# ESE535: Electronic Design Automation

Day 10: February 27, 2008

Partitioning 2

(spectral, network flow, replication)



## Today

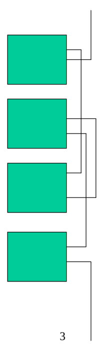
- Alternate views of partitioning
- Two things we can solve optimally
  - (but don't exactly solve our original problem)
- Techniques
  - Linear Placement w/ squared wire lengths
  - Network flow MinCut



## Optimization Target

- Place cells
- In linear arrangement
- Wire length between connected cells:
  - distance=$X_i$ - $X_j$
  - cost is sum of distance squared

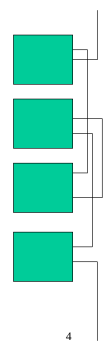
Pick $X_i$'s to minimize cost





## Why this Target?

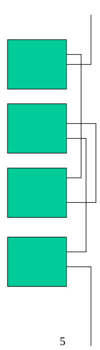
- Minimize sum of squared wire distances
- Prefer:
  - **Area:** minimize channel width
  - **Delay:** minimize critical path length





## Why this Target?

- Our preferred targets are discontinuous and discrete
- Cannot formulate analytically
- Not clear how to drive toward solution
  - Does reducing the channel width at a non-bottleneck help or not?
  - Does reducing a non-critical path help or not?





## Spectral Ordering

Minimize Squared Wire length -- 1D layout

- Start with connection array C ($c_{i,j}$)
- "Placement" Vector X for $x_i$ placement
- **Problem:**
  - Minimize cost = $0.5*\Sigma$ (all i,j) $(x_i - x_j)^2 c_{i,j}$
  - cost sum is $X^TBX$
    - B = D-C
    - D=diagonal matrix, $d_{i,i} = \Sigma$(over j) $c_{i,j}$

## Spectral Ordering

- Constraint: $X^TX=1$
  - prevent trivial solution all $x_i$'s =0
- Minimize cost=$X^TBX$ w/ constraint
  - minimize $L=X^TBX-\lambda(X^TX-1)$
  - $\partial L/\partial X=2BX-2\lambda X=0$
  - $(B-\lambda I)X=0$
  - $X \rightarrow$ Eigenvector of B
  - cost is Eigenvalue $\lambda$



## Spectral Solution

- Smallest eigenvalue is zero
  - Corresponds to case where all $x_i$'s are the same ➔uninteresting
- Second smallest eigenvalue (eigenvector) is the solution we want



## Spectral Ordering

- X ($x_i$'s) continuous
- use to order nodes
  - real problem wants to place at discrete locations
  - this is one case where can solve ILP from LP
    - Solve LP giving continuous $x_i$'s
    - then move back to closest discrete point



## Spectral Ordering Option

- With iteration, can reweigh connections to change cost model being optimized
  - linear
  - $(\text{distance})^{1.X}$
- Can encourage "closeness"
  - by weighting connection between nodes
    - Making $c_{i,j}$ larger
  - (have to allow some nodes to not be close)



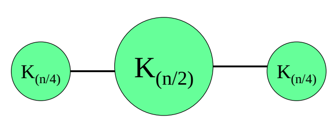
## Spectral Partitioning

- Can form a basis for partitioning
- Attempts to cluster together connected components
- Form cut partition from ordering
  - *E.g.* Left half of ordering is one half, right half is the other



## Spectral Ordering

- Midpoint bisect isn't necessarily best place to cut, consider:



## Fanout

- How do we treat fanout?
- As described assumes point-to-point nets
- For partitioning, pay price when cut something once
  - *I.e.* the accounting did last time for KLFM
- Also a discrete optimization problem
  - Hard to model analytically



## Spectral Fanout

- Typically:
  - Treat all nodes on a single net as fully connected
  - Model links between all of them
  - Weight connections so cutting in half counts as cutting the wire
  - Threshold out high fanout nodes
    - If connect to too many things give no information



## Spectral Partitioning Options

- Can bisect by choosing midpoint
  - (not strictly optimizing for minimum bisect)
- Can relax cut critera
  - min cut w/in some $\delta$ of balance
- Ratio Cut
  - minimize (cut/|A||B|)
    - idea tradeoff imbalance for smaller cut
      - more imbalance $\rightarrow$smaller |A||B|
      - so cut must be much smaller to accept
- Circular bisect/relaxed/ratio cut
  - wrap into circle and pick two cut points
  - How many such cuts?



