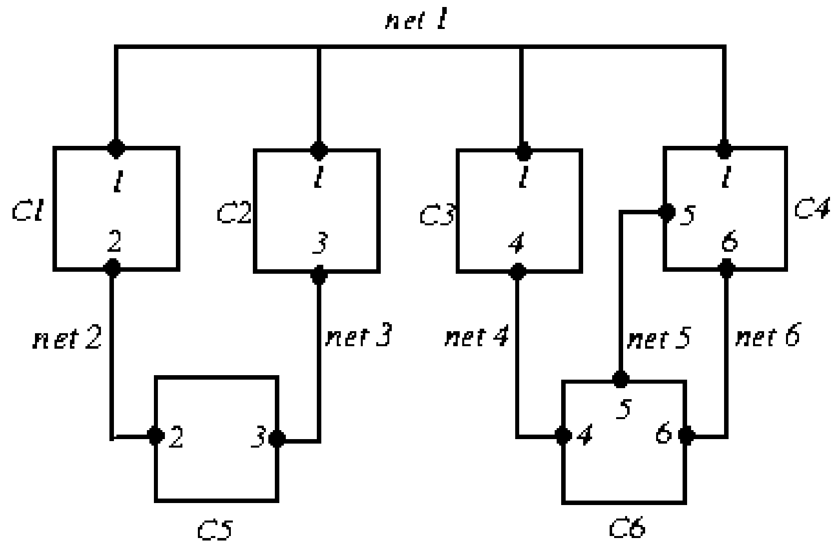
Fiduccia-Mattheyses Heuristic

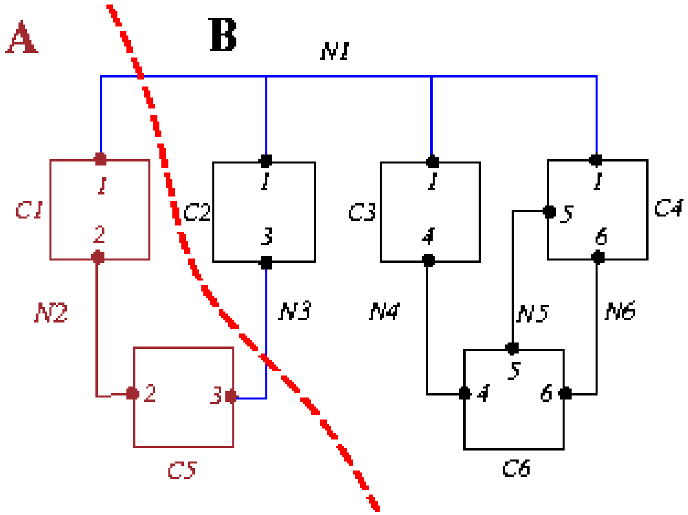
- Fiduccia and Mattheyses, "A linear time heuristic for improving network partitions," DAC-82.
- New features to the K-L heuristic:
  - Aims at **reducing net-cut costs**; the concept of cutsize is extended to hypergraphs.
  - Only a **single vertex** is moved across the cut in a single move.
  - Vertices are weighted.
  - Can handle "unbalanced" partitions; a balance factor is introduced.
  - A special data structure is used to select vertices to be moved across the cut to improve running time.
  - **Time complexity** $O(P)$, where P is the total # of terminals.

F-M Heuristic: Notation

- $n(i)$: # of cells in Net i; e.g., $n(1) = 4$.
- $s(i)$: size of Cell i.
- $p(i)$: # of pin terminals in Cell i; e.g., $p(6)=3$.
- C: total # of cells; e.g., $C=6$.
- N: total # of nets; e.g., $N=6$.
- P: total # of pins; $P = p(1) + \ldots + p(C) = n(1) + \ldots + n(N)$.

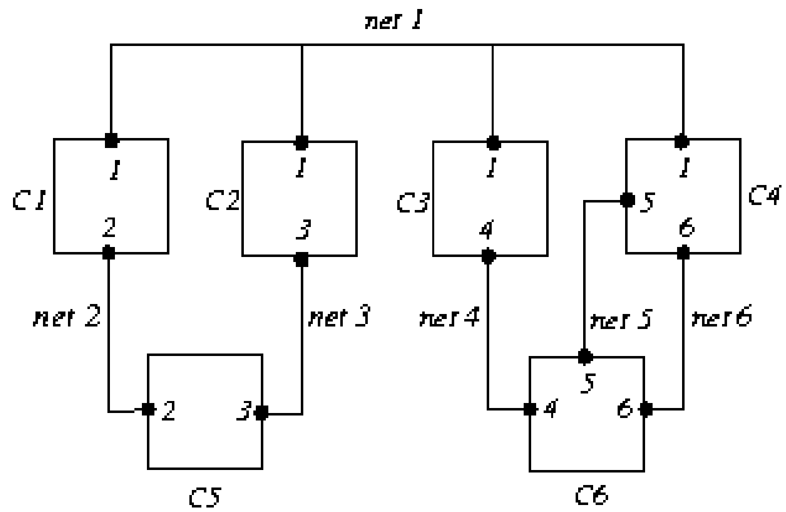


Cut



- **Cutstate** of a net:
  - Net 1 and Net 3 are **cut** by the partition.
  - Net 2, Net 4, Net 5, and Net 6 are **uncut**.
- **Cutset** = {Net 1, Net 3}.
- $|A|$ = size of A = $s(1)+s(5)$; $|B| = s(2)+s(3)+s(4)+s(6)$.
- **Balanced 2-way partition:** Given a fraction r, $0 < r < 1$, partition a graph into two sets A and B such that
  - $\frac{|A|}{|A|+|B|} \approx r$.
  - Size of the cutset is minimized.

Input Data Structures



Cell array		Net array	
C1	Nets 1, 2	Net 1	C1, C2, C3, C4
C2	Nets 1, 3	Net 2	C1, C5
C3	Nets 1, 4	Net 3	C2, C5
C4	Nets 1, 5, 6	Net 4	C3, C6
C5	Nets 2, 3	Net 5	C4, C6
C6	Nets 4, 5, 6	Net 6	C4, C6

- Size of the network: $P = \sum_{i=1}^{6} n(i) = 14$
- Construction of the two arrays takes $O(P)$ time.

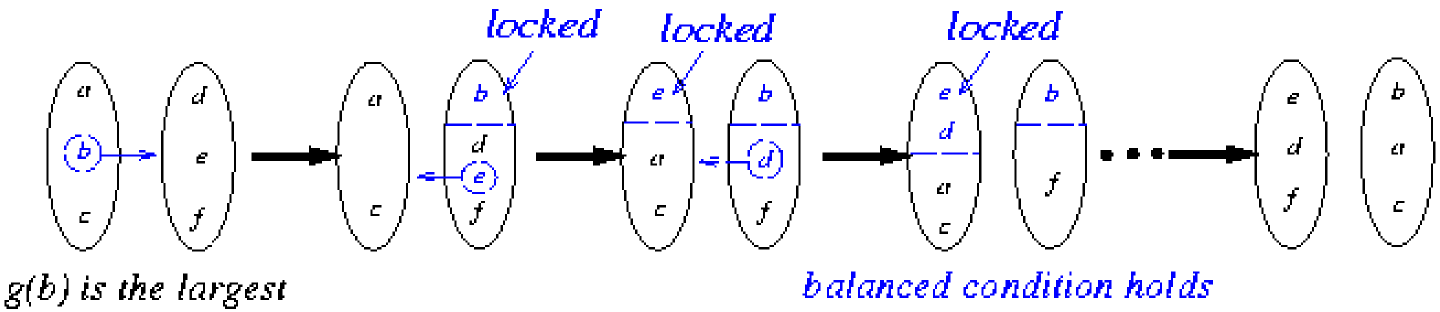
Basic Ideas: Balance and Movement

- Only move a cell at a time, preserving "balance."

$$\frac{|A|}{|A|+|B|} \approx r$$
$$rW - S_{max} \leq |A| \leq rW + S_{max},$$

where $W=|A|+|B|$; $S_{max}=\max_i s(i)$.

