Partitioning

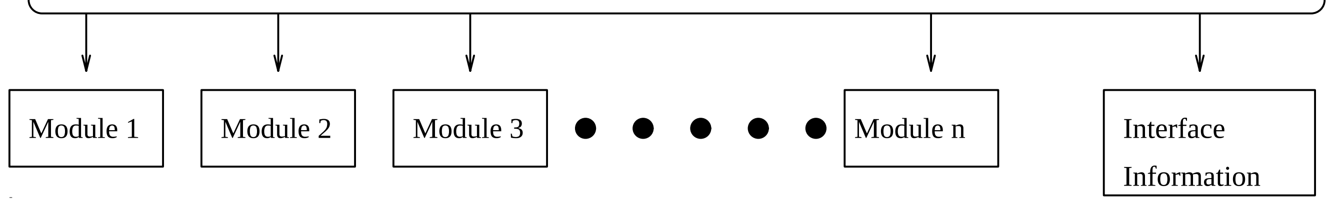
ECE6133
Physical Design Automation of VLSI Systems

Prof. Sung Kyu Lim
School of Electrical and Computer Engineering
Georgia Institute of Technology

Partitioning

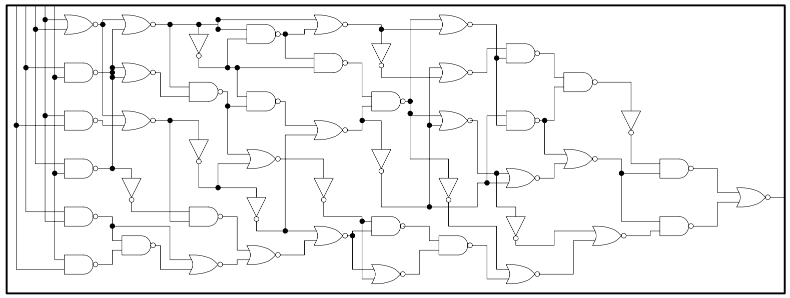
System design

- Decomposition of a complex system into smaller subsystems.
- Each subsystem can be designed independently speeding up the design process.
- Decomposition scheme has to minimize the interconnections between the subsystems.
- Decomposition is carried out hierarchically until each subsystem is of managable size.

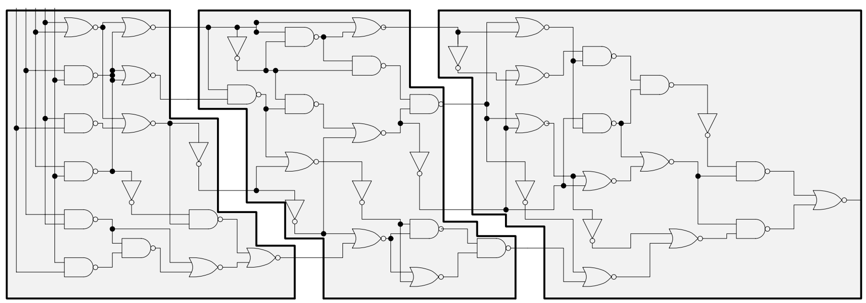


Partitioning of A Circuit

Input size = 48



(a)

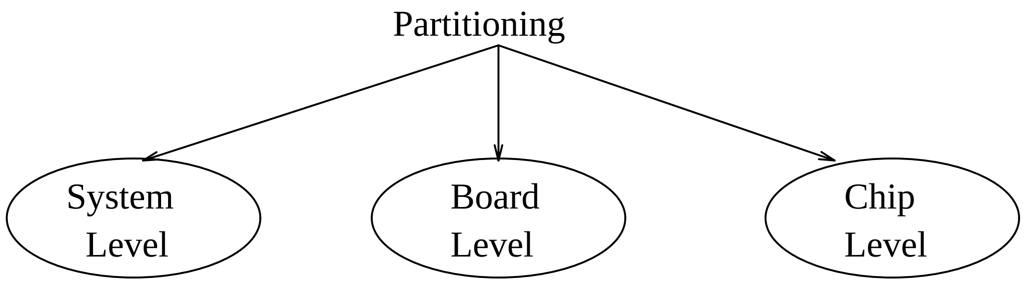


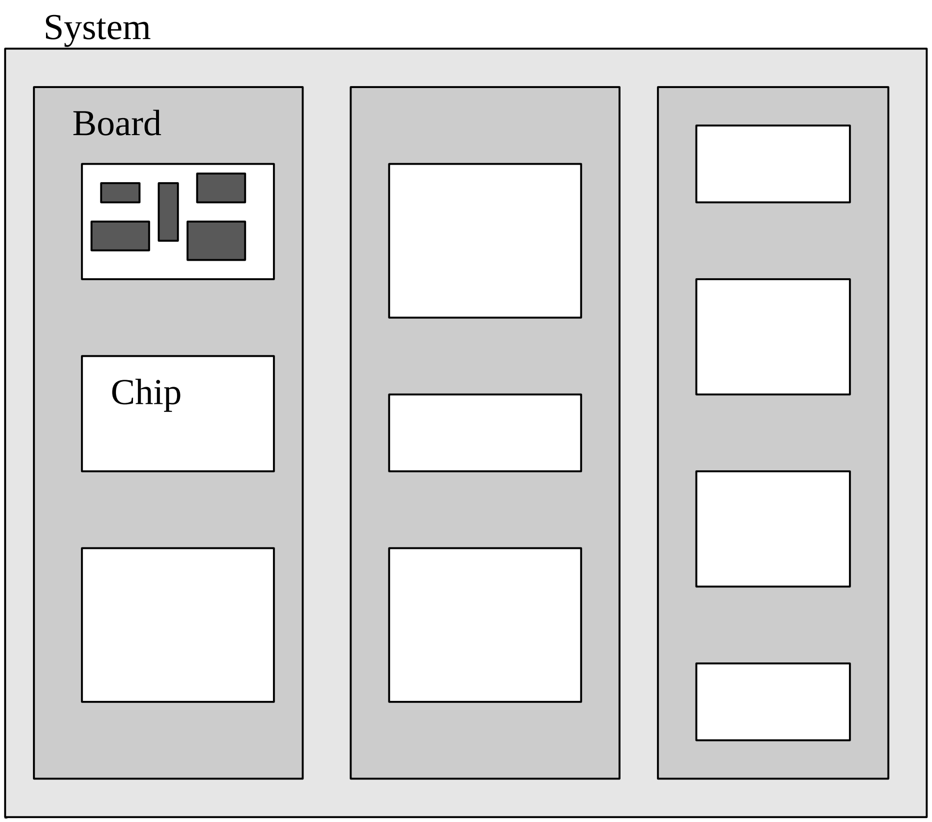
(b)

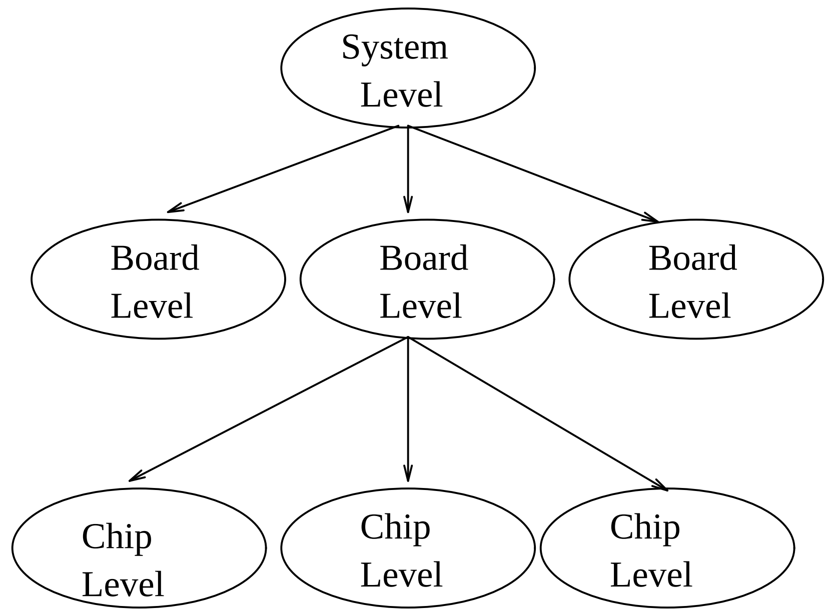
Cut 1 = 4 Cut 2 = 4

Size 1 = 15 Size 2 = 16 Size 3 = 17

Partitioning at different levels







Problem Formulation

1. Interconnections between partitions:

$$Obj_1 : \sum_{i=1}^{k} \sum_{j=1}^{k} c_{ij}, (i \neq j) \quad \text{is minimized}$$

2. Delay due to partitioning:

$$Obj_2 : \max_{p_i \in P}(H(p_i)) \quad \text{is minimized}$$

3. Number of terminals:

$$Cons_1 : Count(V_i) \leq T_i, \quad 1 \leq i \leq k$$

where,

c_{ij} is the cutsize between partitions V_i and V_j.