## Spectral vs. FM

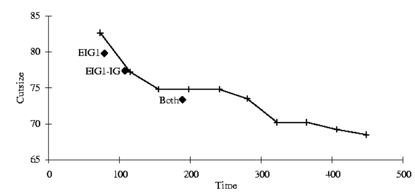


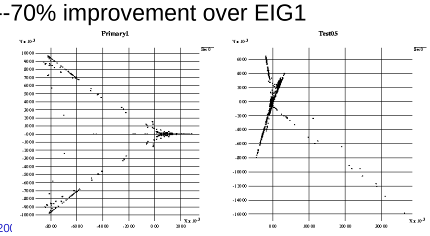

**Figure 5.** Graphs of cutsizes for different numbers of runs of our optimized version of KLFM versus the spectral initialization approaches. Values shown are the geometric means of the results for the 9 test circuits (all but industry3).

From Hauck/Boriello '96



## Improving Spectral

- More Eigenvalues
  - look at clusters in n-d space
    - **But:** 2 eigenvectors is not opt. solution to 2D placement
  - 5--70% improvement over EIG1






## Spectral Note

- Unlike KLFM, attacks **global** connectivity characteristics
- Good for finding "natural" clusters
  - hence use as clustering heuristic for multilevel algorithms

## Spectral Theory

- There are conditions under which spectral is optimal [Boppana/FOCS28 (1987)]
  – B=A+diag(d)
  – g(G,d)=[sum(B)-n*$\lambda(B_S)$]/4
  – $B_S$ mapping Bx to closest point on S
  – h(G) = max d g(G,d)
- h(G) lower bound on cut size



## Spectral Theory

- Boppana paper gives a probabilistic model for graphs
  – model favors graphs with small cuts
  – necessary since *truly* random graph has cut size O(n)
  – shows high likelihood of bisection being lower bound
- In practice
  – known to be **very** weak for graphs with large cuts



## Max Flow

MinCut



## MinCut Goal

- Find maximum flow (mincut) between a source and a sink
  – no balance guarantee



## MaxFlow

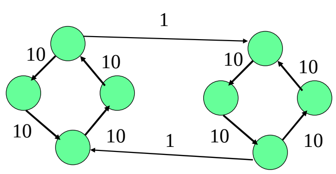
- Set all edge flows to zero
  – F[u,v]=0
- While there is a path from s,t
  – (breadth-first-search)
  – for each edge in path f[u,v]=f[u,v]+1
  – f[v,u]=-f[u,v]
  – When c[v,u]=f[v,u] remove edge from search
- O(|E|*cutsize)
- [Our problem simpler than general case CLR]



## Technical Details

- For min-cut in graphs,
  – Don't really care about directionality of cut
  – Just want to minimize wire crossings
- Fanout
  – Want to charge discretely …cut or not cut
- Pick start and end nodes?

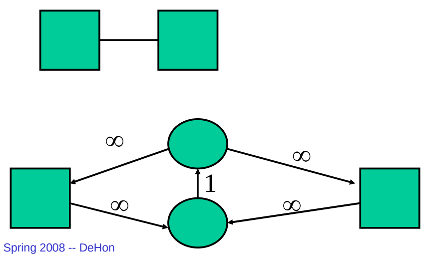
## Directionality




For logic net: cutting a net is the same regardless of which way the signal flows