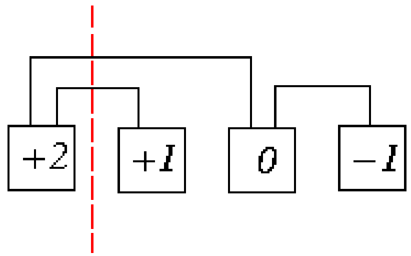
- $g(i)$: gain in moving cell i to the other set, i.e., size of **old** cutset - size of **new** cutset.

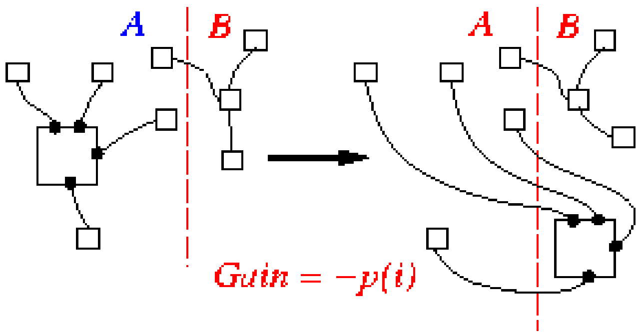


- Suppose $\hat{g}_i$'s: $g(b)$, $g(e)$, $g(d)$, $g(a)$, $g(f)$, $g(c)$ and the largest partial sum is $g(b)+g(e)+g(d)$. Then we should move b, e, d resulting two sets: $\{a, c, e, d\}$, $\{b, f\}$.

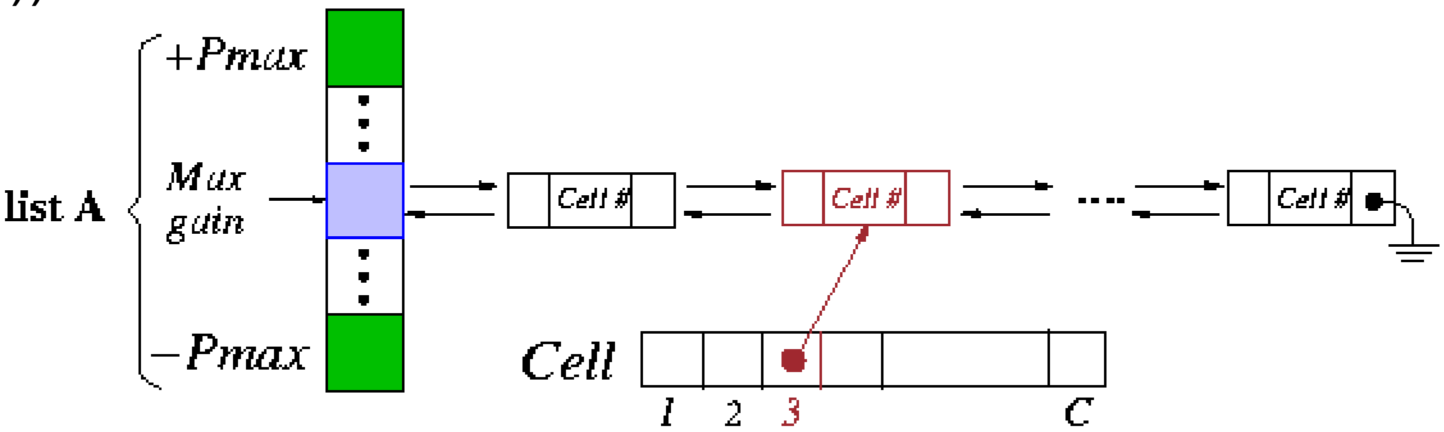
Cell Gains and Data Structure Manipulation

- $-p(i) \leq g(i) \leq p(i)$





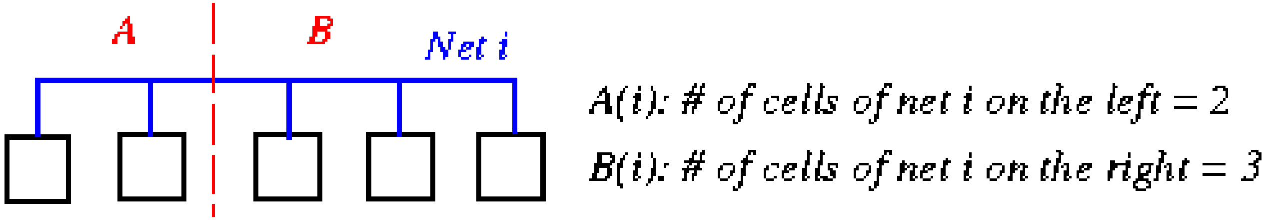
- Two "bucket list" structures, one for set A and one for set B ($P_{max} = \max_i p(i)$).



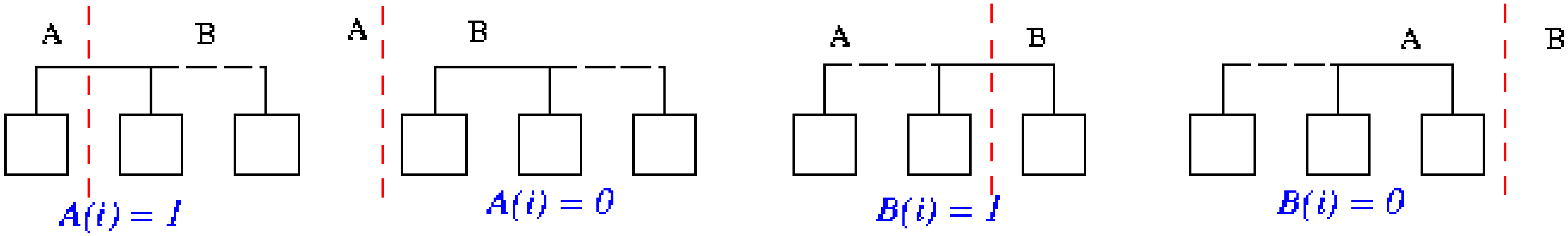
- O(1)-time operations: find a cell with Max Gain, remove Cell i from the structure, insert Cell i into the structure, update $g(i)$ to $g(i)+\Delta$, update the Max Gain pointer.

Net Distribution and Critical Nets

- Distribution of Net *i*: (*A*(*i*), *B*(*i*)) = (2, 3).
  - (*A*(*i*), *B*(*i*)) for all *i* can be computed in *O*(*P*) time.



- **Critical Nets**: A net is critical if it has a cell which if moved will change its cutstate.
  - 4 cases: *A*(*i*) = 0 or 1, *B*(*i*) = 0 or 1.



