《芯片设计自动化与智能优化》 Partitioning

The slides are based on Prof. David Z. Pan’s lecture notes at UT Austin

Yibo Lin

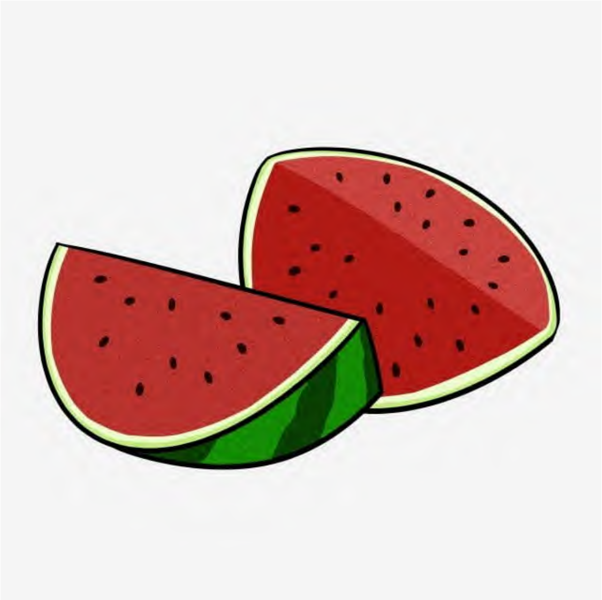
Peking University

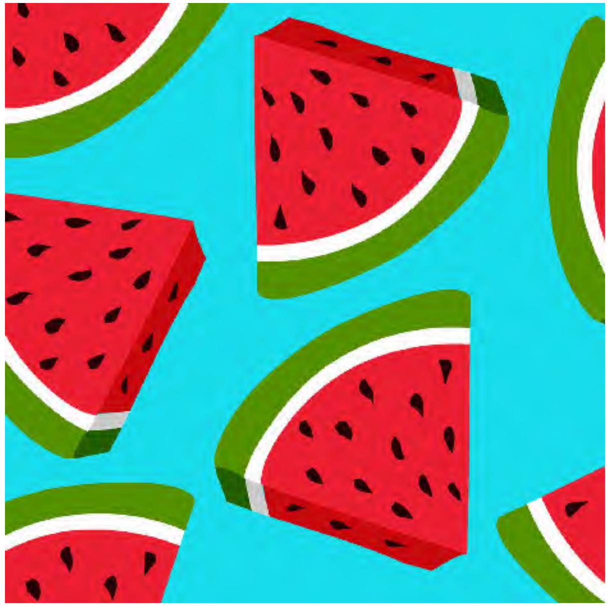
Outline

- What is partitioning
- KL algorithm
- FM algorithm
- Spectral algorithm

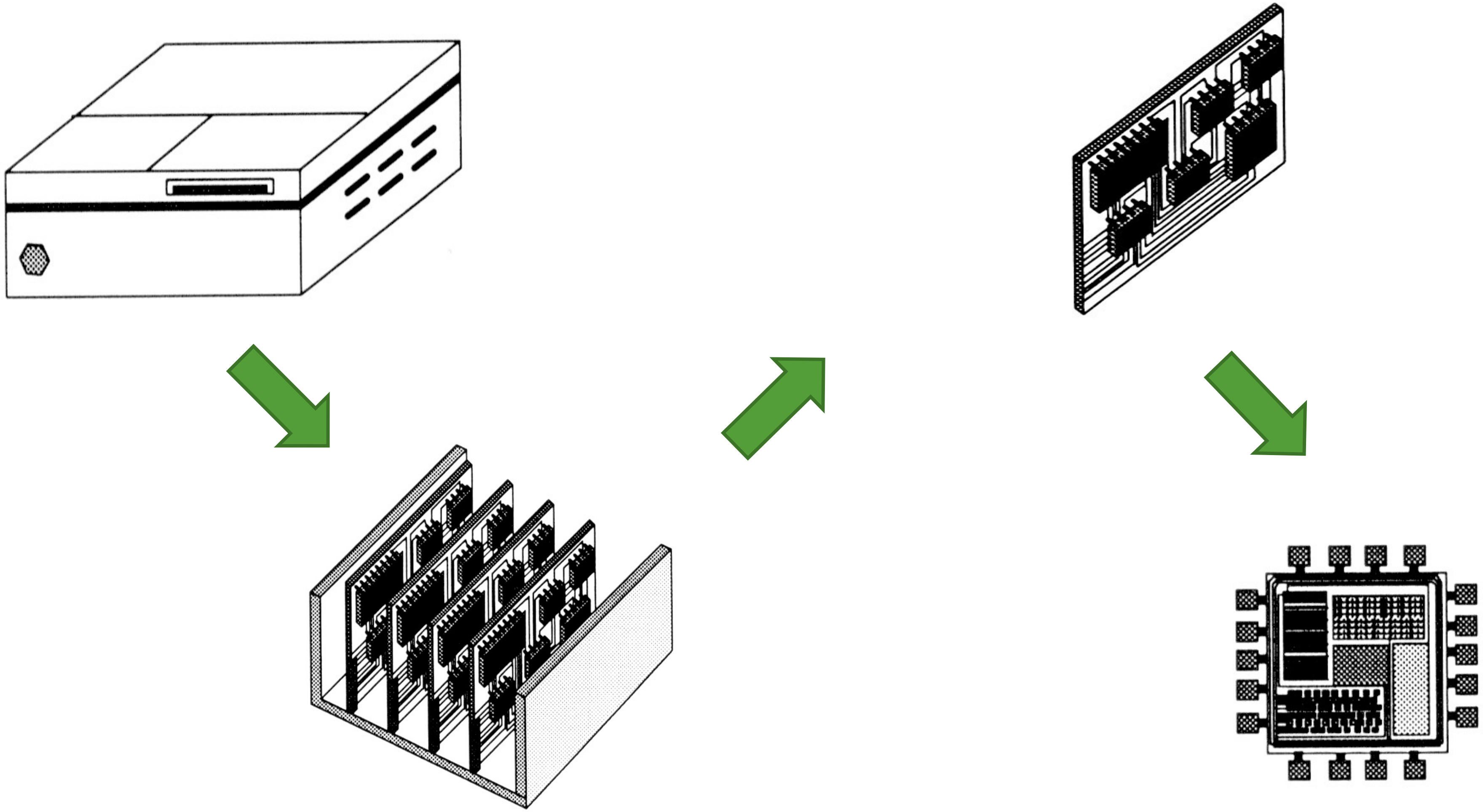
What is Partitioning

- Divide and conquer

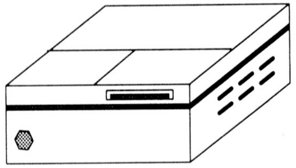




What is Partitioning – System Hierarchy

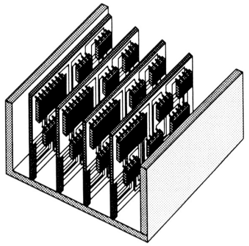


Levels of Partitioning



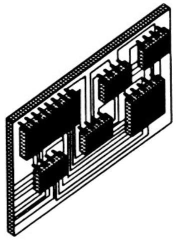
System

System Level Partitioning



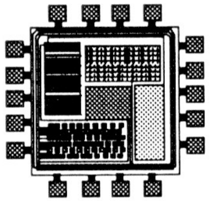
PCBs

Board Level Partitioning



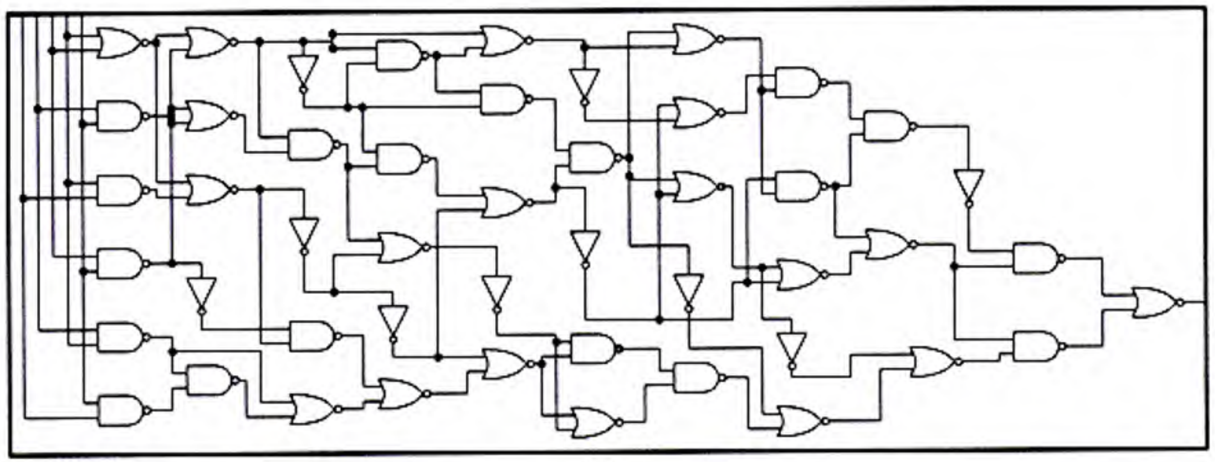
Chips

Chip Level Partitioning

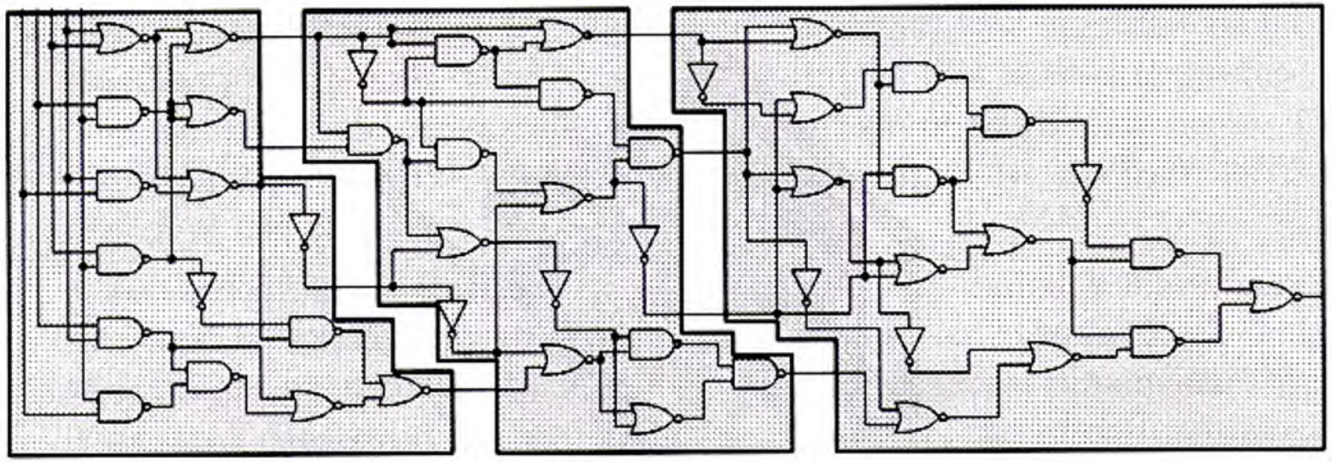


Subcircuits / Blocks

Partitioning of Circuit



(a)



(b)

Importance of Circuit Partitioning

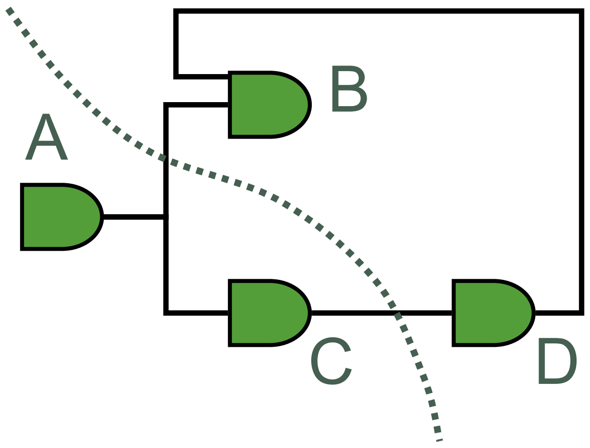
- Divide-and-conquer methodology
  - The most effective way to solve problems of high complexity
  - E.g.: min-cut based placement, partitioning-based test generation,…
- System-level partitioning for multi-chip designs
  - Inter-chip interconnection delay dominates system performance.
- Circuit emulation/parallel simulation
  - Partition large circuit into multiple FPGAs (e.g. Quickturn), or   multiple special-purpose processors (e.g. Zycad).
- Parallel CAD development
  - Task decomposition and load balancing
- In deep-submicron designs, partitioning defines local and global interconnect, and has significant impact on circuit performance
- …… ……

Some Terminology

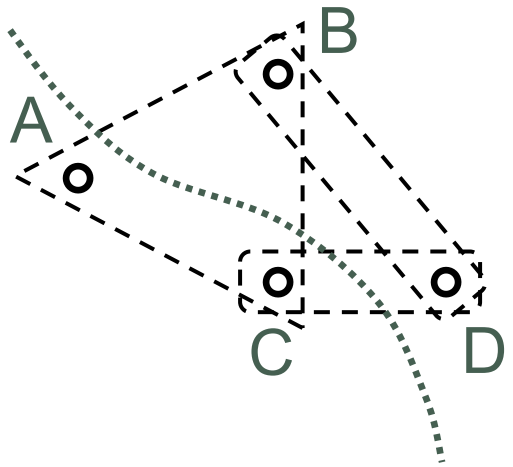
- Partitioning: Dividing bigger circuits into a small number of partitions (top down)
- Clustering: cluster small cells into bigger clusters (bottom up).
- Covering / Technology Mapping: Clustering such that each partitions (clusters) have some special structure (e.g., can be implemented by a cell in a cell library).
- K-way Partitioning: Dividing into k partitions.
- Bipartitioning: 2-way partitioning.
- Bisectioning: Bipartitioning such that the two partitions have the same size.