$H(p_i)$ is the number of times a hyperpath p_i is cut.

$Count(V_i)$ is the terminal count for partition V_i.

Problem Formulation

1. Area of each partition:

$$Cons_2 : A_i^{\min} \le Area(V_i) \le A_i^{\max}, \quad i = 1, 2, \ldots, k$$

2. Number of partitions:

$$Cons_3 : K_{\min} \le k \le K_{\max}$$

The partitioning problem at any level or design style deals with one or more of the above parameters.

Partitioning Methods

- **Top-down Partitioning (cutsize only)**
  - Iterative improvement [KL70, FM82, Kr84, San89]
  - Spectral based [HK92, AZ95]
  - Clustering method [SU72, NOP87, WC92, SS93, CS93, HK95]
  - Network flow based [YW94, YW97]
  - Analytical based [RDJ94, LLC95]
  - Multi-level [CS93, HB95, AHK97, KA+97, KK99]
- **Bottom-up Clustering (delay only)**
  - Unit delay model [LLT69, CD93]
  - General delay model [MBV91, RW93, YW95]
  - Sequential circuits with retiming [PKL98, CLW99, CL00]

Kernighan-Lin Algorithm

- It is a bisectioning algorithm
- The input graph is partitioned into two subsets of equal sizes.
- Till the cutsize keeps improving,
  - Vertex pairs which give the largest decrease in cutsize are exchanged
  - These vertices are then locked
  - If no improvement is possible and some vertices are still unlocked, the vertices which give the smallest increase are exchanged

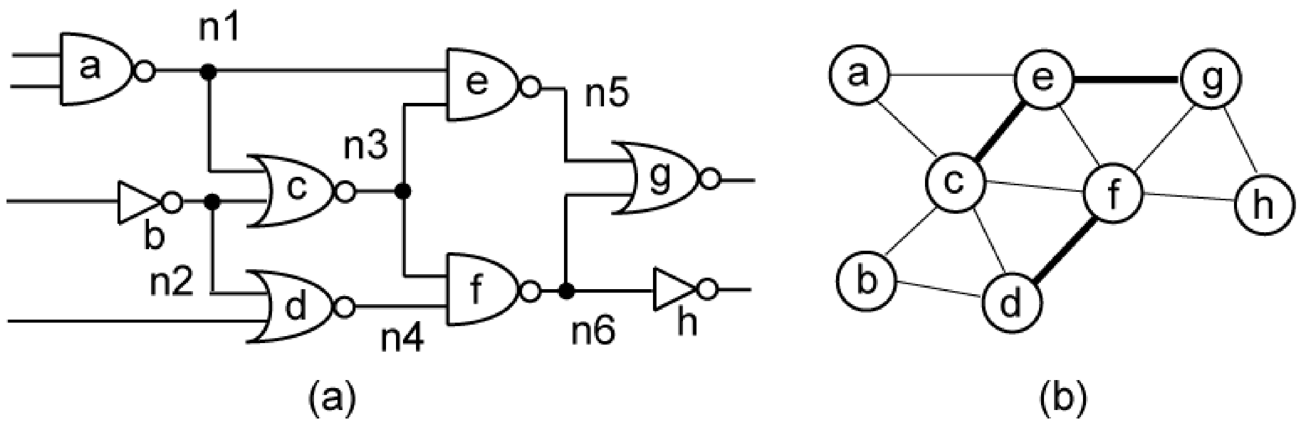
W. Kernighan and S. Lin, Bell System Technical Journal, 1970.

Kernighan-Lin Algorithm

```
Algorithm KL
begin
    INITIALIZE();
    while( IMPROVE(table) = TRUE ) do
    (* if an improvement has been made during last iteration,
    the process is carried out again. *)
        while ( UNLOCK(A) = TRUE ) do
        (* if there exists any unlocked vertex in A,
        more tentative exchanges are carried out. *)
            for ( each a ∈ A ) do
                if (a = unlocked) then
                    for( each b ∈ B ) do
                        if (b = unlocked) then
                            if (D_max < D(a) + D(b)) then
                                D_max = D(a) + D(b);
                                a_max = a;
                                b_max = b;
            TENT-EXCHGE(a_max, b_max);
            LOCK(a_max, b_max);
            LOG(table);
            D_max = −∞;
        ACTUAL-EXCHGE(table);
end.
```

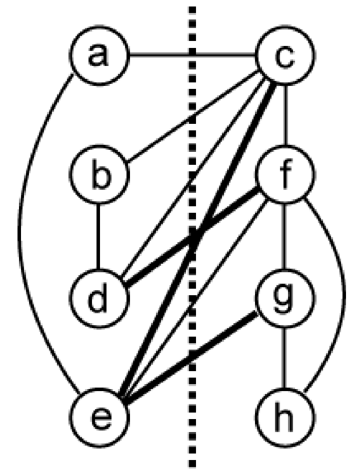
Kernighan-Lin Algorithm

- Perform single KL pass on the following circuit:
  - KL needs undirected graph (clique-based weighting)



First Swap

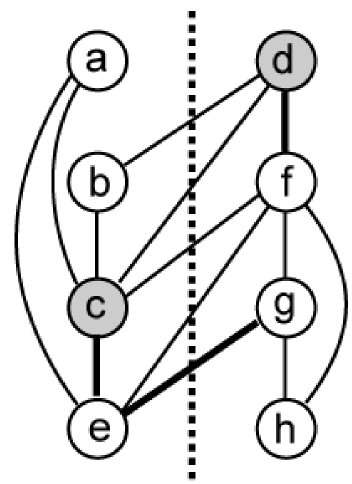
pair	$E_x - I_x$	$E_y - I_y$	$c(x,y)$	gain
(a,c)	$0.5 - 0.5$	$2.5 - 0.5$	0.5	1
(a,f)	$0.5 - 0.5$	$1.5 - 1.5$	0	0
(a,g)	$0.5 - 0.5$	$1 - 1$	0	0
(a,h)	$0.5 - 0.5$	$0 - 1$	0	-1
(b,c)	$0.5 - 0.5$	$2.5 - 0.5$	0.5	1
(b,f)	$0.5 - 0.5$	$1.5 - 1.5$	0	0
(b,g)	$0.5 - 0.5$	$1 - 1$	0	0
(b,h)	$0.5 - 0.5$	$0 - 1$	0	-1
$\mathbf{(d,c)}$	$\mathbf{1.5 - 0.5}$	$\mathbf{2.5 - 0.5}$	$\mathbf{0.5}$	**2**
(d,f)	$1.5 - 0.5$	$1.5 - 1.5$	1	-1
(d,g)	$1.5 - 0.5$	$1 - 1$	0	1
(d,h)	$1.5 - 0.5$	$0 - 1$	0	0
(e,c)	$2.5 - 0.5$	$2.5 - 0.5$	1	2
(e,f)	$2.5 - 0.5$	$1.5 - 1.5$	0.5	1
(e,g)	$2.5 - 0.5$	$1 - 1$	1	0
(e,h)	$2.5 - 0.5$	$0 - 1$	0	1



initial partitioning

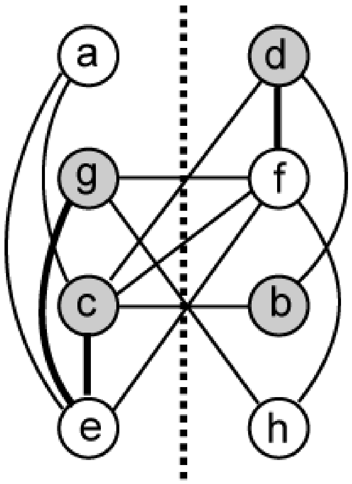
Second Swap

pair	$E_x - I_x$	$E_y - I_y$	$c(x, y)$	gain
(a, f)	$0 - 1$	$1 - 2$	0	-2
(a, g)	$0 - 1$	$1 - 1$	0	-1
(a, h)	$0 - 1$	$0 - 1$	0	-2
(b, f)	$0.5 - 0.5$	$1 - 2$	0	-1
$\mathbf{(b, g)}$	$\mathbf{0.5 - 0.5}$	$\mathbf{1 - 1}$	**0**	**0**
(b, h)	$0.5 - 0.5$	$0 - 1$	0	-1
(e, f)	$1.5 - 1.5$	$1 - 2$	0.5	-2
(e, g)	$1.5 - 1.5$	$1 - 1$	1	-2
(e, h)	$1.5 - 1.5$	$0 - 1$	0	-1