- **Gain of a cell depends only on its critical nets.**

Computing Initial Gains of All Free Cells

- Initialization of all cell gains requires $O(P)$ time (efficient algorithm shown below):

$g(i) \leftarrow 0;$

$F \leftarrow$ the "from block" of Cell i;

$T \leftarrow$ the "to block" of Cell i;

for each net n on Cell i **do**

  if $F(n)=1$ **then** $g(i) \leftarrow g(i)+1$;

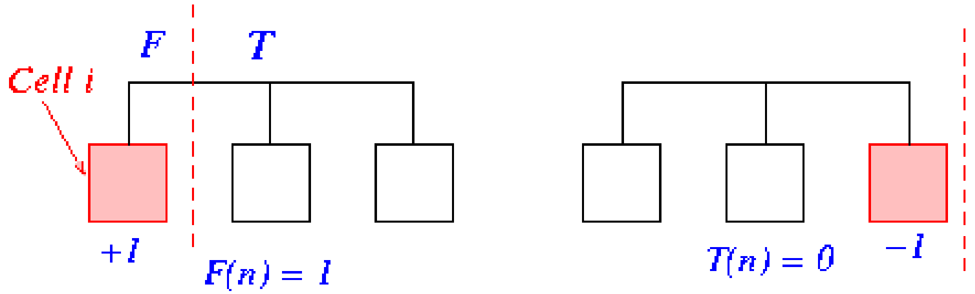
  if $T(n)=0$ **then** $g(i) \leftarrow g(i)-1$;

FS(i): # of nets that have cell i as the only cell in From Block

TE(i): # of nets that contain cell i and are entirely located in From Block

gain(i) = FS(i) – TE(i)

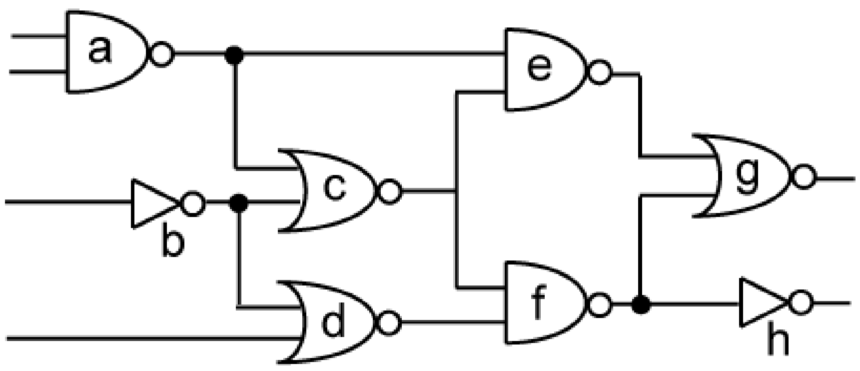
F(n)/T(n): # of cells on net n in the From/To Block



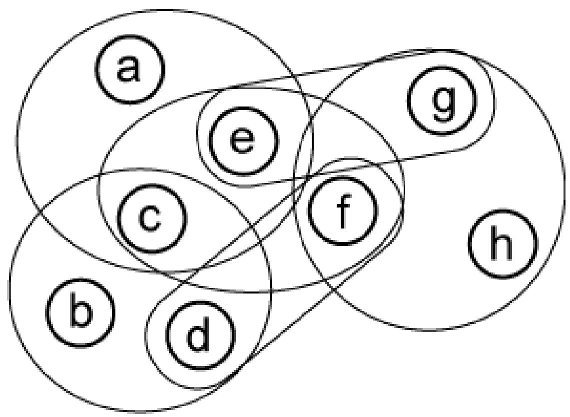
- Will show: Only need $O(P)$ time to maintain all cell gains in one pass.

Fiduccia-Mattheyses Algorithm

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



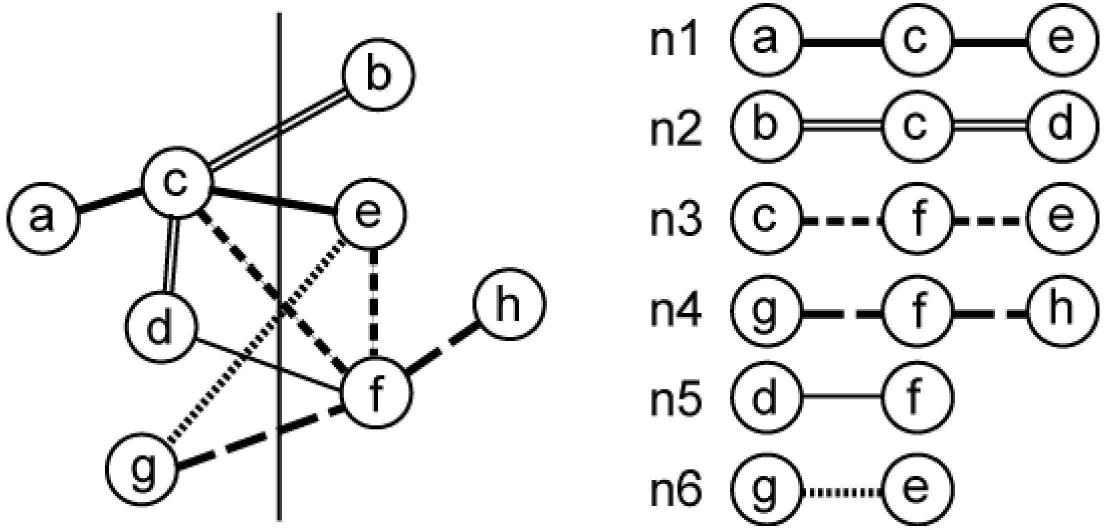
(a)



(b)

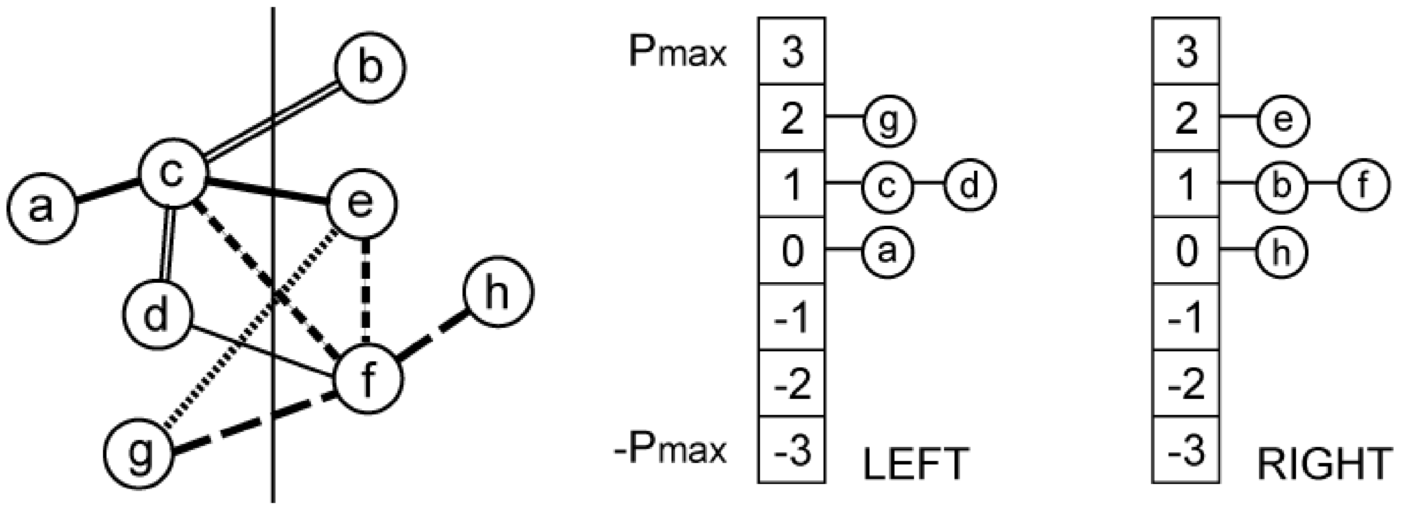
Initial Partitioning

- Random initial partitioning is given.