Circuit Representation

- Netlist:
  - Gates: A, B, C, D
  - Nets: {A,B,C}, {B,D}, {C,D}



- Hypergraph:
  - Vertices: A, B, C, D
  - Hyperedges: {A,B,C}, {B,D}, {C,D}

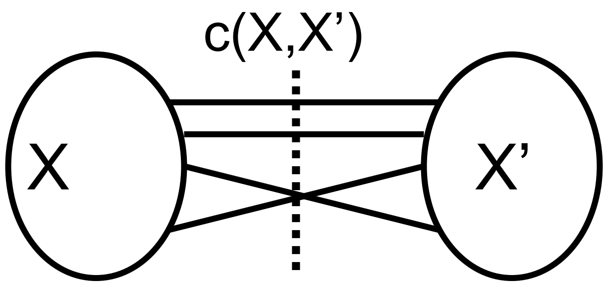


  - Vertex label: Gate size/area
  - Hyperedge label:

    Importance of net (weight)

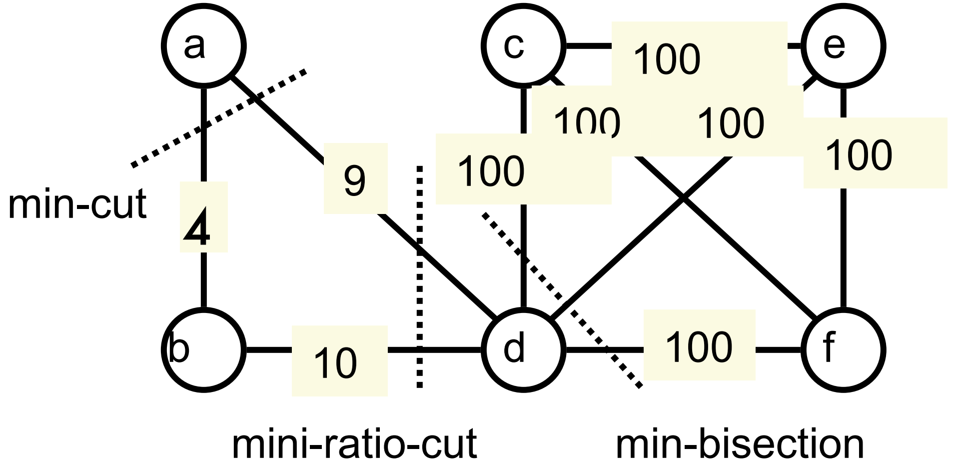
Circuit Partitioning Formulation

- Bi-partitioning formulation:
  - Minimize interconnections between partitions



- Minimum cut: min c(x, x')
- Minimum bisection: min c(x, x') with |x|= |x'|
- Minimum ratio-cut: min c(x, x') / |x||x'|

A Bi-Partitioning Example



Min-cut size=13
Min-Bisection size = 300
Min-ratio-cut size= 19

Ratio-cut helps to identify natural clusters

Circuit Partitioning Formulation (Cont'd)

- General multi-way partitioning formulation:
  - Partitioning a network N into N1, N2, …, Nk such that
- Each partition has an area constraint
  - $\sum_{v \in N_i} a(v) \leq A_i$
- each partition has an I/O constraint
  - $c(N_i, N - N_i) \leq I_i$
- Minimize the total interconnection:
  - $\sum_{N_i} c(N_i, N - N_i)$

Partitioning Algorithms

- Iterative partitioning algorithms
- Spectral based partitioning algorithms
- Net partitioning vs. module partitioning
- Multi-way partitioning
- Multi-level partitioning
- Further study in partitioning techniques (timing-driven …)

Iterative Partitioning Algorithms

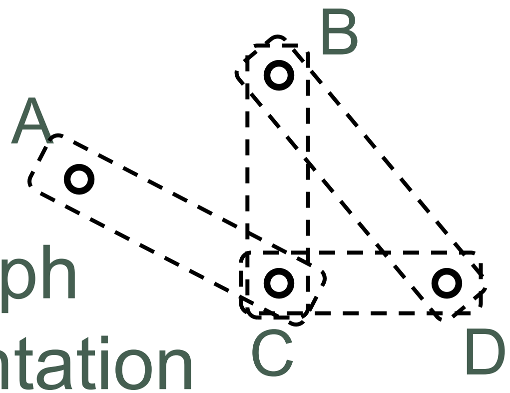
- Greedy iterative improvement method
  - [Kernighan-Lin 1970]
  - [Fiduccia-Mattheyses 1982]
  - [krishnamurthy 1984]
- Simulated Annealing
  - [Kirkpartrick-Gelatt-Vecchi 1983]
  - [Greene-Supowit 1984]

Kernighan-Lin Algorithm

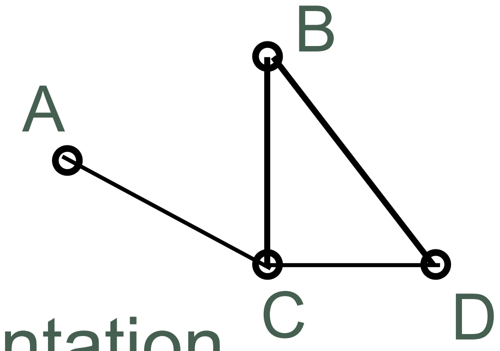
- Restricted Partition Problem
- Restrictions:
  - For Bisectioning of circuit.
  - Assume all gates are of the same size.
  - Works only for 2-terminal nets.
- If all nets are 2-terminal,

  the Hypergraph is called a Graph.

Hypergraph Representation



Graph Representation



"An Efficient Heuristic Procedure for Partitioning Graphs" The Bell System Technical Journal 49(2):291-307, 1970

Problem Formulation

- Input: A graph with
  - Set vertices V. (|V| = 2n)
  - Set of edges E. (|E| = m)
  - Cost c_{AB} for each edge {A, B} in E.
- Output: 2 partitions X & Y such that
  - Total cost of edges cut is minimized.
  - Each partition has n vertices.
- This problem is NP-Complete!!!!!

A Trivial Approach

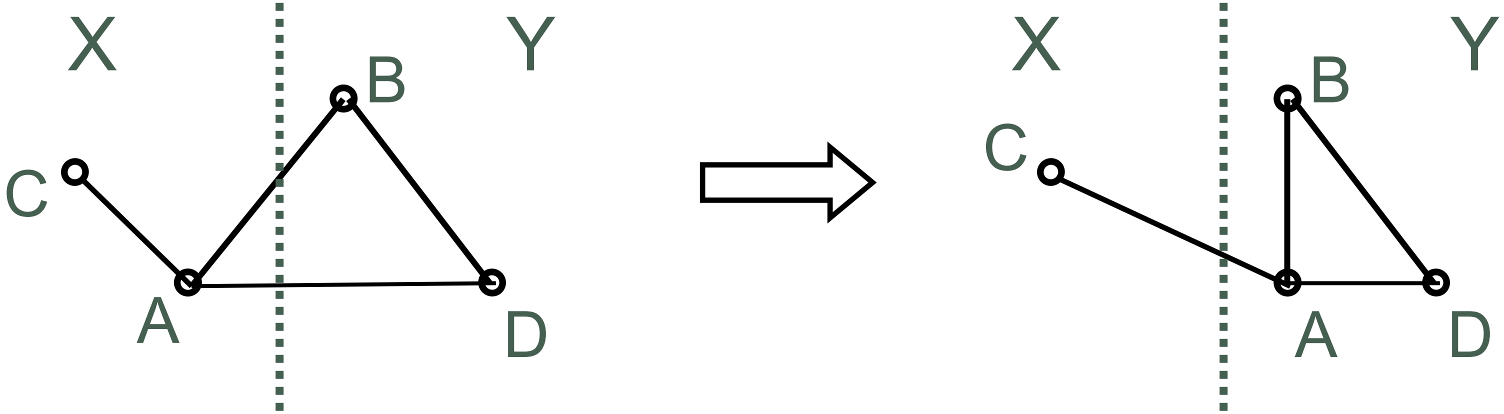
- Try all possible bisections. Find the best one.
- If there are 2n vertices,

  # of possibilities = $(2n)! / n!^2 = n^{O(n)}$

- For 4 vertices (A,B,C,D), 3 possibilities.
  1. X={A,B} & Y={C,D}
  2. X={A,C} & Y={B,D}
  3. X={A,D} & Y={B,C}
- For 100 vertices, $5\text{x}10^{28}$ possibilities.
- Need $1.59\text{x}10^{13}$ years if one can try 100M possbilities per second.

Idea of KL Algorithm

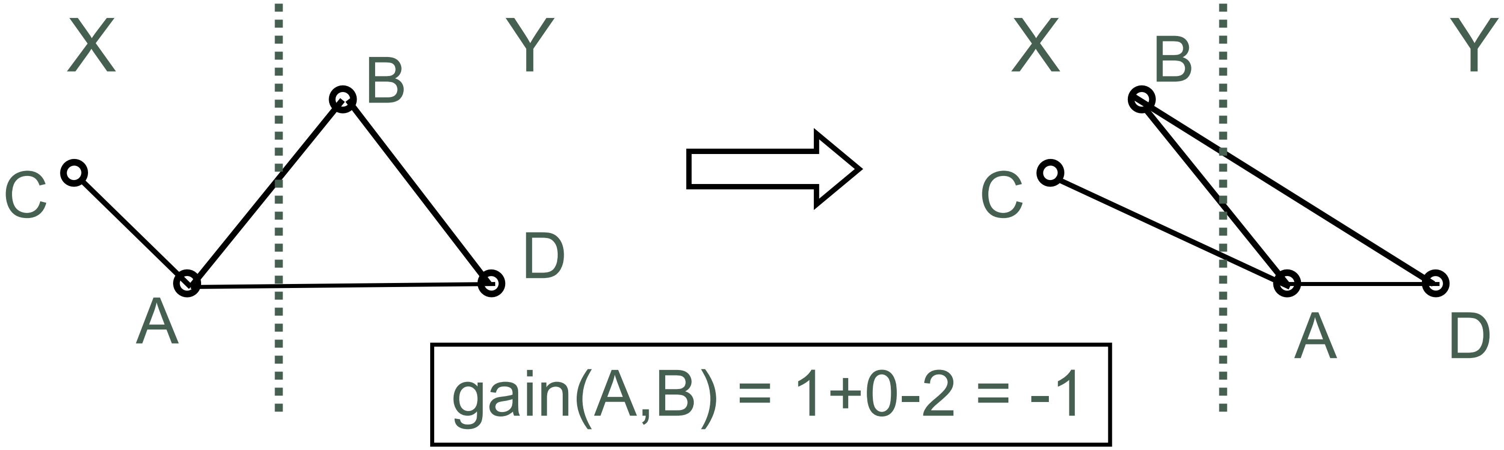
- D_A = Decrease in cut value if moving A
  - External cost (connection) E_A – Internal cost I_A
  - Moving node a from block A to block B would increase the value of the cutset by E_A and decrease it by I_A



$D_A = 2\text{-}1 = 1$
$D_B = 1\text{-}1 = 0$

Idea of KL Algorithm

- Note that we want to balance two partitions
- If switch A & B, gain(A,B) = $D_A+D_B-2c_{AB}$
  - c_{AB} : edge cost for AB



Idea of KL Algorithm

- Start with any initial legal partitions X and Y.
- A <u>pass</u> (exchanging each vertex exactly once) is described below:

  1. For i := 1 to n do

     From the unlocked (unexchanged) vertices,

     choose a pair (A,B) s.t. gain(A,B) is largest.

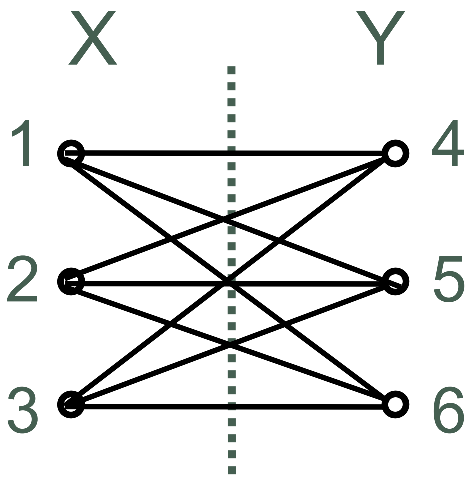
     Exchange A and B. Lock A and B.

     Let g_i = gain(A,B).

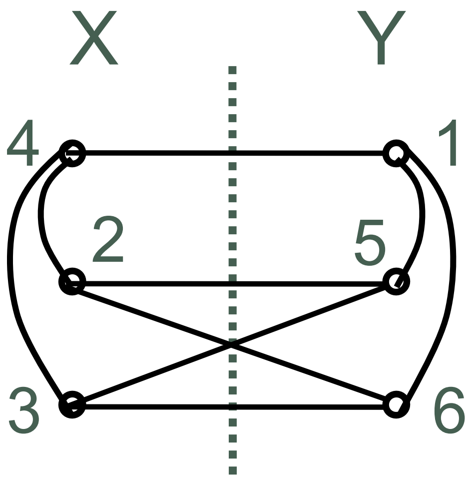
  2. Find the k s.t. $G=g_1+\ldots+g_k$ is maximized.
  3. Switch the first k pairs.

- Repeat the pass until there is no improvement (G=0).

Example



Original Cut Value = 9



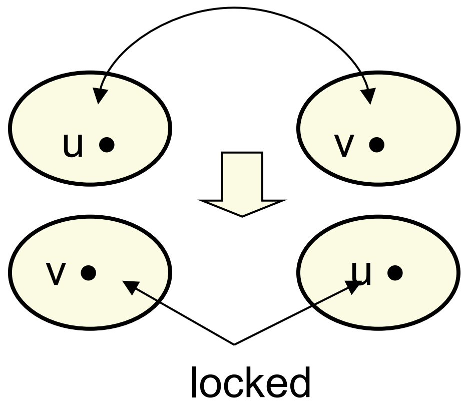
Optimal Cut Value = 5

Time Complexity of KL

- For each pass,
  - $O(n^2)$ time to find the best pair to exchange.
  - n pairs exchanged.
  - Total time is $O(n^3)$ per pass.
- Better implementation can get $O(n^2 \log n)$ time per pass.
- Number of passes is usually small.

Recap of Kernighan-Lin's Algorithm

- Pair-wise exchange of nodes to reduce cut size
- Allow cut size to increase temporarily within a pass
- Compute the gain of a swap
- Repeat
  - Perform a feasible swap of max gain
  - Mark swapped nodes "locked";
  - Update swap gains;
- Until no feasible swap;
- Find max prefix partial sum in gain sequence g1, g2, …, gm
- Make corresponding swaps permanent.



- Start another pass if current pass reduces the cut size
  - (usually converge after a few passes)

Charles Alpert and Andrew Kahng, "Recent Directions in Netlist Partitioning: A Survey", Integration: the VLSI Journal, 19(1-2), 1995, pp. 1-81.

Fiducciа-Mattheyses Algorithm

- Modification of KL Algorithm:
  - Can handle non-uniform vertex weights (areas)
  - Allow unbalanced partitions
  - Extended to handle hypergraphs
  - Clever way to select vertices to move, run much faster.

"A Linear-time Heuristics for Improving Network Partitions" 19th DAC, pages 175-181, 1982.