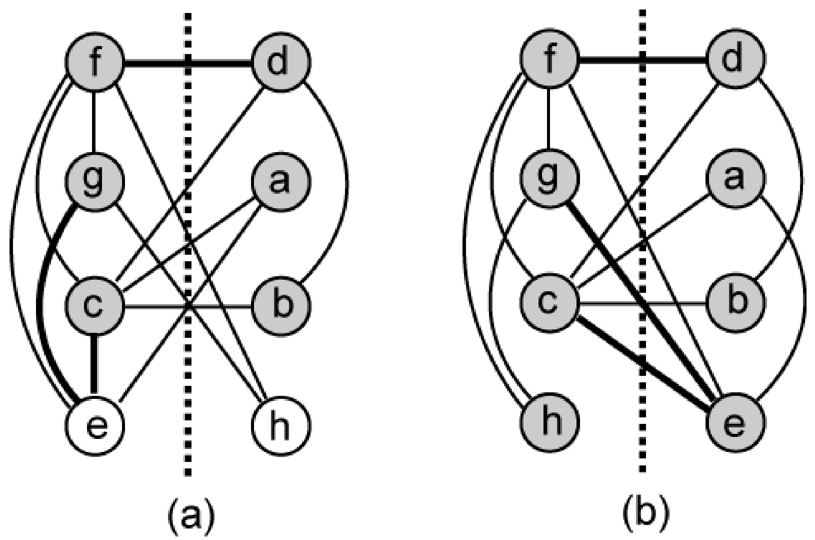
Third Swap

pair	$E_x - I_x$	$E_y - I_y$	$c(x, y)$	gain
$\boldsymbol{(a, f)}$	$\mathbf{0 - 1}$	$\mathbf{1.5 - 1.5}$	$\mathbf{0}$	**-1**
(a, h)	$0 - 1$	$0.5 - 0.5$	0	-1
(e, f)	$0.5 - 2.5$	$1.5 - 1.5$	0.5	-3
(e, h)	$0.5 - 2.5$	$0.5 - 0.5$	0	-2



Fourth Swap

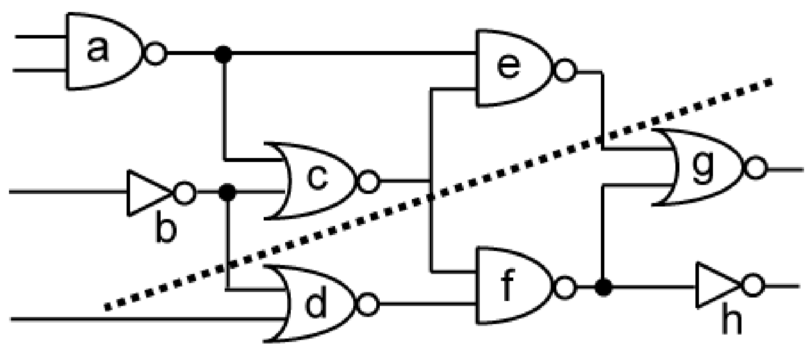
- Last swap does not require gain computation



Summary

- Cutsize reduced from 5 to 3
  - Two best solutions found (solutions are always area-balanced)

i	pair	$gain(i)$	$\sum gain(i)$	cutsize
0	-	-	-	5
1	$\mathbf{(d, c)}$	**2**	**2**	**3**
2	$\mathbf{(b, g)}$	**0**	**2**	**3**
3	(a, f)	-1	1	4
4	(e, h)	-1	0	5



Drawbacks of K-L Algorithm

- K-L algorithm considers balanced partitions only.
- As vertices have unit weights, it is not possible to allocate a vertex to a partition.
- The K-L algorithm considers edges instead of hyperedges.
- High, $O(n^3)$ complexity.

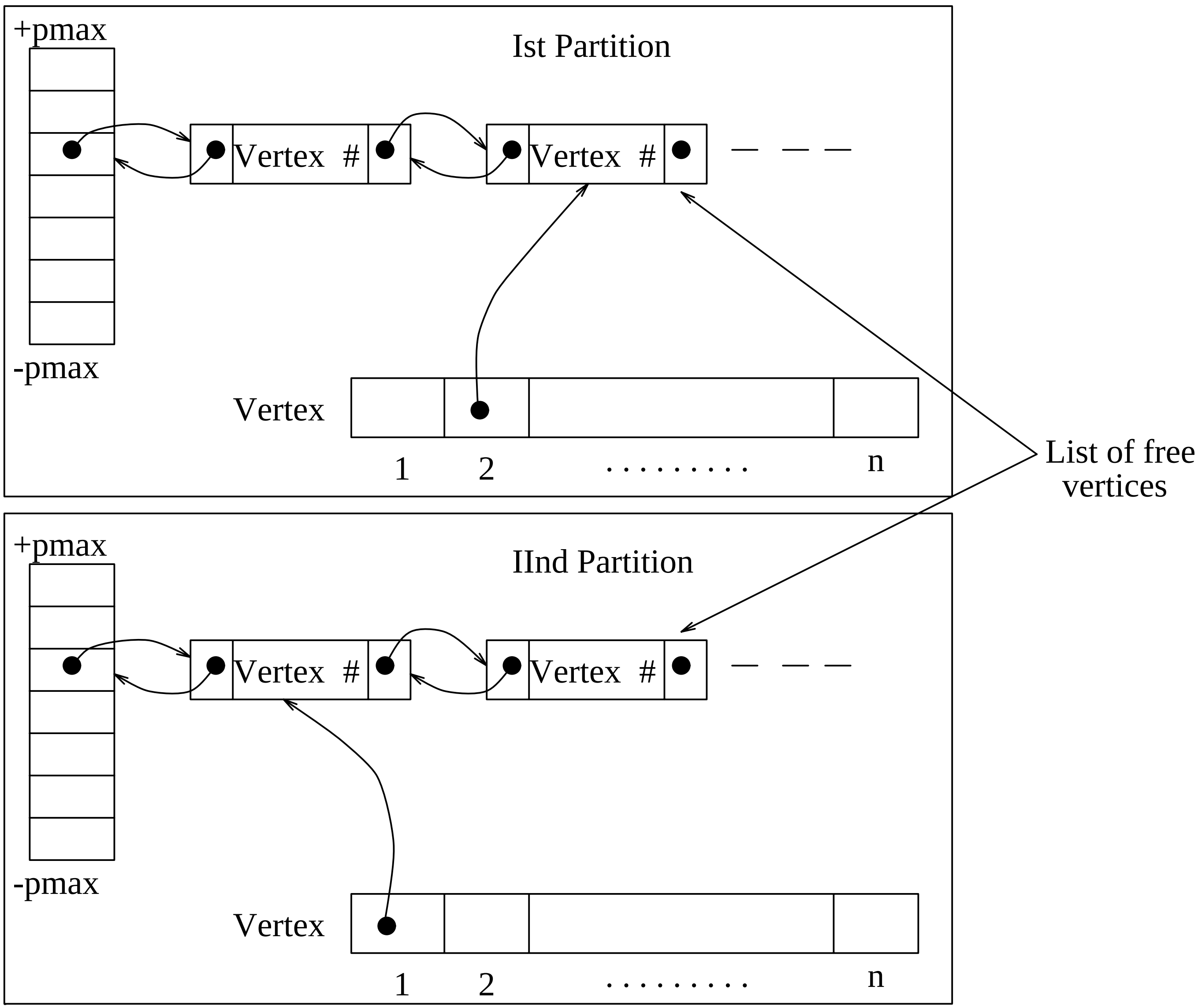
Fiduccia-Mattheyses Algorithm

This algorithm is a modified version of Kernighan-Lin Algorithm.