Gain Computation and Bucket Set Up

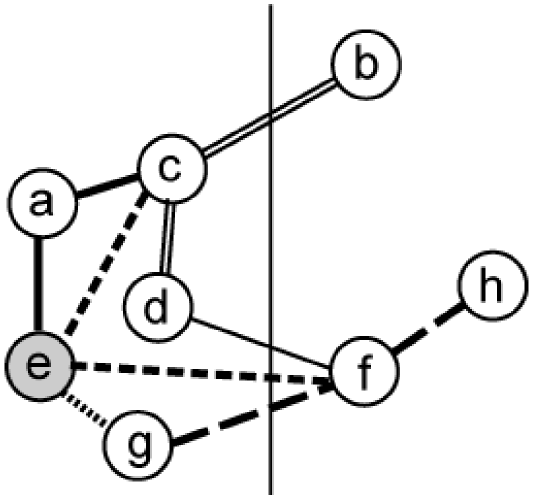
cell c: c is contained in net $n_1 = \{a, c, e\}$, $n_2 = \{b, c, d\}$, and $n_3 = \{c, f, e\}$. n_3 contains c as its only cell located in the left partition, so $FS(c) = 1$. In addition, none of these three nets are located entirely in the left partition. So, $TE(c) = 0$. Thus, $gain(c) = 1$.



FS(x): # of nets that have x as the only cell in LEFT
TE(x): # of nets that contain x and are entirely located in LEFT
gain(x) = FS(x) – TE(x)

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: $gain(a) = FS(a) - TE(a) = 0 - 1 = -1$, $gain(c) = 0 - 1 = -1$, $gain(g) = 1 - 1 = 0$, $gain(f) = 2 - 0 = 2$.

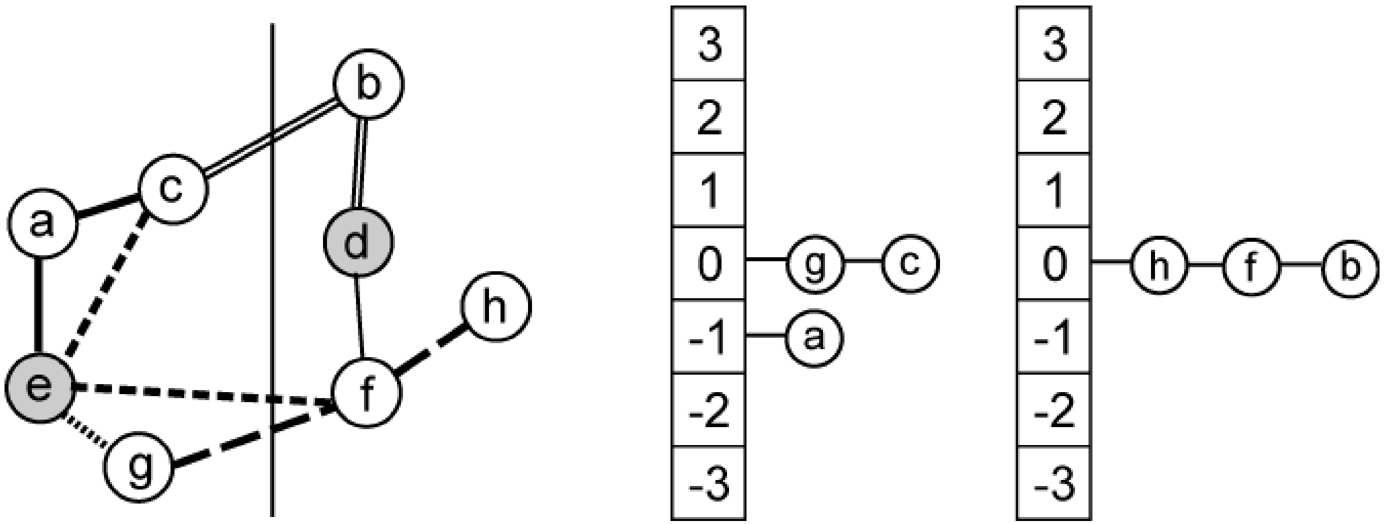


3			
2			
1	d		
0	g		
-1	a	c	
-2			
-3			

3	
2	f
1	b
0	h
-1	
-2	
-3	

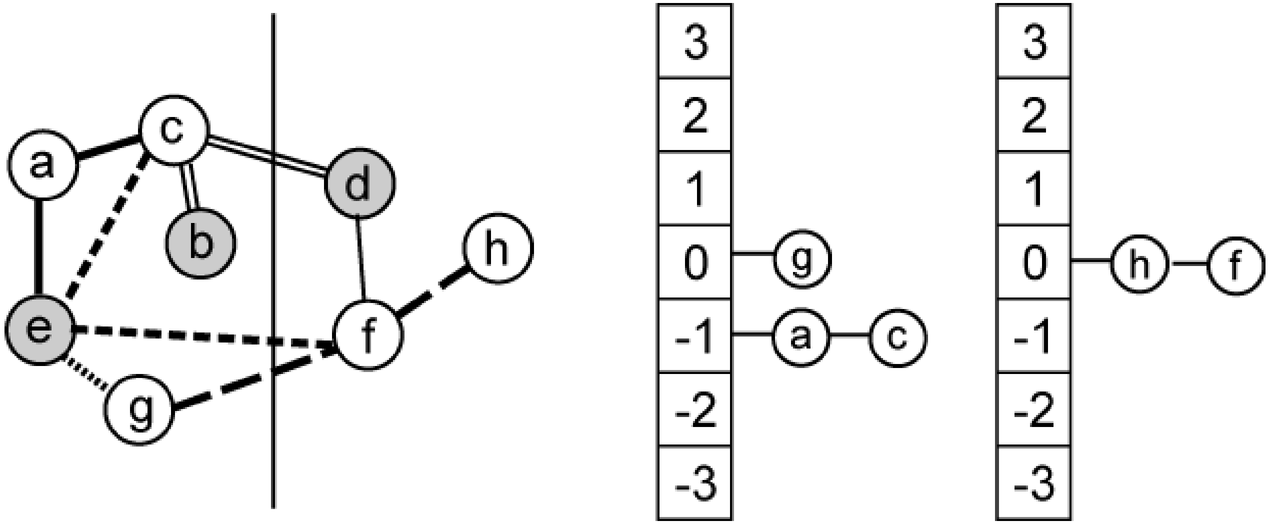
Second Move

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: $gain(b) = 0 - 0 = 0$, $gain(c) = 1 - 1 = 0$, $gain(f) = 1 - 1 = 0$.



