

## **ECE-256a**

### **lecture 6**

Malgorzata Marek-Sadowska  
Electrical and Computer Engineering Department  
Engineering I, room 4111  
mms@ece.ucsb.edu.

ECE 256A

1

## **References**

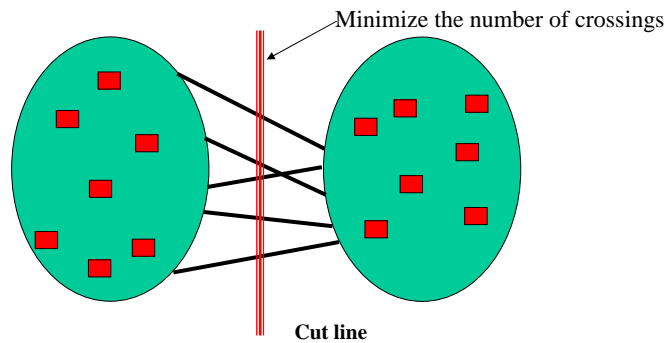
- C.M.Fiduccia and R.M. Mattheyses, "A Linear-Time Heuristic for Improving Network Partitions", Design Automation Conference, 1982, pp. 175-181.
- B.W. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs" Bell System Technical Journal, Vol. 49, Feb. 1970, pp. 291-307.

ECE 256A

2

## Partitioning

- Partition the gates between two regions so that
  - Capacity (# of gates allowed) on each side is not exceeded
  - Cost (for example, the number) of wires across the cut is minimized
- Classical problem: bi-partitioning

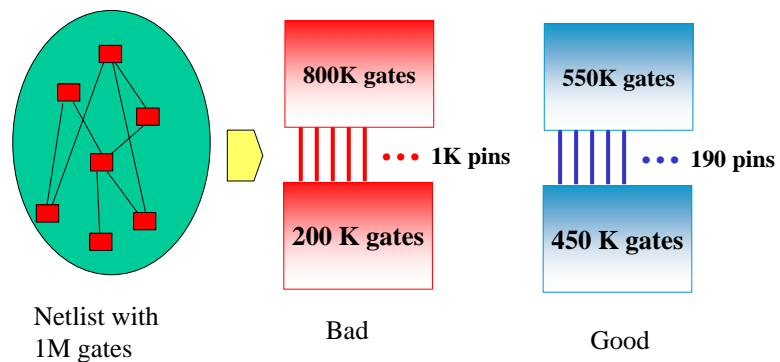


ECE 256A

3

## Example

- Circuit has capacity constraint  $\leq 500\text{K}$  gates/partition
- I/O constraints  $\leq 200$  pins (connections on the boundary)



ECE 256A

4

## GRAPH PARTITIONING

G: a graph of n nodes each of size s(i)

P: a positive integer such that  $\forall i \ 0 < s(i) \leq P$

C:  $(c_{ij}) \ i, j = 1, 2, \dots, n$  be a weighted connectivity matrix

A k-way partition of G: a set of non-empty, disjoint subsets of nodes of G:

$V_1, V_2, \dots, V_k$  such that  $\bigcup_{i=1}^k V_i = V$

A partition is admissible, if

$\forall i \ |V_i| \leq P$

Cost = cost of external edges, between partitions.

ECE 256A

5

## Exact solutions

Enumerate all partitions.

Suppose  $\forall i \ s(i) = 1$  and  $kP = n$

There are:  $\binom{n}{p}$  ways to choose the 1-st subset

$\binom{n-p}{p}$  ways to choose the 2nd subset

$\vdots$

The ordering of sets is not important, so the number of different partitions is

$$\frac{1}{k!} \binom{n}{p} \binom{n-p}{p} \dots \binom{2p}{p} \binom{p}{p}$$

For  $n = 40$

$P = 10$

$k = 4$

# cases  $> 10^{20}$

ECE 256A

6

### Heuristics which did not work:

#### 1. Random solutions

Low probability of finding a good solution.