- A single vertex is moved across the cut in a single move which permits handling of unbalanced partitions.
- The concept of cutsize is extended to hypergraphs.
- Vertices to be moved are selected in a way to improve time complexity.
- A special data structure is used to do this.
- Overall time complexity of the algorithm is $O(n^2)$.

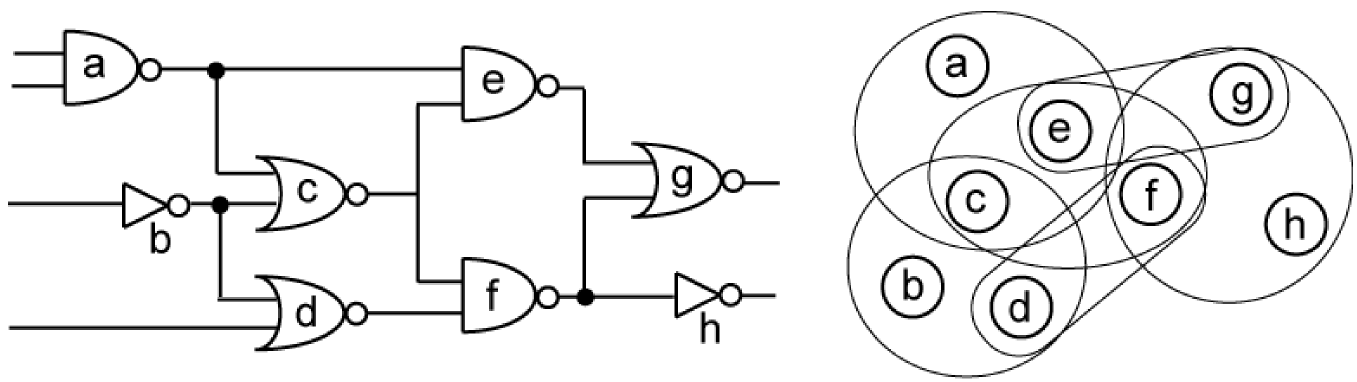
C. M. Fiduccia and R. M. Mattheyses, 19th DAC, 1982.

Data Structure Used in Fiduccia-Mattheyses Algorithm



Fiduccia-Mattheyses Algorithm

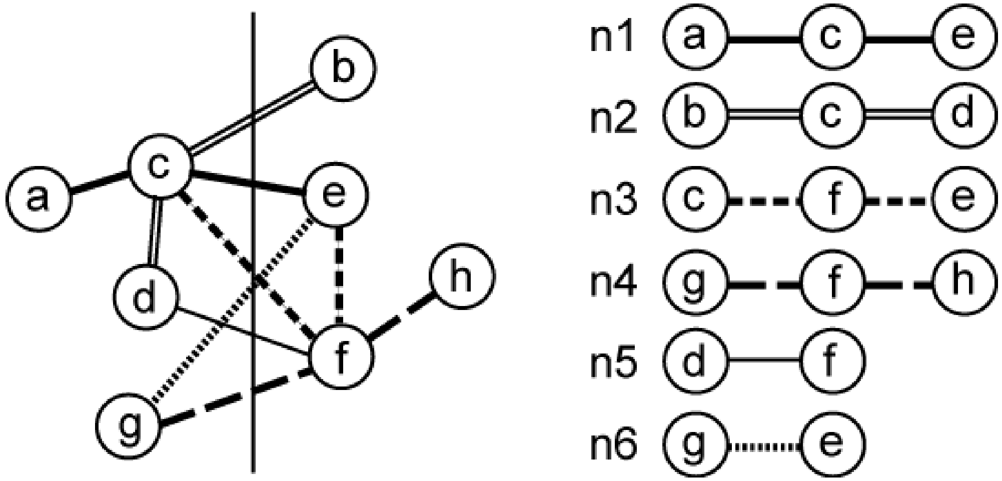
- Perform FM algorithm on the following circuit:
  - Area constraint = [3,5]
  - Break ties in alphabetical order.



(a) (b)

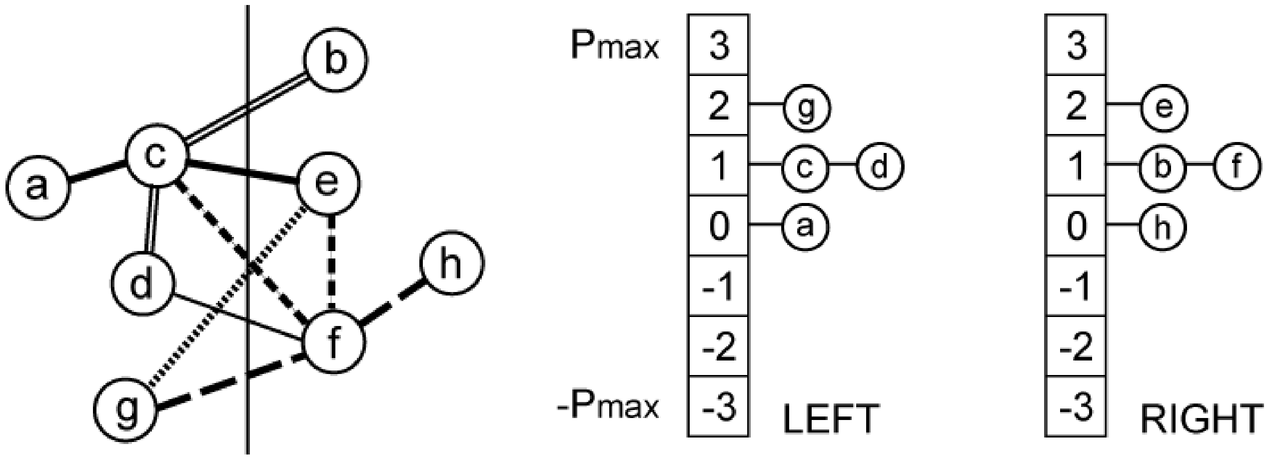
Initial Partitioning

- Random initial partitioning is given.



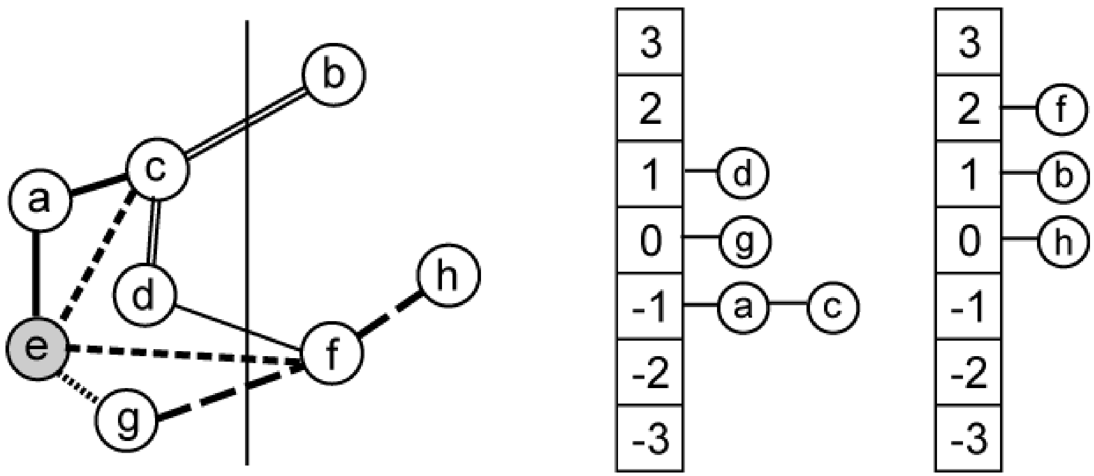
Gain Computation and Bucket Set Up

cell c: c is contained in net $n_1 = \{a, c, e\}$, $n_2 = \{b, c, d\}$, and $n_3 = \{c, f, e\}$. n_3 contains c as its only cell located in the left partition, so $FS(c) = 1$. In addition, none of these three nets are located entirely in the left partition. So, $TE(c) = 0$. Thus, $gain(c) = 1$.