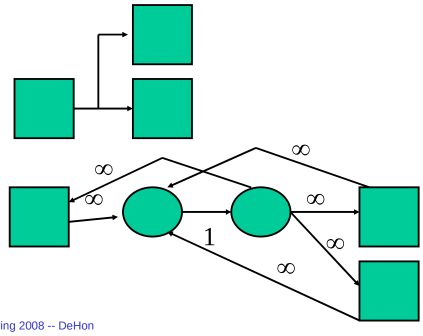


## Directionality Construct






## Fanout Construct






## Extend to Balanced Cut

- Pick a start node and a finish node
- Compute min-cut start to finish
- If halves sufficiently balanced, done
- else
  - collapse all nodes in smaller half into one node
  - pick a node adjacent to smaller half
  - collapse that node into smaller half
  - repeat from min-cut computation

FBB -- Yang/Wong ICCAD'94



## Observation

- Can use residual flow from previous cut when computing next cuts
- Consequently, work of multiple network flows is only O(|E|*final_cut_cost)



## Picking Nodes

- Optimal:
  - would look at all s,t pairs
    - Just for first cut is merely N-1 “others”
      - …N/2 to guarantee something in second half
    - Anything you pick **must** be in separate halves
    - Assuming thereis perfect/ideal bisection
      - If pick randomly, probability in different halves is 50%
      - Few random selections likely to yield s,t in different halves
  - would also look at all nodes to collapse into smaller
  - could formulate as branching search

## Picking Nodes

- Randomly pick
  - (maybe try several starting points)
- With small number of adjacent nodes,
  - could afford to branch on all



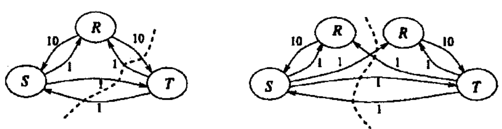
## Approximation

- Can find 1/3 balanced cuts within O(log(n)) of best cut in polynomial time
  - algorithm due to Leighton and Rao
  - exposition in Hochbaum's *Approx. Alg.*
- Bound cut size
  - Boppana -- lower bound h(G)
  - Boppana/Spectral cut -- one upper bound
  - Leighton/Rao -- upper bound



## Min Cut Replication

- Noted last time could use replication to reduce cut size
  - Observed could use FM to replicate
- Can solve unbounded replication optimally with mincut




[Liu,Kuo,Cheng TRCAD v14n5p623]

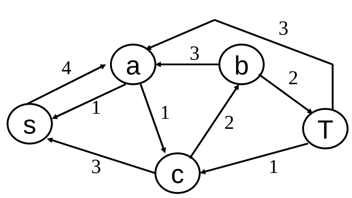


## Min-Cut Replication

- Key Idea:
  - Create two copies of net
    - Connect super src/sink to both
    - reverse links on "to" second copy
    - Provide **directional** "free" links between them
  - Take mincut
    - S=reachable from src; T=reachable from sink
    - R=rest ➔ replication set
  - Allows nodes to be associated with both ends

