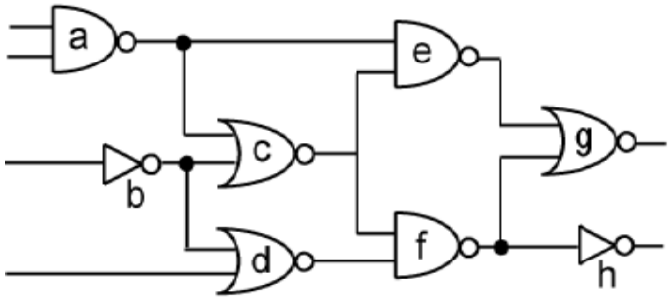
Fiduccia and Mattheyses Algorithm

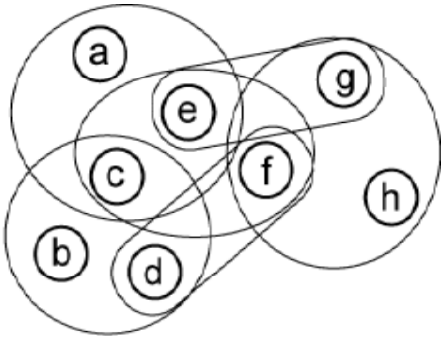
ECE6133

Wanik Cho

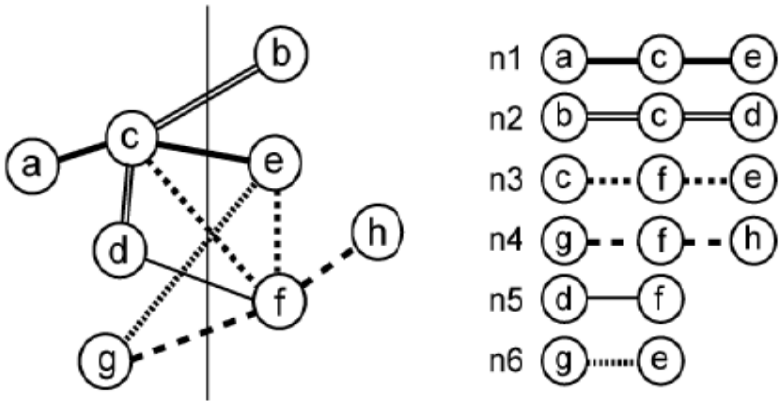
04. 24. 08



gate-level circuit



hypergraph model



initial partitioning

Gain = FS(x) – TE(x)

FS(x) : the # of nets that contain x as the only cell in one part

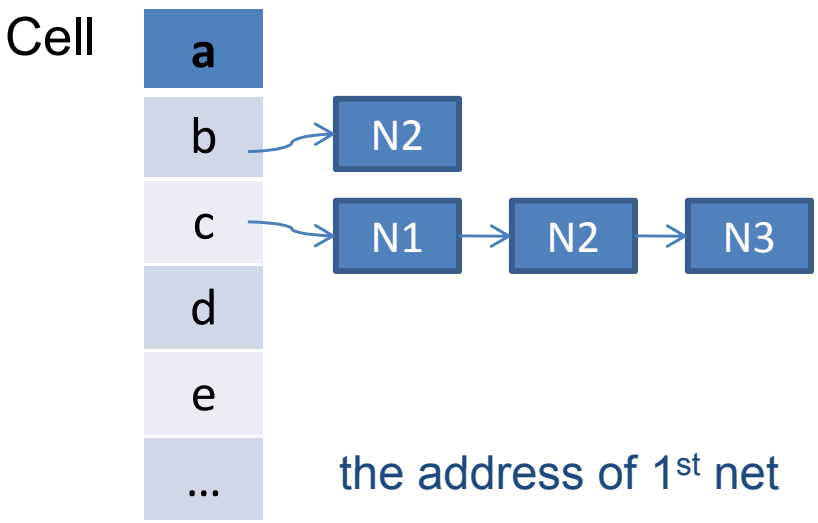
TE(x) : the # of nets that contain x entirely located in one part