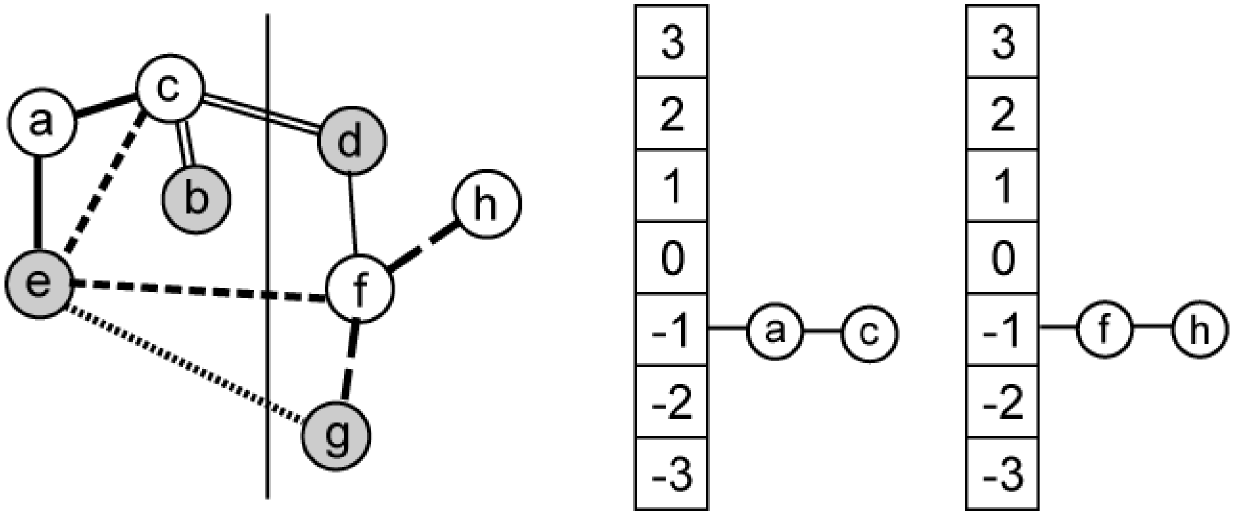
Third Move

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: $gain(c) = 0 - 1 = -1$.



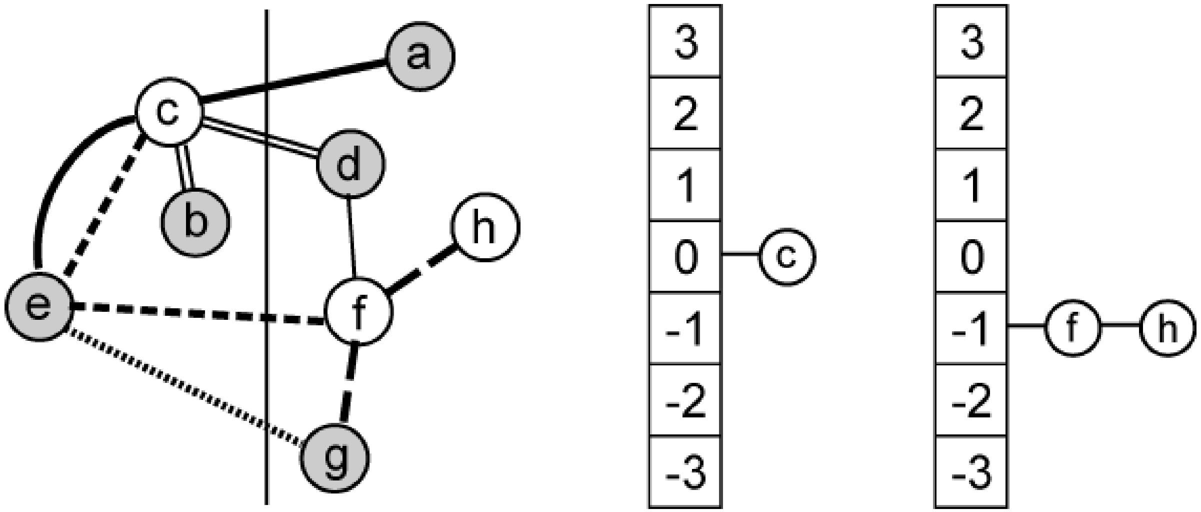
Fourth Move

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: $gain(f) = 1 - 2 = -1$, $gain(h) = 0 - 1 = -1$.



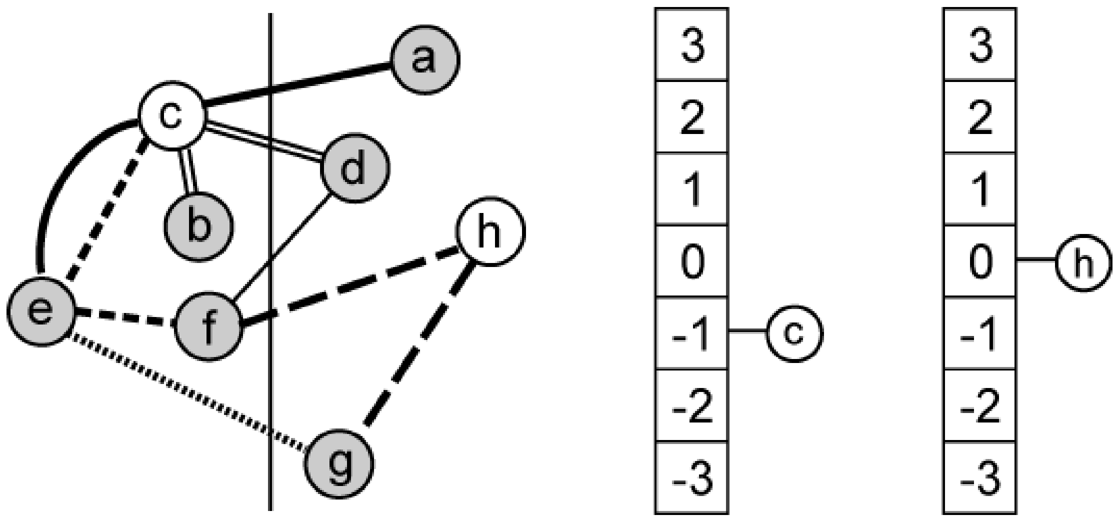
Fifth Move

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



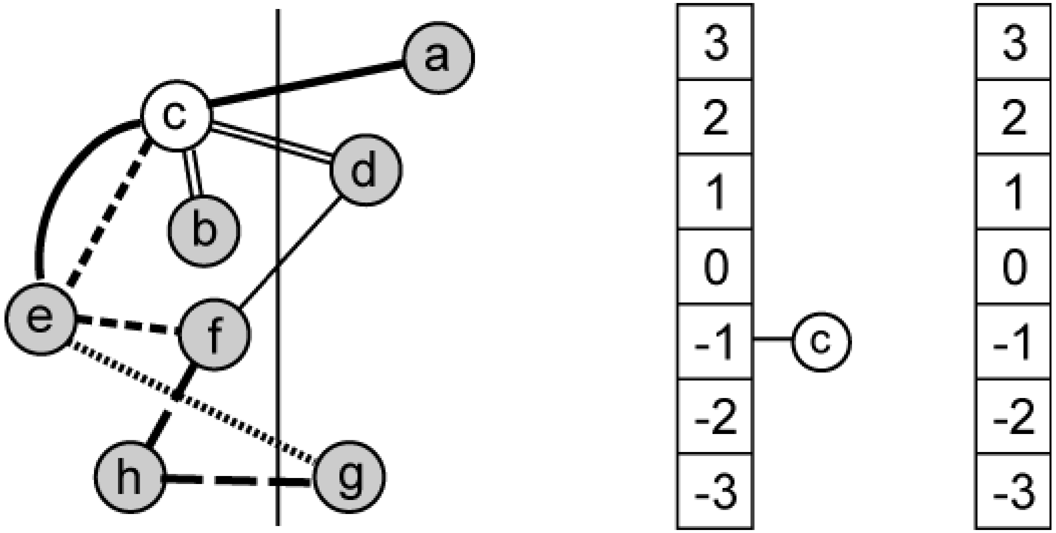
Sixth Move

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: $gain(h) = 0 - 0 = 0$, $gain(c) = 0 - 1 = -1$.