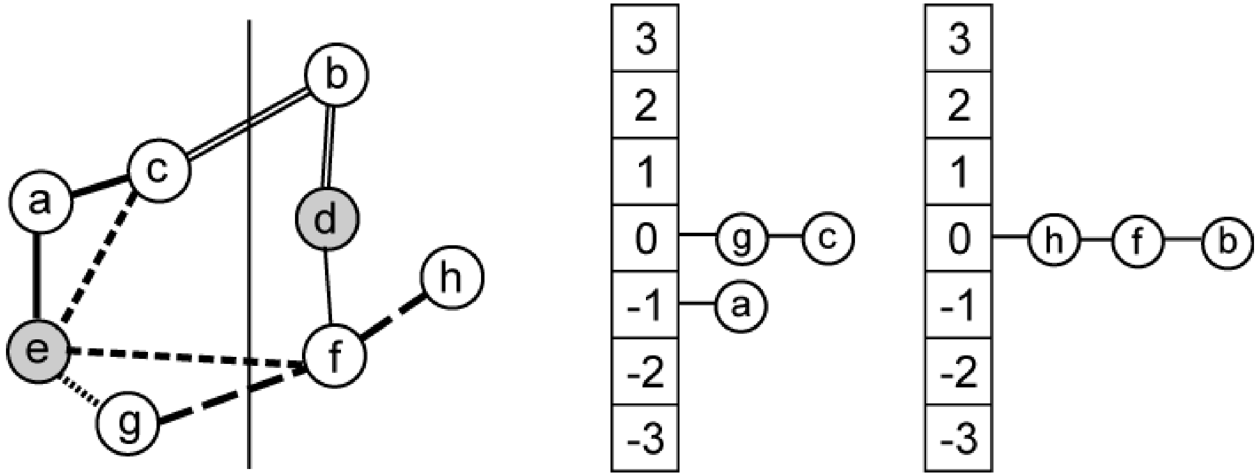
First Move

move 1: From the initial bucket we see that both cell g and e have the maximum gain and can be moved without violating the area constraint. We move e based on alphabetical order. We update the gain of the unlocked neighbors of e, $N(e) = \{a, c, g, f\}$, as follows: $gain(a) = FS(a) - TE(a) = 0 - 1 = -1$, $gain(c) = 0 - 1 = -1$, $gain(g) = 1 - 1 = 0$, $gain(f) = 2 - 0 = 2$.



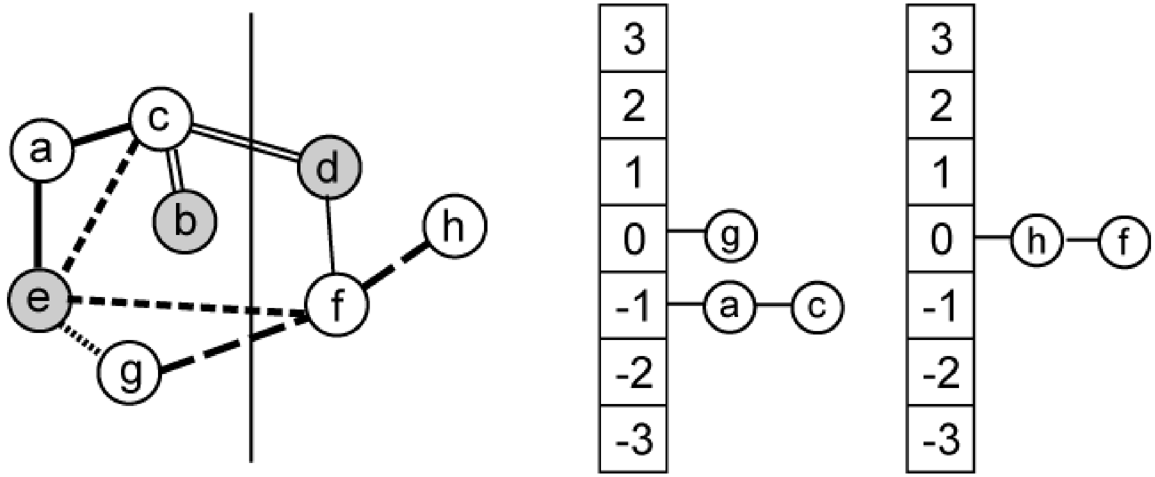
Second Move

move 2: f has the maximum gain, but moving f will violate the area constraint. So we move d. We update the gain of the unlocked neighbors of d, $N(d) = \{b, c, f\}$, as follows: $gain(b) = 0 - 0 = 0$, $gain(c) = 1 - 1 = 0$, $gain(f) = 1 - 1 = 0$.



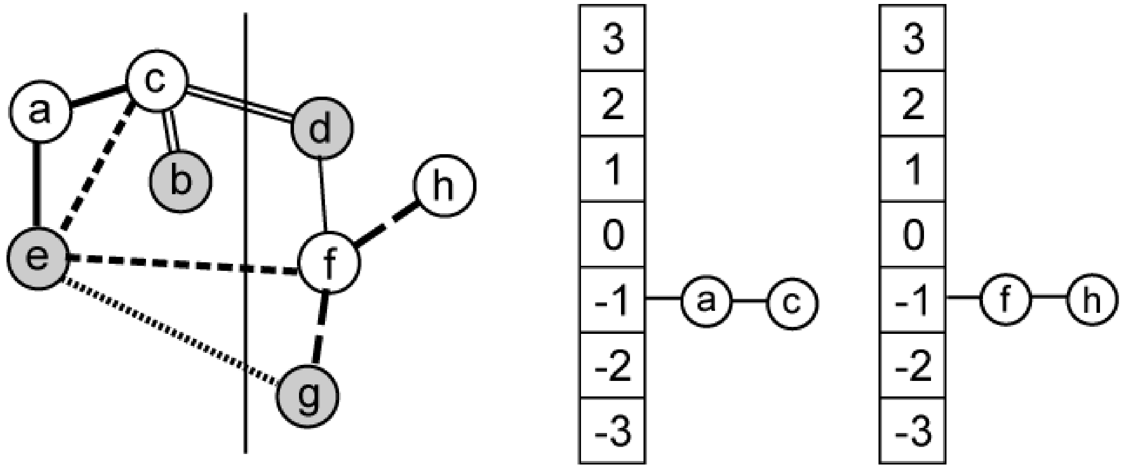
Third Move

move 3: Among the maximum gain cells $\{g, c, h, f, b\}$, we choose b based on alphabetical order. We update the gain of the unlocked neighbors of b, $N(b) = \{c\}$ as follows: $gain(c) = 0 - 1 = -1$.



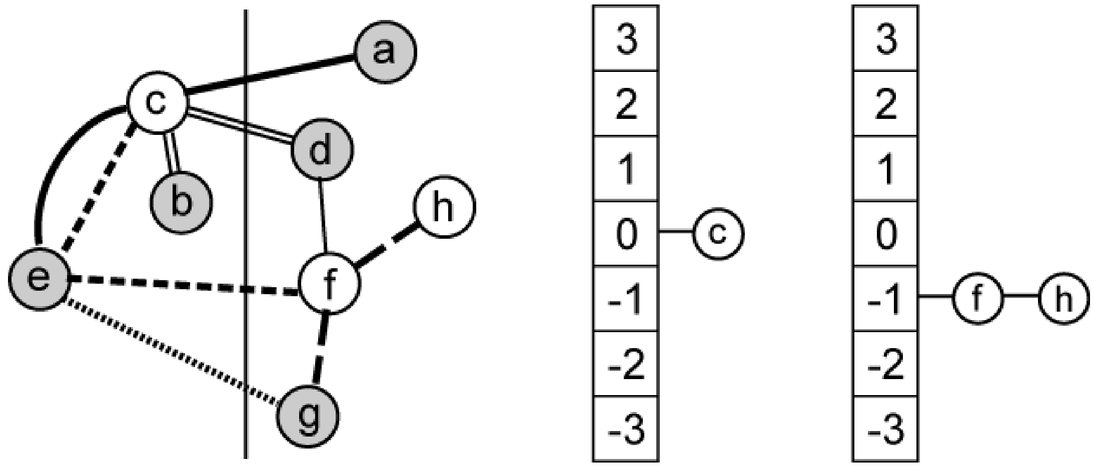
Forth Move

move 4: Among the maximum gain cells $\{g, h, f\}$, we choose g based on the area constraint. We update the gain of the unlocked neighbors of g, $N(g) = \{f, h\}$, as follows: $gain(f) = 1 - 2 = -1$, $gain(h) = 0 - 1 = -1$.