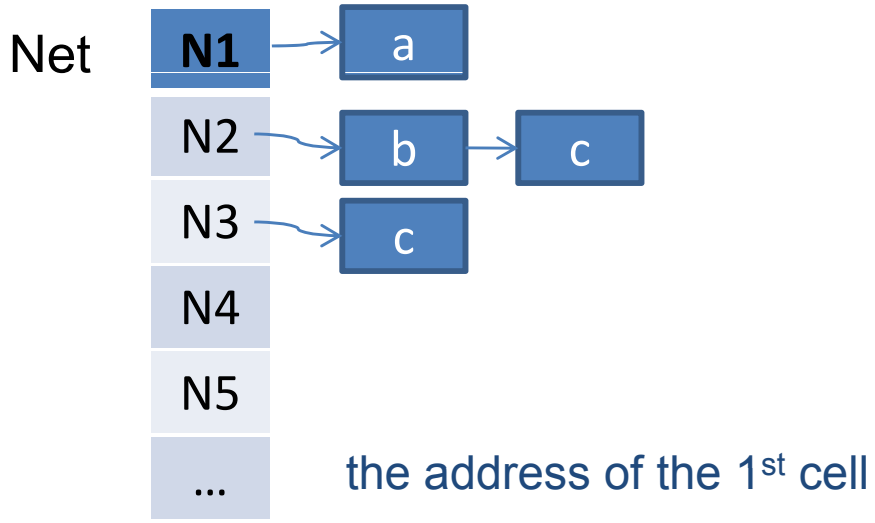
- Example : e cell

FS(e) = 2, TE(e) = 0
Gain = 2 – 0 = 2

Reduce cutsize !!

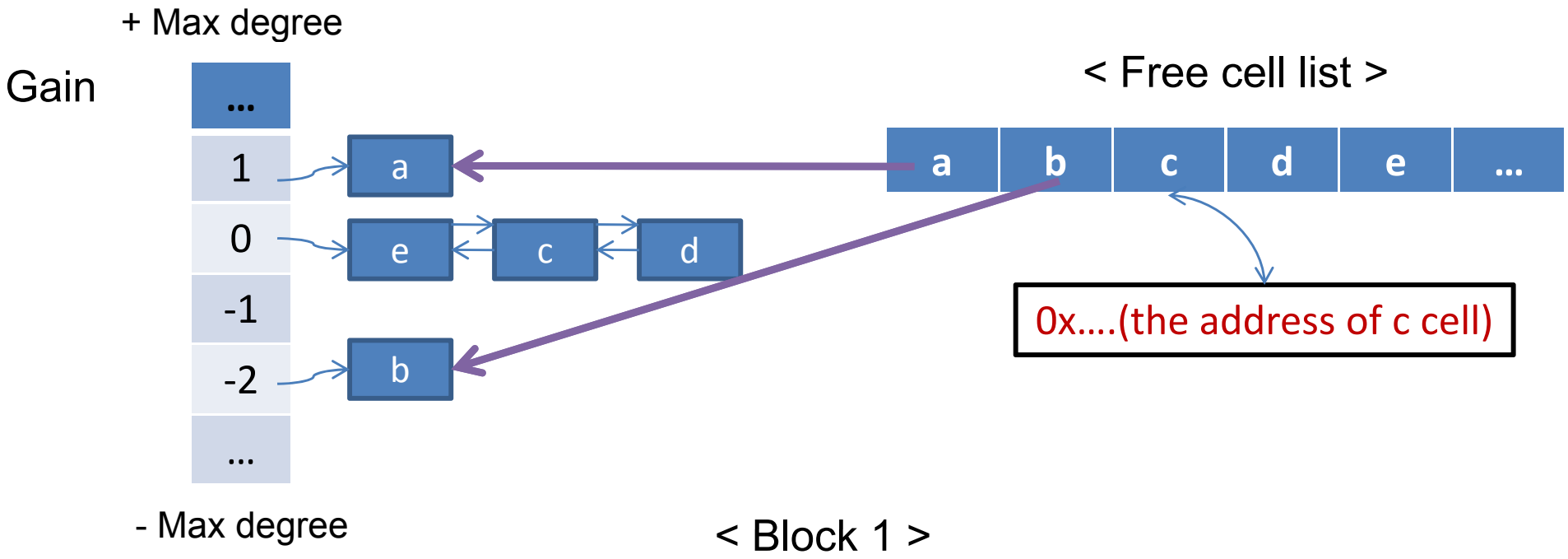
1. Array and single-linked list for nets and cells