Problem Formulation

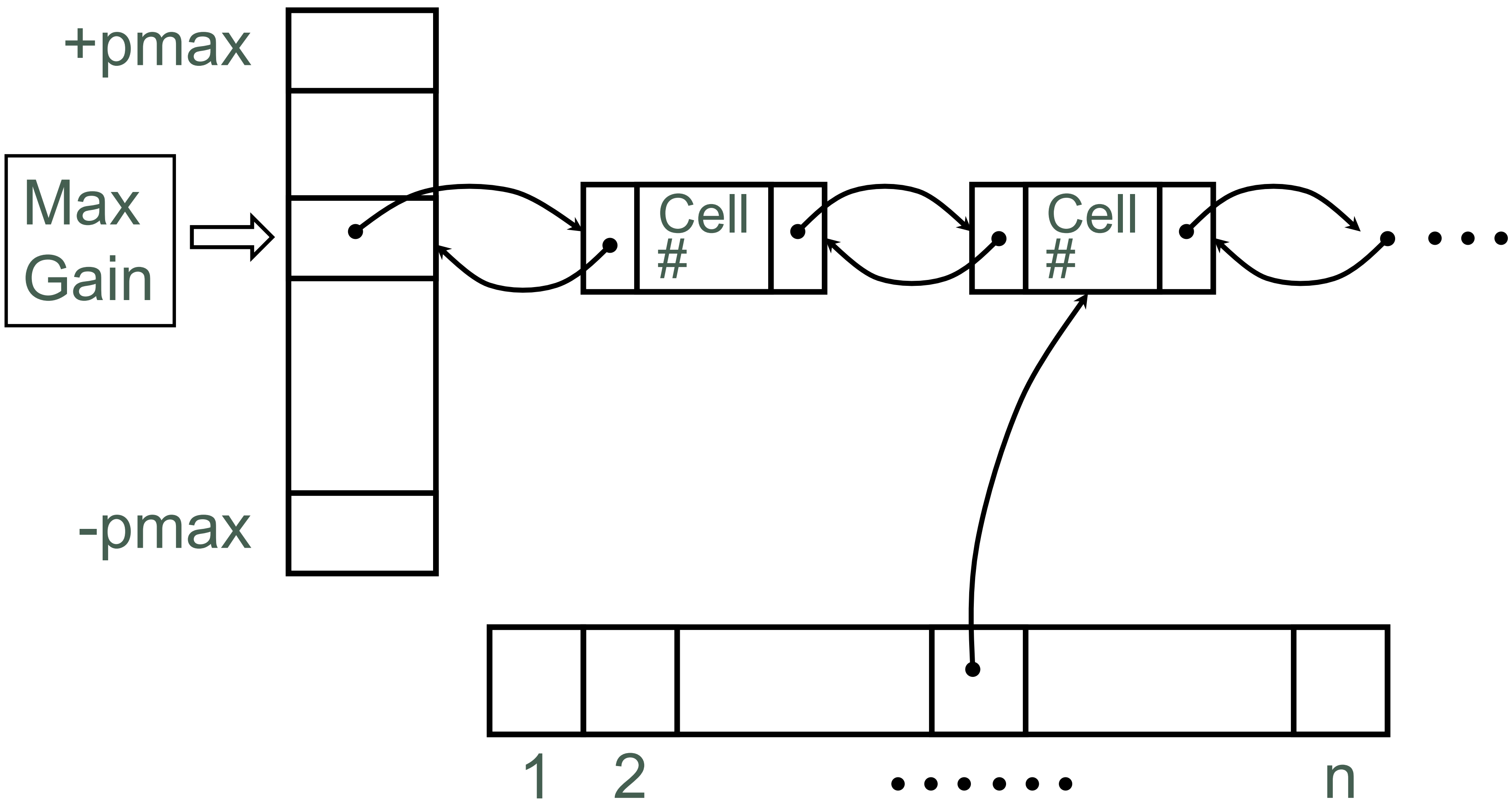
- Input: A hypergraph with
  - Set vertices V. (|V| = n)
  - Set of hyperedges E. (total # pins in netlist = p)
  - Area a_u for each vertex u in V.
  - Cost c_e for each hyperedge in e.
  - An area ratio r.
- Output: 2 partitions X & Y such that
  - Total cost of hyperedges cut is minimized.
  - area(X) / (area(X) + area(Y)) is about r.
- This problem is NP-Complete!!!

Ideas of FM Algorithm

- Similar to KL:
  - Work in passes.
  - Lock vertices after moved.
  - Actually, only move those vertices up to the maximum partial sum of gain.

- Difference from KL:
  - Not exchanging pairs of vertices.
    Move only one vertex at each time.
  - The use of gain bucket data structure.

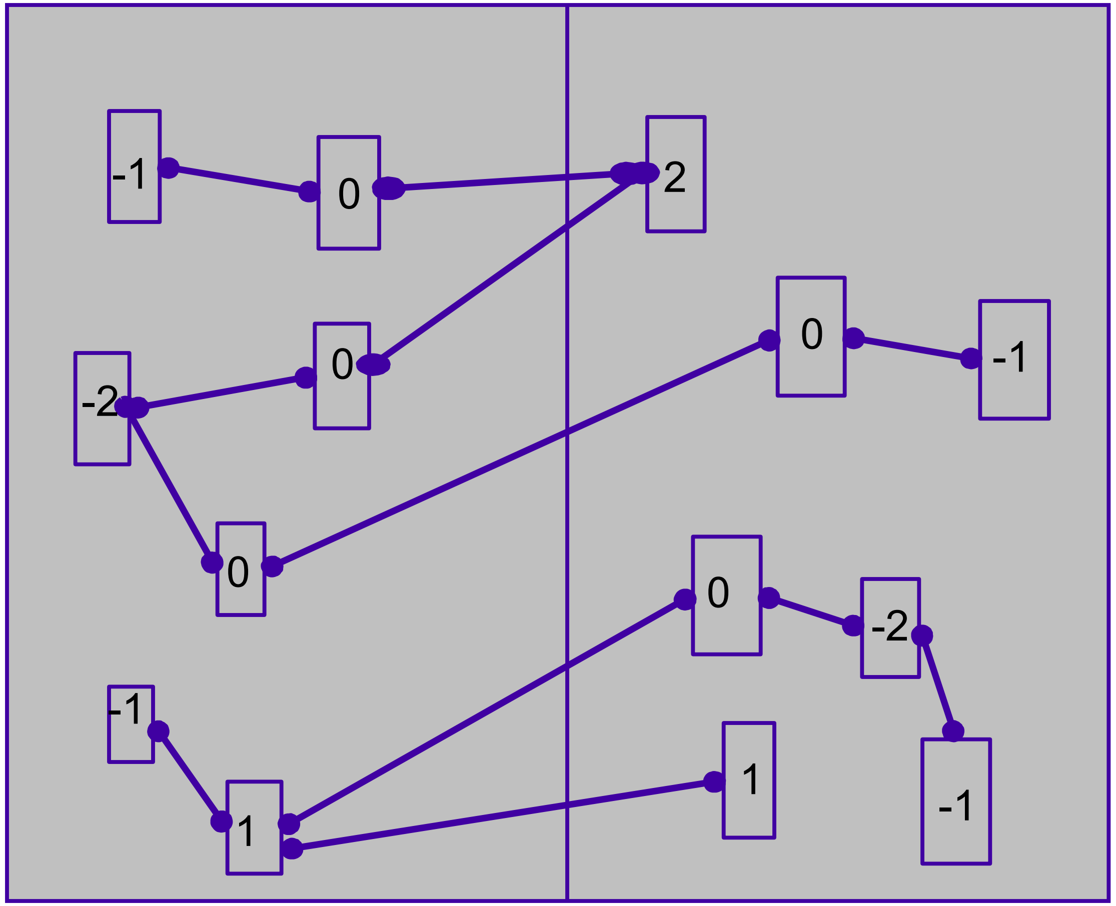
Gain Bucket Data Structure



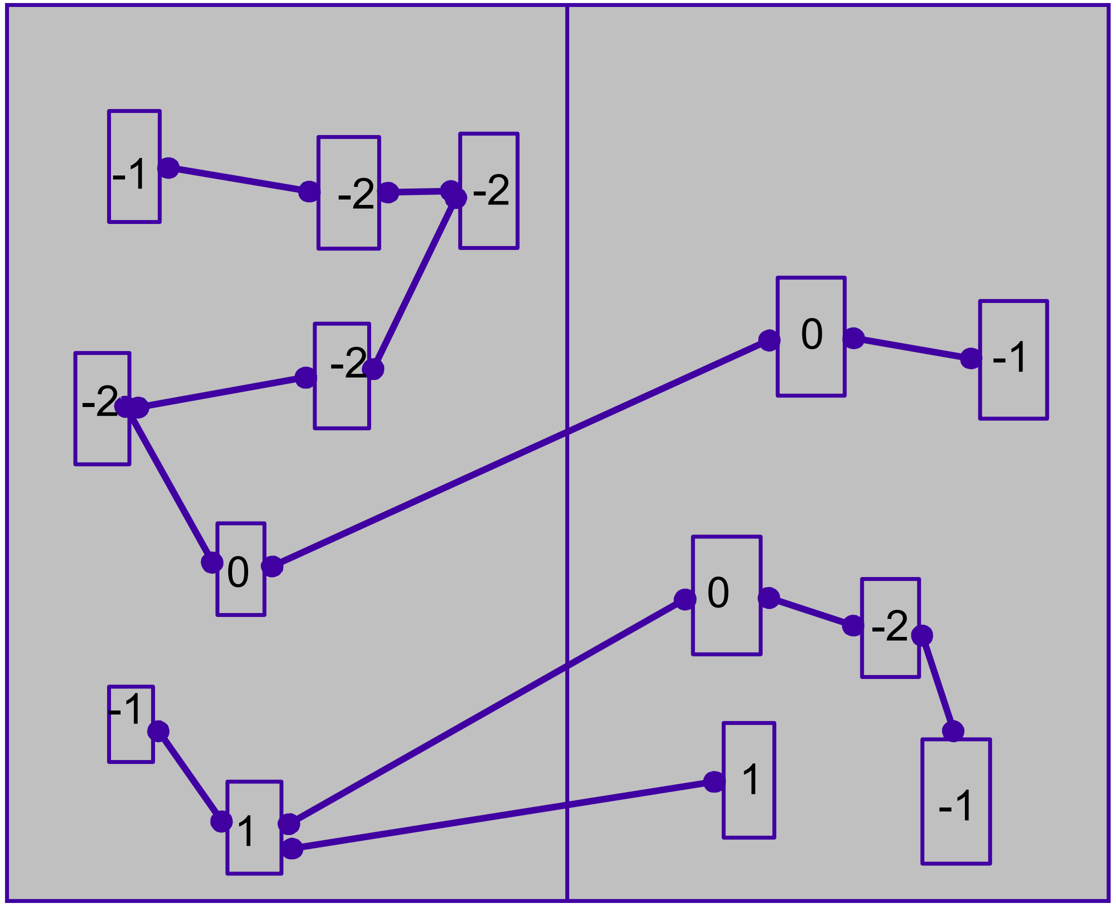
FM Partitioning

- Moves are made based on object gain.
- Object Gain
  - The amount of change in cut crossings
  - that will occur if an object is moved from
  - its current partition into the other partition
- Procedure
  - each object is assigned a gain
  - objects are put into a sorted gain list
  - the object with the highest gain from the larger of the two sides is selected and moved
  - the moved object is "locked"
  - gains of "touched" objects are recomputed
  - gain lists are resorted