Experiments with 2-way partitions of 32 node graphs indicate 2-5 optimal partitions one of  $\frac{1}{2} \binom{32}{16}$  partitions; probability of success on any trial is less than  $10^{-7}$

#### 2. Max Flow-Min Cut

No way to control the sizes of partitions.

#### 3. Clustering - difficulties in systematic assignments of nodes that do not obviously belong to any particular set.

ECE 256A

7

## 2-way partition

Let  $c$ (# of cells) =  $2n$ ,

$C = (C_{ij})$  is a cost matrix;  $i, j = 1, 2, \dots, 2n$ .

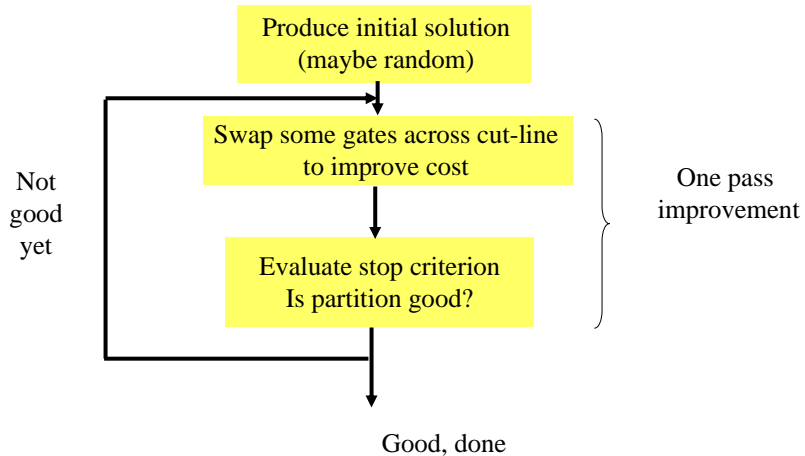
$C_{ii} = 0 \forall_i$

We wish to partition  $S$  ( the set of  $2n$  cells and 2-pin interconnects described by  $C$ ) into 2 sets:  $A$  and  $B$ , each containing  $n$  cells. So  $\forall_i s(i) = 1, |A| = |B| = n$ . External cost  $T = \sum_{A \times B} c_{ab}$

ECE 256A

8

## Solution approach



ECE 256A

9

## Outline of the Kernighan-Lin method.

1. Start with any partition  $A, B$  of  $S$ .
2. Try to decrease the initial external cost  $T$  by a series of interchanges of subsets of  $A$  and  $B$ .
3. When no further improvement is possible, the resulting partition  $A', B'$  is locally minimum with respect to the algorithm.
4. The process may be repeated with different starting partitions.

ECE 256A

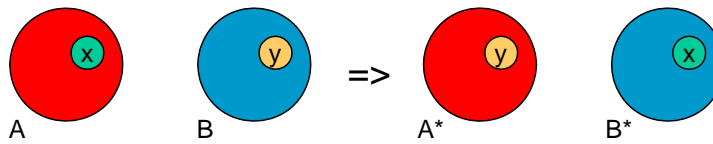
10

Let  $A^*, B^*$  be a minimum cost 2-way partition;  $A, B$  is an arbitrary 2-way partition.

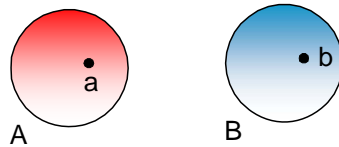
$\exists X \subset A \wedge \exists Y \subset B$  such that  $|X| < |Y| \leq \frac{n}{2}$  that

$$A^* = A - X + Y$$

$$B^* = B - Y + X$$



How to identify  $X$  and  $Y$ ?