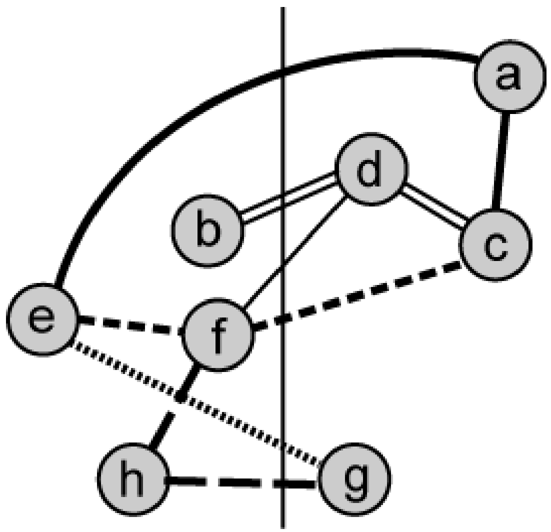
Seventh Move

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



Last Move

move 8: We move c.



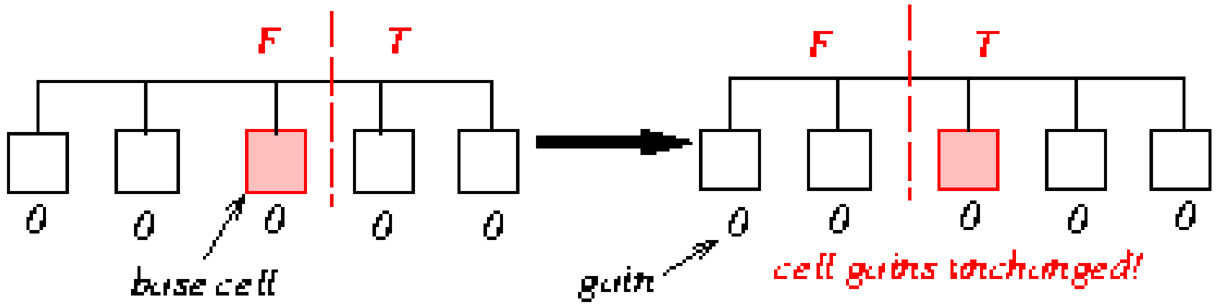
Summary

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

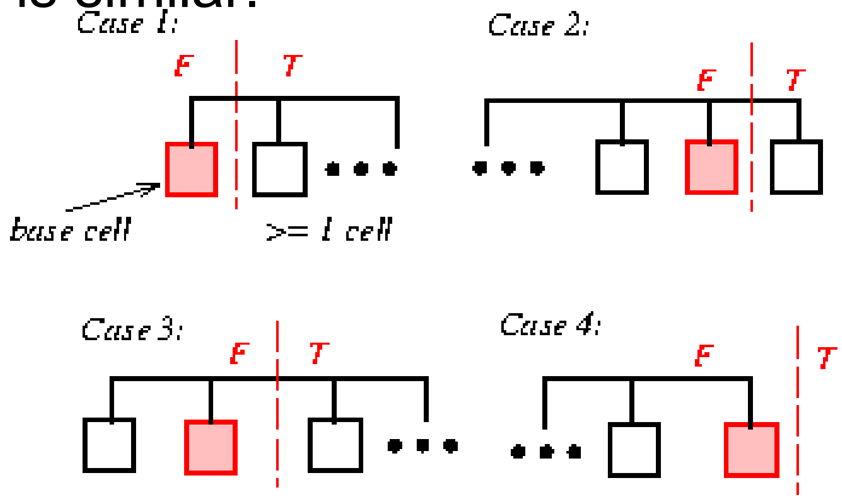
i	cell	$g(i)$	$\sum g(i)$	cutsize
0	-	-	-	6
1	e	2	2	4
2	$\boldsymbol{d}$	**1**	**3**	**3**
3	$\boldsymbol{b}$	**0**	**3**	**3**
4	$\boldsymbol{g}$	**0**	**3**	**3**
5	a	-1	2	4
6	f	-1	1	5
7	h	0	1	5
8	c	-1	0	6

Updating Cell Gains

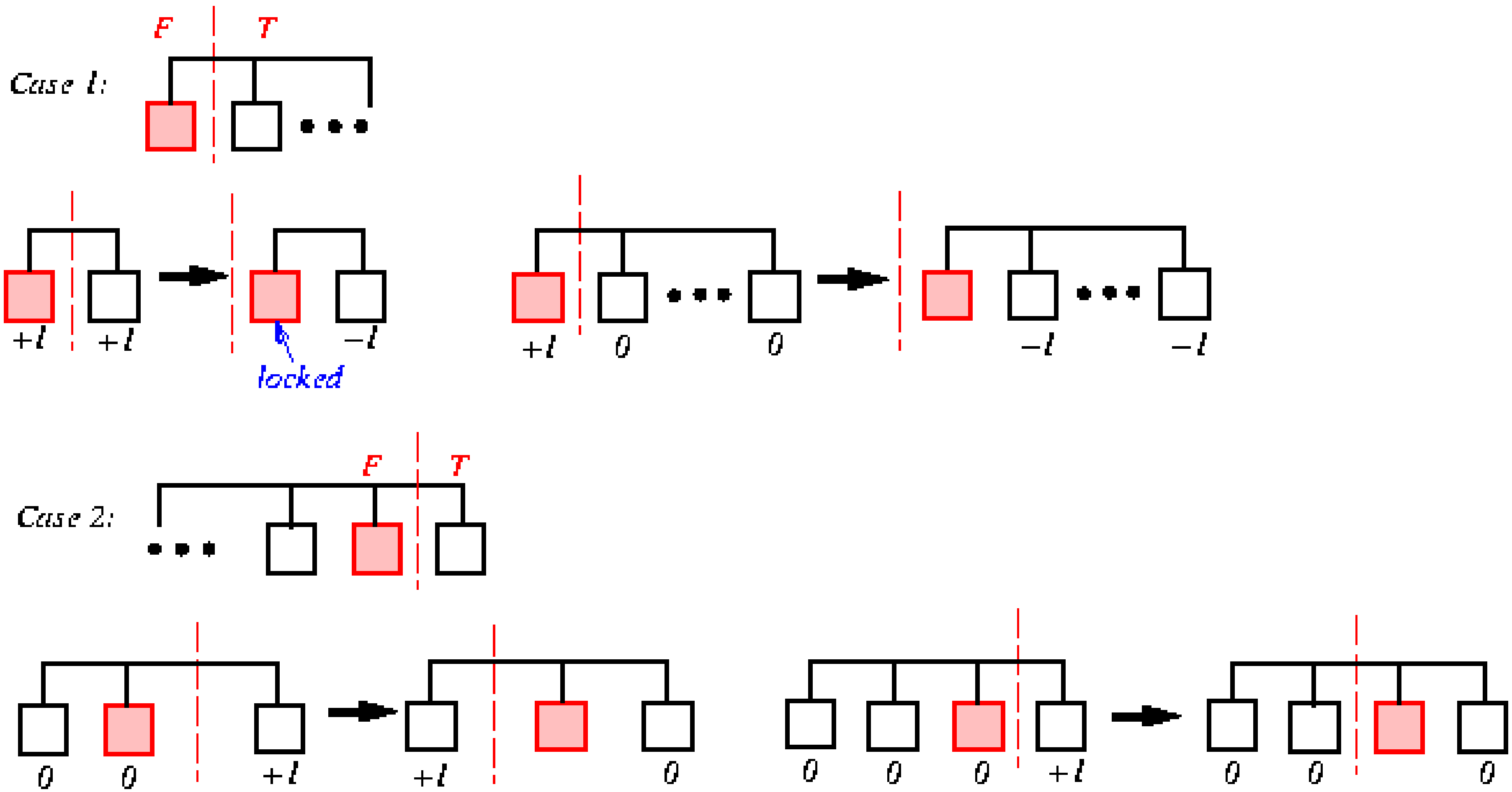
- To update the gains, we only need to look at those nets, connected to the base cell, which are critical **before** or **after** the move.
- **Base cell**: The cell selected for movement from one set to the other.