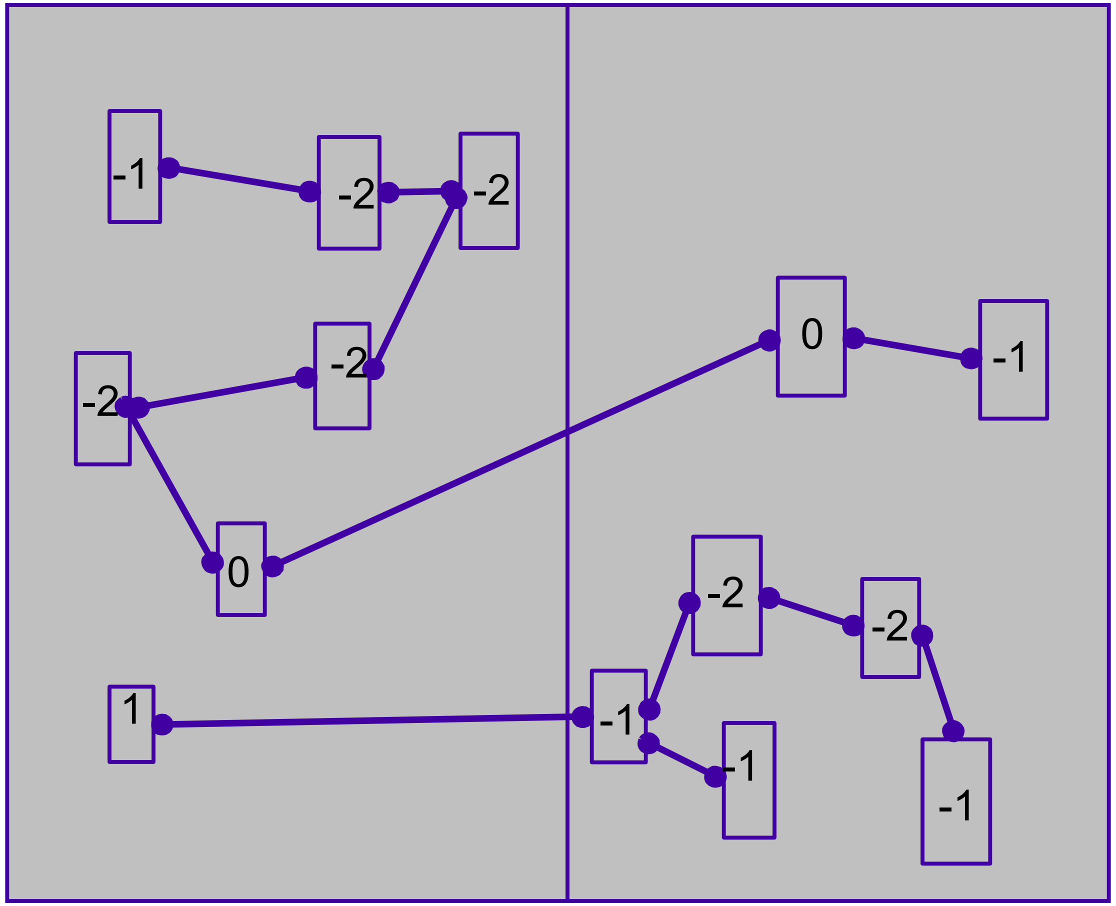
FM Partitioning



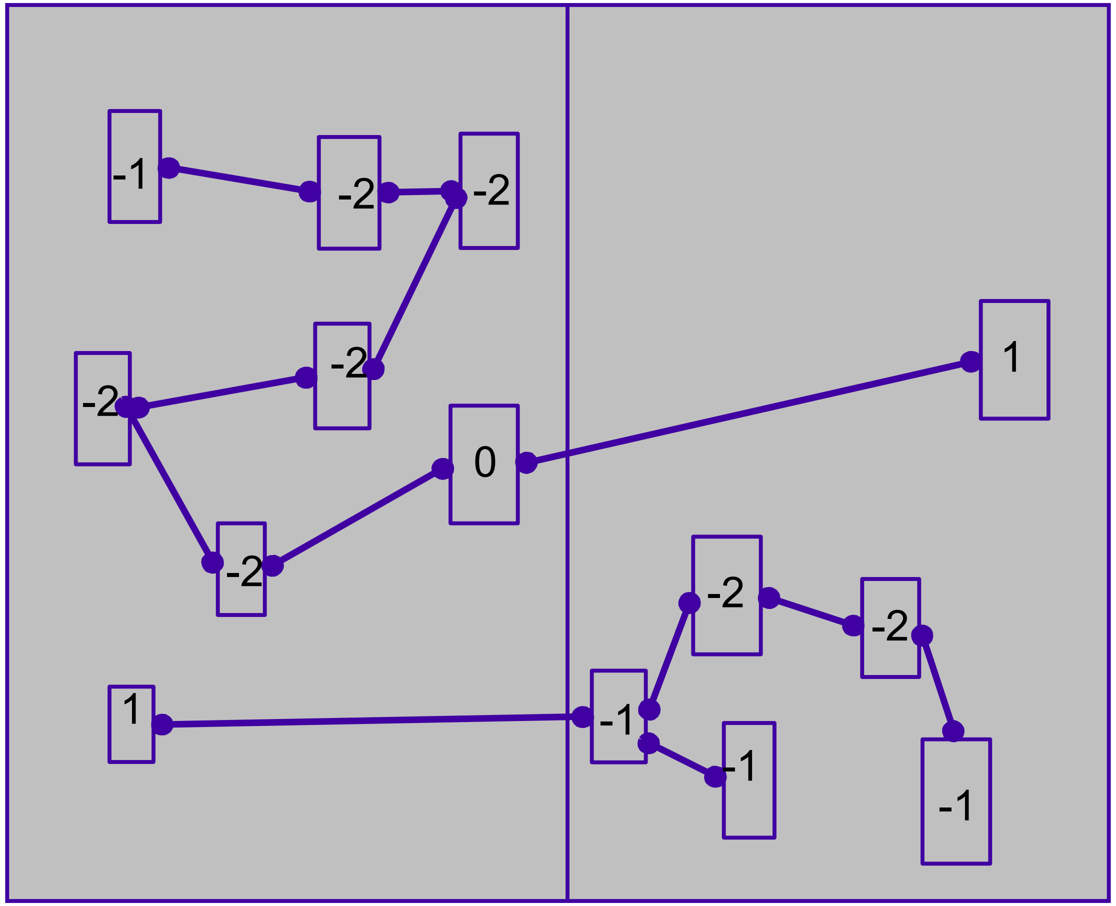
FM Partitioning



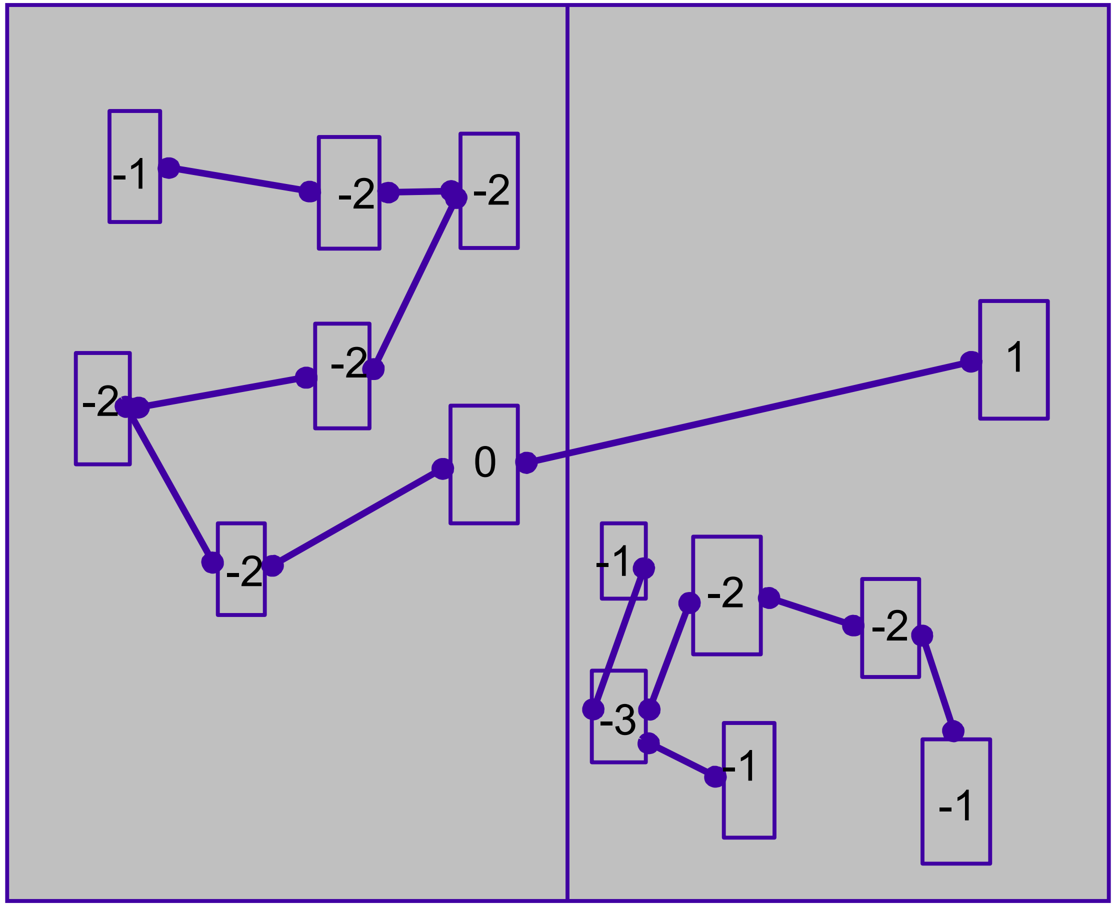
FM Partitioning



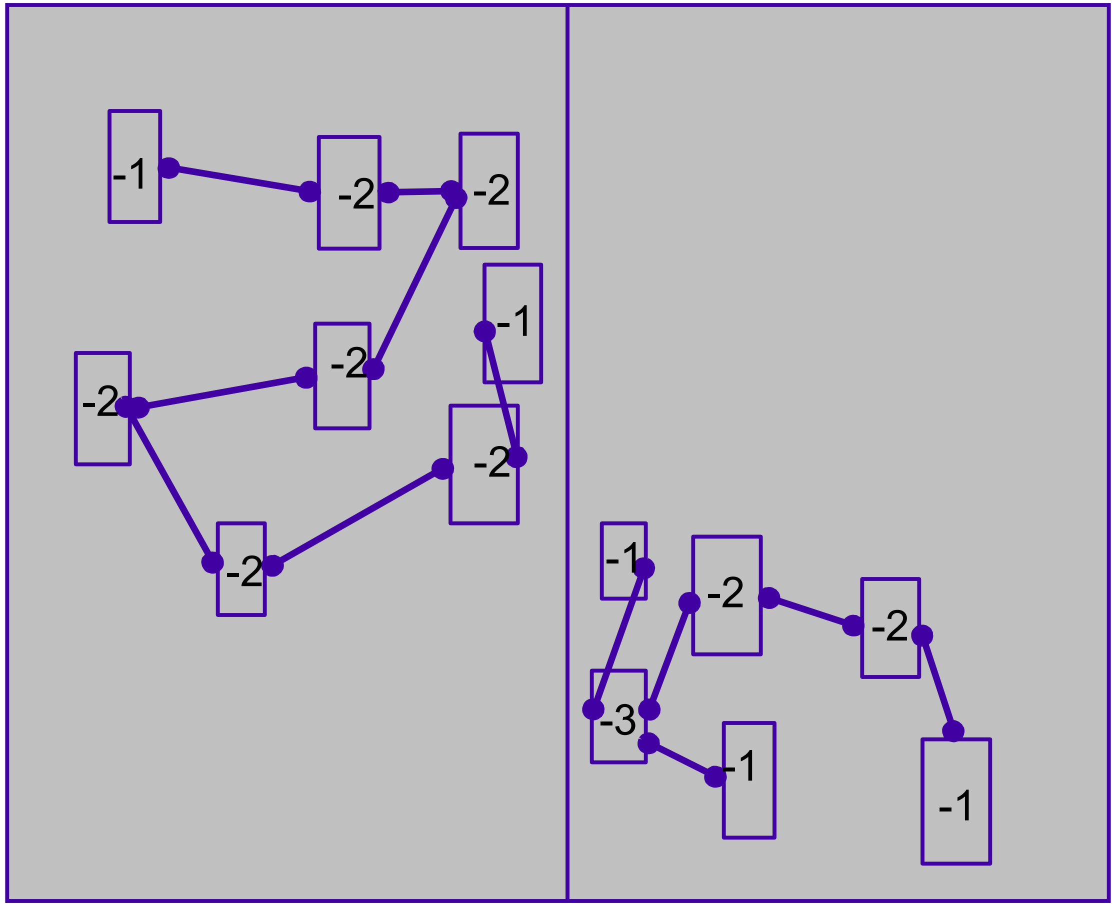
FM Partitioning



FM Partitioning



FM Partitioning



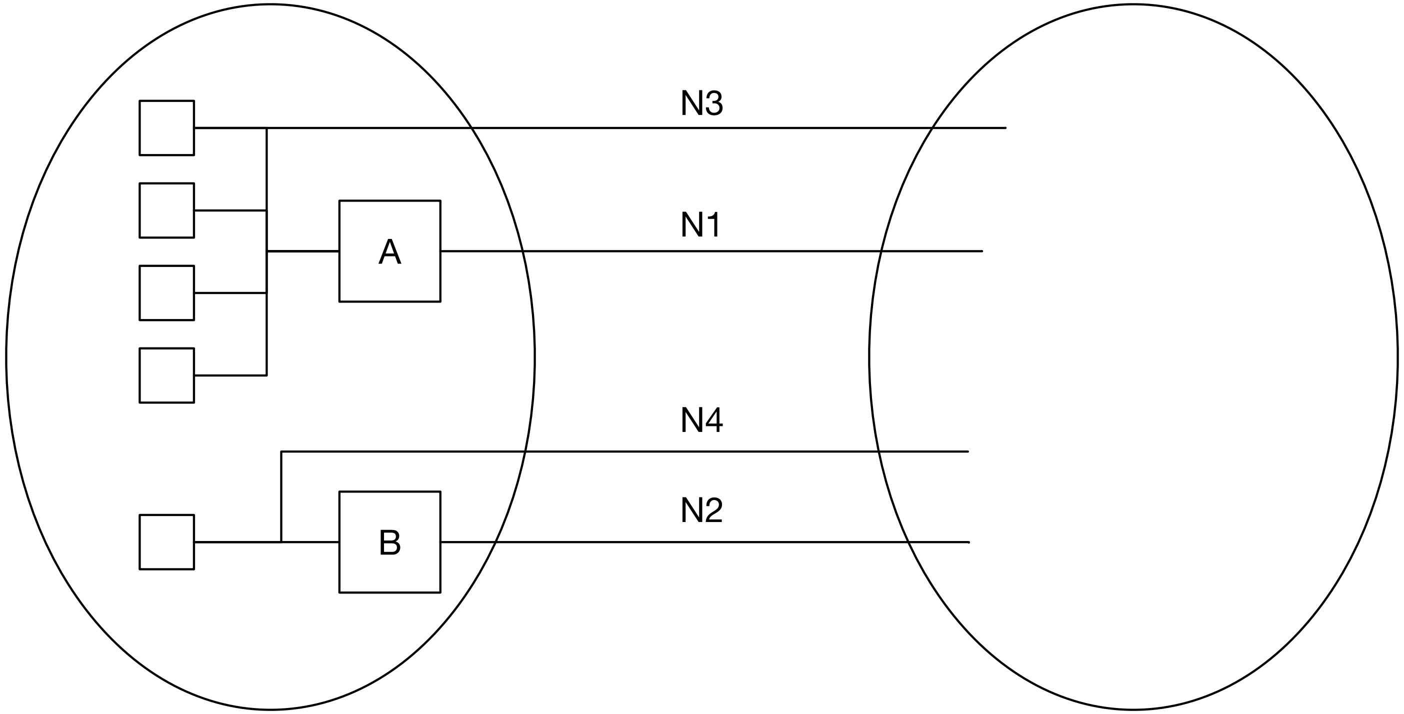
Time Complexity of FM

- For each pass,
  - Constant time to find the best vertex to move.
  - After each move, time to update gain buckets is proportional to degree of vertex moved.
  - Total time is O(p), where p is total number of pins

- Number of passes is usually small.

Extension by Krishnamurthy

- Problem with FM
  - Too greedy
  - Sensitive to the initial partitions



“An Improved Min-Cut Algorithm for Partitioning VLSI Networks”, IEEE Trans. Computer, 33(5):438-446, 1984.

Extension by Krishnamurthy

- Tie-Breaking Strategy
- For each vertex, instead of having a gain bucket, a gain vector is used.
- Gain vector is a sequence of potential gain values corresponding to numbers of possible moves into the future.
- Therefore, r^{th} entry looks r moves ahead.
- Time complexity is O(pr), where r is max # of look-ahead moves stored in gain vector.
- If ties still occur, some researchers observe that LIFO order improves solution quality.

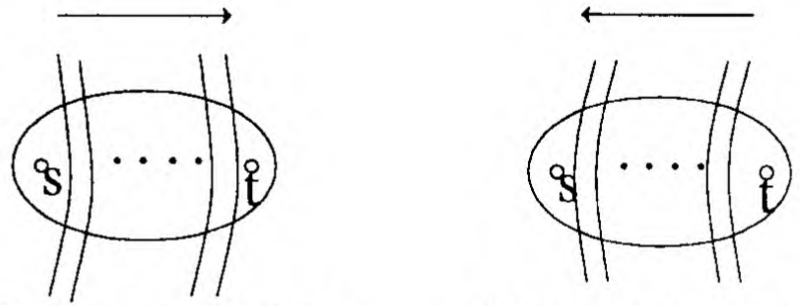
"An Improved Min-Cut Algorithm for Partitioning VLSI Networks", IEEE Trans. Computer, 33(5):438-446, 1984.

Ratio Cut Objective by Wei and Cheng

- It is not desirable to have some pre-defined ratio on the partition sizes.
- Wei and Cheng proposed the Ratio Cut objective.
- Try to locate natural clusters in circuit and force the partitions to be of similar sizes at the same time.
- Ratio Cut $R_{XY} = CXY/(|X|\times|Y|)$
- A heuristic based on FM was proposed.

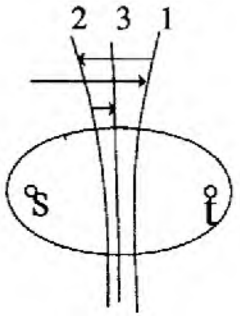
Dual problem of multi-commodity flow

Initial partitioning

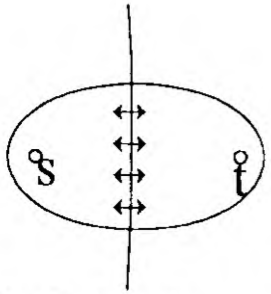


(a) From source s to sink t. (b) From sink t to source s.

Right/left Shifting



Modified FM



“Towards Efficient Hierarchical Designs by Ratio Cut Partitioning”, ICCAD, pages 1:298-301, 1989.

Sanchis Algorithm

- Multi-Way Partitioning
- Dividing into more than 2 partitions.
- Algorithm by extending the idea of FM + Krishnamurthy.

“Multiple-way Network Partitioning”, IEEE Trans. Computers, 38(1):62-81, 1989.