External cost of  $a \in A$ :

$$E_a = \sum_{y \in B} c_{ay}$$

Internal cost of  $a \in A$ :

$$I_a = \sum_{x \in A} c_{ax}$$

$$E_b = \sum_{x \in A} c_{bx}$$

$$I_b = \sum_{y \in B} c_{by}$$

Let

$$D_z = E_z - I_z \quad \forall z \in S$$

Lemma. Consider any  $a \in A$ ,  $b \in B$ . If  $a$  and  $b$  are interchanged, the gain (cost reduction) is  $D_a + D_b - 2c_{ab}$



Let  $z$  be the total cost due to all connections between  $A$  and  $B$  that do not involve  $a$  or  $b$ .

Then:

$$T = z + E_a + E_b - c_{ab}$$

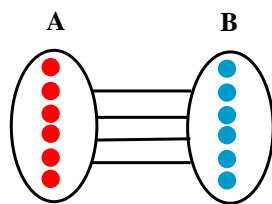
exchange  $a$  and  $b$ :

$$T' = z + I_a + I_b + c_{ab}$$

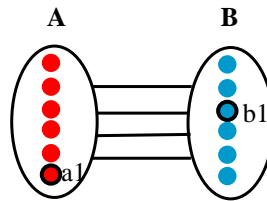
gain = old cost - new cost =  $T - T' =$

$$D_a + D_b - 2c_{ab}$$

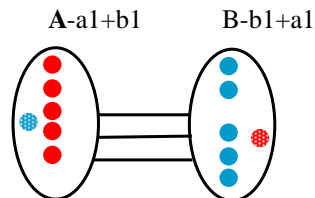
## K&L Improvement Procedure



1. Start with any partition



2. Identify  $a_1$  in  $A$  and  $b_1$  in  $B$  so that swapping them will give maximum gain



3. Swap  $a_1$  and  $b_1$  and lock them in place, so they can't be swapped again

4. Continue identifying  $a_2, b_2, \dots$

## K&L Algorithm: critical ideas

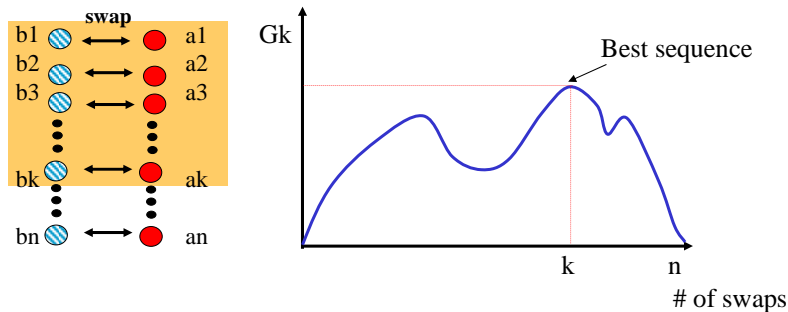
- **Gain**
  - Gain is the change in cost that results from swapping one gate in the A-side with one gate in the B-side
  - Compute it as  $\sum_{cut} C_{ij} \text{ (after swap)} - \sum_{cut} C_{ij} \text{ (before swap)}$
- **Greedy decision**
  - Make the best next swap
  - Do this swap even if it's negative
    - Biggest positive gain
    - Smallest (closest to zero) negative gain
- **Do all n swaps**

ECE 256A

15

## K&L: Picking Swap Sequence

- Facts:
  - Gain from doing k swaps in sequence is  $G_k = \sum_{i=1}^k g_i$
  - $G_k$  is not monotonic function of k