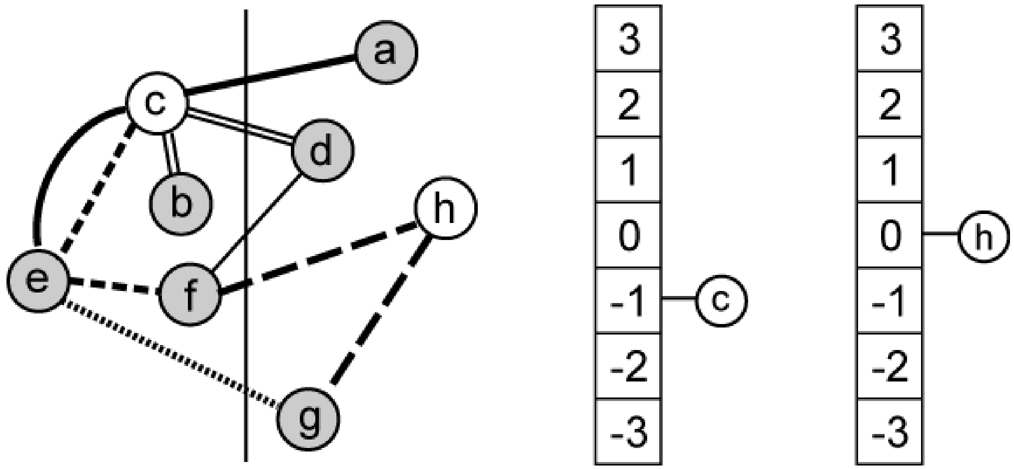
Fifth Move

move 5: We choose a based on alphabetical order. We update the gain of the unlocked neighbors of a, $N(a) = \{c\}$, as follows: $gain(c) = 0 - 0 = 0$.



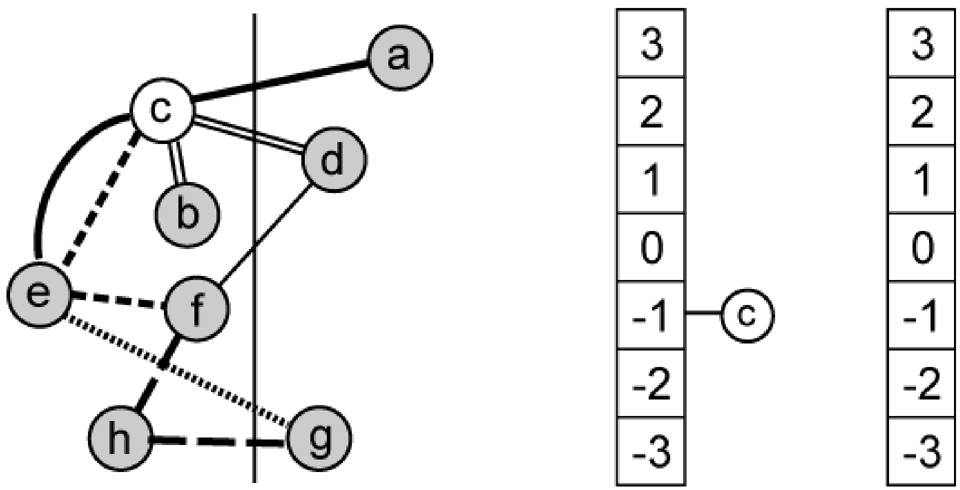
Sixth Move

move 6: We choose f based on the area constraint and alphabetical order. We update the gain of the unlocked neighbors of f, $N(f) = \{h, c\}$, as follows: $gain(h) = 0 - 0 = 0$, $gain(c) = 0 - 1 = -1$.



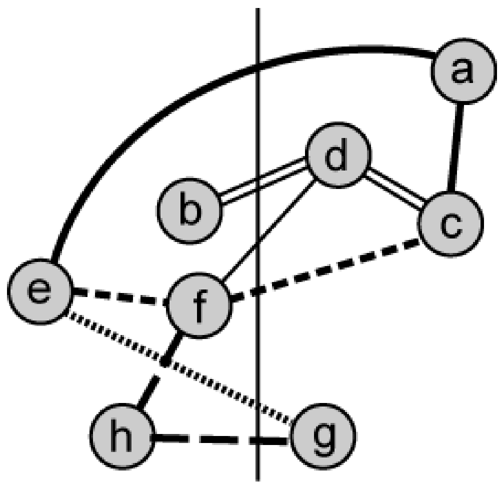
Seventh Move

move 7: We move h. h has no unlocked neighbor.



Last Move

move 8: We move c.



Summary

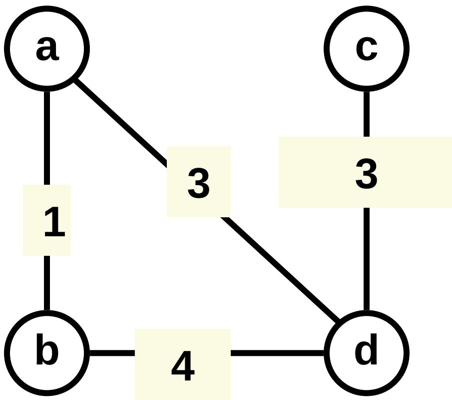
- Found three best solutions.
  - Cutsize reduced from 6 to 3.
  - Solutions after move 2 and 4 are better balanced.

i	cell	$g(i)$	$\sum g(i)$	cutsize
0	-	-	-	6
1	e	2	2	4
2	$\boldsymbol{d}$	**1**	**3**	**3**
3	$\boldsymbol{b}$	**0**	**3**	**3**
4	$\boldsymbol{g}$	**0**	**3**	**3**
5	a	-1	2	4
6	f	-1	1	5
7	h	0	1	5
8	c	-1	0	6

Probing Further

- FM Algorithm
  - [Krishnamurthy, 1984]: developed “look-ahead” gain concept, where gain is now a vector.
  - [Sanchis, 1989]: perform “flat” multi-way partitioning, where gain considers all possible destinations
  - [Cong and Lim, 1998]: showed that recursive is way better than flat multi-way partitioning, improved flat method
  - [Dutt and Deng, 1996]: encourages neighboring cell move, effective in avoiding cutting clusters
  - [Hagen et al, 1997]: showed that LIFO bucket works better than FIFO
  - [Hauck and Borriello, 1997]: evaluated all existing FM extensions and proposed the “best” combination

Spectral Based Partitioning Algorithms



$$A = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 0 & 3 \\ b & 1 & 0 & 0 & 4 \\ c & 0 & 0 & 0 & 3 \\ d & 3 & 4 & 3 & 0 \end{array}$$

$$D = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 4 & 0 & 0 & 0 \\ b & 0 & 5 & 0 & 0 \\ c & 0 & 0 & 3 & 0 \\ d & 0 & 0 & 0 & 10 \end{array}$$

D: degree matrix; A: adjacency matrix; D-A: Laplacian matrix

Eigenvectors of D-A form the Laplacian spectrum of G

Some Applications of Laplacian Spectrum

- **Placement and floorplan**
  [Hall 1970]
  [Otten 1982]
  [Frankle-Karp 1986]
  [Tsay-Kuh 1986]

- **Bisection lower bound and computation**
  [Donath-Hoffman 1973]
  [Barnes 1982]
  [Boppana 1987]

- **Ratio-cut lower bound and computation**
  [Hagen-Kahng 1991]
  [Cong-Hagen-Kahng 1992]

Eigenvalues and Eigenvectors

$$\begin{array}{ccc} \mathbf{A} & \underline{\mathbf{x}} & \mathbf{A}\underline{\mathbf{x}} \\ \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ \vdots & & & \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} & \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = & \begin{pmatrix} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n \end{pmatrix} \end{array}$$