Partitioning: Simulated Annealing

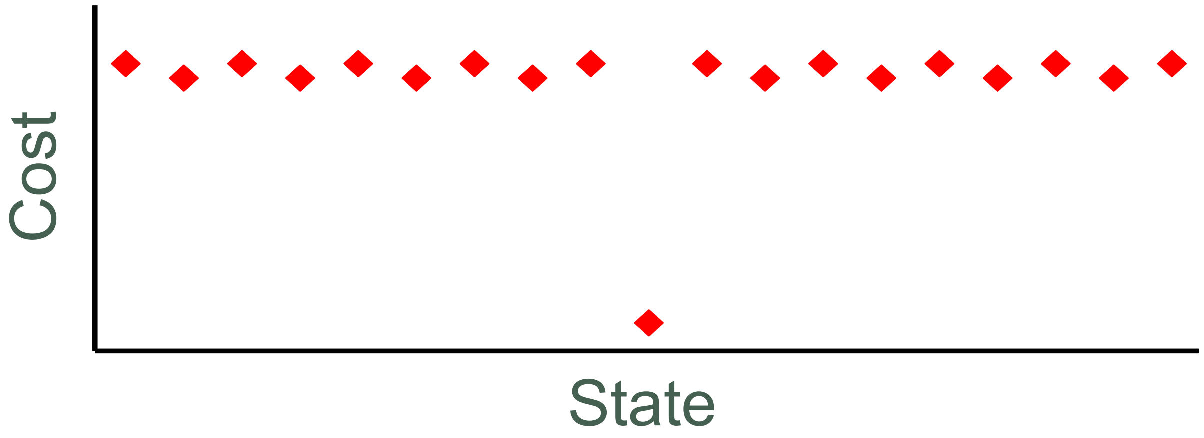
- State Space Search Problem
- Combinatorial optimization problems (like partitioning) can be thought as a State Space Search Problem.
- A <u>State</u> is just a configuration of the combinatorial objects involved.
- The <u>State Space</u> is the set of all possible states (configurations).
- A <u>Neighbourhood Structure</u> is also defined (which states can one go in one step).
- There is a cost corresponding to each state.
- Search for the min (or max) cost state.

Greedy Algorithm

- A very simple technique for State Space Search Problem.
- Start from any state.
- Always move to a neighbor with the min cost (assume minimization problem).
- Stop when all neighbors have a higher cost than the current state.

Problem with Greedy Algorithms

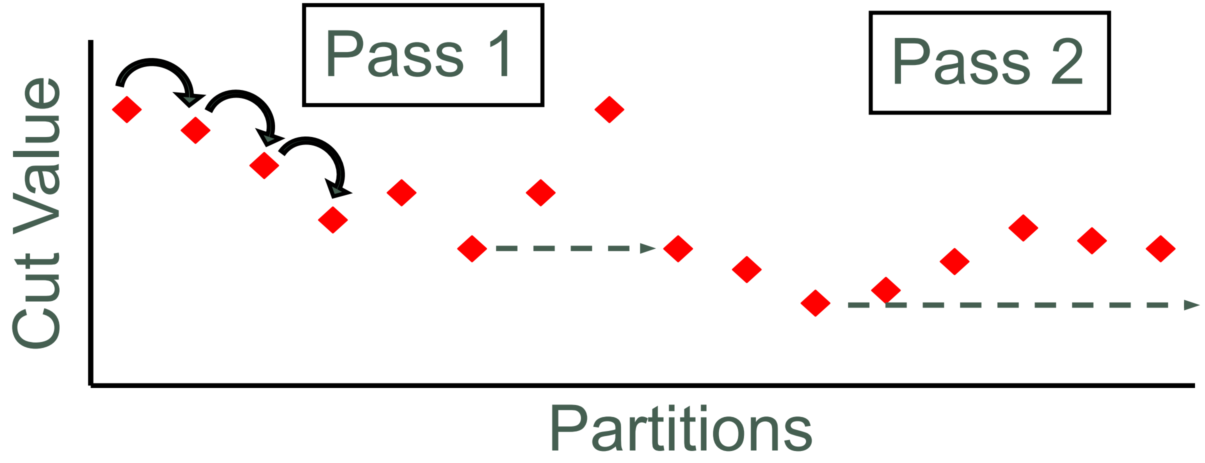
- Easily get stuck at local minimum.
- Will obtain non-optimal solutions.



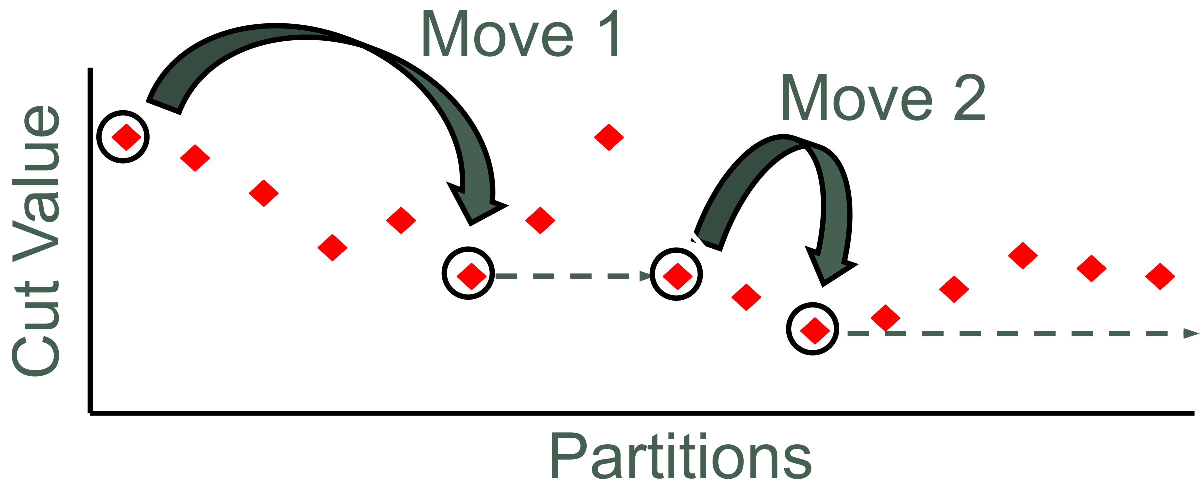
- Optimal only for convex (or concave for maximization) funtions.

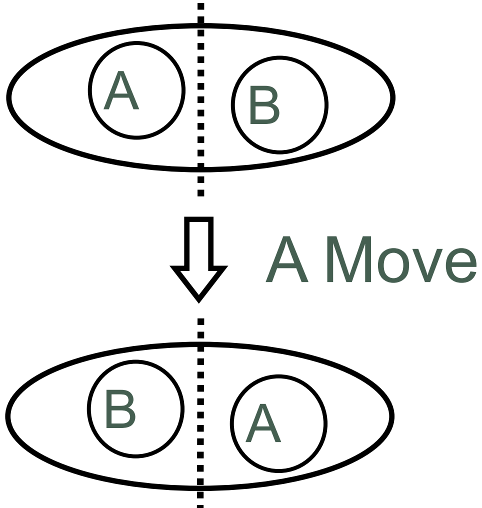
Greedy Nature of KL & FM

- KL and FM are *almost* greedy algorithms.



- Purely greedy if we consider a pass as a “move”.





Simulated Annealing

- Very general search technique.
- Try to avoid being trapped in local minimum by making probabilistic moves.
- Popularize as a heuristic for optimization by:
  - Kirkpatrick, Gelatt and Vecchi, “Optimization by Simulated Annealing”, Science, 220(4598):498-516, May 1983.

Basic Idea of Simulated Annealing

- Inspired by the *Annealing Process*:
  - The process of carefully cooling molten metals in order to obtain a good crystal structure.
  - First, metal is heated to a very high temperature.
  - Then slowly cooled.
  - By cooling at a proper rate, atoms will have an increased chance to regain proper crystal structure.
- Attaining a min cost state in simulated annealing is analogous to attaining a good crystal structure in annealing.

The Simulated Annealing Procedure

Let t be the initial temperature.

Repeat

Repeat

- Pick a neighbor of the current state randomly.
- Let $c = cost$ of current state.
  Let $c' = cost$ of the neighbour picked.
- If $c' < c$, then move to the neighbour (downhill move).
- If $c' > c$, then move to the neighbour with probablility $e^{-(c'-c)/t}$ (uphill move).

Until equilibrium is reached.

Reduce t according to cooling schedule.

Until Freezing point is reached.

Things to decide when using SA

- When solving a combinatorial problem,

  we have to decide:
  - The state space
  - The neighborhood structure
  - The cost function
  - The initial state
  - The initial temperature
  - The cooling schedule (how to change t)
  - The freezing point

Common Cooling Schedules

- Initial temperature, Cooling schedule, and freezing point are usually experimentally determined.
- Some common cooling schedules:
  - $t = \alpha t$, where α is typically around 0.95
  - $t = e^{-\beta t} t$, where β is typically around 0.7
  - ……

Paper by Johnson, Aragon, McGeoch and Schevon on Bisectioning using SA

- An extensive empirical study of Simulated Annealing versus Iterative Improvement Approaches.
- Conclusion: SA is a competitive approach, getting better solutions than KL for random graphs.

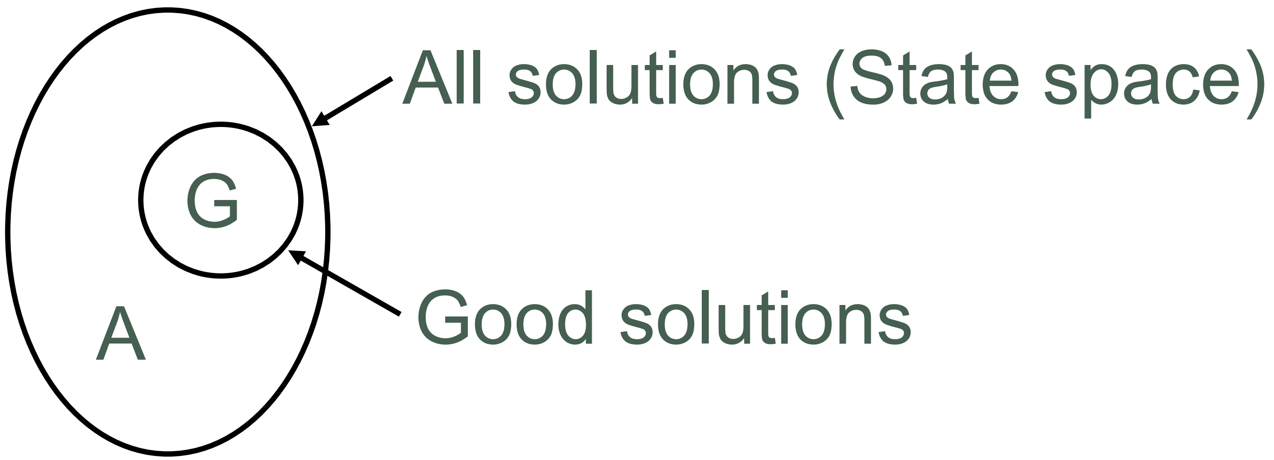
Remarks:

- Netlists are not random graphs, but sparse graphs with local structure.
- SA is too slow. So KL/FM variants are still most popular.
- Multiple runs of KL/FM variants with random initial solutions may be preferable to SA.

“Optimization by Simulated Annealing: An Experimental Evaluation Part I, Graph Partitioning”, Operations Research, 37:865-892, 1989.

The Use of Randomness

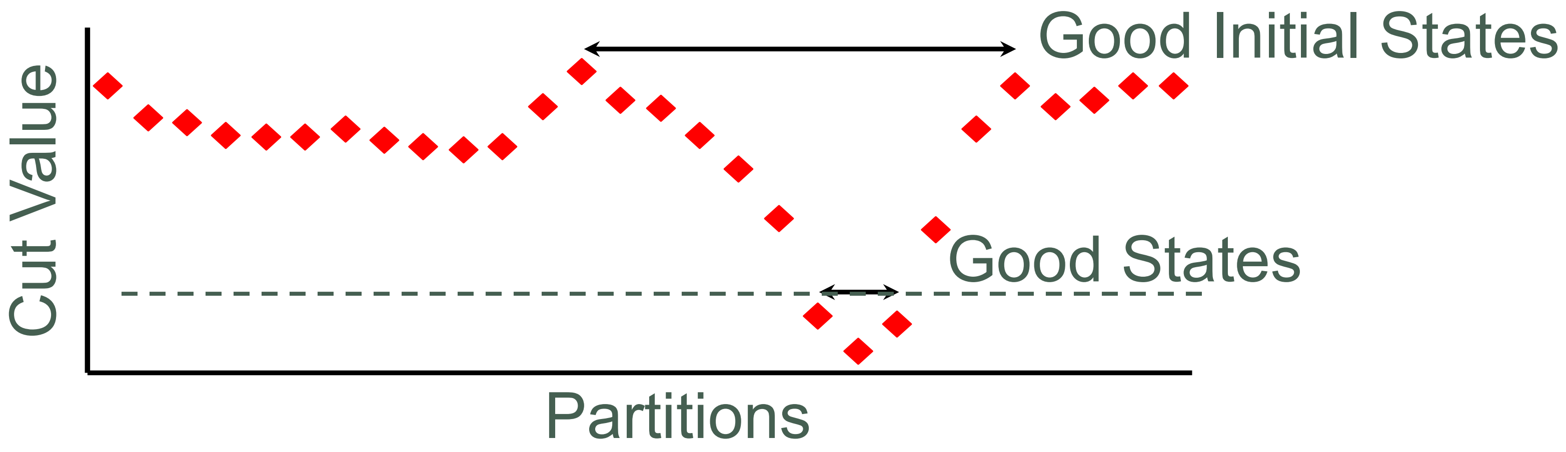
- For any partitioning problem:



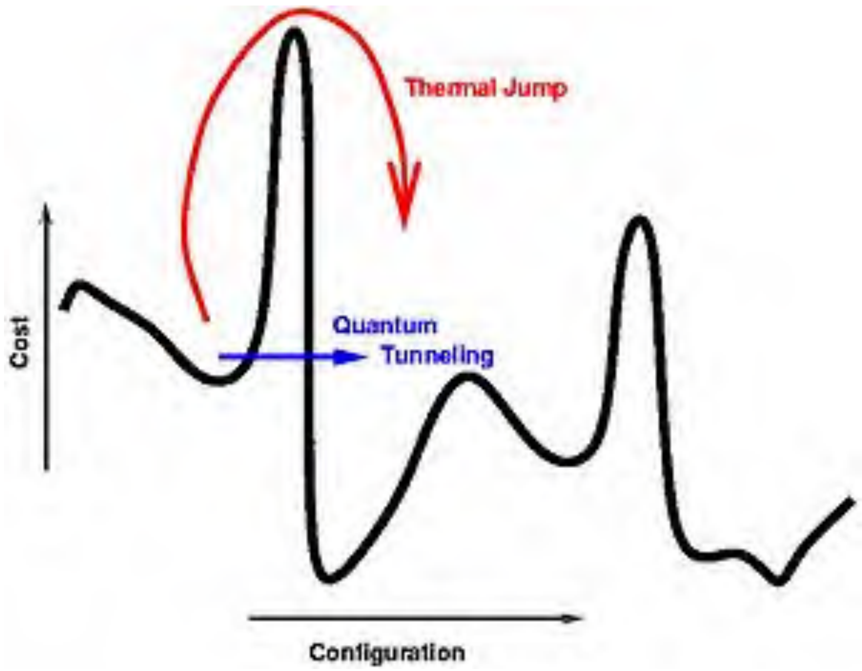
- Suppose solutions are picked randomly.
- If $|G|/|A| = r$, Pr(at least 1 good in $5/r$ trials) = $1-(1-r)^{5/r}$
- If $|G|/|A| = 0.001$, Pr(at least 1 good in 5000 trials) = $1-(1-0.001)^{5000} = 0.9933$

Adding Randomness to KL/FM

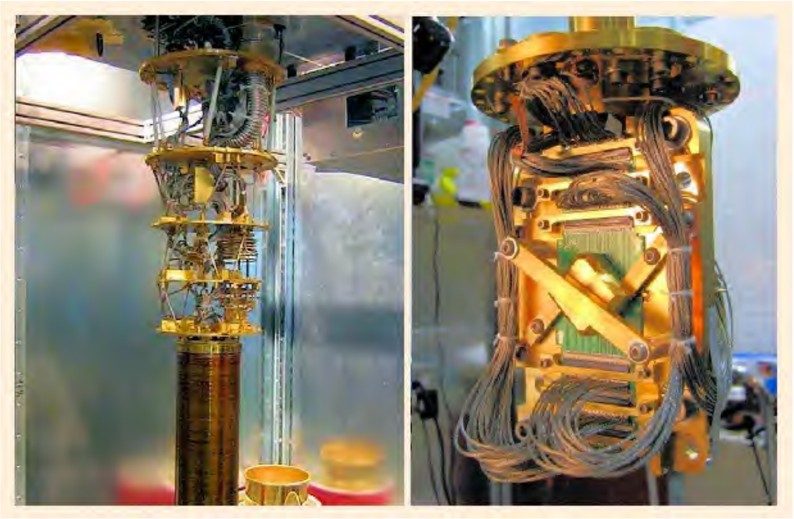
- In fact, # of good states are extremely few. Therefore, r is extremely small.
- Need extremely long time if just picking states randomly (without doing KL/FM).
- Running KL/FM variants several times with random initial solutions is a good idea.



