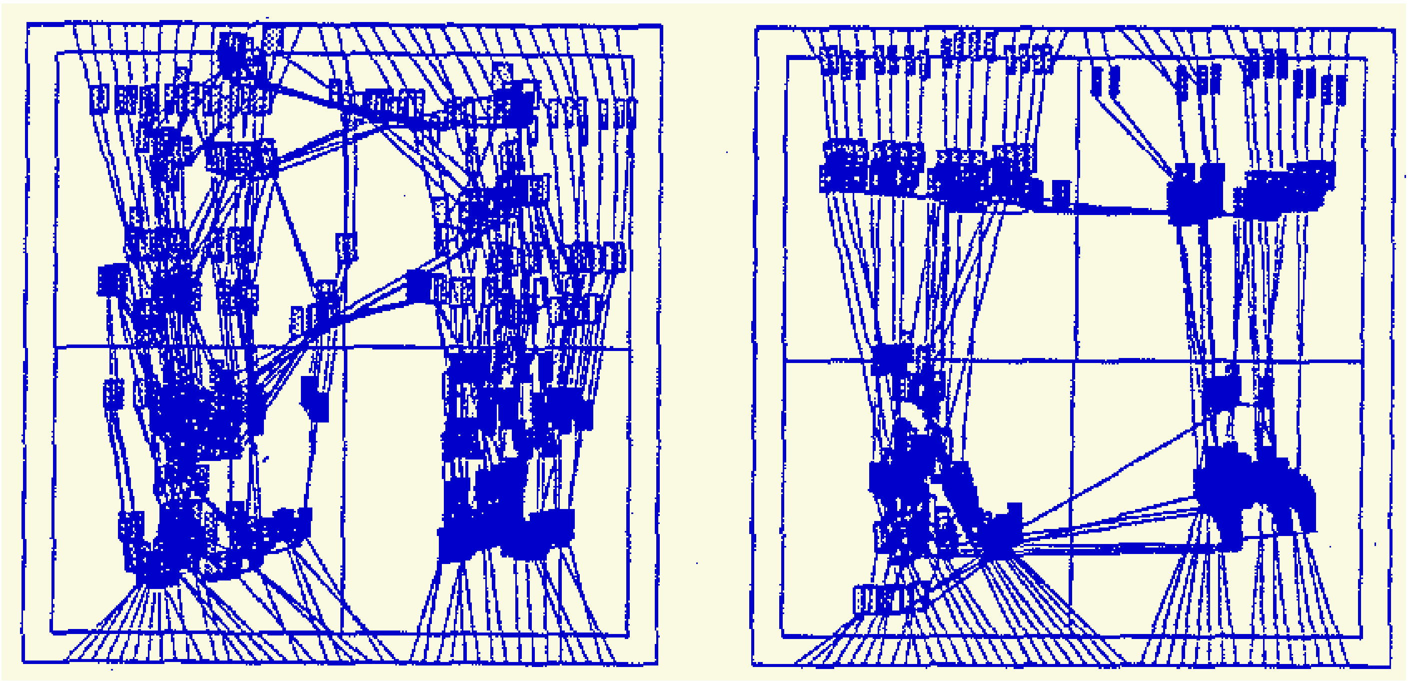
Analytical Placement Results



(a) Global placement with 1 region

Analytical Placement Results

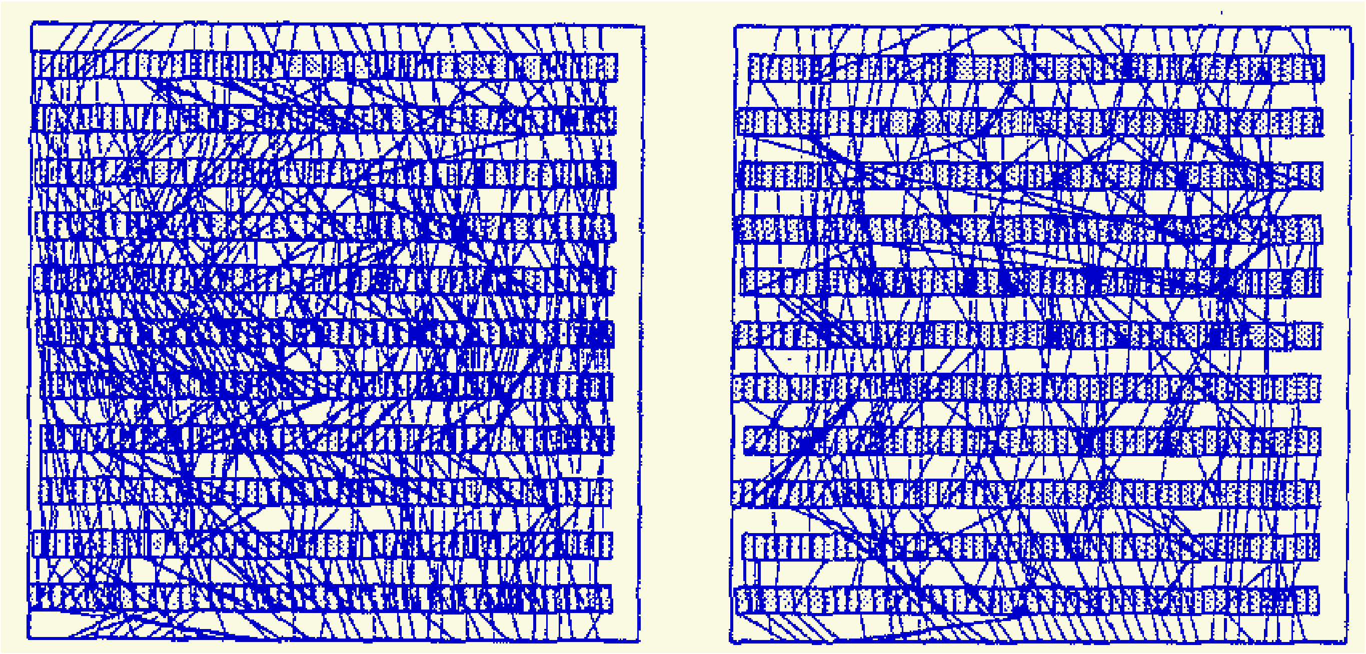
Quadratic objective function **Linear objective function**



(b) Global placement with 4 regions

Analytical Placement Results

Quadratic objective function | Linear objective function



(c) Final placements