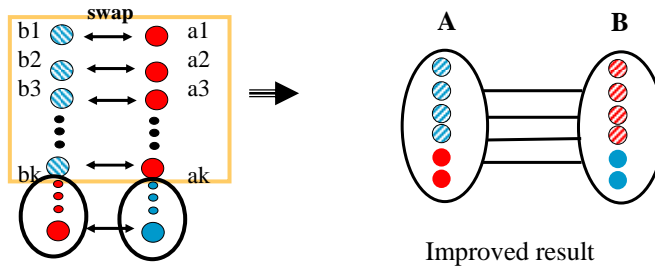


ECE 256A

16

## K&L: Doing the swaps

- Interpretation:
  - We will do only those k swaps, since they maximize gain



ECE 256A

17

## The exchange algorithm.

1. Compute D values for all elements of S.
2. Choose  $a_i \in A$  and  $b_i \in B$  such that  $g_i = D_{a_i} + D_{b_i} - 2c_{a_i b_i}$  is maximum.
3. Move  $a_i$  to B,  $b_i$  to A and lock them. Store  $g_i, (a_i, b_i)$ .
4. If A,B have any movable (unlocked) elements do
  - a. Update D values
  - b. go to step 2

end
- else go to 5.
5. Choose k to maximize partial sum  $\sum_{i=1}^k g_i = G$ .

ECE 256A

18

6. Move the first  $k$  elements from the sequence  $(a_1, b_1)(a_2, b_2) \dots (a_x, b_x) \dots (a_n, b_n)$  to the other side of the partitions.
7. Treat the resulting partition as a new partition and repeat the process until the partitions can not be improved ( $k = 0$ )

## **A Linear-Time Heuristic for Improving Network Partitions**

Fiduccia and Mattheyses algorithm.

### Problem:

Given a network consisting of a set of modules connected by a set of nets, the mincut partitioning problem is to find a partition of the set of modules into 2 blocks A and B such that the number of nets having modules in both blocks is minimal. In general, size constraints are imposed on A and B.

The network consists of  $c$  modules (cells) and  $N$  nets.

- \* A net is defined as a set consisting of at least two cells.
- \* Each cell is contained in at least one net.
- \*  $n(i)$  denotes the # of cells in  $net(i)$ .
- \* Any 2 cells which share a net are called neighbors.
- \* Each cell is assumed to have size  $s(i)$ .
- \*  $p(i)$  denotes the number of pins in  $cell(i)$

At input:

nets are presented one at a time, in any order, each net being completely given before the next one is started.

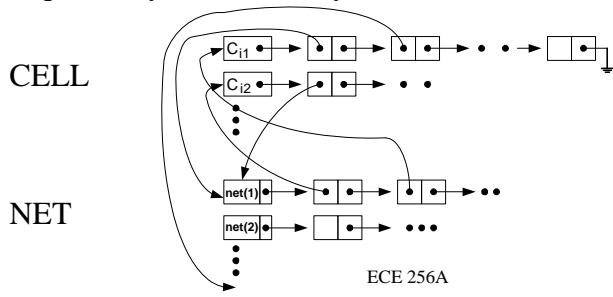
$$P = \sum p(i) = \text{total \# pins}$$

$P$  is the measure of the input and can be interpreted as a “size” of the network.

$C$  is  $O(P)$  and  $N$  is  $O(P)$

Input routine

Cells are identified by integers  $1 \div C$ , nets are numbered sequentially  $1, 2, \dots, N$  as they are entered.



Net-list input:

For each net  $n = 1, \dots, N$  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

$O(P)$  will suffice to do this work.

ECE 256A

23

Cutstate of a net:  $\begin{cases} \text{cut} \\ \text{uncut} \end{cases}$

cut: has at least one cell in each side of the partition;

uncut: all cells of a net are on one side of the partition.

Cutset of a partition  $\equiv$  set of nets which are cut.

The size  $|X|$  of a block of cells  $X$  is the sum of the sizes  $s(i)$  of its constituent cells.

User can specify  $0 < r < 1$  and a mincut partition with

$\frac{|A|}{|A| + |B|} \approx r$  is sought.

Some cells may be pre assigned to a specific side of a partition.

ECE 256A

24

Basic Idea: move cells, one at a time from one block to the other such that the cutset is minimized.