If $\quad A\underline{x}=\lambda\underline{x}$

then λ **is an eigenvalue of A**

$\underline{x}$ **is an eignevector of A w.r.t.** λ

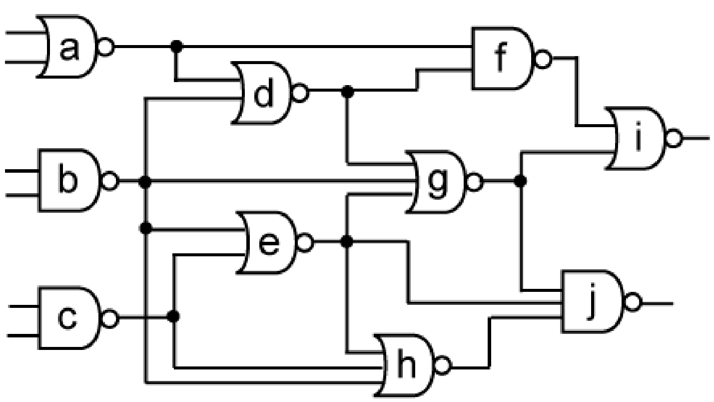
(note that K$\underline{x}$ **is also a eigenvector, for any constant K).**

Spectral Partitioning

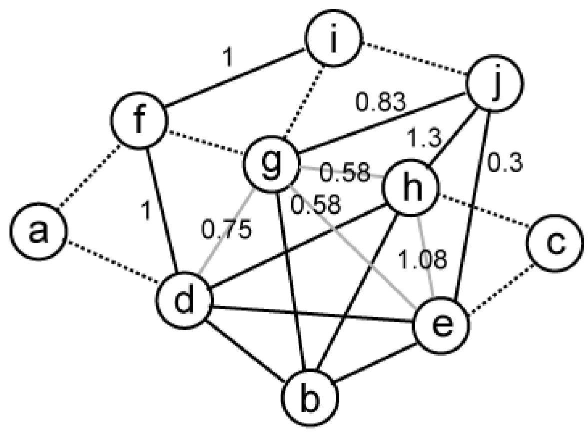
- **Hall's Results [1970]**
  - **Given an undirected edge weighted graph G**
  - **Important property about the Laplacian Matrix Q of G**
  - **Eigenvector of the 2nd smallest eigenvalue of Q gives 1-dimensional placement of nodes in V**
  - **Sum of the squared length of the edges are minimized**
  - **Under $\Sigma\ x^{2=}=1$**
- **Hagen and Kahng's Results [1992]**
  - **2nd smallest eigenvalue of Q is a tight lower bound of ratio-cut**
  - **Derive partitioning from 1-dimensional placement for ratio-cut minimization**

Hagen-Kahng EIG Partitioning

- Perform EIG partitioning and minimize ratio cut cost.
  - Clique-based graph model: dotted edge has weight of 0.5, and solid edge with no label has weight of 0.25.



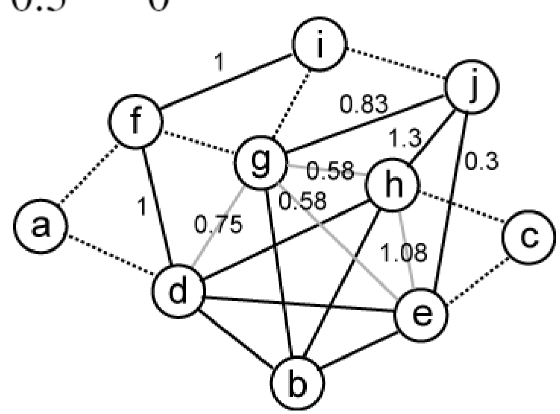
circuit



clique-based model

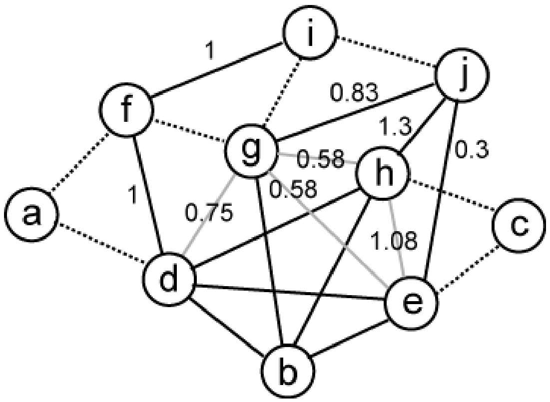
Adjacency Matrix

	a	b	c	d	e	f	g	h	i	j
a	0	0	0	0.5	0	0.5	0	0	0	0
b	0	0	0	0.25	0.25	0	0.25	0.25	0	0
c	0	0	0	0	0.5	0	0	0.5	0	0
d	0.5	0.25	0	0	0.25	1.0	0.75	0.25	0	0
e	0	0.25	0.5	0.25	0	0	0.58	1.08	0	0.33
f	0.5	0	0	1.0	0	0	0.5	0	1.0	0
g	0	0.25	0	0.75	0.58	0.5	0	0.58	0.5	0.83
h	0	0.25	0.5	0.25	1.08	0	0.58	0	0	1.33
i	0	0	0	0	0	1.0	0.5	0	0	0.5
j	0	0	0	0	0.33	0	0.83	1.33	0.5	0



Degree Matrix

	a	b	c	d	e	f	g	h	i	j
a	1.0	0	0	0	0	0	0	0	0	0
b	0	1.0	0	0	0	0	0	0	0	0
c	0	0	1.0	0	0	0	0	0	0	0
d	0	0	0	3.0	0	0	0	0	0	0
e	0	0	0	0	2.99	0	0	0	0	0
f	0	0	0	0	0	3.0	0	0	0	0
g	0	0	0	0	0	0	3.99	0	0	0
h	0	0	0	0	0	0	0	3.99	0	0
i	0	0	0	0	0	0	0	0	2.0	0
j	0	0	0	0	0	0	0	0	0	2.99



Laplacian Matrix

- We obtain $Q = D - A$

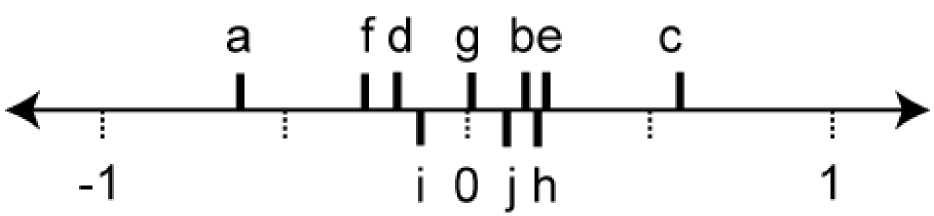
	a	b	c	d	e	f	g	h	i	j
a	1.0	0	0	-0.5	0	-0.5	0	0	0	0
b	0	1.0	0	-0.25	-0.25	0	-0.25	-0.25	0	0
c	0	0	1.0	0	-0.5	0	0	-0.5	0	0
d	-0.5	-0.25	0	3.0	-0.25	-1.0	-0.75	-0.25	0	0
e	0	-0.25	-0.5	-0.25	2.99	0	-0.58	-1.08	0	-0.33
f	-0.5	0	0	-1.0	0	3.0	-0.5	0	-1.0	0
g	0	-0.25	0	-0.75	-0.58	-0.5	3.99	-0.58	-0.5	-0.83
h	0	-0.25	-0.5	-0.25	-1.08	0	-0.58	3.99	0	-1.33
i	0	0	0	0	0	-1.0	-0.5	0	2.0	-0.5
j	0	0	0	0	-0.33	0	-0.83	-1.33	-0.5	2.99

Eigenvalue/vector Computation

The second smallest eigenvalue is 0.6281, and its eigenvector is: [-0.6346, 0.1605, 0.5711, -0.1898, 0.2254, -0.2822, 0.0038, 0.1995, -0.1641, 0.1104]^T. We observe the following:

- The squared sum of the values in the vector is 1 as shown by Hall [Hall, 1970].
- These values define a one-dimensional placement of the 10 nodes within the range of $[-1, 1]$, where the sum of the squared length of all edges is minimized. Figure 2.21 shows this placement.
- These values define the following ordering among the nodes:

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

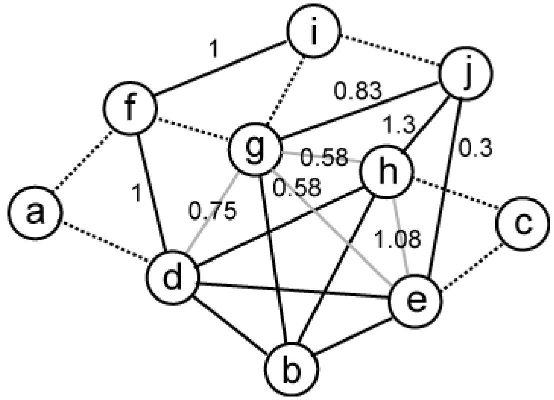


EIG Partitioning

(a) Partitioning $(\{a\}, \{f, d, i, g, j, b, h, e, c\})$:
The cut edges are (a, f) and (a, d). Thus, the cutsize is $0.5+0.5 = 1.0$. The ratio cut is $1.0/(1 \cdot 9) = 0.1111$.