Something Even Fancier – Quantum Annealing



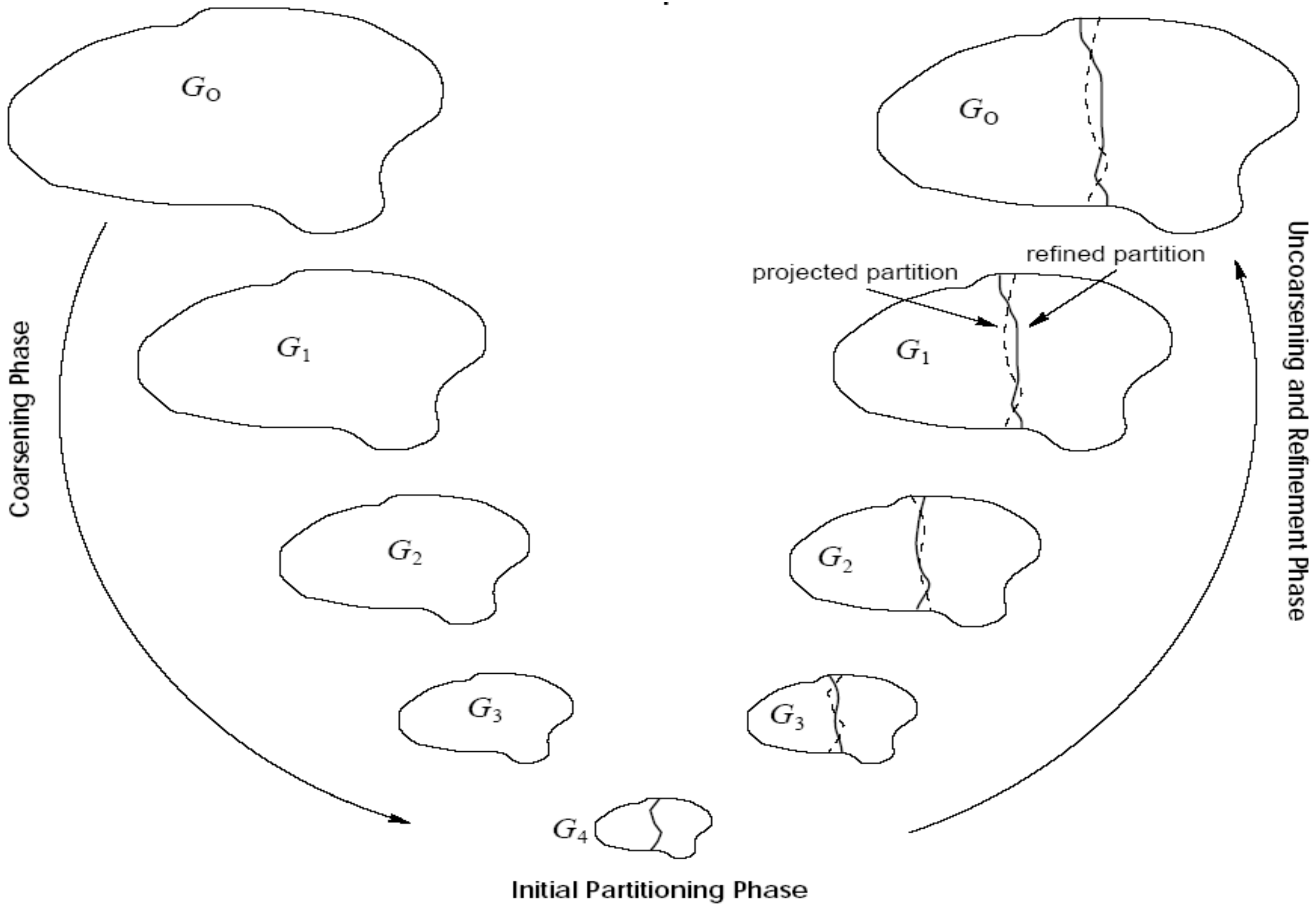
Analytical and numerical evidence suggests that quantum annealing outperforms simulated annealing under certain conditions – Wikipedia



Some Other Approaches

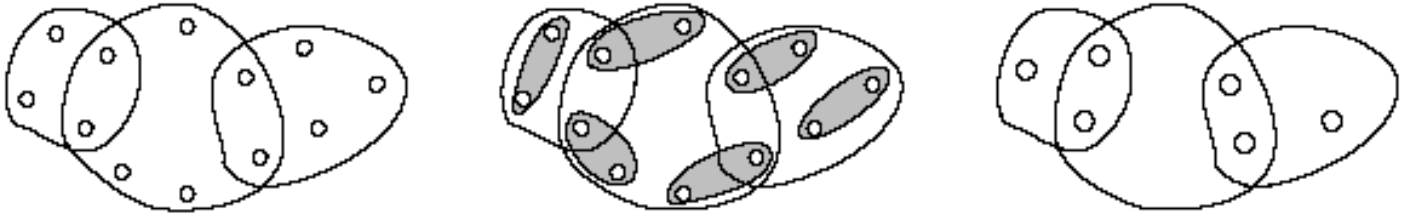
- KL/FM-SA Hybrid: Use KL/FM variant to find a good initial solution for SA, then improve that solution by SA at low temperature.
- Tabu Search
- Genetic Algorithm
- Spectral Methods (finding Eigenvectors)
- Network Flows
- Quadratic Programming
- ......

Partitioning: Multi-Level Technique

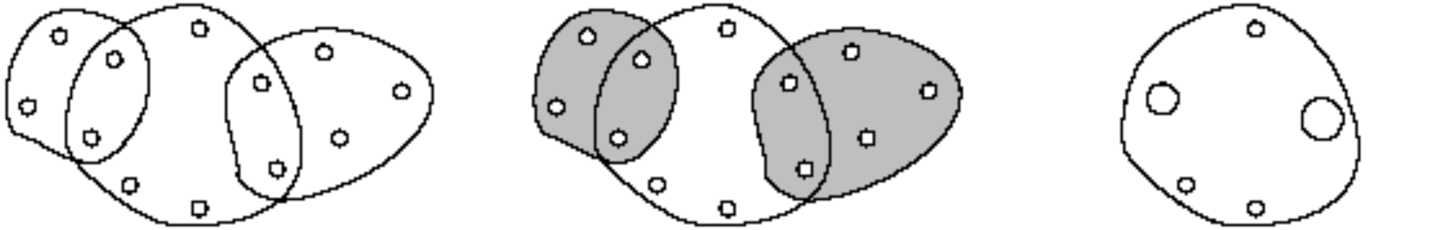


Coarsening Phase

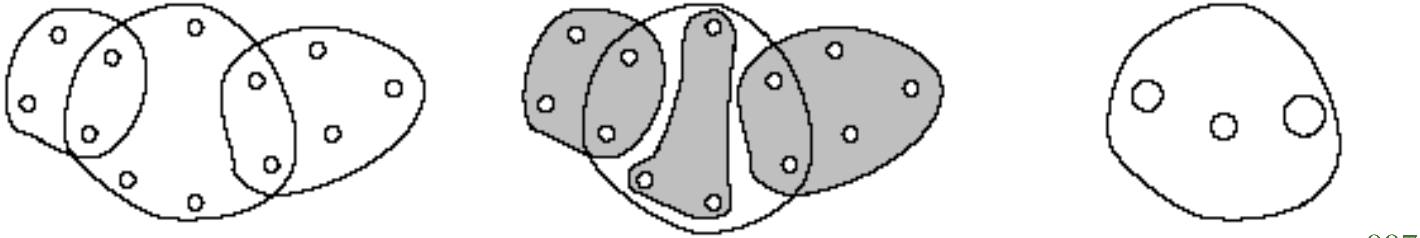
- Edge Coarsening



- Hyper-edge Coarsening (HEC)



- Modified Hyperedge Coarsening (MHEC)



G. Karypis, R. Aggarwal, V. Kumar and S. Shekhar, "Multilevel Hypergraph Partitioning: Applications in VLSI Domain" DAC 1997.

Uncoarsening and Refinement Phase

1. FM:

- Based on FM with two simplifications:
  - Limit number of passes to 2
  - Early-Exit FM (FM-EE), stop each pass if k vertex moves do not improve the cut

2. HER (Hyperedge Refinement)

- Move a group of vertices between partitions so that an entire hyperedge is removed from the cut

G. Karypis, R. Aggarwal, V. Kumar and S. Shekhar, "Multilevel Hypergraph Partitioning: Applications in VLSI Domain" DAC 1997.

hMETIS Algorithm

- Software implementation available for free download from Web
- hMETIS-EE_{20}
  - 20 random initial partitions
  - with 10 runs using HEC for coarsening
  - with 10 runs using MHEC for coarsening
  - FM-EE for refinement
- hMETIS-FM_{20}
  - 20 random initial partitions
  - with 10 runs using HEC for coarsening
  - with 10 runs using MHEC for coarsening
  - FM for refinement

G. Karypis, R. Aggarwal, V. Kumar and S. Shekhar, "Multilevel Hypergraph Partitioning: Applications in VLSI Domain" DAC 1997.

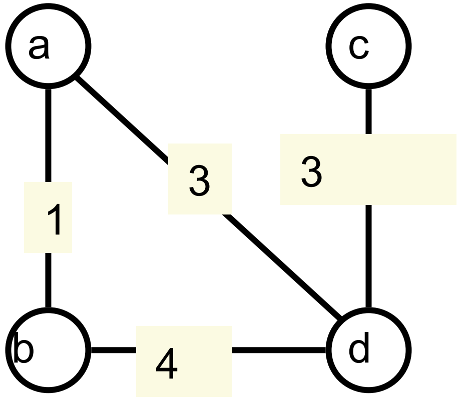
Experimental Results

- Compared with five previous algorithms
- hMETIS-EE_{20} is:
  - 4.1% to 21.4% better
  - On average 0.5% better than the best of the 5 algorithms
  - Roughly 1 to 15 times faster
- hMETIS-FM_{20} is:
  - On average 1.1% better than hMETIS-EE_{20}
  - Improve the best-known bisections for 9 out of 23 test circuits
  - Twice as slow as hMETIS-EE_{20}

Spectral and Flow Algorithms

- Two elegant partition algorithms
  - although not the fastest
- Learn how to formulate the problem!
  - Key to VLSI CAD
  - Spectral based partitioning algorithms
  - Max-flow based partition algorithm

Spectral Based Partitioning Algorithms



$$A = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 0 & 3 \\ b & 1 & 0 & 0 & 4 \\ c & 0 & 0 & 0 & 3 \\ d & 3 & 4 & 3 & 0 \end{array} \qquad D = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 4 & 0 & 0 & 0 \\ b & 0 & 5 & 0 & 0 \\ c & 0 & 0 & 3 & 0 \\ d & 0 & 0 & 0 & 10 \end{array}$$

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

Eigenvectors of D-A form the Laplacian spectrum of Q

Eigenvalues and Eigenvectors

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

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).

A Basic Property

$$\underline{x}^T A \underline{x} = (x_1, \cdots, x_n) \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ & \vdots & \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

$$= \left(\sum_{i=1}^{n} x_i a_{i1}, \quad \cdots \quad \sum_{i=1}^{n} x_i a_{in}, \right) \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

$$= \sum_{i,j} x_i x_j a_{ij}$$

Basic Idea of Laplacian Spectrum Based Graph Partitioning

- Given a bisection (X, X'), define a partitioning vector

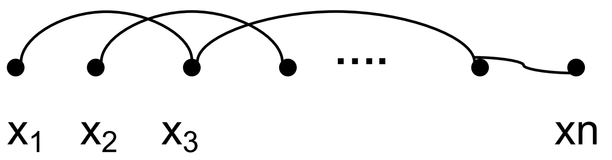
$$\underline{x} = (x_1, x_2, \cdots, x_n) \text{ s.t. } x_i = \begin{cases} -1 & i \in X \\ 1 & i \in X' \end{cases}$$

clearly, $\underline{x} \perp \underline{1}, \underline{x} \neq \underline{0}$ ($\underline{1} = (1, 1, \ldots, 1)$, $\underline{0} = (0, 0, \ldots, 0)$)

- For a partition vector x:
- Let $S = \{\underline{x} \perp \underline{1}\ and\ \underline{x} \neq \underline{0}\}$. Finding best partition vector $\underline{x}$ such that the total edge cut $C(X, X')$ is minimum is relaxed to finding $\underline{x}$ in S such that $\frac{1}{4}\sum_{(i,j)\in E}(x_i - x_j)^2$ is minimum

- Linear placement interpretation:

minimize total squared wirelength



Property of Laplacian Matrix

(1) $$x^T Qx = \sum Q_{ij} x_i x_j$$

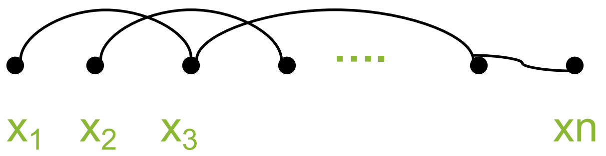
$$= x^T Dx - x^T Ax$$

$$= \sum d_i x_i^2 - \sum A_{ij} x_i x_j$$

$$= \sum d_i x_i^2 - \sum_{(i,j)\in E} 2 a_{ij} x_i x_j$$

$$= \sum_{(i,j)\in E} (x_i - x_j)^2$$

= 4 * C(X, X')



squared wirelength

If *x* is a partition vector

Therefore, we want to minimize $x^T Qx$

Property of Laplacian Matrix (Cont'd)

(2) Q is symmetric and semi-definite, i.e.