Base cell (cell to be moved) chosen based on balance condition and cutset.

Gain(i) of cell(i) = #nets by which cutset decreases when cell(i) moves.

$$-p(i) < g(i) < +p(i)$$

After a move, a cell is locked in its new block for the remainder of the pass. Only free cells can move. Stop: no free cells or balancing criterion. The best partition encountered during the pass is returned.

ECE 256A

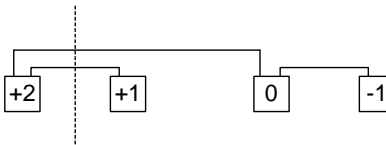
25

Computational effort:

- \* select the base cell
- \* move it
- \* adjust the gains of its free neighbors

Naive approach:  $O(P^2)$  gain computations per pass.

Cell gains:

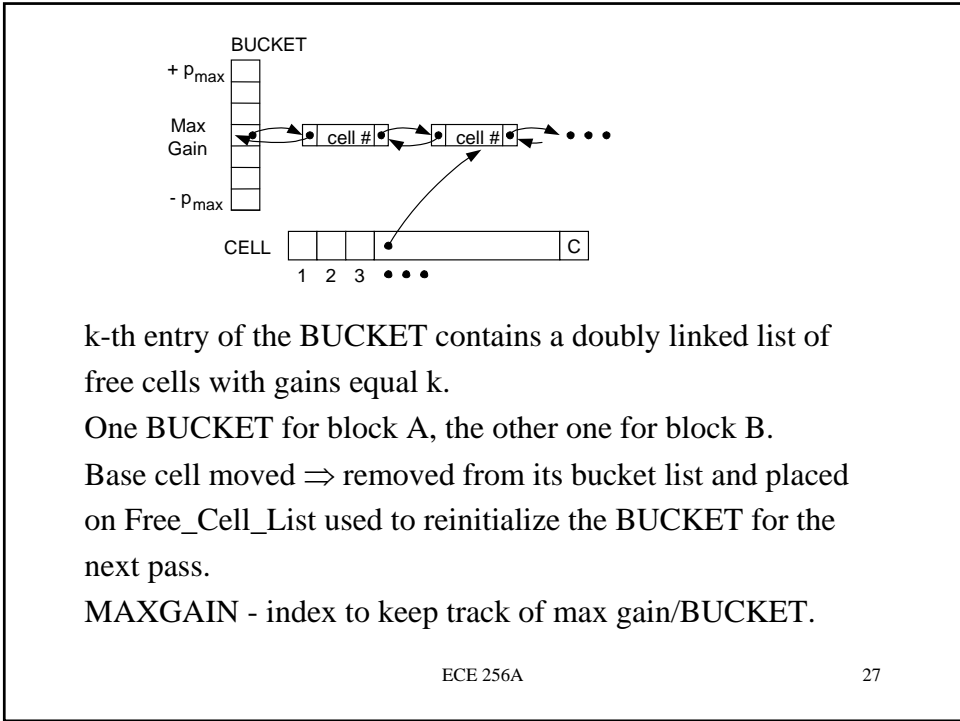


$$-p(i) \leq g(i) \leq p(i)$$

$$p_{\max} = \max \{ p(i) \mid \text{cell}(i) \text{ is initially free} \}$$

ECE 256A

26



\* The total amount of work required to maintain each BUCKET array is  $O(P)$  per pass.

Initialization:  $O(p_{\max}) + O(f) = O(P)$

↑  
free cells

g - total # of gain adjustments

$O(g)$  - work to move all free cells to their bucket lists.

Later we will show that  $g = O(P)$

R - sum of all amounts by which MAXGAIN is reset. The total time/pass used to search down for non-empty bucket and to remove a cell of highest gain is  $O(R + p_{\max}) + O(f) = O(R) + O(P)$ .

Later, we will see that  $R = O(g)$ .