- Consider only the case where the base cell is in the left partition. The other case is similar.

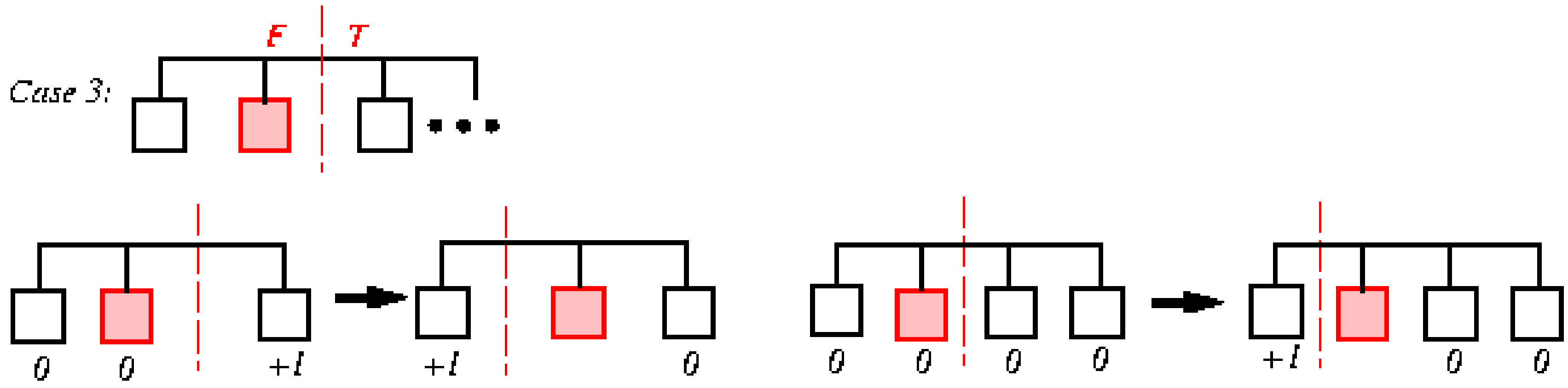


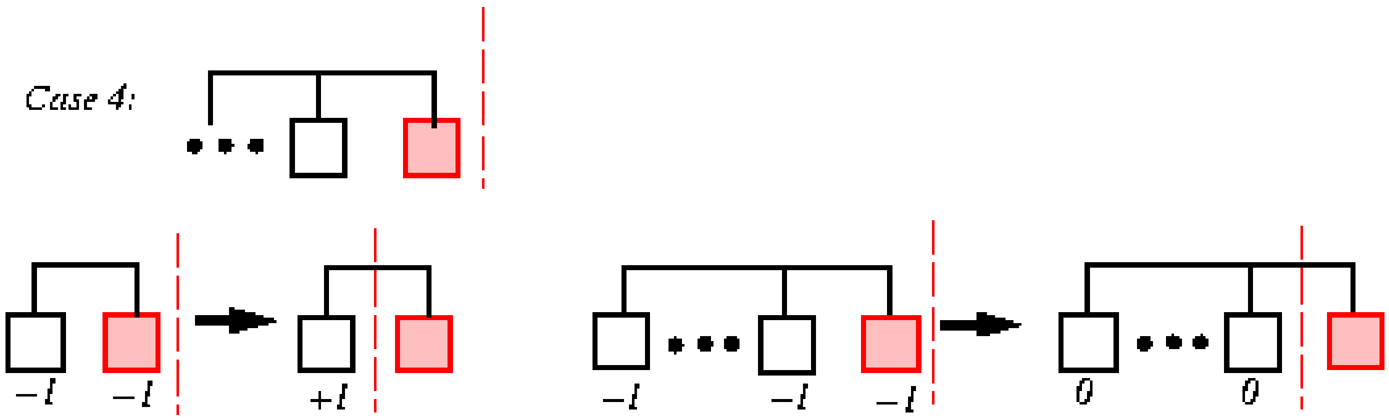
Updating Cell Gains (cont'd)



Slides Courtesy of Prof. H.-R. Jiang

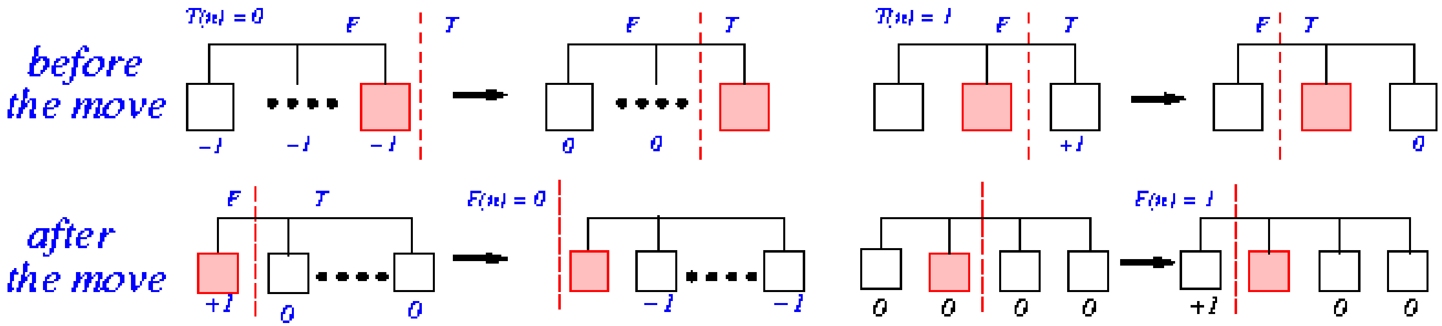
Updating Cell Gains (cont'd)





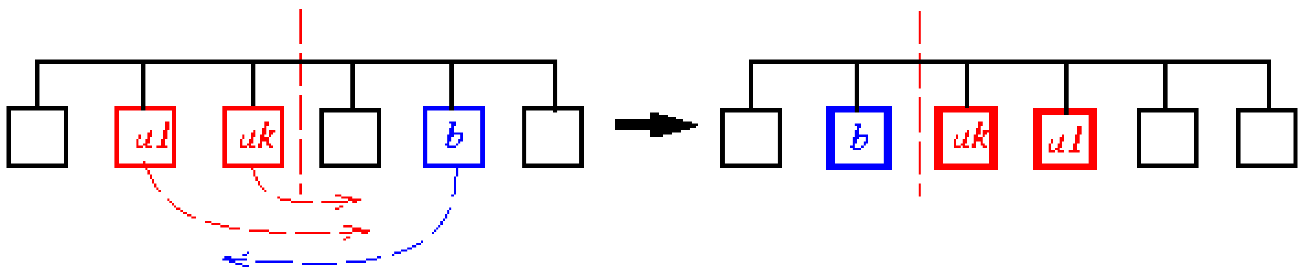
Algorithm for Updating Cell Gains

Algorithm: Update_Gain
1 **begin** /* *move base cell and update neighbors' gains* */
2 $F \leftarrow$ the *Front Block* of the base cell;
3 $T \leftarrow$ the *To Block* of the base cell;
4 Lock the base cell and complement its block;
5 **for** each net n on the base cell **do**
/* check critical nets before the move */
6 **if** $T(n) = 0$ **then** increment gains of all free cells on n
else if $T(n)=1$ **then** decrement gain of the only T cell on n,
if it is free
/* change $F(n)$ and $T(n)$ to reflect the move */
7 $F(n) \leftarrow F(n) - 1$; $T(n) \leftarrow T(n)+1$;
/* *check for critical nets after the move* */
8 **if** $F(n)=0$ **then** decrement gains of all free cells on n
else if $F(n) = 1$ **then** increment gain of the only F cell on n,
if it is free
9 **end**



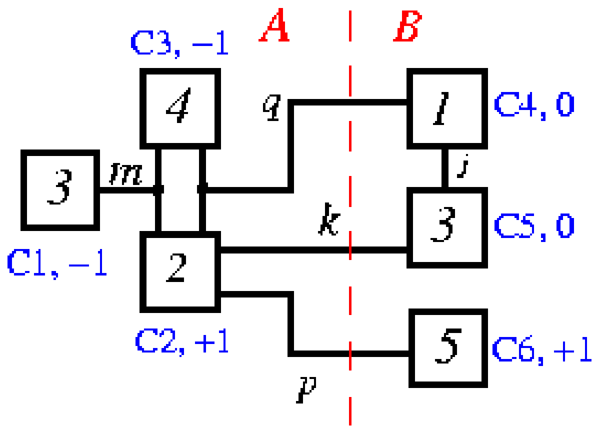
Complexity of Updating Cell Gains

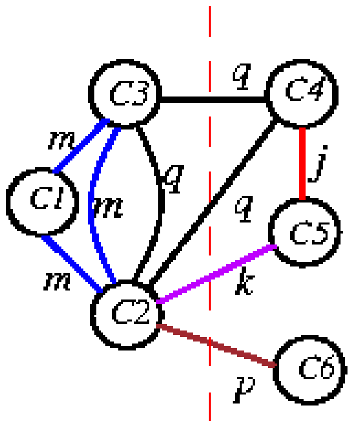
- Once a net has “locked’ cells at both sides, the net will remain cut from now on.
- Suppose we move $a_1, a_2, \ldots, a_k$ from left to right, and then move b from right to left At most only moving $a_1, a_2, \ldots, a_k$ and b need updating!



- To update the cell gains, it takes $O(n(i))$ work for Net i.
- Total time = $n(1)+n(2)+\ldots+n(N) = O(P)$.

F-M Heuristic: An Example





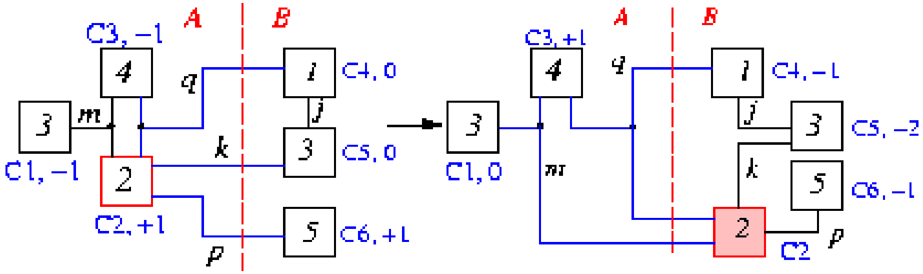
- Computing cell gains: $F(n) = 1 \quad g(i) + 1$; $T(n)=0 \quad g(i) - 1$

-

Cell	m F	m T	q F	q T	k F	k T	p F	p T	j F	j T	$g(i)$
c1	0	−1									−1
c2	0	−1	0	0	+1	0	+1	0			+1
c3	0	−1	0	0							−1
c4			+1	0					0	−1	0
c5					+1	0			0	−1	0
c6							+1	0			+1

- Balanced criterion: $r|V| - S_{max} \leq |A| \leq r|V| + S_{max}$. Let $r = 0.4 \quad |A| = 9, |V|= 18, S_{max} = 5, r|V|=7.2 \quad$ Balanced: $2.2 \leq 9 \leq 12.2$!
- maximum gain: c_2 and balanced: $2.2 \leq 9\text{-}2 \leq 12.2 \quad$ Move c_2 from A to B (use size criterion if there is a tie).

F-M Heuristic: An Example (cont'd)



- Changes in net distribution:

Net	Before move		After move	
	F	T	F'	T'
k	1	1	0	2
m	3	0	2	1
q	2	1	1	2
p	1	1	0	2

- Updating cell gains on critical nets (run Algorithm Update_Gain):

Cells	Gains due to $T(n)$				Gain due to $F(n)$				Gain changes	
	k	m	q	p	k	m	q	p	Old	New
c_1		$+1$							-1	0
c_3		$+1$					$+1$		-1	$+1$
c_4			-1						0	-1
c_5	-1				-1				0	-2
c_6				-1				-1	$+1$	-1

- Maximum gain: c_3 and balanced! ($2.2 \leq 7\text{-}4 \leq 12.2$) $\rightarrow$ Move c_3 from *A* to *B* (use size criterion if there is a tie).

Summary of the Example

Step	Cell	Max gain	$\|A\|$	Balanced?	Locked cell	A	B
0	-	-	9	-	$\emptyset$	1, 2, 3	4, 5, 6
1	c_2	$+1$	7	yes	c_2	1, 3	2, 4, 5, 6
2	c_3	$+1$	3	yes	c_2, c_3	1	2, 3, 4, 5, 6
3	c_1	$+1$	0	no	-	-	-
$3'$	c_6	-1	8	yes	c_2, c_3, c_6	1, 6	2, 3, 4, 5
4	c_1	$+1$	5	yes	c_1, c_2, c_3, c_6	6	1, 2, 3, 4, 5
5	c_5	-2	8	yes	c_1, c_2, c_3, c_5, c_6	5, 6	1, 2, 3, 4
6	c_4	0	9	yes	all cells	4, 5, 6	1, 2, 3

- $\widehat{g_1} = 1, \widehat{g_2} = 1, \widehat{g_3} = -1, \widehat{g_4} = 1, \widehat{g_5} = -2, \widehat{g_6} = 0$ Maximum partial sum G_k = +2, k = 2 or 4.
- Since k=4 results in a better balanced Move c_1, c_2, c_3, c_6 A={6}, B={1, 2, 3, 4, 5}.
- **Repeat the whole process until new $G_k \leq 0$.**