(b) Partitioning $(\{a, f\}, \{d, i, g, j, b, h, e, c\})$:
The cut edges are (f, i), (f, g), (f, d) and (a, d). Thus, the cutsize is $1.0 + 0.5 + 1.0 + 0.5 = 3.0$. The ratio cut is $3.0/(2 \cdot 8) = 0.1875$.

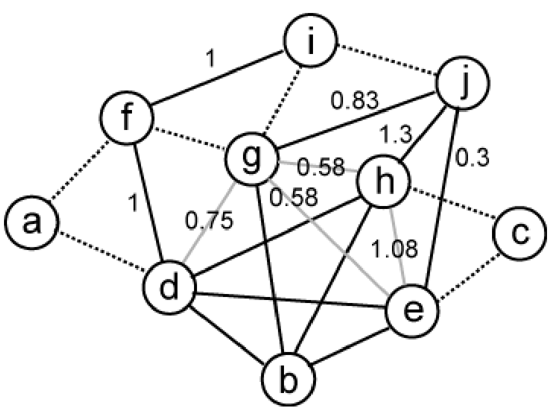
(c) Partitioning $(\{a, f, d\}, \{i, g, j, b, h, e, c\})$:
The cut edges are (f, i), (f, g), (d, g), (d, h), (d, e), and (d, b). Thus, the cutsize is $1.0 + 0.5 + 0.75 + 3 \cdot 0.25 = 3.0$. The ratio cut is $3.0/(3 \cdot 7) = 0.1429$.



EIG Partitioning (cont)

(d) Partitioning $(\{a, f, d, i\}, \{g, j, b, h, e, c\})$:
The cut edges are (i, j), (i, g), (f, g), (d, g), (d, h), (d, e), and (d, b). Thus, the cutsize is $0.5 \cdot 3 + 0.75 + 3 \cdot 0.25 = 3.0$. The ratio cut is $3.0/(4 \cdot 6) = 0.125$.

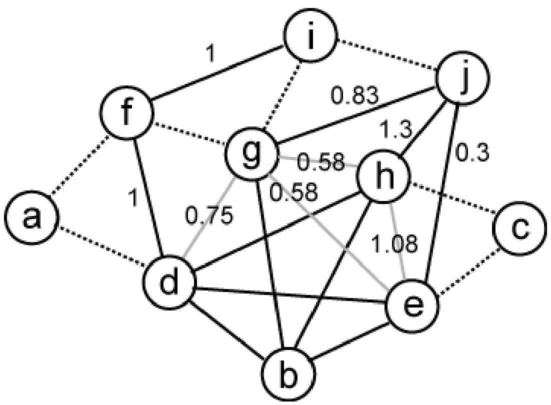
(e) Partitioning $(\{a, f, d, i, g\}, \{j, b, h, e, c\})$:
The cut edges are (i, j), (g, j), (g, h), (g, e), (g, b), (d, h), (d, e), and (d, b). Thus, the cutsize is $0.5 + 0.83 + 0.58 \cdot 2 + 0.25 \cdot 4 = 3.49$. The ratio cut is $3.49/(5 \cdot 5) = 0.1396$.



EIG Partitioning (cont)

(f) Partitioning $(\{a, f, d, i, g, j\}, \{b, h, e, c\})$:
The cut edges are (j, e), (j, h), (g, h), (g, e), (g, b), (d, h), (d, e), and (d, b). Thus, the cutsize is $0.33 + 1.33 + 0.58 \cdot 2 + 0.25 \cdot 4 = 3.82$. The ratio cut is $3.82/(6 \cdot 4) = 0.1592$.

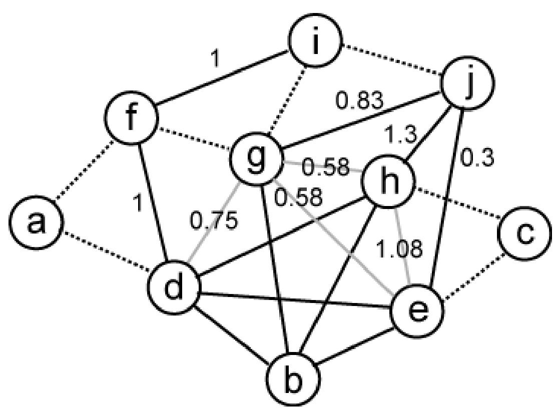
(g) Partitioning $(\{a, f, d, i, g, j, b\}, \{h, e, c\})$:
The cut edges are (j, e), (j, h), (g, h), (g, e), (d, h), (d, e), (b, h), and (b, e). Thus, the cutsize is $0.33 + 1.33 + 0.58 \cdot 2 + 0.25 \cdot 4 = 3.82$. The ratio cut is $3.82/(7 \cdot 3) = 0.1819$.



EIG Partitioning (cont)

(h) Partitioning $(\{a, f, d, i, g, j, b, h\}, \{e, c\})$:
The cut edges are (h, c), (h, e), (j, e), (g, e), (d, e), and (b, e). Thus, the cutsize is $0.5 + 1.08 + 0.33 + 0.58 + 0.25 + 0.25 = 2.99$. The ratio cut is $2.99/(8 \cdot 2) = 0.1869$.

(i) Partitioning $(\{a, f, d, i, g, j, b, h, e\}, \{c\})$:
The cut edges are (h, c) and (e, c). Thus, the cutsize is $0.5 + 0.5 = 1.0$. The ratio cut is $1.0/(9 \cdot 1) = 0.1111$.



Summary

- Good solution found:
  - {(a,f,d,g,i), (j,b,h,e,c)} is well-balanced and has low RC cost.

P_A	P_B	cutsize	ratio cut
$\{a\}$	$\{f, d, i, g, j, b, h, e, c\}$	1.0	$1.0/(1 \cdot 9) = 0.1111$
$\{a, f\}$	$\{d, i, g, j, b, h, e, c\}$	3.0	$3.0/(2 \cdot 8) = 0.1875$
$\{a, f, d\}$	$\{i, g, j, b, h, e, c\}$	3.0	$3.0/(3 \cdot 7) = 0.1429$
$\{a, f, d, i\}$	$\{g, j, b, h, e, c\}$	3.0	$3.0/(4 \cdot 6) = 0.125$
$\{a, f, d, i, g\}$	$\{j, b, h, e, c\}$	3.49	$3.49/(5 \cdot 5) = 0.1396$
$\{a, f, d, i, g, j\}$	$\{b, h, e, c\}$	3.82	$3.82/(6 \cdot 4) = 0.1592$
$\{a, f, d, i, g, j, b\}$	$\{h, e, c\}$	3.82	$3.82/(7 \cdot 3) = 0.1819$
$\{a, f, d, i, g, j, b, h\}$	$\{e, c\}$	2.99	$2.99/(8 \cdot 2) = 0.1869$
$\{a, f, d, i, g, j, b, h, e\}$	$\{c\}$	1.0	$1.0/(9 \cdot 1) = 0.1111$

Theorem

Verify that the second smallest eigenvalue is a tight lower bound of the ratio cut metric.

The eigenvalue is $\lambda = 0.6281$. It is shown in [Hagen and Kahng, 1992] that $c \geq \lambda/n$, where c is the ratio cut cost, and n is the number of nodes in the graph. Since $n = 10$ in our case, we see that $\lambda/n = 0.06281$ is smaller than all of the ratio cut values shown in Table 1.6.

Probing Further

- EIG Algorithm
  - [Chan et al, 1994]: extended EIG to multi-way partitioning, uses k-smallest eigenvalues/eigenvectors
  - [Riess et al, 1994]: use GORDIAN-L placement to derive partitioning solution that minimizes ratio-cut
  - [Alpert and Yao, 1995]: presented a new vertex ordering scheme based on eigenvectors
  - [Alpert and Khang, 1995]: used dynamic programming to split vertex ordering and obtain multi-way partitioning
  - [Li at al, 1996]: studied linear vs quadratic objectives, and proposed α-order objective F^{α}, $(1 \leq \alpha \leq 2)$