(i) $x^T Qx = \sum Q_{ij} x_i x_j \geq 0$

(ii) all eigenvalues of Q are ≥0

(3) The smallest eigenvalue of Q is 0

corresponding eigenvector of Q is x0= (1, 1, …, 1)

(not interesting, all modules overlap and x0∉S)

(4) According to Courant-Fischer minimax principle:

the 2nd smallest eigenvalue satisfies:

$$\lambda = \min_{x \text{ in } S} \frac{x^T Qx}{|x|^2}$$

Results on Spectral Based Graph Partitioning

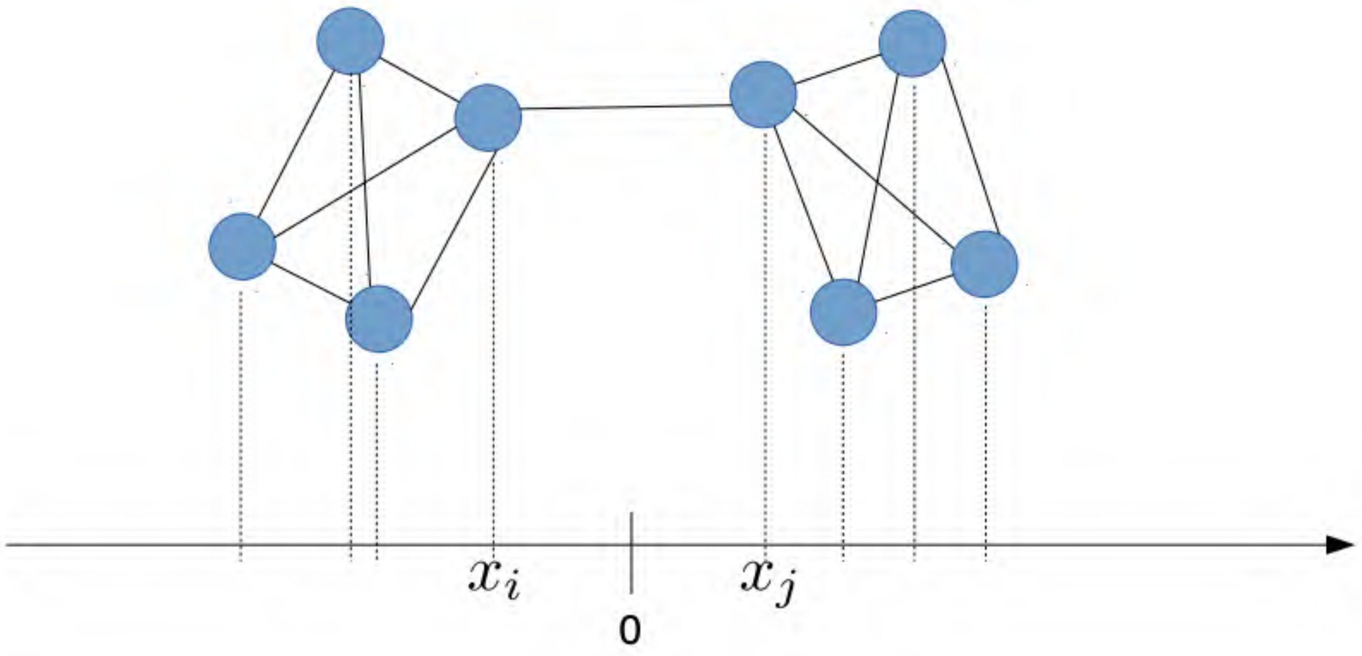
- Min bisection cost $C(X, X') \geq n \cdot \lambda / 4$
- Min ratio-cut cost $C(X, X') / |X| \cdot |X'| \geq \lambda / n$
- The second smallest eigenvalue gives the best linear placement
- Compute the best bisection or ratio-cut
    - based on the second smallest eigenvector

Computing the Eigenvector

- Only need one eigenvector
  - (the second smallest one)
- Q is symmetric and sparse
- Use block Lanczos Algorithm

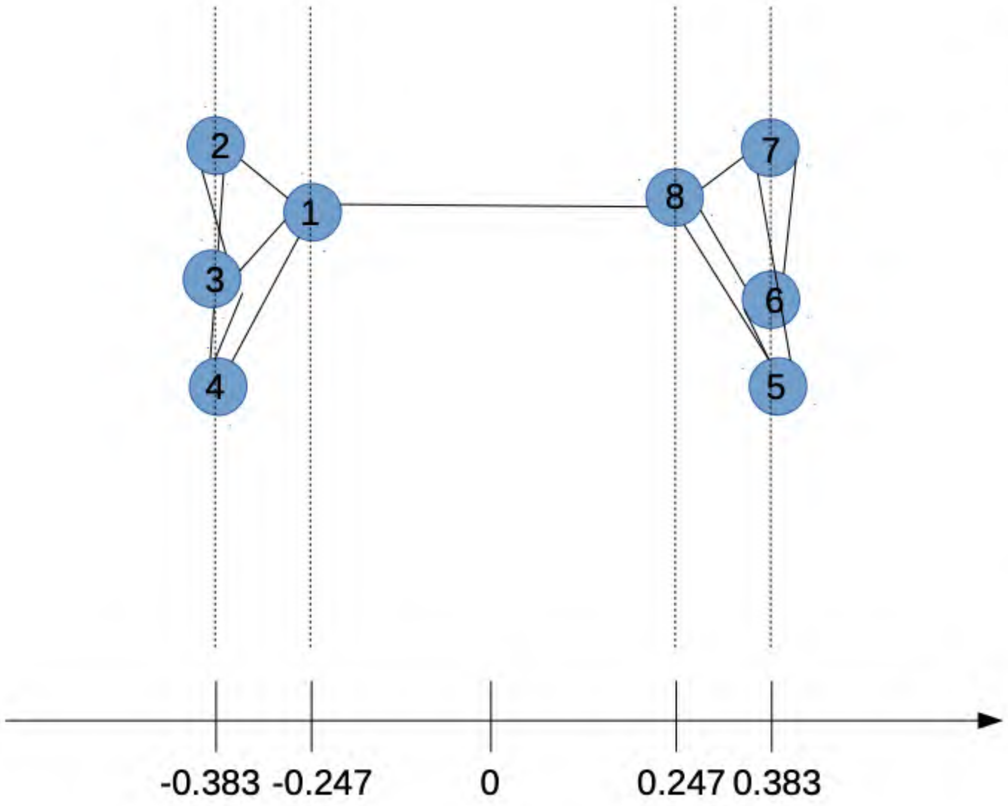
What Does This Mean

$$\lambda_2 = \min_{x:\sum x_i=0} \sum_{(i,j)\in E} (x_i - x_j)^2$$



Evimaria Terzi: "Clustering: graph cuts and spectral graph partitioning"
Daniel A. Spielman: "The Laplacian"
Jure Leskovec: "Defining the graph laplacian"

What Does This Mean



$$\lambda_1 = 0$$
$$\lambda_2 = 0.354$$

$$v_2 = \begin{bmatrix} 0.247 \\ 0.383 \\ 0.383 \\ 0.383 \\ -0.383 \\ -0.383 \\ -0.383 \\ -0.247 \end{bmatrix}$$

Evimaria Terzi: "Clustering: graph cuts and spectral graph partitioning"
Daniel A. Spielman: "The Laplacian"
Jure Leskovec: "Defining the graph laplacian"

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]

Zhuo Feng
Associate Professor
Department of Electrical and Computer Engineering
Michigan Technological University, Houghton, MI

Biography

Zhuo Feng received the Ph.D. degree in Electrical and Computer Engineering from Texas A&M University, College Static
University of Singapore, Singapore, in 2005 and the B.Eng. degree in Information Engineering from Xi'an Jiaotong University,
and Computer Engineering, Michigan Technological University, Houghton, MI, where he is affiliated with the Computer Engi
National Science Foundation (NSF) in 2014, a Best Paper Award from ACM/IEEE Design Automation Conference (DAC) in
Computer-Aided Design (ICCAD) in 2006 and 2008. He has served on the technical program committees of major internation
DAC, ISQED, and VLSI-DAT, and has been a technical referee for many leading IEEE/ACM journals in VLSI and parallel co
Department of Energy (DoE). He is a Senior Member of IEEE. In 2016, he became a co-founder of LeapLinear Solutions to
(networks) with billions of elements, based on the latest breakthroughs in spectral graph theory.

Research

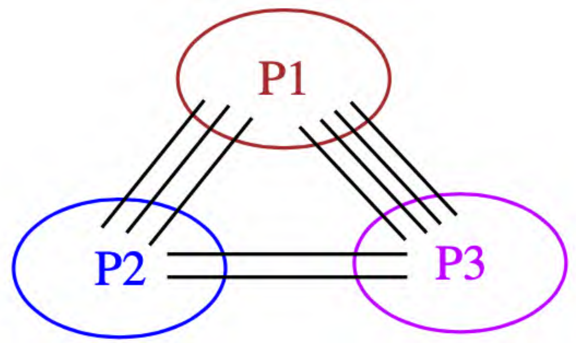
- High-performance spectral methods for numerical and graph problems
  1. Spectral sparsification of undirected graphs (DAC'16, DAC'18, software) and directed graphs (arXiv:1812.04165)
  2. Sparsified algebraic multigrid (SAMG) for solving SDD matrices (ICCAD'17)
  3. Spectral graph reduction for scalable graph partitioning and data visualization (arXiv: 1812.08942, DAC'19)
  4. Spectral methods for data clustering and network reduction (arXiv:1710.04584, ICCAD'14)

https://pages.mtu.edu/~zhuofeng/
https://dblp.org/pid/81/4441.html

Network Flow Based Partitioning

- Min-cut balanced partitioning: Yang and Wong, ICCAD-94.
  - Based on max-flow min-cut theorem.

Selected in The Best of ICCAD (20 years of Excellence in Computer Aided Design), 2003



- Gate replication for partitioning: Yang and Wong, ICCAD-95.
- Gate replication for partitioning: Yang and Wong, ICCAD-95.
- Multi-way partitioning with area and pin constraints: Liu and Wong, ISPD-97.
- Multi-resource partitioning: Liu, Zhu, and Wong, FPGA-98.
- Partitioning for time-multiplexed FPGAs: Liu and Wong, ICCAD-98.

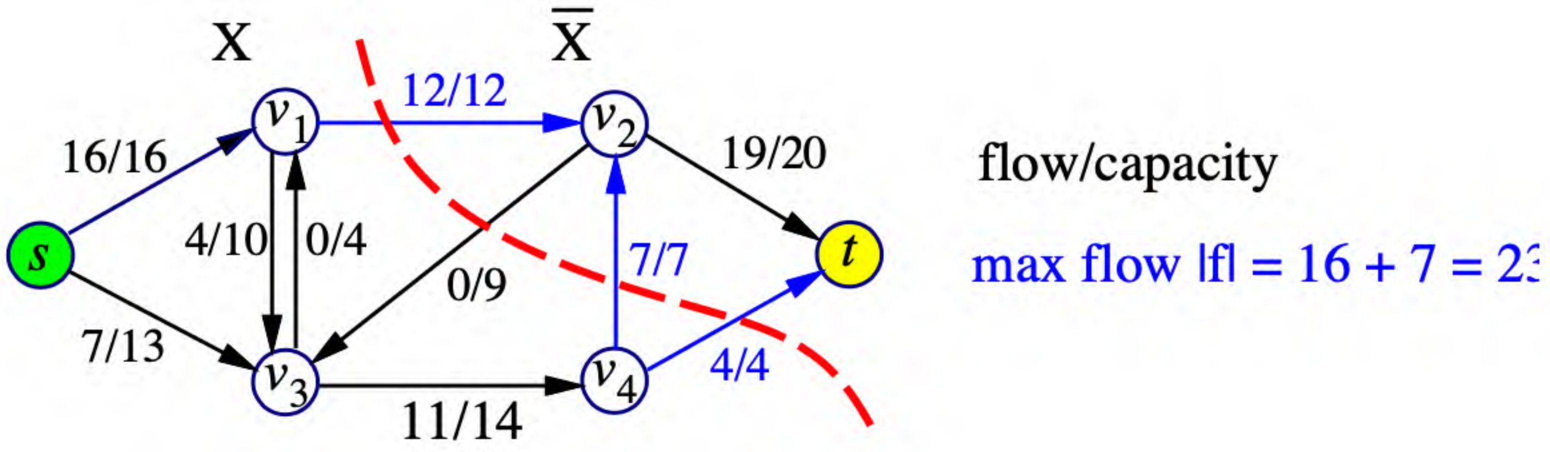
H. Yang and D. F. Wong, "Efficient Network Flow Based Min-cut Balanced Partitioning", IEEE Transactions on Computer-Aided Design, pp 1533-1540, 1996

Flow Networks

- A **flow network** $G = (V, E)$ is a **directed** graph in which each edge $(u, v) \in E$ has a **capacity** $c(u, v) > 0$.
- There is exactly one node with no incoming (outgoing) edges, called the **source** s (**sink** t).
- A **flow** $f : V \times V \rightarrow R$ satisfies
  - **Capacity constraint:** $f(u, v) \leq c(u, v), \forall u, v \in V$.
  - **Skew symmetry:** $f(u, v) = -f(v, u), \forall u, v \in V$.
  - **Flow conservation:** $\sum_{v \in V} f(u, v) = 0, \forall u \in V - \{s, t\}$.
- The **value** of a flow f: $|f| = \sum_{v \in V} f(s, v) = \sum_{v \in V} f(v, t)$

Flow Networks

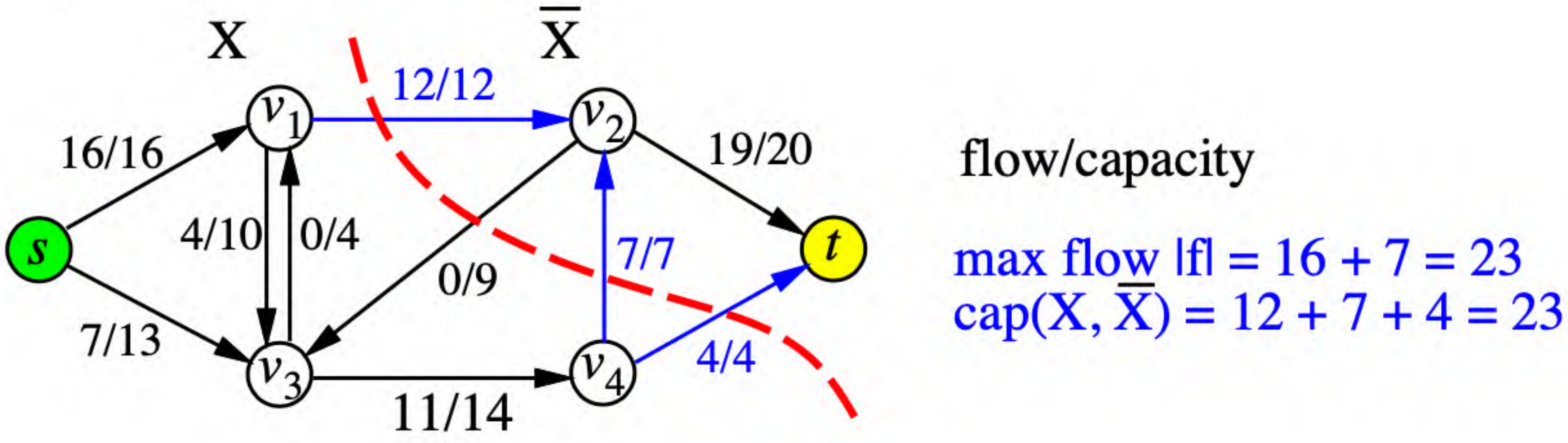
- **Maximum-flow problem:** Given a flow network G with source s and sink t, find a flow of maximum value from s to t.