ECE 256A 28

### Balance

(A, B) is balanced when

$$rW - s_{\max} \leq |A| \leq rW + s_{\max}$$

$$W = |A| + |B|$$

$$r = \frac{|A|}{|A| + |B|}$$

$s_{\max}$  = the size of the largest cell which is initially free

- \* Initial pass needed to establish the balance.
- \* The tolerance of  $s_{\max}$  allows to maintain the balance.

### The basic idea:

1. Consider the first cell (if any) of highest gain from BUCKET, rejecting it if move causes imbalance. If neither block has a qualifying cell, no more moves will be attempted.
2. Choose a cell of highest gain, breaking ties by choosing the best balance.
3. This is the base cell; remove it from the bucket list; place it on the FREE CELL List.

## Computing and maintaining cell gains.

Given a partition (A,B), the distribution of

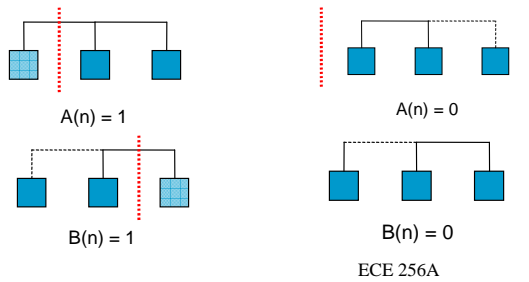
$$n = (A(n), B(n))$$

#cells of net n in A

# cells in B

It can be computed in  $O(P)$  for all nets.

A net is critical, if there exists a cell on it, which if moved would change the net's cutstate.  $A(n)$  or  $B(n)$  is 0 or 1.



ECE 256A

31

Useful observations:

- \* gain of a cell depends on its critical nets
- \* a net which is not critical before or after a move cannot influence the gains of any of its cells.

$$g(i) = FS(i) - TE(i)$$

cell #i

#nets which contain cell i as their only cell in "From" block

$TE(i)$  = #nets which contain cell i and have empty "To" block.

ECE 256A

32

Compute cell gains:

for each 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  $g(i) ++$

if  $T(n) = 0$  then  $g(i) --$

end for

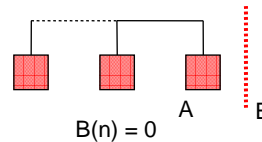
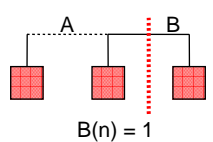
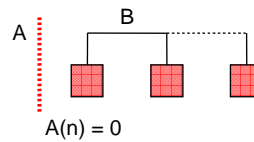
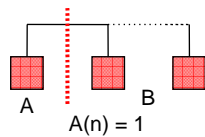
end for

Initialization of all cell gains requires  $O(P)$  work.

ECE 256A

33

Critical nets:



A net is critical before the move if and only if:

$F(n) = 1, T(n) = 0, T(n) = 1$

A net is critical after the move if and only if:

$T(n) = 1, F(n) = 0$  or  $F(n) = 1$

ECE 256A

34

$F(n) = 1$  before the move  $\equiv F(n) = 0$  after  
 $T(n) = 1$  after the move  $\equiv T(n) = 0$  before  
Move base cell and update neighbor's gain:  
 F: "from" block of base cell  
 T: = "to" block of base cell  
 "Move cell" = Lock it and complement its block ;  
for each net n on the base cell do  
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(i), if it is free.  
     decrement  $F(n)$  /\* change distribution\*/  
     increment  $T(n)$   
If  $F(n) = 0$  then decrement gains of all free cells on net(n)  
else if  $F(n) = 1$  then increment gain of the only F cell on

ECE 256A

35

net(n), if it is free

end for

If a net has n cells  $\rightarrow O(n)$  work/update.

\* No more than 4 update operations/net are performed during  
1 pass.