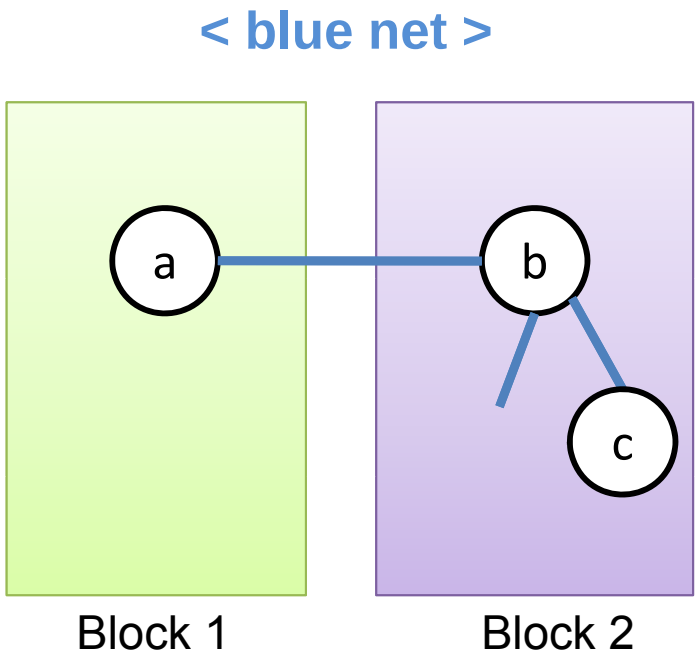

```
FOR each net n =1 … N DO
      FOR each (cell, pin) pair
            (I, j) on net n DO
            IF net n is not at the front of the net-list for cell i
                THEN insert cell i into the cell-list of net n and insert net n into the net-list of cell i
       END FOR
END FOR
```

2. Array and double-linked list for the cell gain



- For 2 blocks, we need such two data structures
- Able to access cells without searching every single array

```
/* compute initial cell gains */
FOR each free cell i DO
        g(i) ← 0
        F    ← the “From block” of cell(i)
        T    ← the “to block” of cell(i)
        FOR each net n on cell i DO
                IF F(n) = 1 THEN increment g(i)
                IF T(n) = 0 THEN then decrement g(i)
        END FOR
END FOR
```



- Cell a : n(F[block1]) = 1 n(T[block2]) = 2 → gain = 1
- Cell b(c) : n(F[block2]) = 2 n(T[blcok1]) = 1 → gain = 0

→ All cells : n(F[block2) = 3 n(T(block1]) = 0 → gain = -1

Move the base cell and update **neighbors**' gains

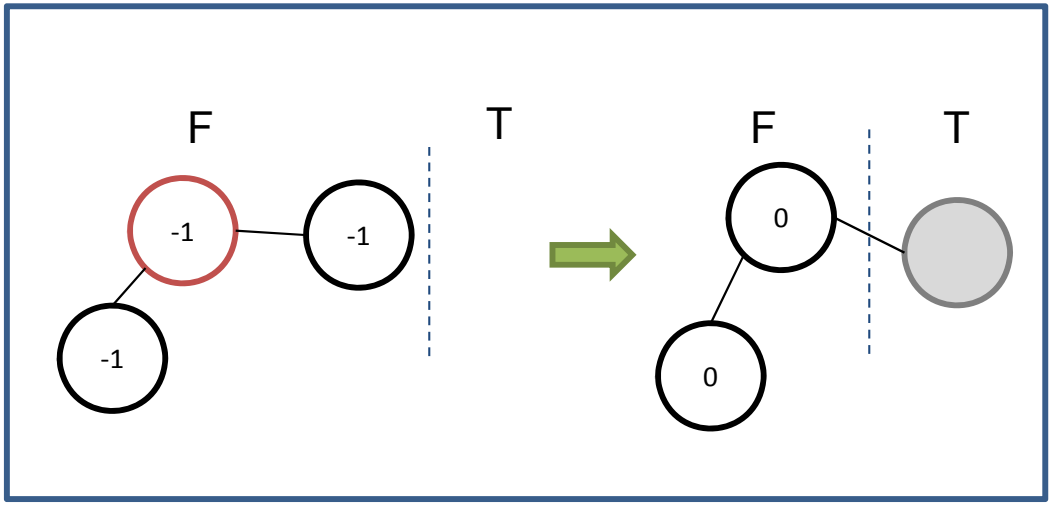
FOR each net n on the base cell DO

/* check for critical nets before the move

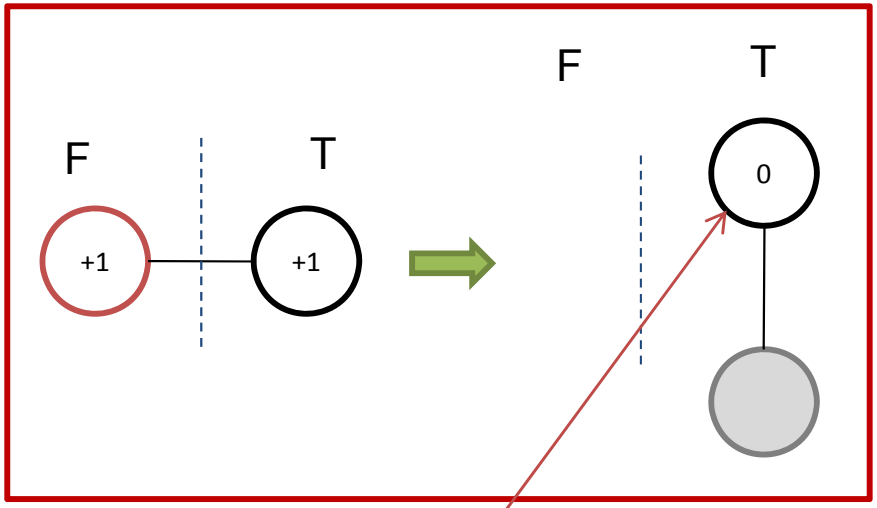
IF T(n) = 0 THEN increment gains of all free cells on net(n)

ELSE IF T(n) = 1 THEN decrement gain of the only T cell on net(n) if it is free

T(n) = 0



T(n) = 1



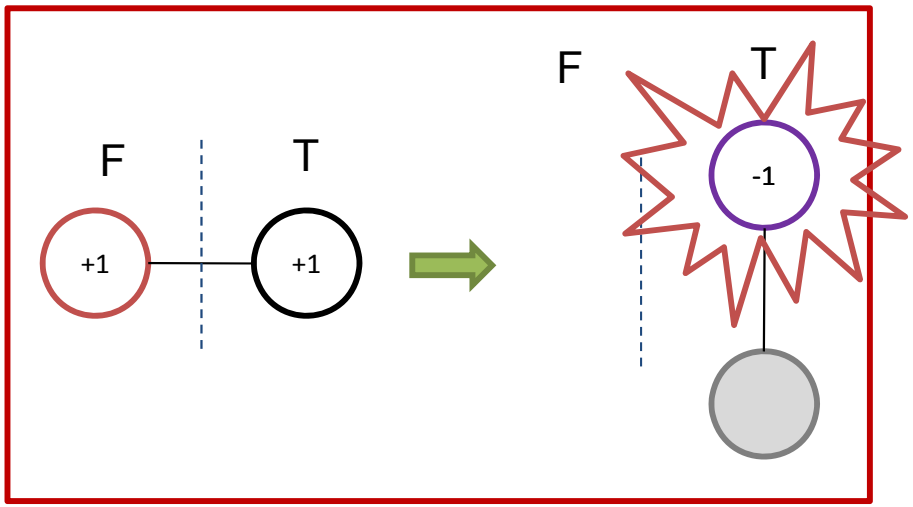
Should be -1

/* change the net distribution to reflect the move*/

/* check for critical nets after the move

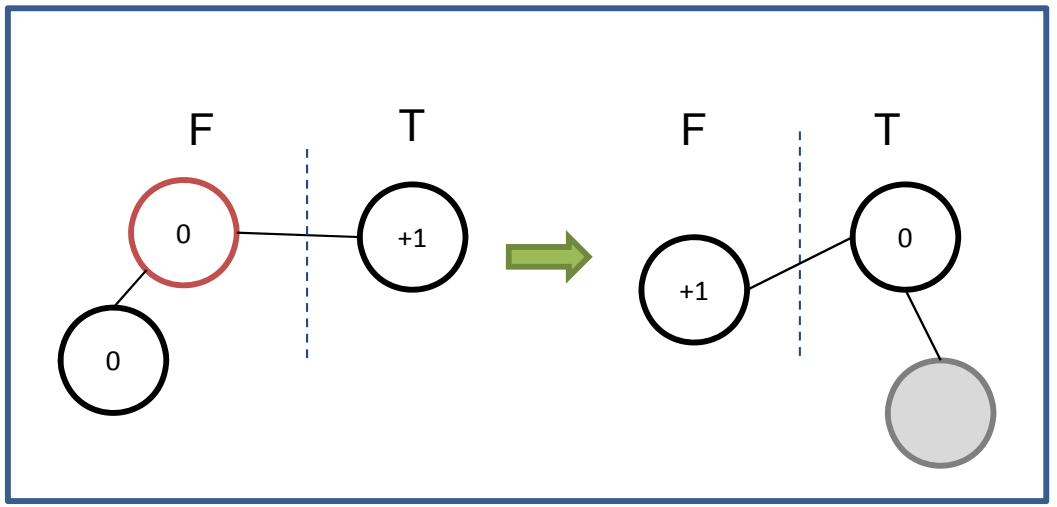
IF F(n) = 0 THEN decrement gains of all free cells on net(n)

F(n) = 0



ELSE IF F(n) = 1 THEN increment gain of the only F cell on net(n) if it is free

F(n) = 1



END FOR

Simulation Result

```
NET LIST INFORMATION
====================
NET TOTAL                = 6
CELL TOTAL               = 8
THE MAXIMUM OF NET SIZE = 3
THE MAXIMUM OF DEGREE    = 3

OPTION
========
INPUT FILE               = ex1.hgr
AREA SKEW                = 0.05
THE NUMBER OF RUNS       = 5
CONSTRAIN                = 0
[ 4 : 4 ]
pass: 0 >> gain: 1 cut: 4 -> 3 at: 1  mcnt: 7
pass: 1 >> gain: 0 cut: 3 -> 3 at: 0  mcnt: 6
**run1: area: [ 5 : 3 ] gain: 1(4->3)
pass: 0 >> gain: 1 cut: 4 -> 3 at: 1  mcnt: 7
pass: 1 >> gain: 0 cut: 3 -> 3 at: 0  mcnt: 6
**run2: area: [ 5 : 3 ] gain: 1(4->3)
pass: 0 >> gain: 0 cut: 3 -> 3 at: 0  mcnt: 7
**run3: area: [ 5 : 3 ] gain: 0(3->3)
pass: 0 >> gain: 1 cut: 4 -> 3 at: 2  mcnt: 7
pass: 1 >> gain: 0 cut: 3 -> 3 at: 0  mcnt: 7
**run4: area: [ 4 : 4 ] gain: 1(4->3)
pass: 0 >> gain: 2 cut: 5 -> 3 at: 2  mcnt: 7
pass: 1 >> gain: 0 cut: 3 -> 3 at: 0  mcnt: 7
**run5: area: [ 4 : 4 ] gain: 2(5->3)
wcho7@ccblincad23.ece.gatech.edu>
```

```
OPTION
========
INPUT FILE               = ex1.hgr
AREA SKEW                = 0.05
THE NUMBER OF RUNS       = 1
CONSTRAIN                = 0
[ 4 : 4 ]
** initial net_array **
1 5 -> 1 3 -> 1 1 ->
1 4 -> 1 3 -> 0 2 ->
1 5 -> 0 6 -> 1 3 ->
0 8 -> 0 6 -> 0 7 ->
0 6 -> 1 4 ->
1 5 -> 0 7 ->

pass: 0 >> gain: 1 cut: 4 -> 3 at: 1  mcnt: 7
** final net_array **
1 5 -> 1 3 -> 1 1 ->
0 4 -> 1 3 -> 0 2 ->
1 5 -> 0 6 -> 1 3 ->
0 8 -> 0 6 -> 0 7 ->
0 6 -> 0 4 ->
1 5 -> 0 7 ->

<Moved cells>
4 -> 2 -> 5 -> 8 -> 3 -> 7 -> 1 ->
```

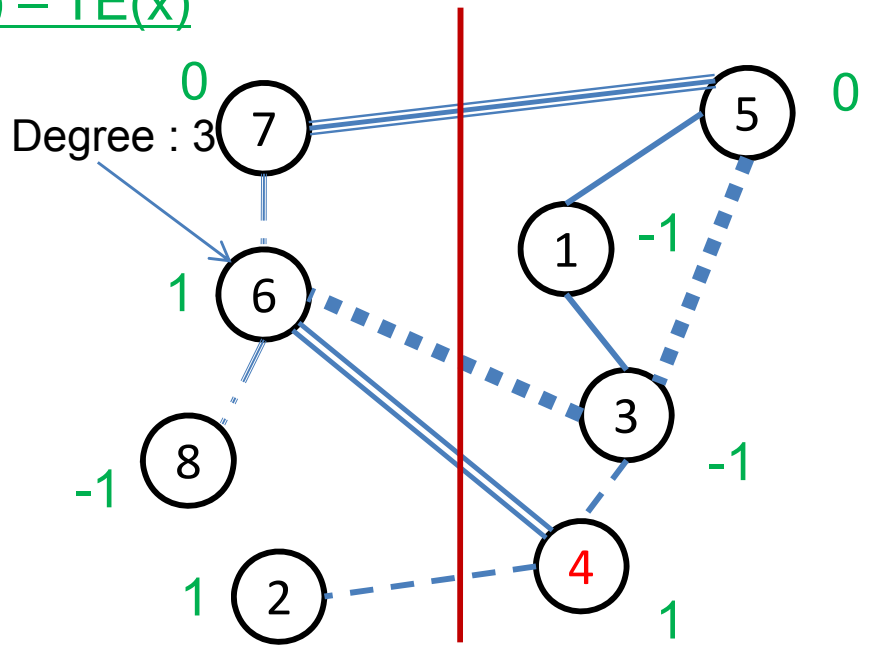
```
** initial net_array **
1 5 -> 1 3 -> 1 1 ->
1 4 -> 1 3 -> 0 2 ->
1 5 -> 0 6 -> 1 3 ->
0 8 -> 0 6 -> 0 7 ->
0 6 -> 1 4 ->
1 5 -> 0 7 ->
```

```
** Bucket Array (block1) **
Cannot print
Cannot print
6 -> 2
7
8
Cannot print
Cannot print
```

```
** Bucket Array (block2) **
Cannot print
Cannot print
4
5
3 -> 1
Cannot print
Cannot print
```

```
** cell_array **
0 1 ->
0 2 ->
0 3 -> 0 2 -> 0 1 ->
0 5 -> 0 2 ->
0 6 -> 0 3 -> 0 1 ->
0 5 -> 0 4 -> 0 3 ->
0 6 -> 0 4 ->
0 4 ->
```

Gain = FS(x) – TE(x)

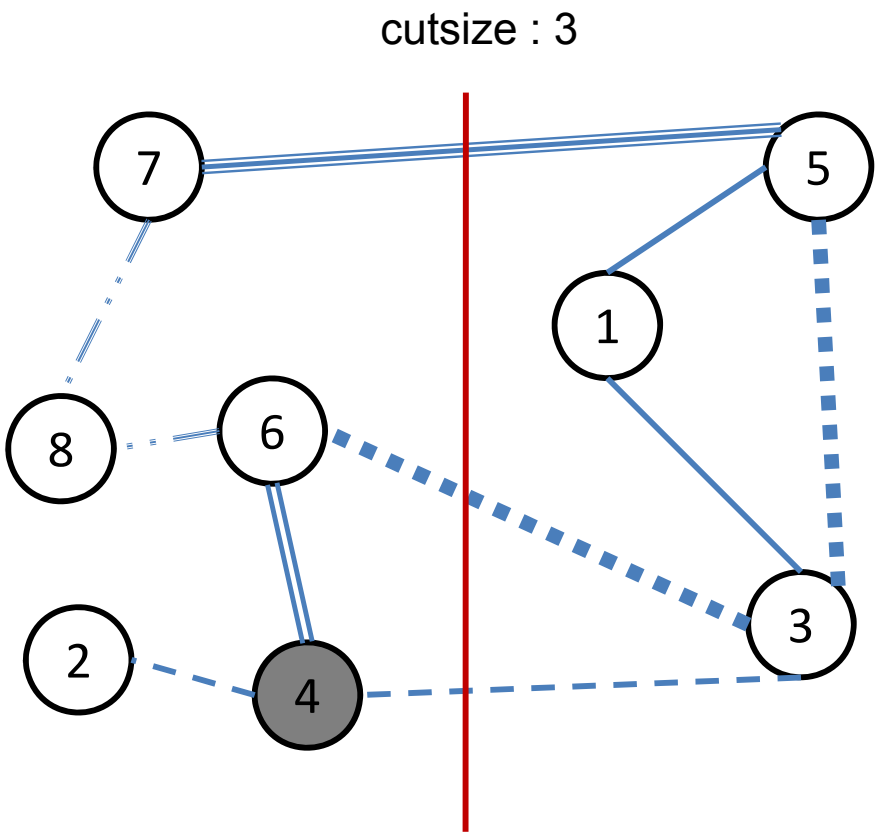


cutsize : 4

- Move cell 4-

```
** After Cell 4 Bucket Array (block1)**
Cannot print
Cannot print
Cannot print
2 -> 7
6 -> 8
Cannot print
Cannot print
```

```
** After Cell 4 Bucket Array (block2)**
Cannot print
Cannot print
Cannot print
3 -> 5
1
Cannot print
Cannot print
```