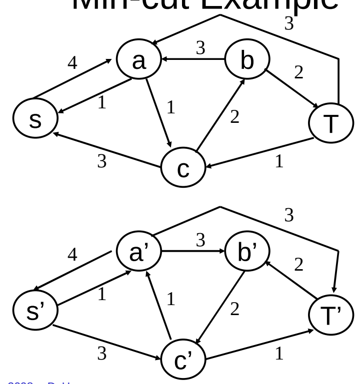


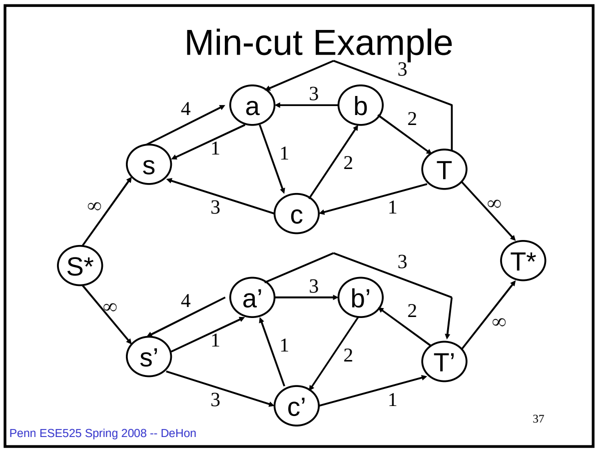
## Min-cut Example

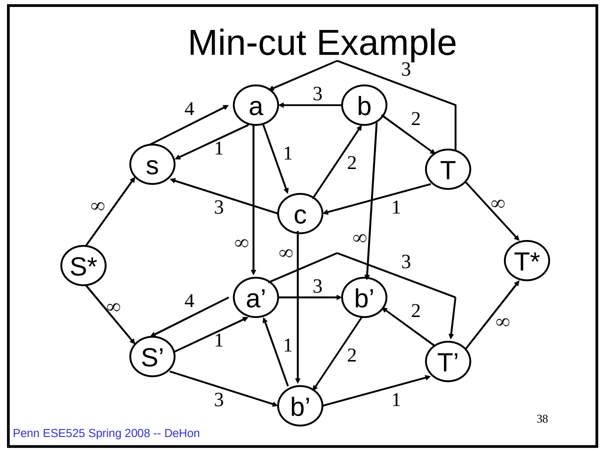


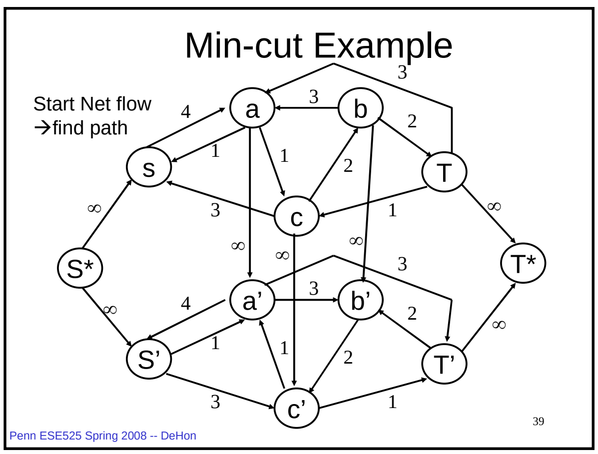


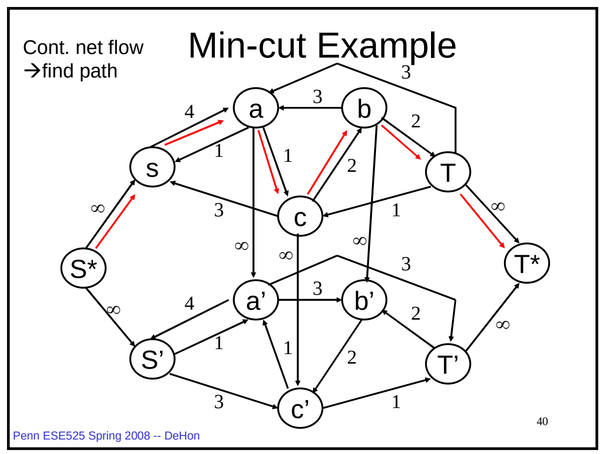

## Min-cut Example

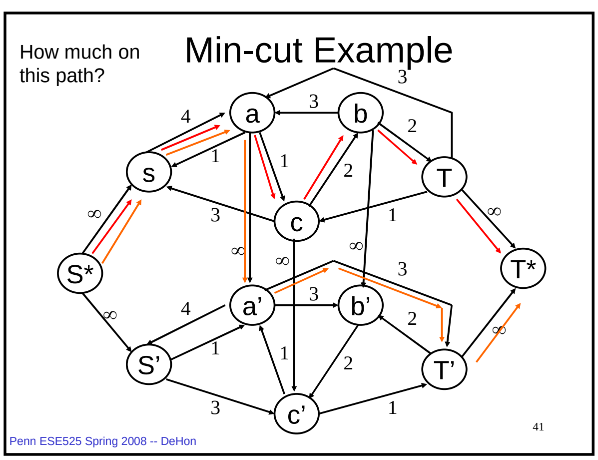


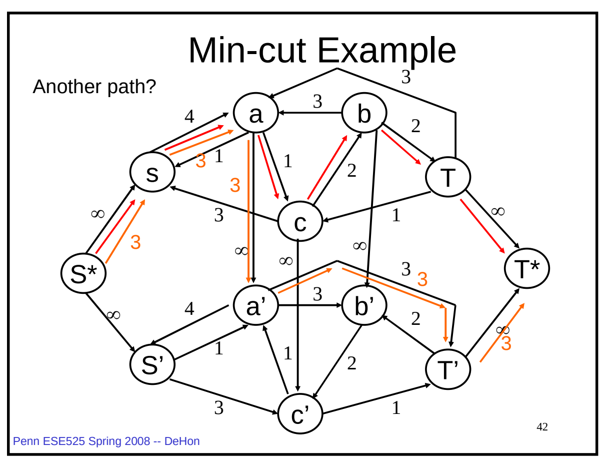

Min-cut Example
S*
s
a
b
c
T
T*
s'
a'
b'
c'
T'
Penn ESE525 Spring 2008 -- DeHon
37


Min-cut Example
S*
s
a
b
c
T
T*
S'
a'
b'
T'
Penn ESE525 Spring 2008 -- DeHon