Max-Flow Min-Cut

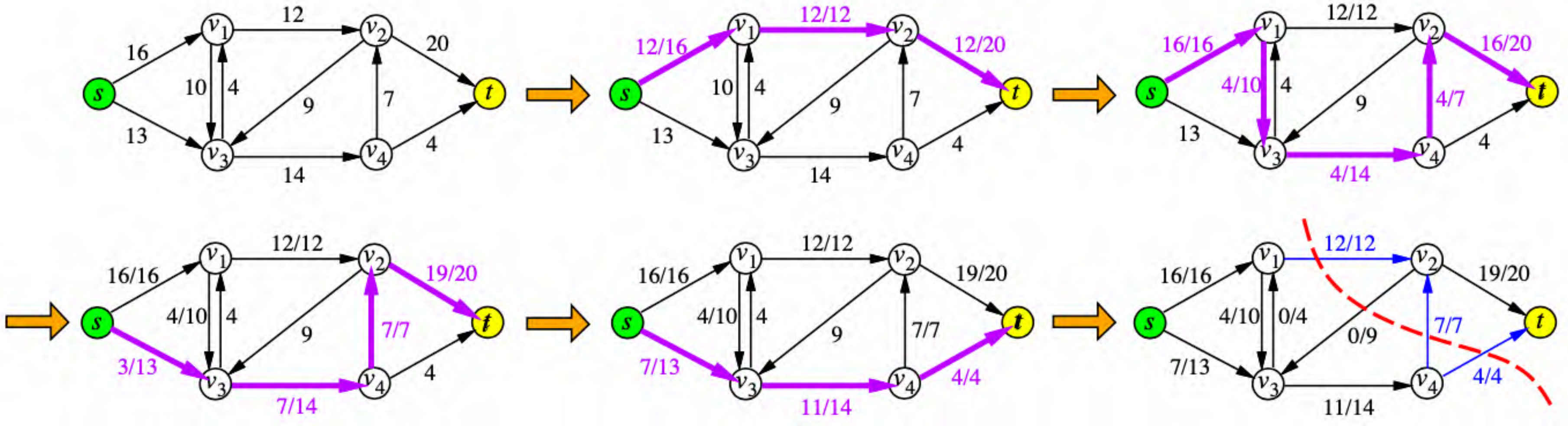
- A **cut** $(X, \bar{X})$ of flow network $G = (V, E)$ is a partition of V into X and $\bar{X} = V - X$ such that $s \in X$ and $t \in \bar{X}$.
  - **Capacity of a cut:** $cap(X, \bar{X}) = \sum_{u \in X, v \in \bar{X}} c(u, v)$ (Count only **forward** edges!)
- **Max-flow min-cut theorem** Ford & Fulkerson, 1956.
  - f is a max-flow $\Leftrightarrow |f| = cap(X, \bar{X})$ for some min-cut $(X, \bar{X})$.

Special case of **duality** theorem for linear programs



Network Flow Algorithms

- An **augmenting path** p is a simple path from s to t with the following properties:
  - For every edge $(u, v) \in E$ on p in the **forward** direction (a **forward edge**), we have $f(u, v) < c(u, v)$.
  - For every edge $(u, v) \in E$ on p in the **reverse** direction (a **backward edge**), we have $f(u, v) > 0$.
- f is a max-flow $\Leftrightarrow$ no more augmenting path.



Network Flow Algorithms

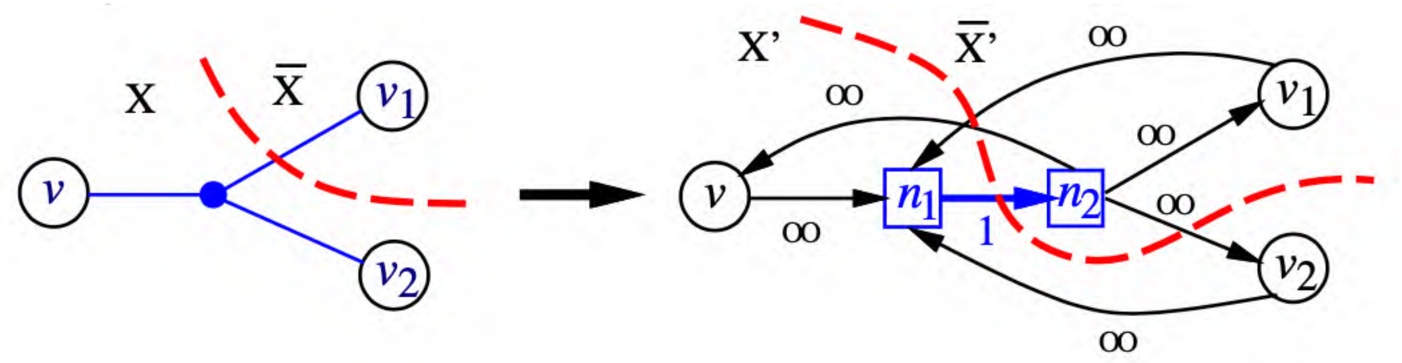
- An augmenting path p is a simple path from s to t with the following properties:
  - For every edge $(u, v) \in E$ on p in the forward direction (a forward edge), we have $f(u, v) < c(u, v)$.
  - For every edge $(u, v) \in E$ on p in the reverse direction (a backward edge), we have $f(u, v) > 0$.
- f is a max-flow $\Leftrightarrow$ no more augmenting path.
- First algorithm by Ford & Fulkerson in 1959: $O(|E||f|)$
- First **polynomial-time** algorithm by Edmonds & Karp in 1969: $O(|E|^2|V|)$
- Goldberg & Tarjan in 1985: $O(|E||V| \log(|V|^2/|E|))$, etc.

Network Flow Based Partitioning

- Why was the technique not wisely used in partitioning?
  - Works on graphs, not hypergraphs.
  - Results in unbalanced partitions; repeated min-cut for balance: $|V|$ max-flows, time-consuming!
- Yang & Wong, ICCAD-94.
  - Exact **net** modeling by flow network.
  - Optimal algorithm for min-net-cut bipartition (unbalanced).
  - Efficient implementation for repeated min-net-cut: same asymptotic time as **one** max-flow computation.

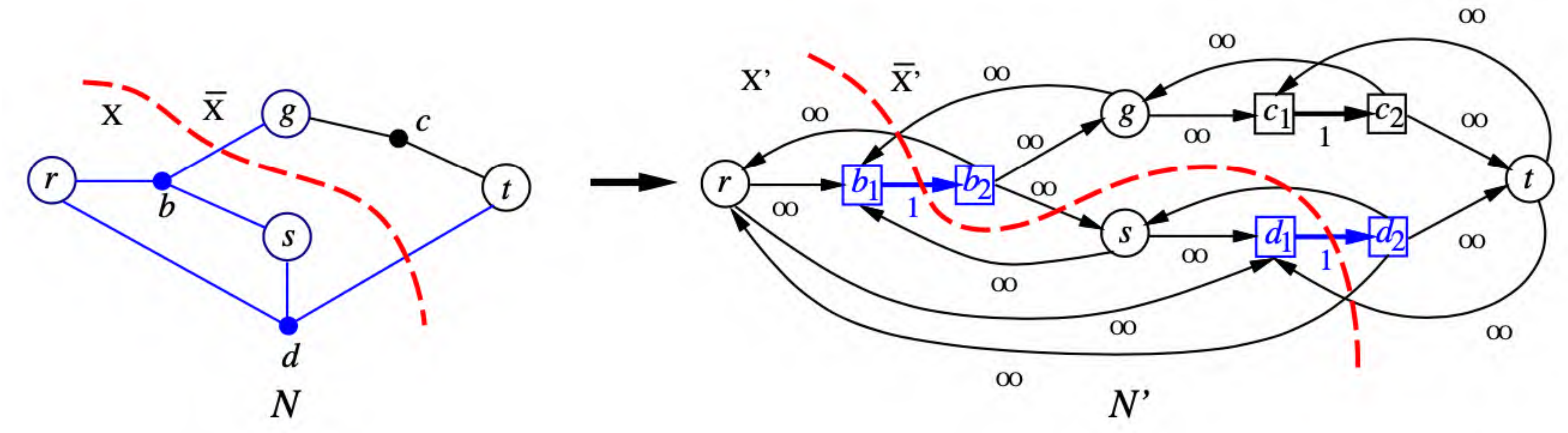
Min-Net-Cut Bipartition

- Net modeling by flow network:



- A min-net-cut $(X, \bar{X})$ in N $\Leftrightarrow$ A min-capacity-cut $(X, \bar{X})$ in N'.
- Size of flow network: $|V'| \leq 3|V|, |E'| \leq 2|E| + 3|V|$.
- Time complexity: $O(\mathrm{min{-}net{-}cut{-}size}) \times |E| = O(|V||E|)$.

Min-Net-Cut Bipartition

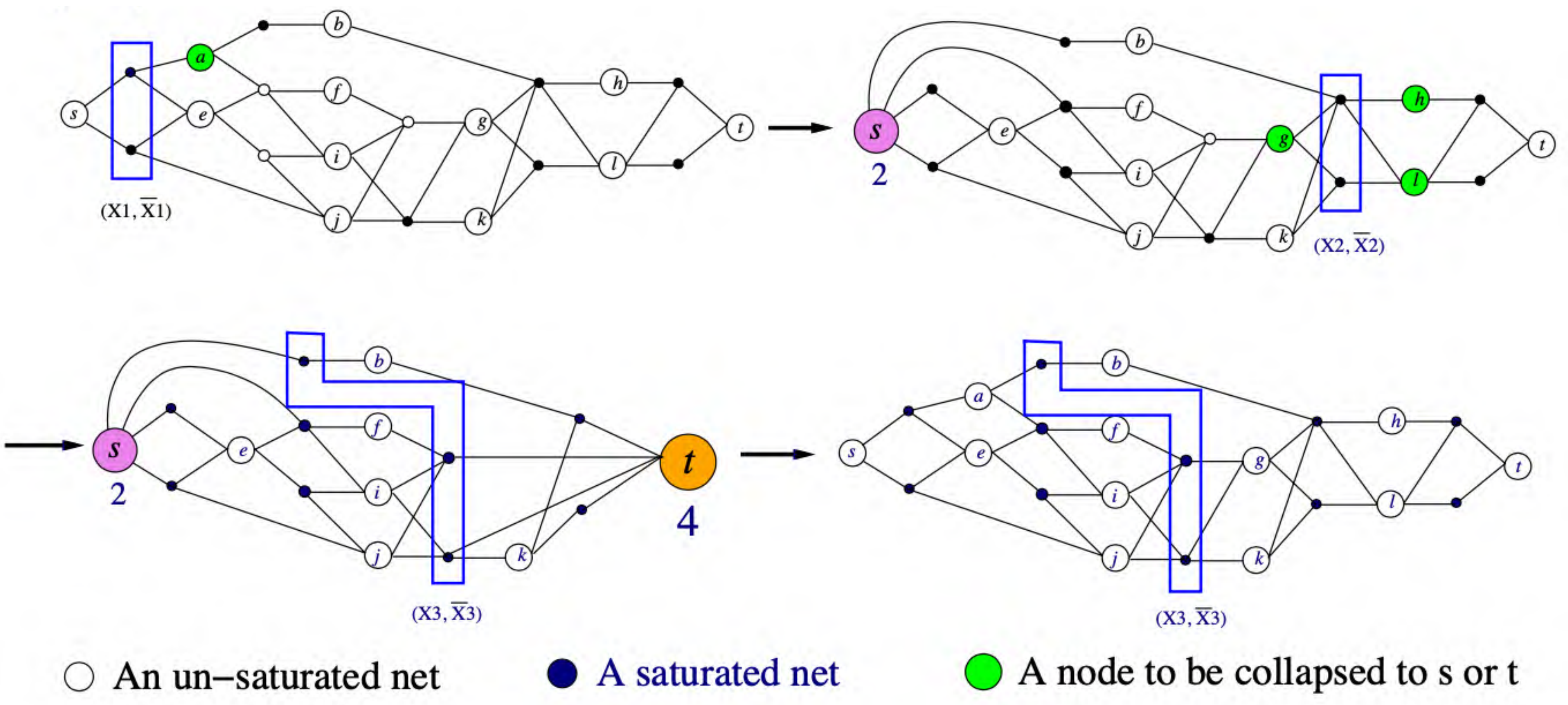


4 nodes: {d, g, r, s, t}
3 nets
(r; g, s)
(s; r, t)
(g; t)

New constructed graph

Repeated Min-Cut for Flow Balanced Bipartition (FBB)

- Allow component weights to deviate from $(1-\varepsilon)W/2$ to $(1+\varepsilon)W/2$.

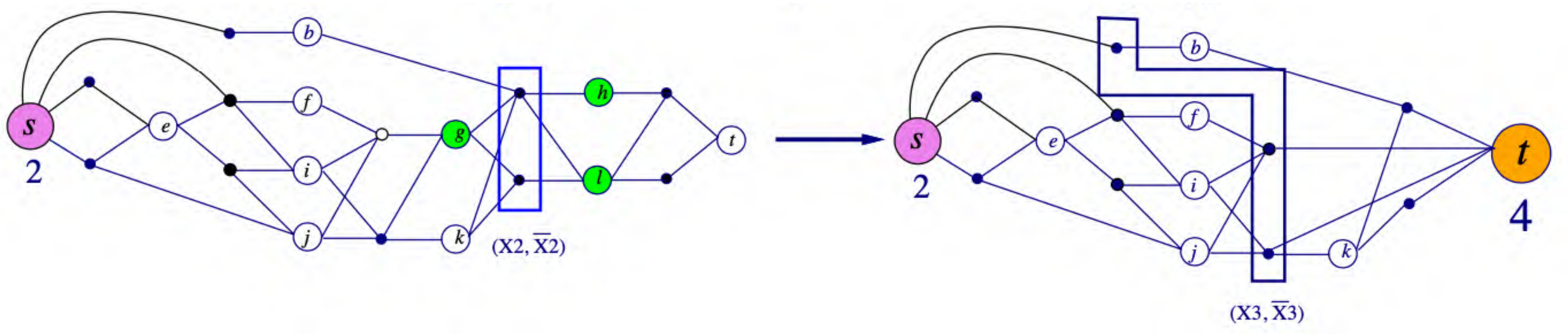


Incremental Flow

- Repeatedly compute max-flow: very time-consuming.
- No need to compute max-flow from scratch in each iteration.
- Retain the flow function computed in the previous iteration.
- Find additional flow in each iteration. Still correct.
- FBB time complexity: $O(|V||E|)$, same as **one** max-flow.
  - At most $2|V|$ augmenting path computations.
    - At each augmenting path computation, either an augmenting path is found, or a new cut is found, and at least 1 node is collapsed to s or t.
    - At most $|f| \leq |V|$ augmenting paths found, since bridging edges have unit capacity.

Incremental Flow

– An augmenting path computation: O(|E|) time.



H. Yang and D. F. Wong, "Efficient Network Flow Based Min-cut Balanced Partitioning", IEEE Transactions on Computer-Aided Design, pp 1533-1540, 1996

Partitioning Summary

- Greedy
  - KL
  - FM
  - Multi-level: hMETIS
- Search
  - Simulated annealing
- Analytical
  - Spectral method
  - Flow-based method

Fast & flexible, but quality varies

Theoretically sound,
but may be inefficient