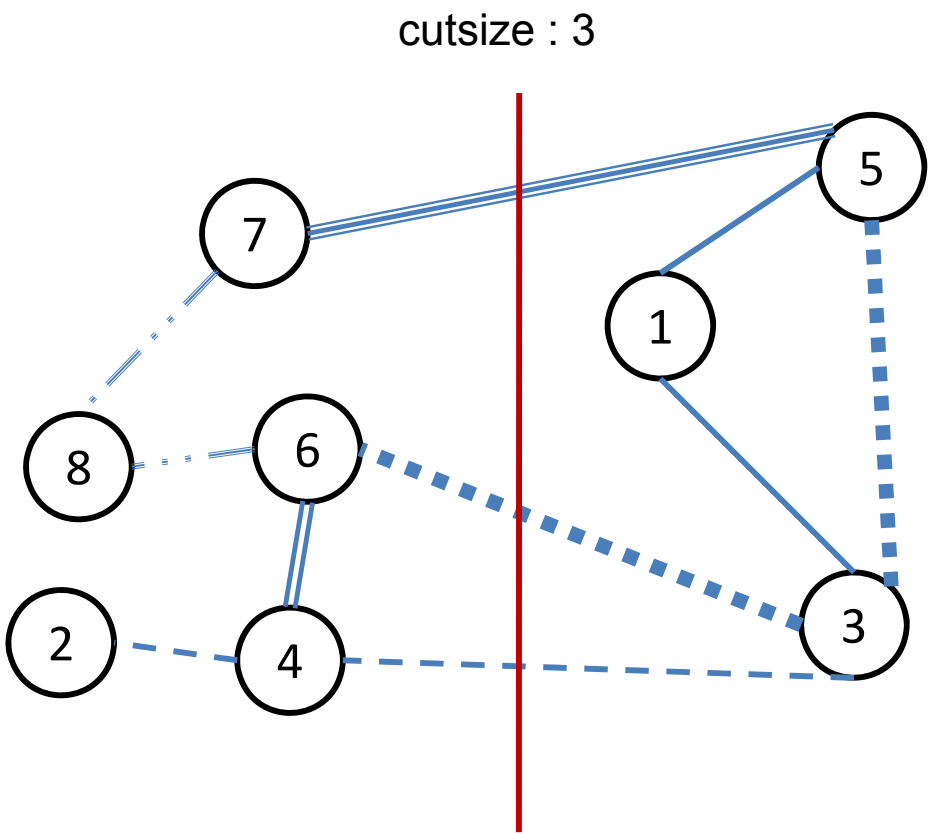


- Cell 4 disappears in bucket
- The cell is locked

- At best cutsize

```
** initial net_array **
1 5 -> 1 3 -> 1 1 ->
1 4 -> 1 3 -> 0 2 ->
1 5 -> 0 6 -> 1 3 ->
0 8 -> 0 6 -> 0 7 ->
0 6 -> 1 4 ->
1 5 -> 0 7 ->

pass: 0 >> gain: 1 cut: 4 -> 3 at: 1  mcnt: 7
** final net_array **
1 5 -> 1 3 -> 1 1 ->
0 4 -> 1 3 -> 0 2 ->
1 5 -> 0 6 -> 1 3 ->
0 8 -> 0 6 -> 0 7 ->
0 6 -> 0 4 ->
1 5 -> 0 7 ->
```



Simulation Result

Input Files	Total net	Total cell	Max degree	Initial cutsize	Final cutsize	Gain
avq.large.hgr	25384	25178	7	15089	785	14304
ibm01.hgr	14111	12752	39	9195	335	8860
ibm04.hgr	31970	27507	526	20545	1007	19538
ibm07.hgr	48117	45926	98	32286	1562	30724
ibm10.hgr	75196	69429	137	50769	2142	48627
ibm13.hgr	99666	84199	180	66130	3481	62649
ibm16.hgr	190048	183484	177	130118	3795	126323
industry2.hgr	13419	12637	315	8056	681	7375
s13207P.hgr	8651	8772	5	4700	114	4586
biomedP.hgr	5742	6514	6	3414	120	3294
p1.hgr	902	833	9	567	68	499
p2.hgr	3029	3014	9	1957	257	1700
structP.hgr	1920	1952	4	1203	54	1149
fract.hgr	147	149	7	106	11	95

Under Linux, Run Time = 5, Skew = 0.05

Thank you for your attention!