38


Min-cut Example
Start Net flow
→find path
S*
s
a
b
c
T
T*
S'
a'
b'
c'
T'
Penn ESE525 Spring 2008 -- DeHon
39

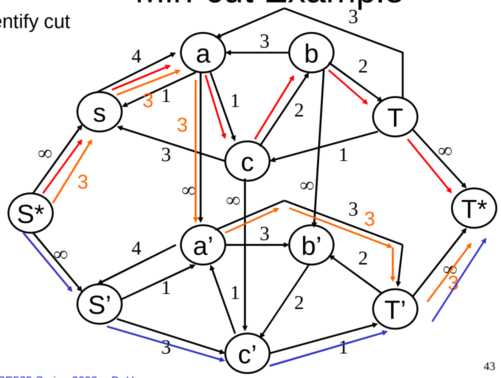

Cont. net flow
→find path
Min-cut Example
S*
s
a
b
c
T
T*
S'
a'
b'
c'
T'
Penn ESE525 Spring 2008 -- DeHon
40


How much on
this path?
Min-cut Example
S*
s
a
b
c
T
T*
S'
a'
b'
c'
T'
Penn ESE525 Spring 2008 -- DeHon
41


Min-cut Example
Another path?
S*
s
a
b
c
T
T*
S'
a'
b'
c'
T'
Penn ESE525 Spring 2008 -- DeHon
42

## Min-cut Example

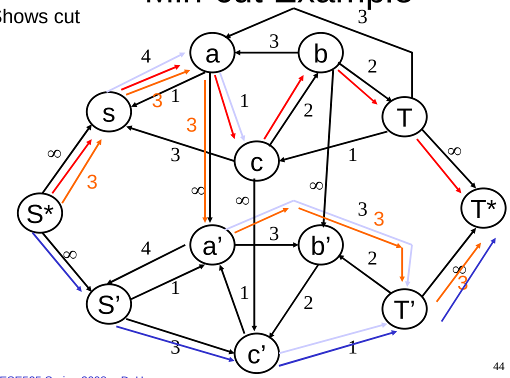
Identify cut






## Min-cut Example

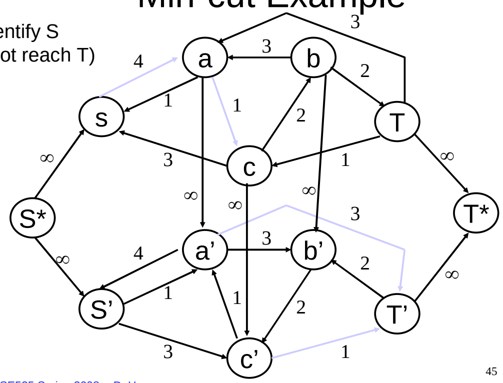
Shows cut






## Min-cut Example

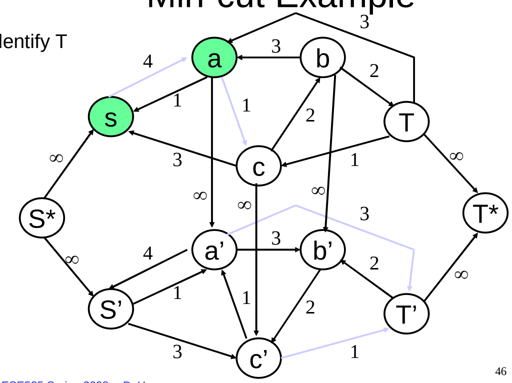
Identify S
(not reach T)






## Min-cut Example

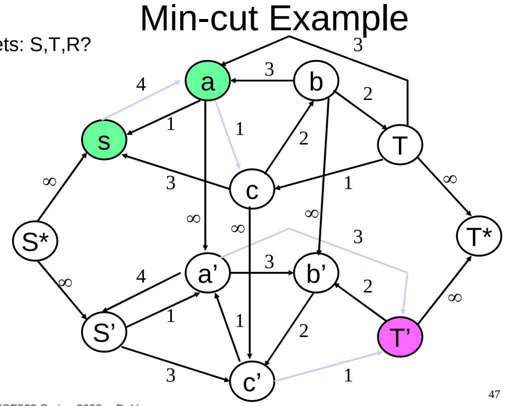
Identify T






## Min-cut Example

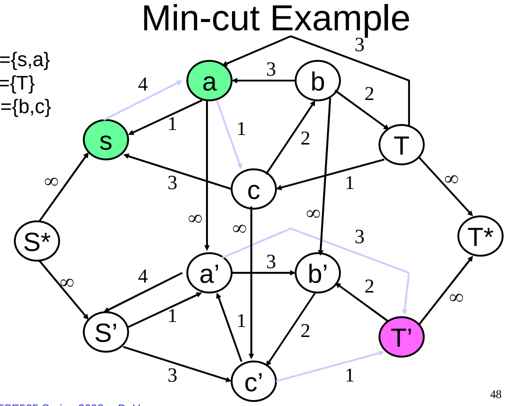
Sets: S,T,R?





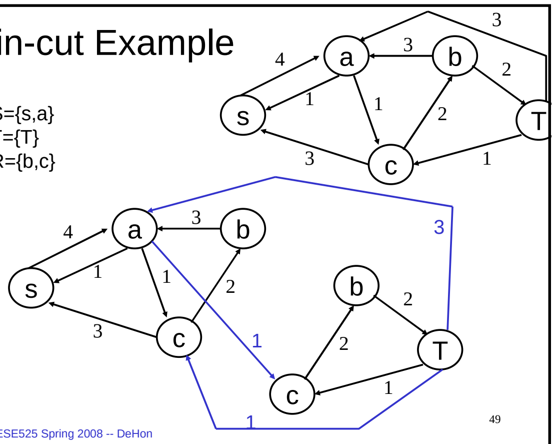

## Min-cut Example

S={s,a}
T={T}
R={b,c}



## Min-cut Example

S={s,a}
T={T}
R={b,c}






## Replication Note

- Cut of minimum width is not unique
  - Similar to phenomenon saw in LUT covering with network flow
- Want to identify **minimum** size replication set for given flow
- Can do by reweighing graph and another min-cut
  - **Idea:** weight on replication connections
    - Minimize cut → minimize replication set
  - See Mak/Wong TRCAD v16n10p1221



## Admin

- Monday reading online
- Homework 3 due Monday



## Big Ideas

- Divide-and-Conquer
- Techniques
  - flow based
  - numerical/linear-programming based
  - Transformation constructs
- Exploit problems we can solve optimally
  - Mincut
  - Linear ordering