LF(n) locked cells on net(n) on the "from" side

FF(n) free cells on net(n) on the "from" side

LT(n) locked cells of net(n) on the "to" side

FT(n) free cells of net(n) on the "to" side

$T(n) = 0$  requires  $LT(n) = FT(n) = 0$

$T(n) = 1$  requires  $LT(n) = 1 \wedge FT(n) = 0$

or  $LT(n) = 0 \wedge FT(n) = 1$ ;

the update is performed only if  $LT(n) = 0$

ECE 256A

36

```

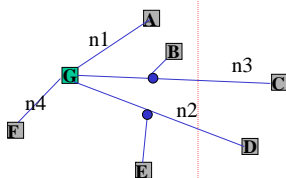
/* check for critical nets before the move*/
if LT(n) = 0
  then if FT(n) = 0 then update gains
  else if FT(n) = 1 then update gains
/* change the net distribution to reflect the move*/
decrement FF(n)
increment LT(n)
/* check for critical nets after the move*/
if LF(n) = 0
  then if FF(n) = 0 then update gains
  else if FF(n) = 1 then update gains

```

ECE 256A

37

### Example



Initial situation:

$g(A)=-1$	$F(n1)=2, T(n1)=0$
$g(B)=0$	$F(n2)=2, T(n2)=1$
$g(C)=1$	$F(n3)=2, T(n3)=1$
$g(D)=1$	$F(n4)=2, T(n4)=0$
$g(E)=0$	
$g(F)=-1$	
$g(G)=-2$	

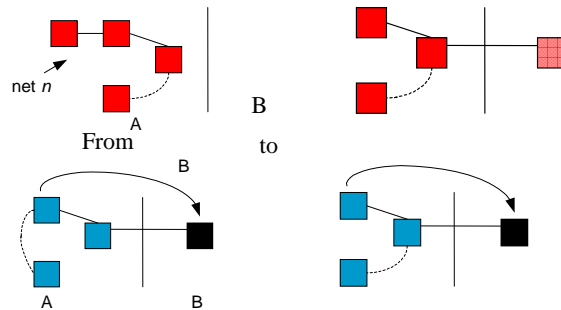
n1:	$T(n1)=0, g(A)=0, F(n1)=1, T(n1)=1$
	$g(G)=-1$
	$g(A)=1$
n2:	$T(n2)=1, g(D)=0, F(n2)=1, T(n2)=2$
	$g(E)=1$
n3:	$T(n3)=1, g(C)=0, F(n3)=1, T(n3)=2$
	$g(B)=1$
n4:	$T(n4)=0, g(G)=0, F(n4)=1, T(n4)=1$
	$g(F)=0$
	$g(F)=1$

ECE 256A

38

After both blocks A and B have served as “T” side for a net  $n$ , no further operations will occur for  $n$ . All cells of such a net are locked on both sides.

Consider



The B side having 0 or 1 cell can cause an update of only the first move in the sequence. Afterwards  $LB(n) > 0$ . Updates can occur only for  $FA(n) = 1$  and  $FA(n) = 0$ , once. 1 more update for  $B = F$ .

ECE 256A

39

So, we have:

total# of gain adjustments/pass is  $O(f)$

$f$  is # of initially free cells.

Thus,  $g = O(f) = O(P)$

Each time a net is updated, the total gain of any cell on that net can be incremented at most  $2\times$ , so during 1 update the value of  $MAXGAIN$  can be reset at most to  $MAXGAIN + 2$ .

So  $R$  is  $O(N) = O(P)$ .

↓

\* The total work required to initialize and maintain cell gains is  $O(P)$  per pass.

ECE 256A

40

After each “pass”, we find  $k$ , which maximizes

$$g_{\max} = \sum_{i=1}^k g_i$$

if  $g_{\max} > 0$

exchange  $a_1, a_2, \dots, a_k$  with  $b_1, \dots, b_k$

Note, that the partial sum

$$\sum g_i \text{ may be } < 0$$

in early stages.

Example:

