Partitioning

Some contributions from
Lei He
Andrew B. Kahng
Igor Markov
Mohammad Tehranipoor

Logistics

- Remember: labs are to be done by yourself. They are NOT a team effort

Hierarchical Partitioning

- **System Level Partitioning**: A system is partitioned into a set of subsystems whereby each sub-system can be *designed* and *fabricated* independently on a PCB or MCM. The criterion for partitioning is the functionality and each PCB/MCM serves a specific task within a system.

If PCB is too large:

- **Board Level Partitioning**: The circuit assigned to a PCB is partitioned into sub-circuits such that each sub-circuit can be fabricated as a VLSI chip.

If chip is too large:

- **Chip Level Partitioning**: The circuit assigned to a chip is partitioned into smaller subcircuits.

System Level Partitioning

- The circuit assigned to PCB must meet certain constraints:
  - E.g., Fixed area, i.e. *32 cm X 15cm*
    - Fixed number of terminals, i.e. 64
- **Objectives:**
  - Minimize the number of boards:
    - The reliability of the system is inversely proportional to the number of PCBs in the systems.
  - Optimize the system performance:
    - Partitioning must minimize any degradation of the performance caused by the delay due to the connections between components in different boards. System bus is slow!

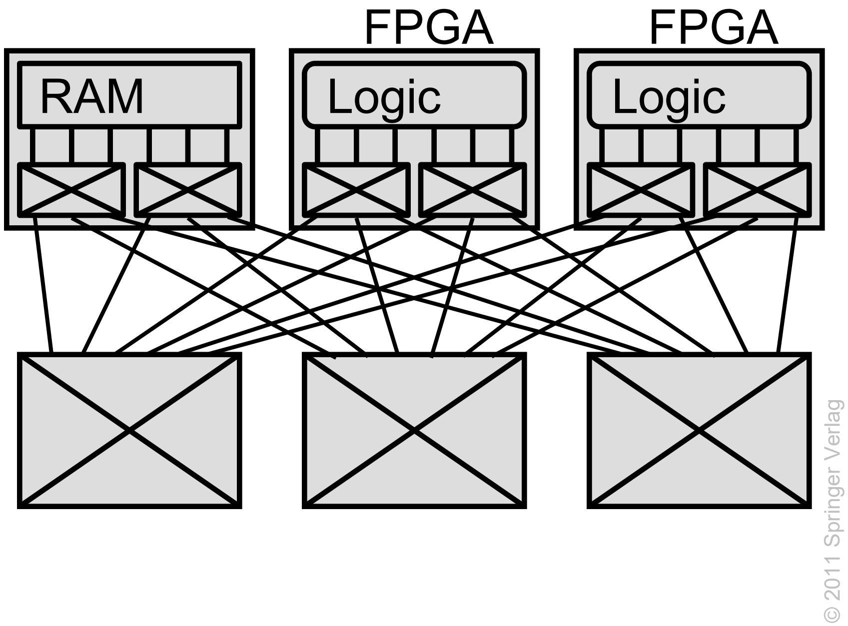
Board Level Partitioning

- Unlike system level partitioning, board level partitioning faces different set of constraints and fulfills different set of objectives.

- chips can have different sizes and different number of terminals.
  - Size: i.e. from *2mm X 2mm* to *25mm X 25mm*
  - Terminal: i.e. from *64 to 300*

- **Objective:**
  - Minimize the number of chips in each board.
  - Minimize the area of each chip.
  - Optimize the board performance.

Another Example: System Partitioning onto Multiple FPGAs



Mapping of a typical system architecture onto multiple FPGAs

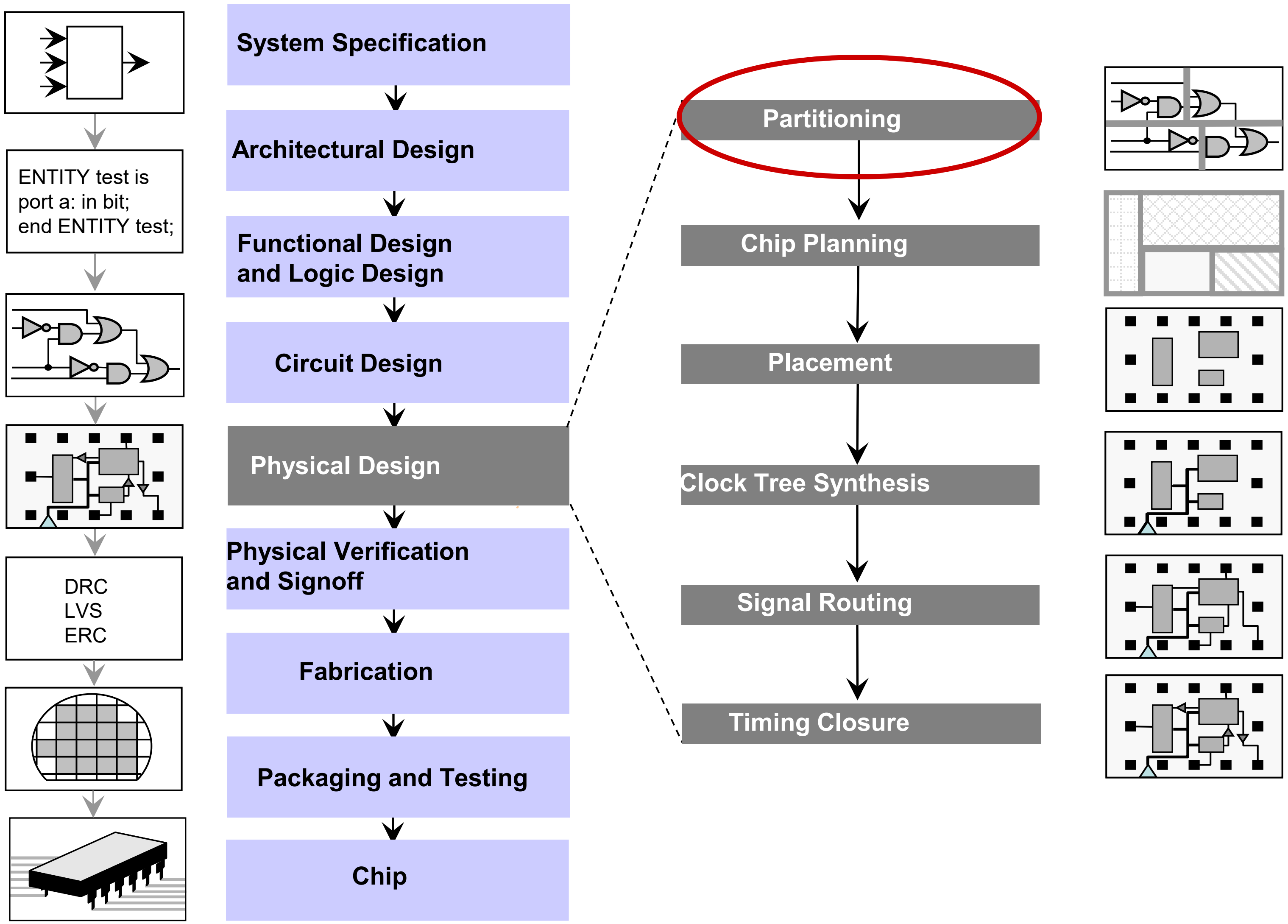
Chip Level Partitioning

- Each block can be independently designed.
- There is no area constraint for any partition.
- The number of nets between blocks (partitions) cannot be greater than the terminal count of the partition.
  - The number of pins is based on the block size

- **Objective**:
  - The number of nets cut by partitioning should be minimized.
    - It simplifies the routing task.
    - It mostly results in minimum degradation of performnace.

- **Drawback:** Partitioning may degrade the performance.

Introduction

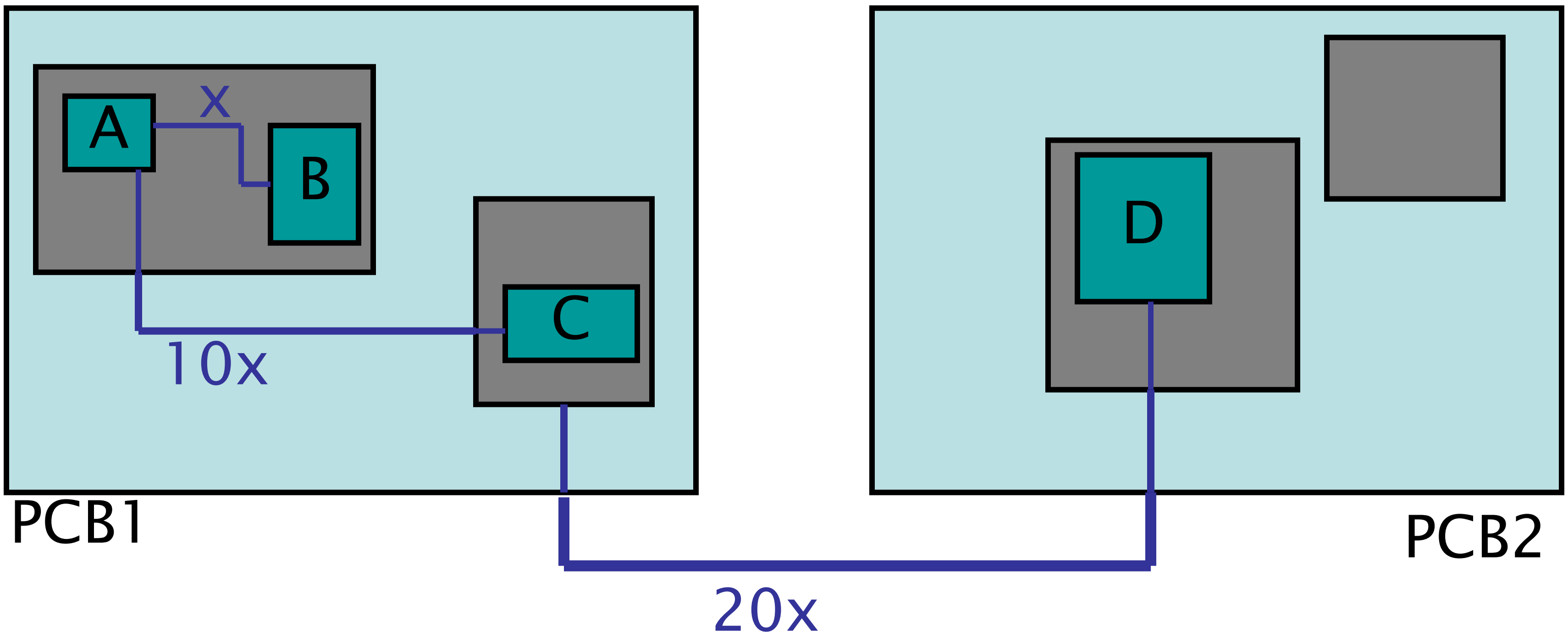


Circuit Partitioning

- **Partitioning**:
  - The process of decomposing a circuit/system into smaller subcircuits/subsystems, which are called **block**, is called ***partitioning***.

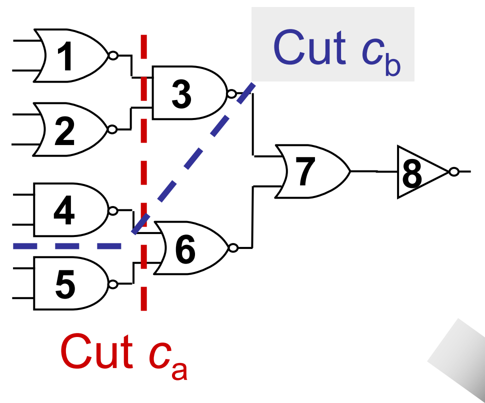
- The partitioning **speeds up** the design process.
- Blocks can be designed independently.
- Original functionality of system remains intact.
- An interface specification is generated during the decomposition.
- The decomposition must ensure minimization of interconnections.
- Time required for decomposition must be a small fraction of total design time.
- There may be more than 15 units working on Intel uP.

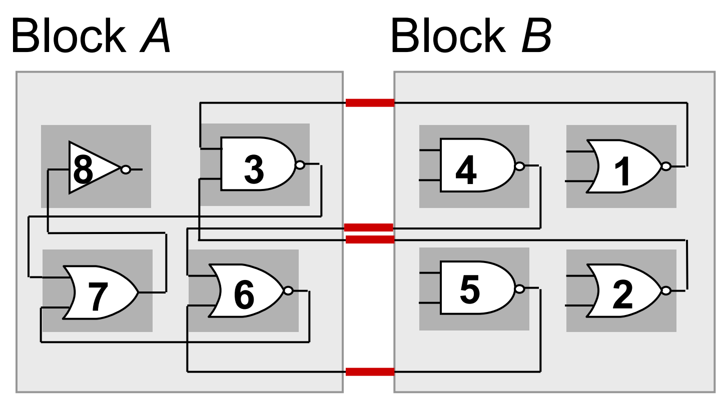
Delay at Different Levels of Partitions



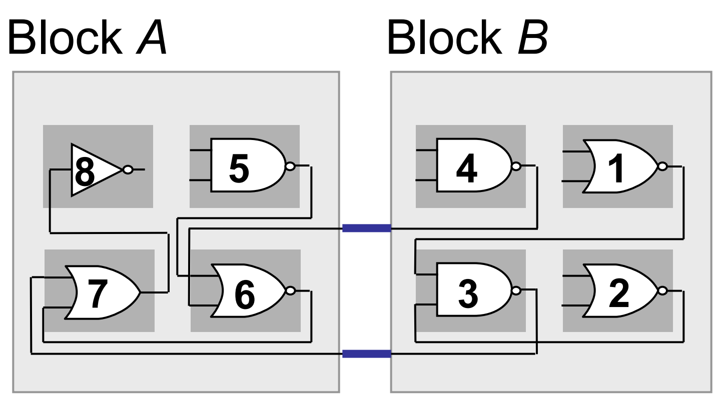
An Example

Circuit:



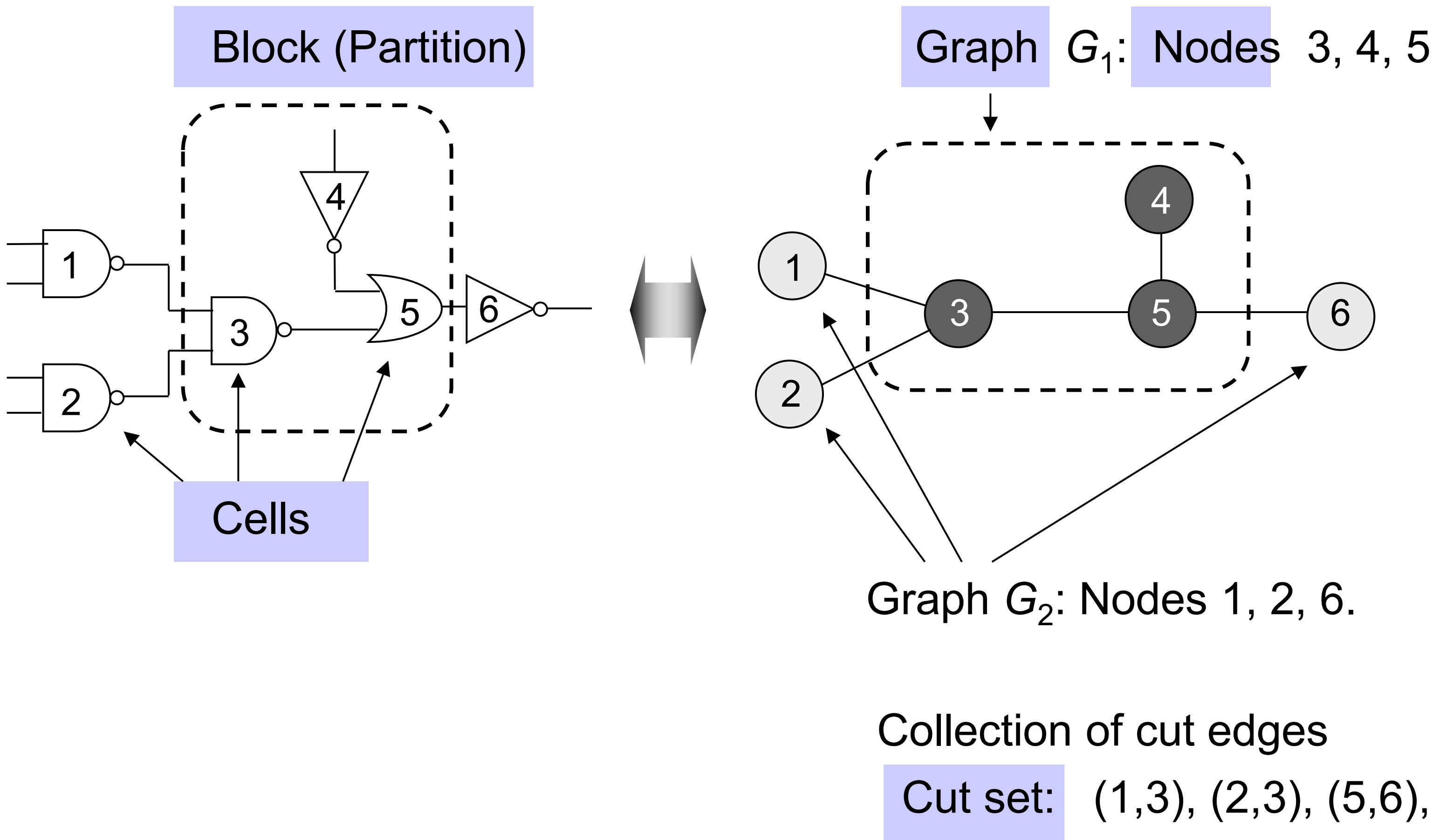


Cut c_a: four external connections



Cut c_b: two external connections

Terminology



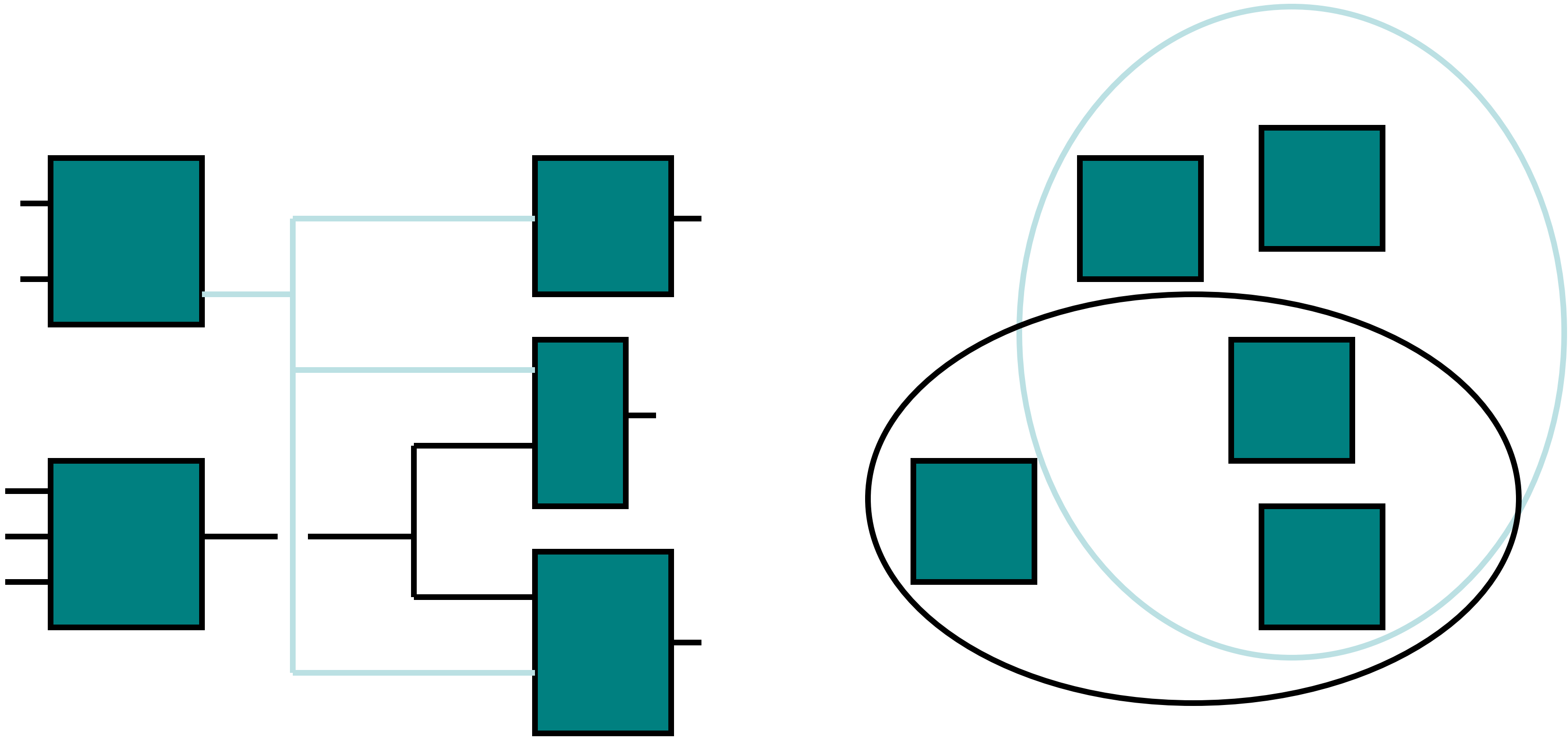
Puneet Gupta (puneet@ee.ucla.edu)

Optimization Goals

- Given a graph $G(V,E)$ with $|V|$ nodes and $|E|$ edges where each node $v \in V$ and each edge $e \in E$.
- Each node has area $s(v)$ and each edge has cost or weight $w(e)$.
- The objective is to divide the graph G into k disjoint subgraphs such that all optimization goals are achieved and all original edge relations are respected.
  - Number of connections between partitions is minimized
  - Each partition meets all design constraints (size, number of external connections..)
  - Balance every partition as well as possible
- Unfortunately, this problem is NP-hard
  - Efficient heuristics developed in the 1970s and 1980s. They are high quality and in low-order polynomial time.

Hypergraphs in VLSI CAD

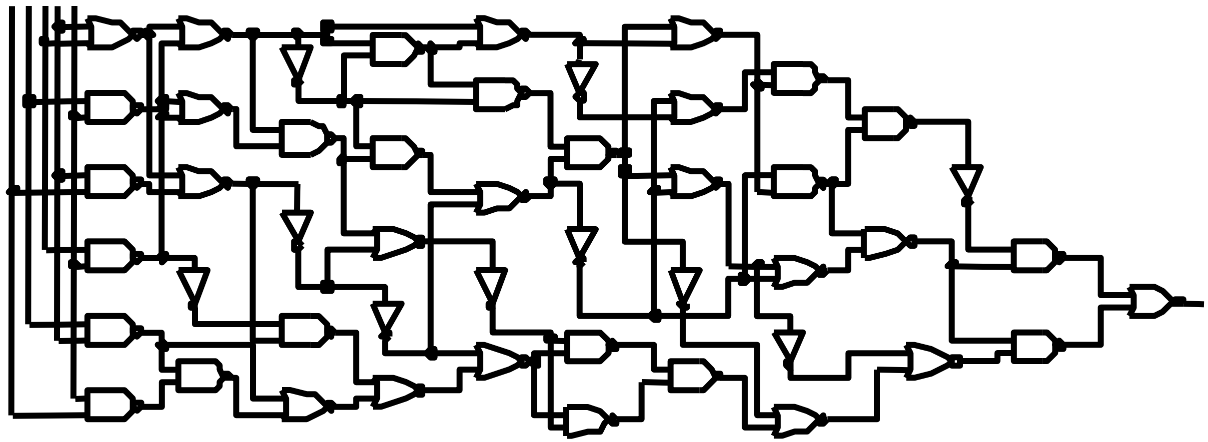
- Circuit netlist represented by hypergraph



Courtesy K. Yang, UCLA

Example: Partitioning of a Circuit

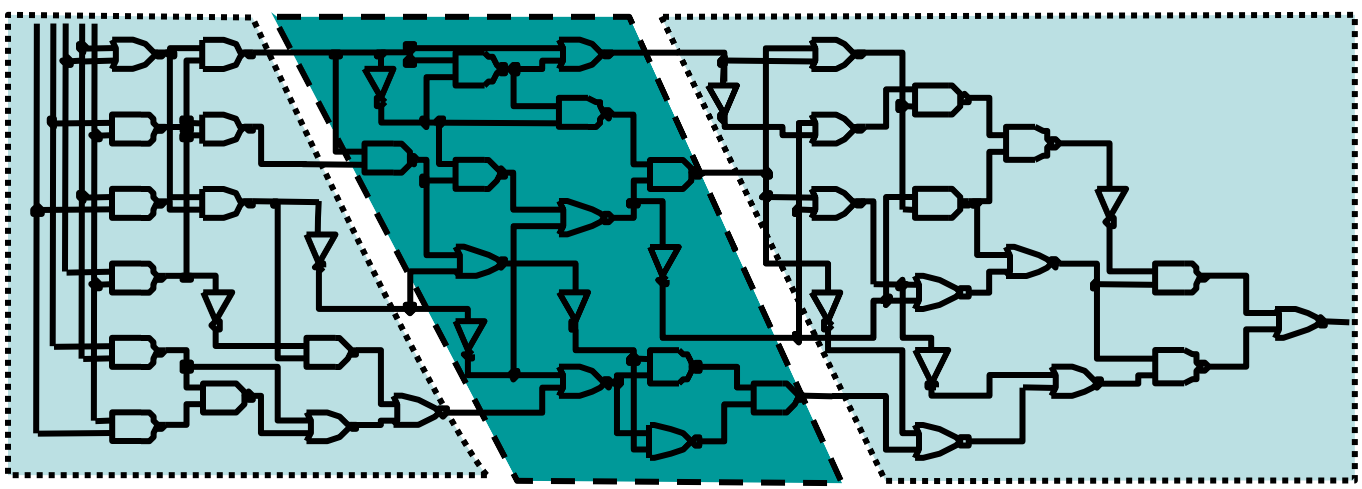
#vertices = 48



Hyperedge Cut = 4
Partition Size = 15

Hyperedge Cut = 4
Partition Size = 16

Partition Size = 17



Notice that the edge cut is different from hyperedge cut

Courtesy K. Yang, UCLA

Fiduccia-Mattheyses (FM) Approach

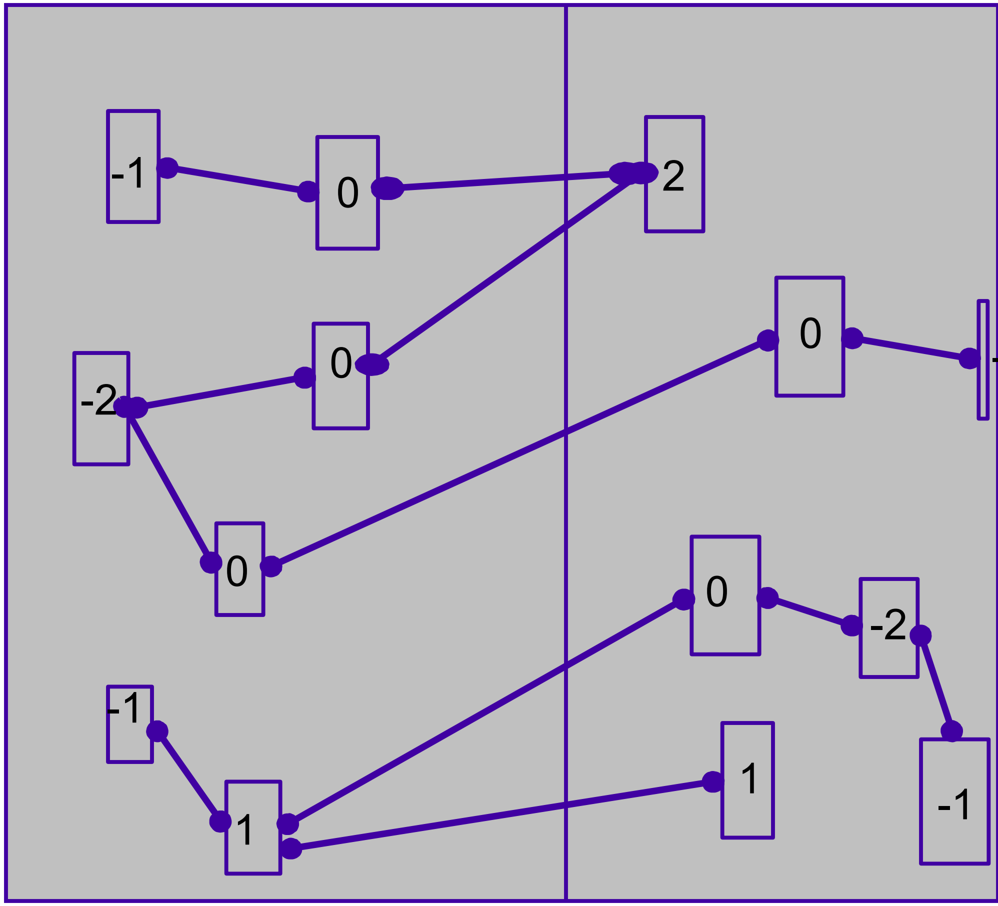
- **Pass**:
  - start with all vertices free to move (*unlocked*)
  - label each possible move with immediate change in cost that it causes (*gain*)
  - iteratively select and execute a move with highest gain, *lock* the moving vertex (i.e., cannot move again during the **pass**), and update affected gains
  - best solution seen during the pass is adopted as starting solution for **next pass**
- FM:
  - start with some initial solution
  - perform **passes** until a **pass** fails to improve solution quality
  - FM algorithm minimizes cut costs based on nets (or hyperedges)
  - A "balance" constraint for node weight sum (e.g., total partition area) can be easily enforced

FM Partitioning

Moves are made based on ***object gain***

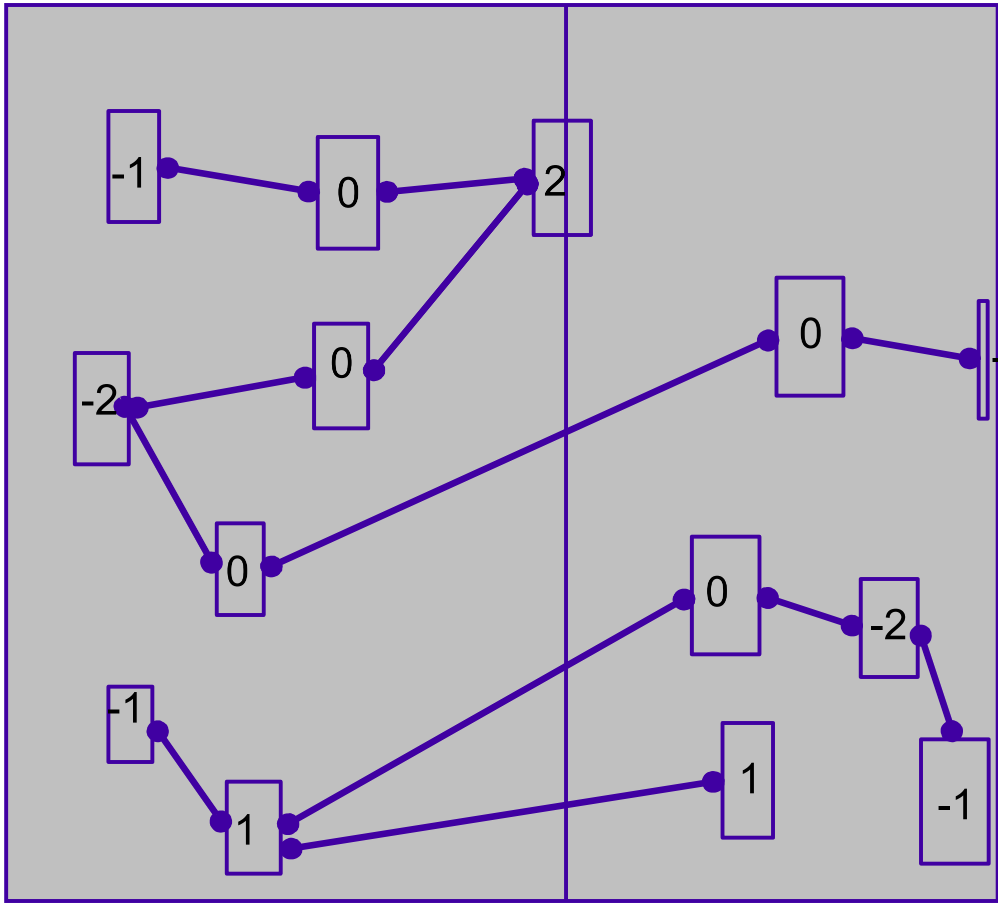
Object Gain: The amount of change in cut crossings
that will occur if an object is moved from
its current partition into the other partition

- 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

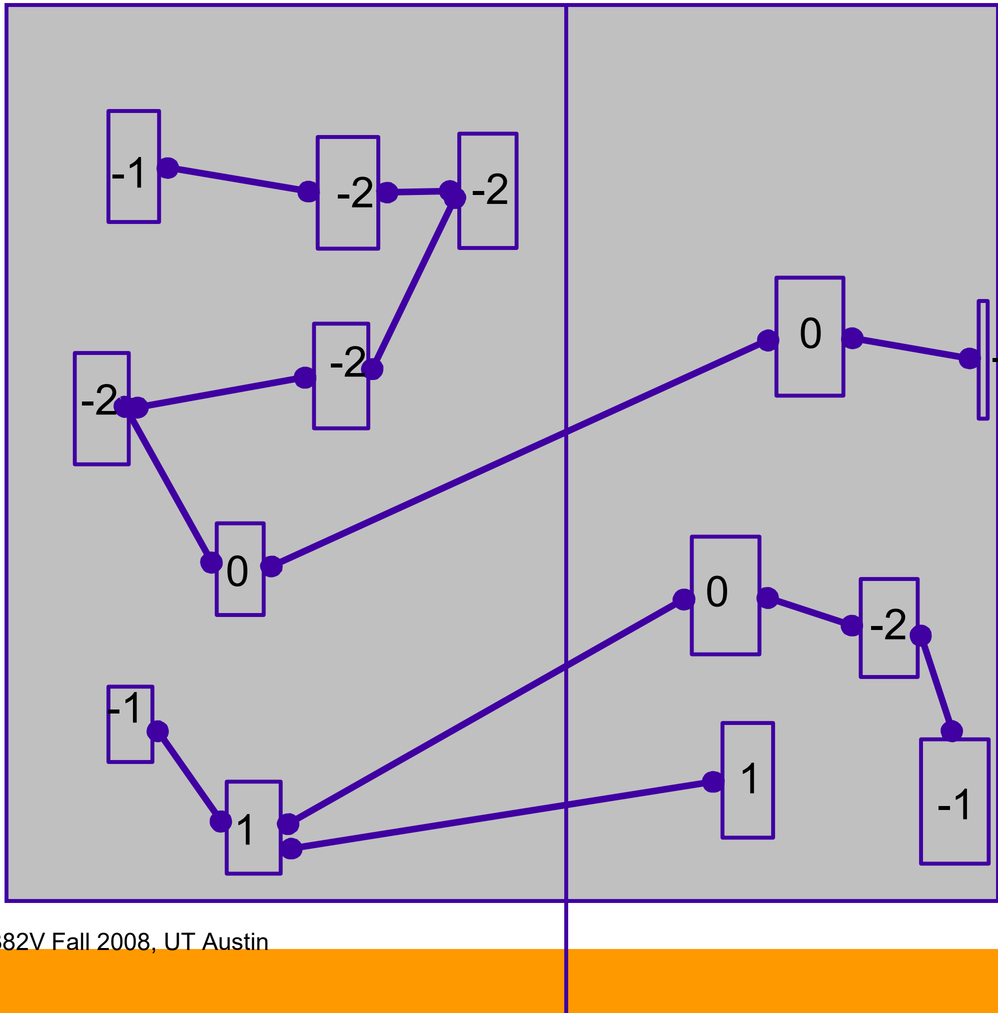


From D. Pan, EE382V Fall 2008, UT Austin

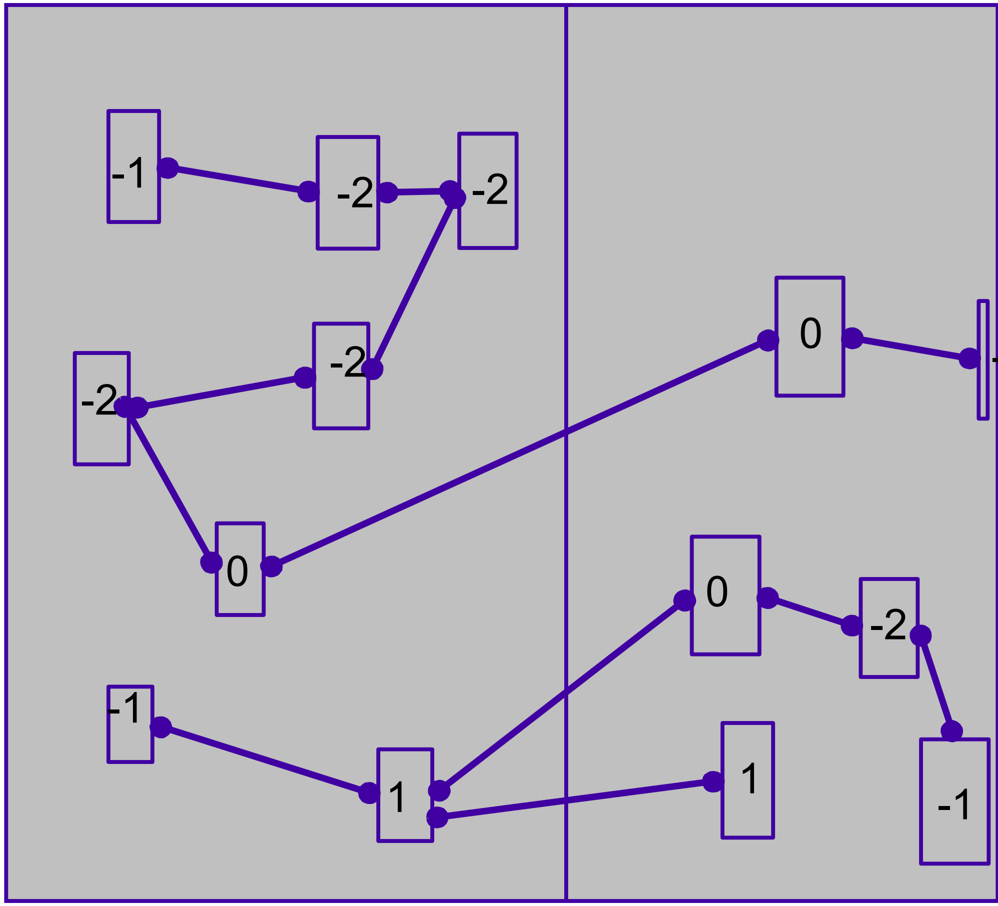
FM Partitioning



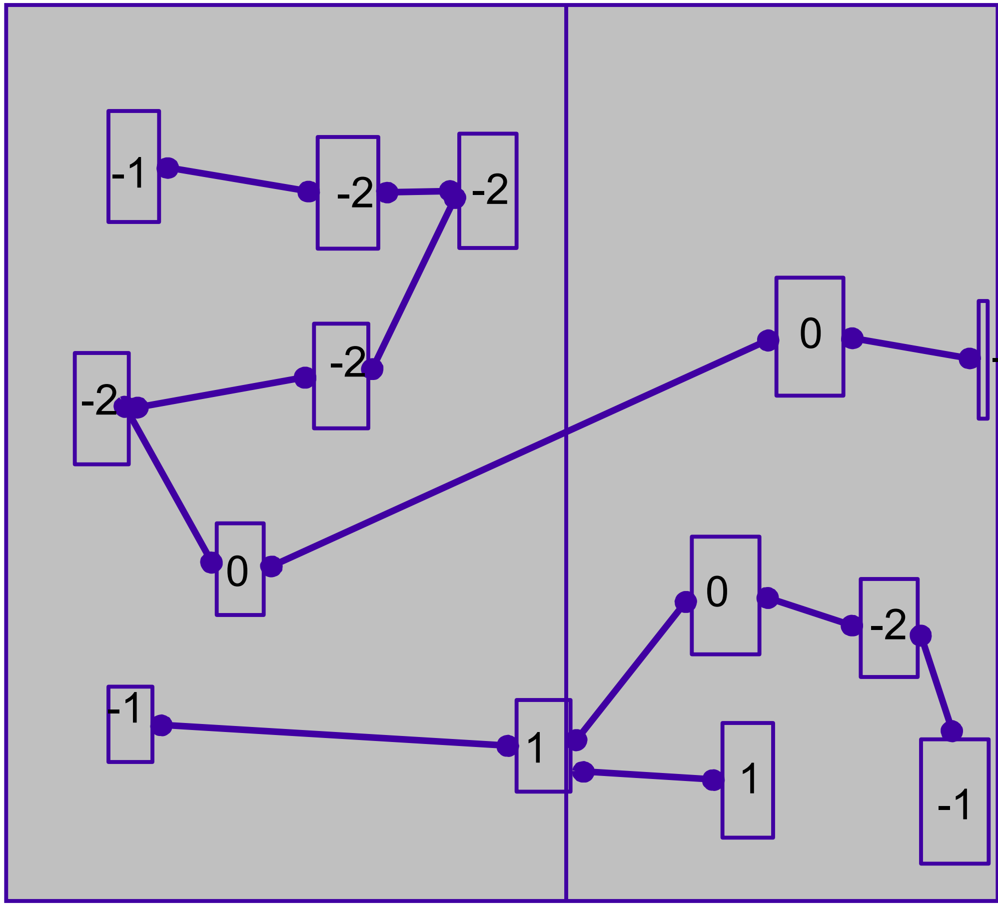
From D. Pan, EE382V Fall 2008, UT Austin



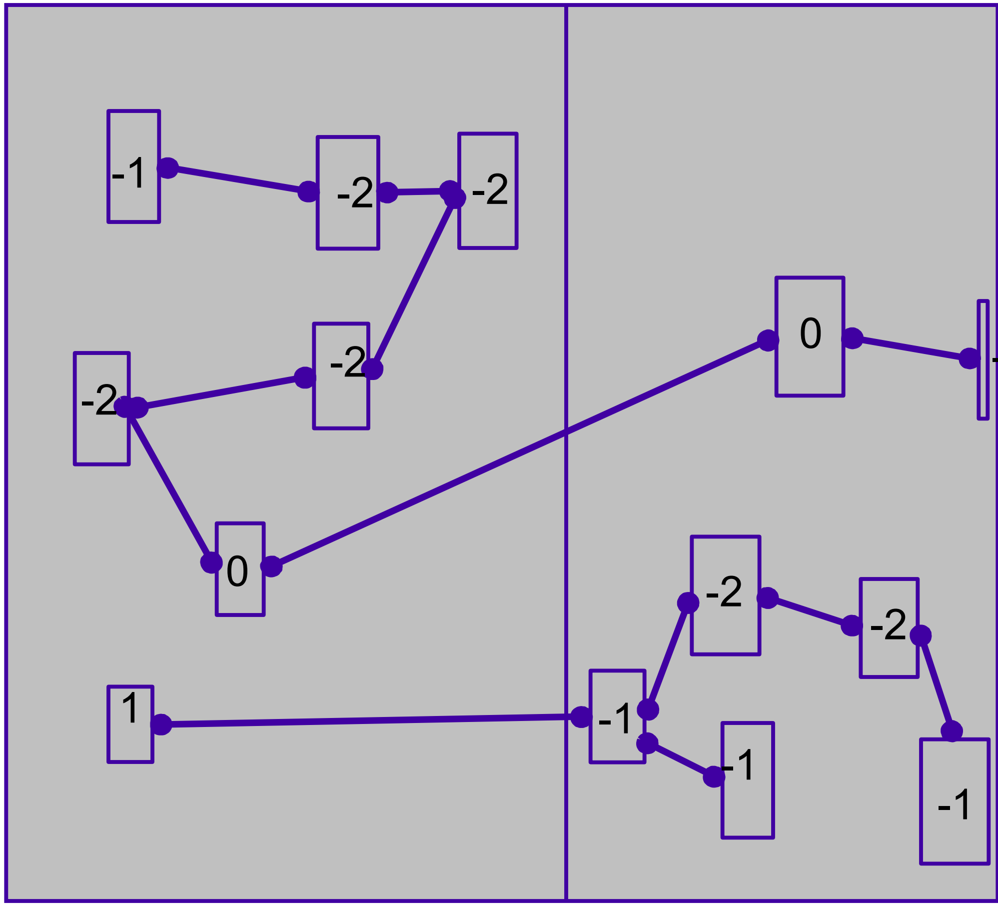
From D. Pan, EE382V Fall 2008, UT Austin



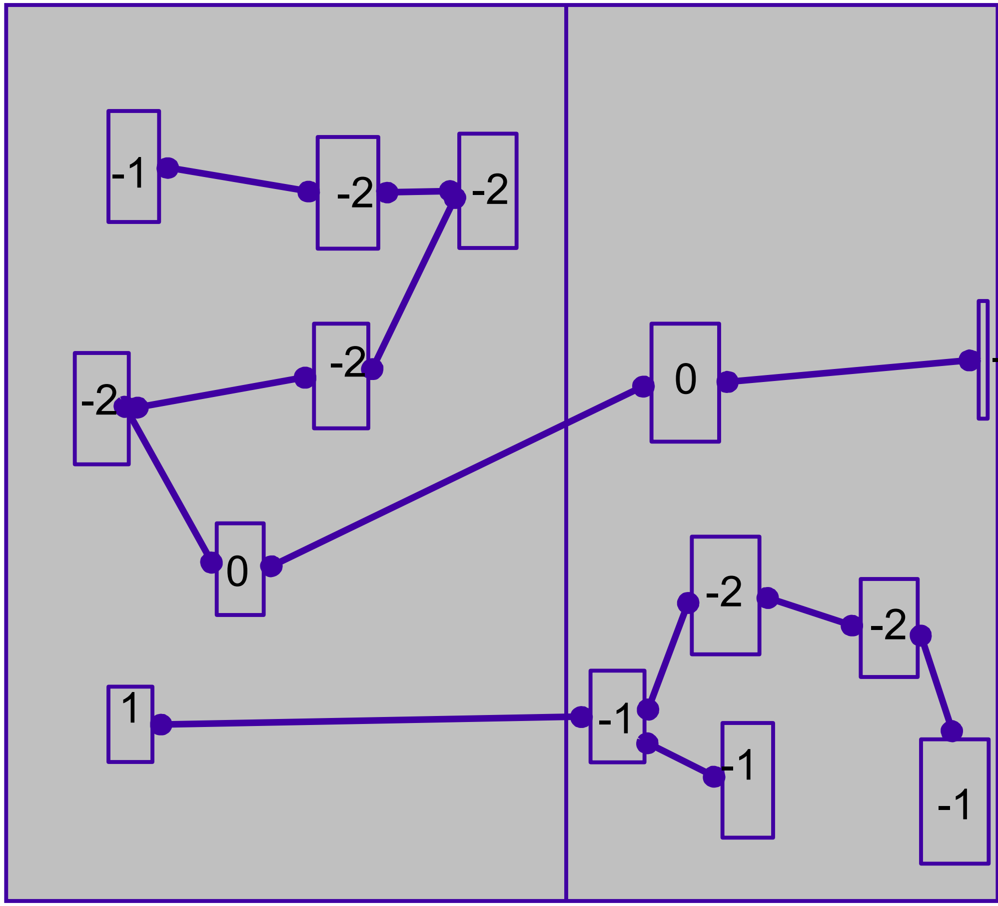
From D. Pan, EE382V Fall 2008, UT Austin



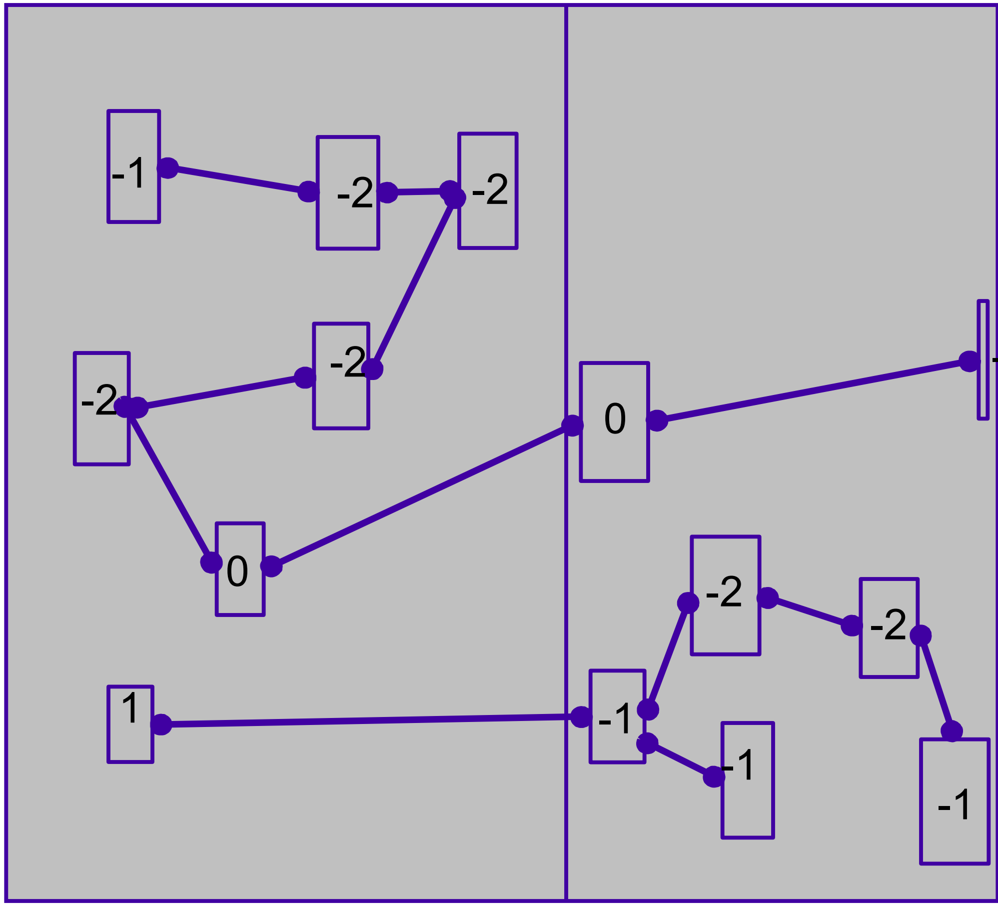
From D. Pan, EE382V Fall 2008, UT Austin



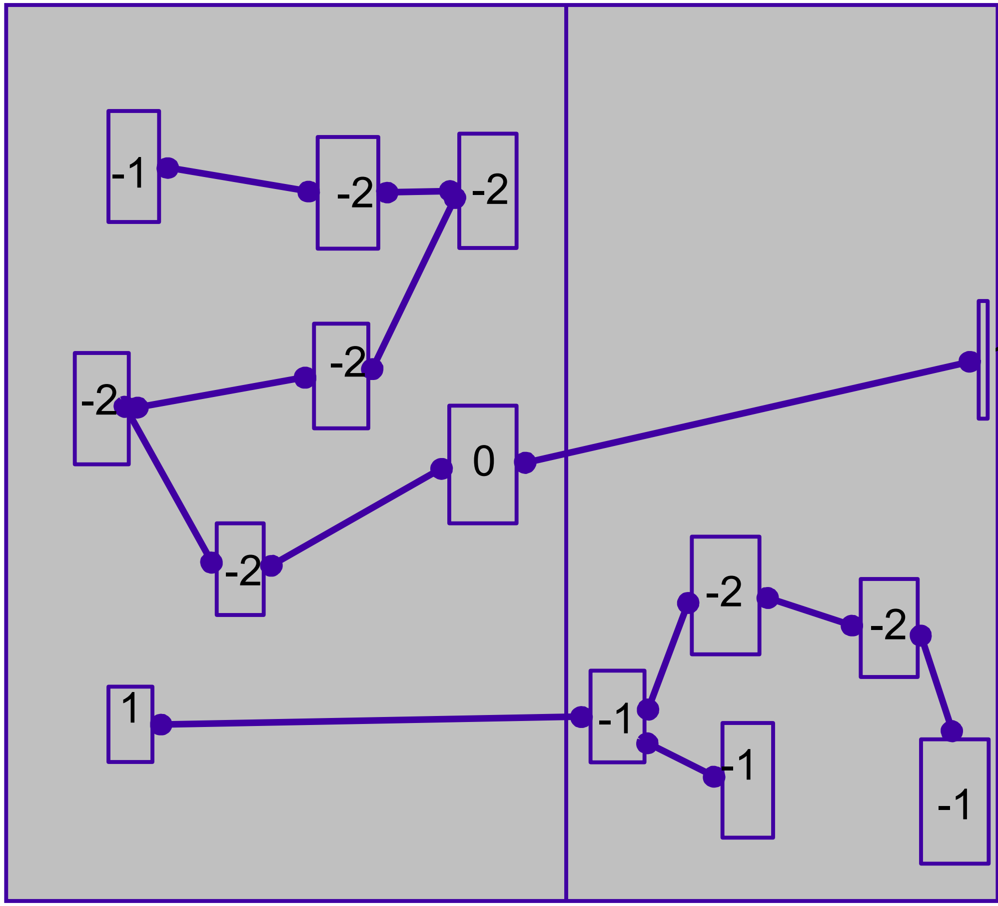
From D. Pan, EE382V Fall 2008, UT Austin



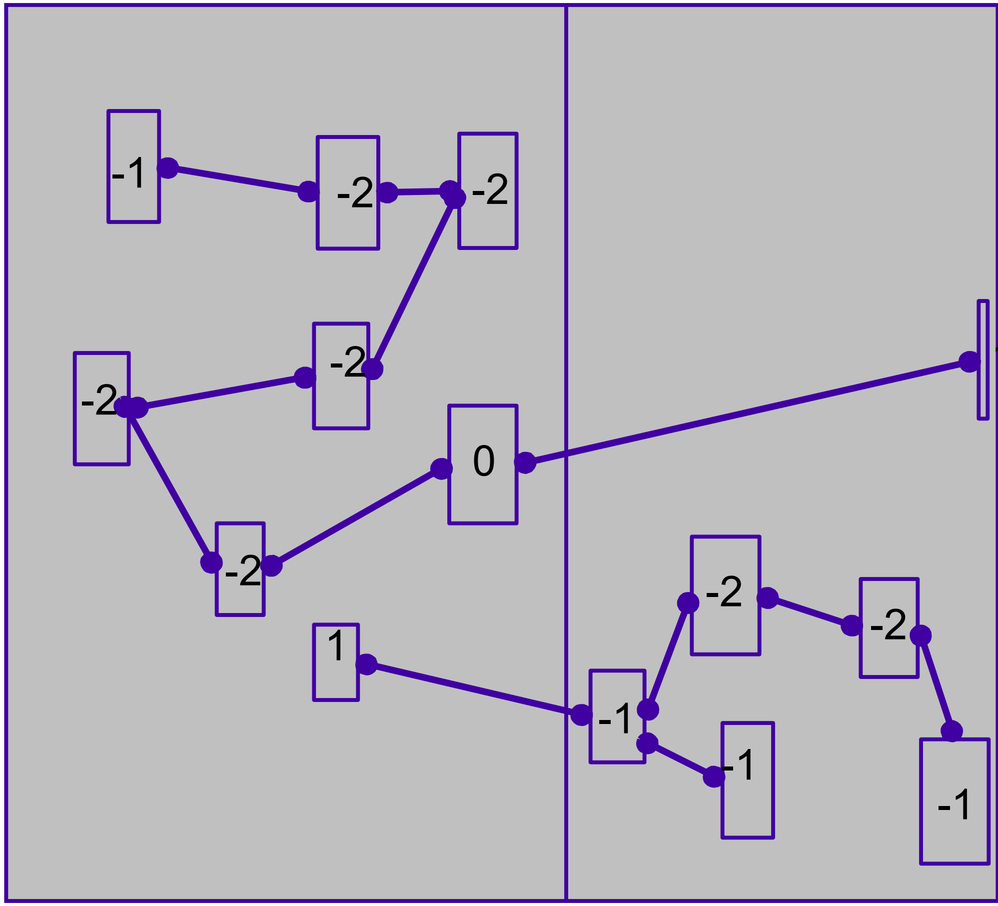
From D. Pan, EE382V Fall 2008, UT Austin



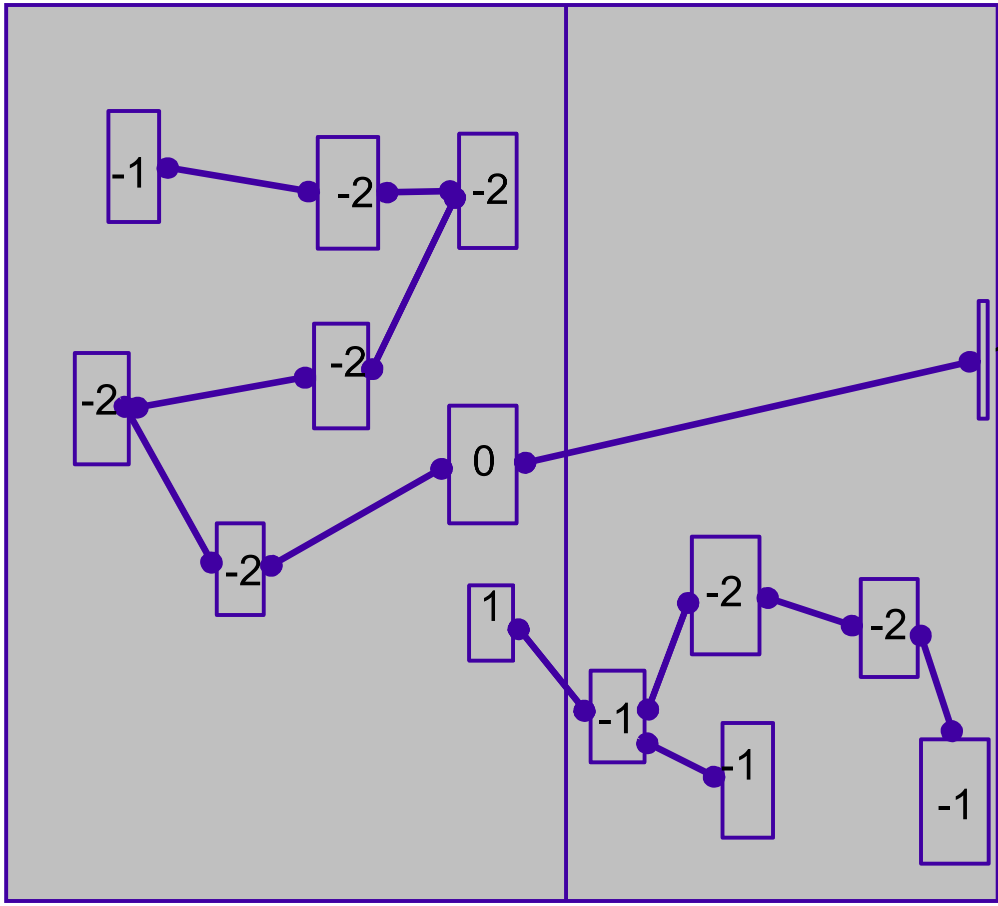
From D. Pan, EE382V Fall 2008, UT Austin



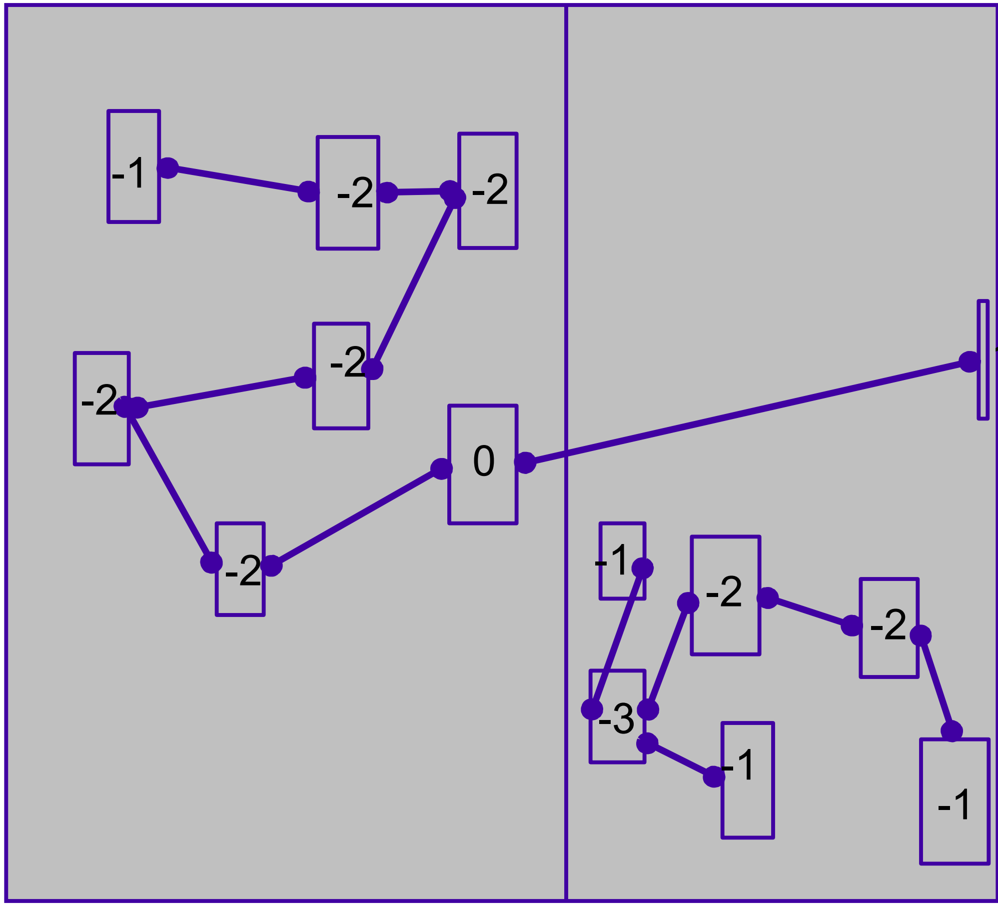
From D. Pan, EE382V Fall 2008, UT Austin



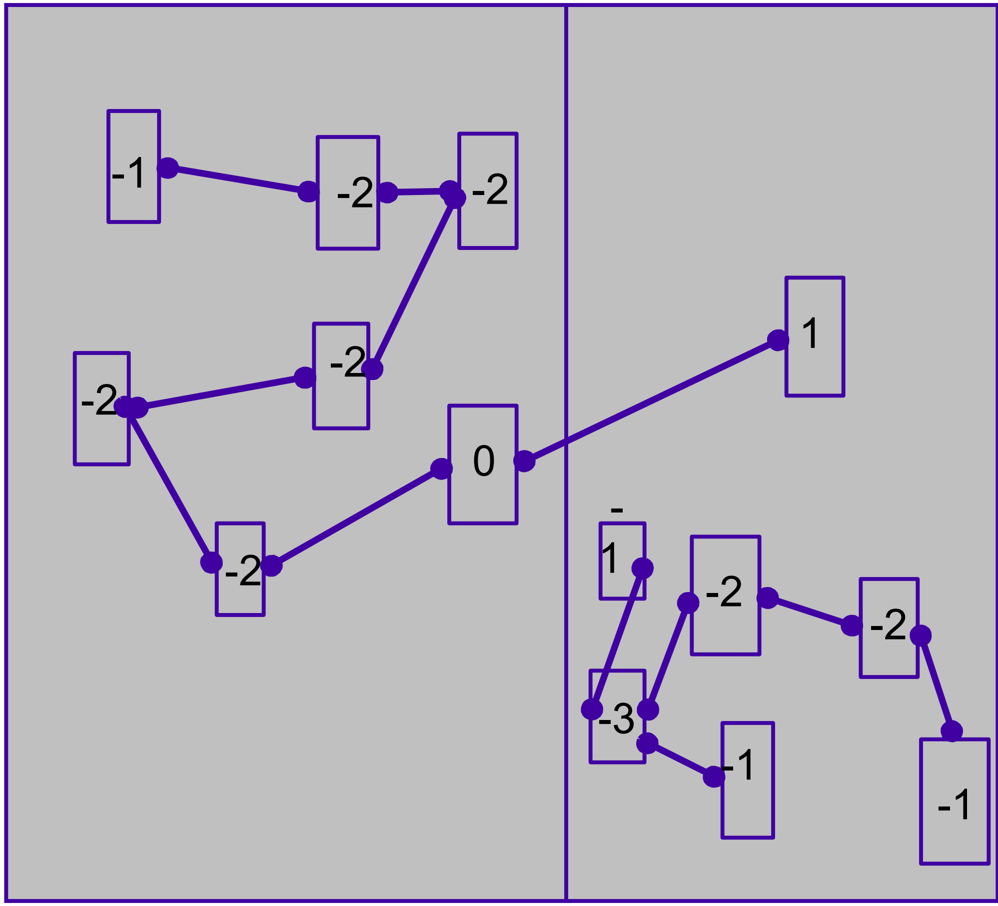
From D. Pan, EE382V Fall 2008, UT Austin



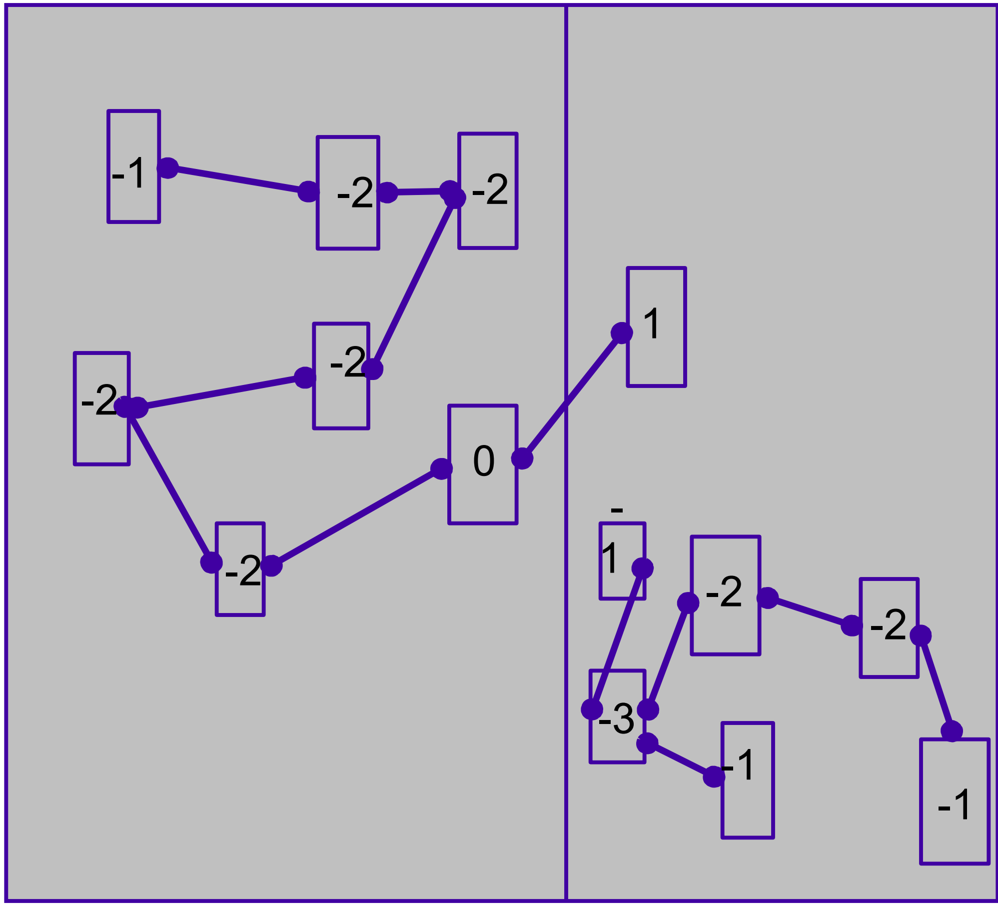
From D. Pan, EE382V Fall 2008, UT Austin



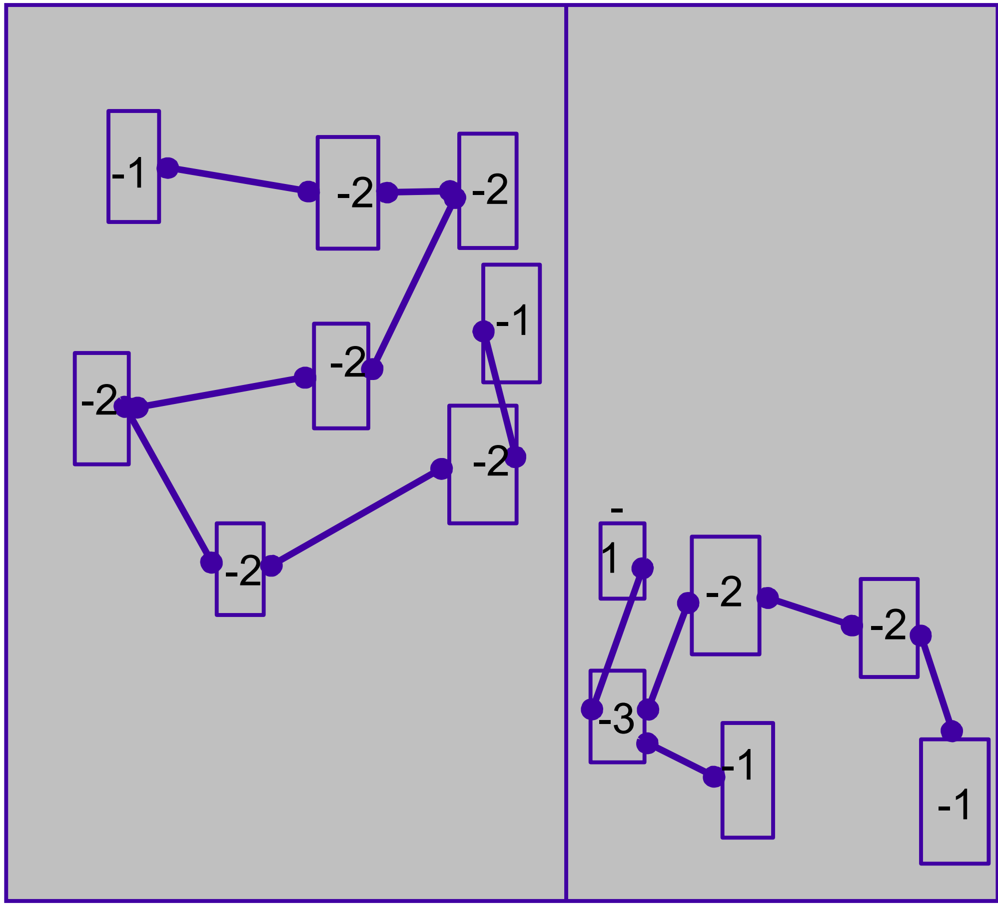
From D. Pan, EE382V Fall 2008, UT Austin



From D. Pan, EE382V Fall 2008, UT Austin

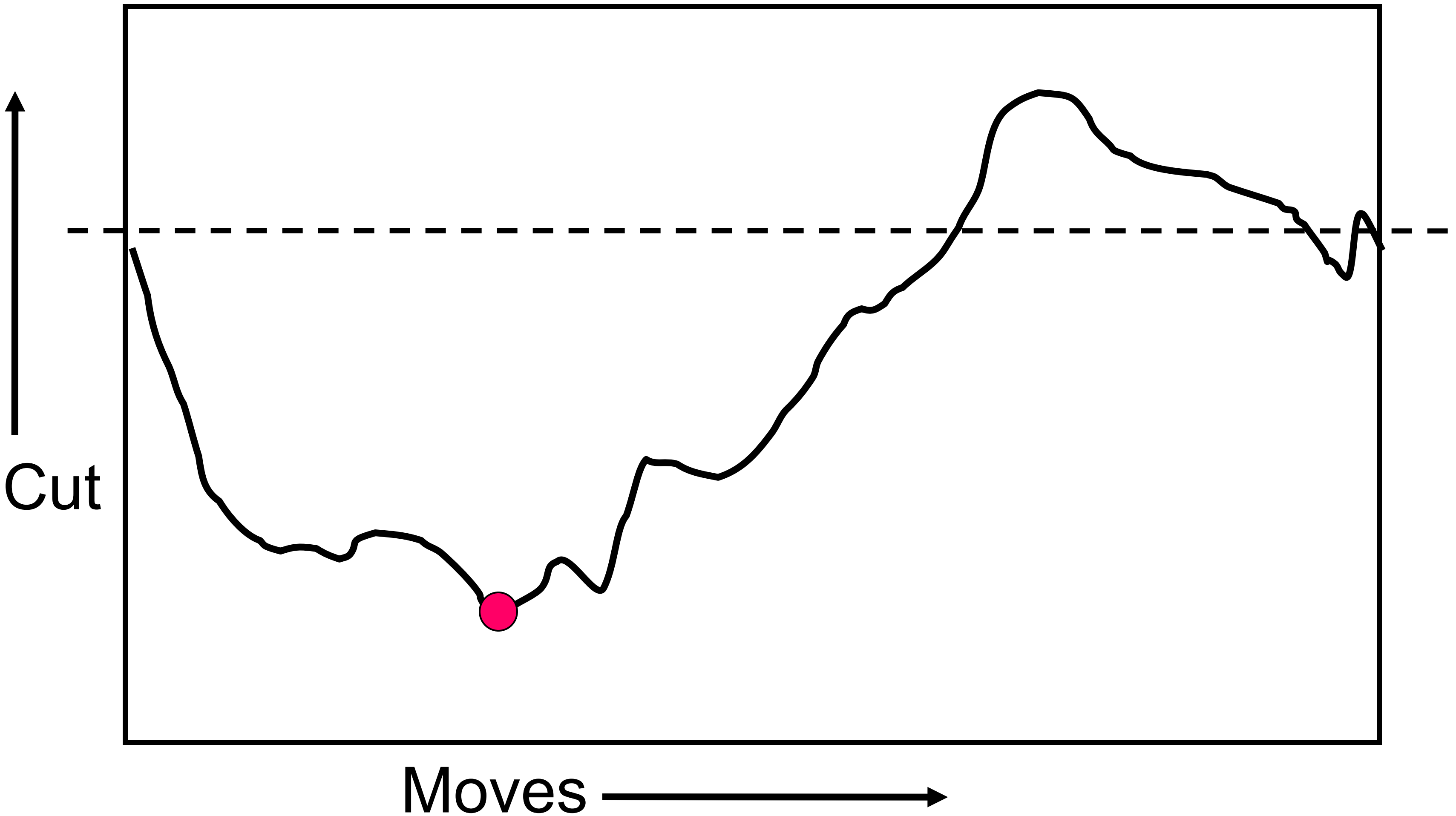


From D. Pan, EE382V Fall 2008, UT Austin

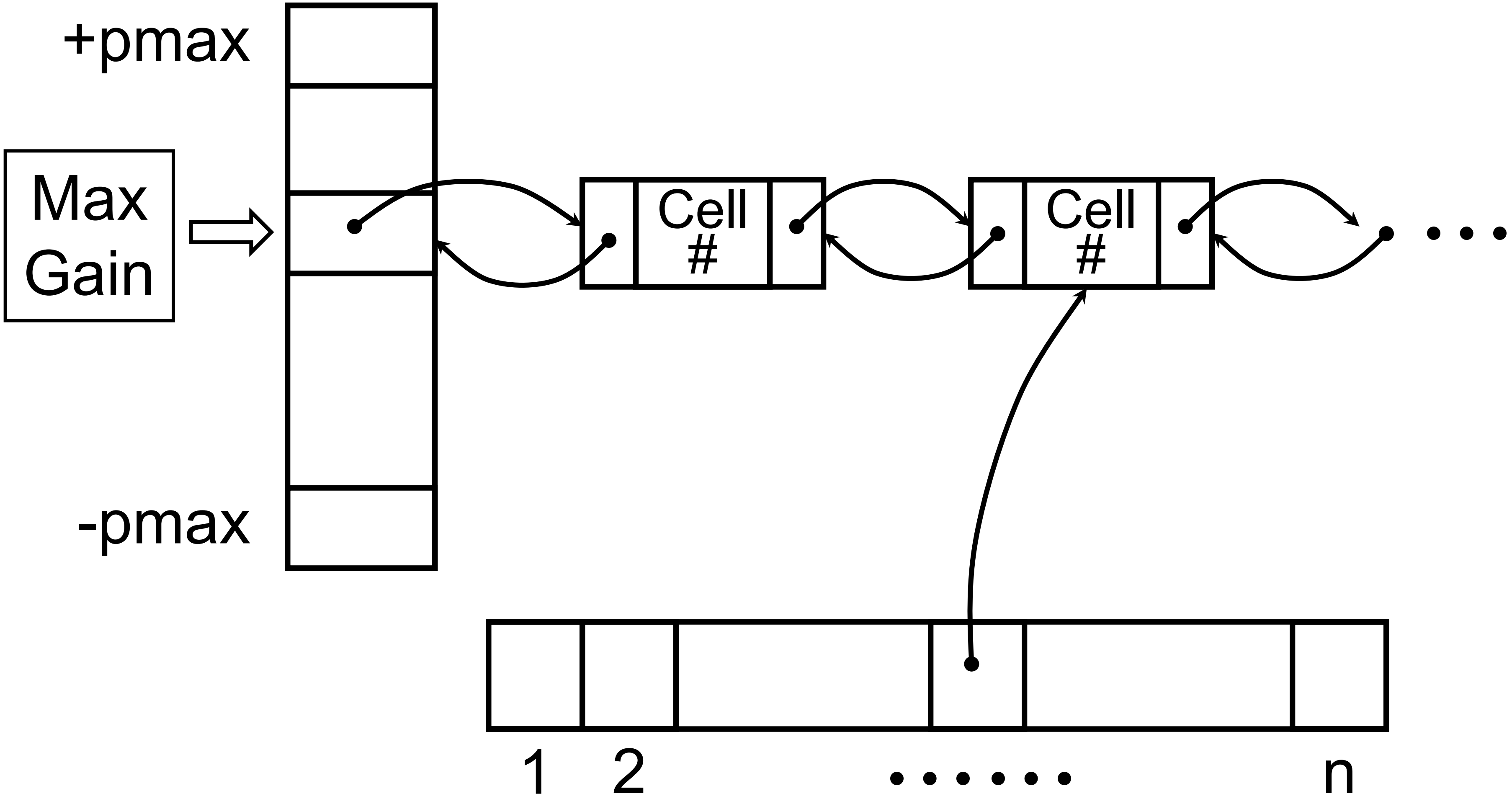


From D. Pan, EE382V Fall 2008, UT Austin

Cut During One Pass (Bipartitioning)



Gain Bucket Data Structure



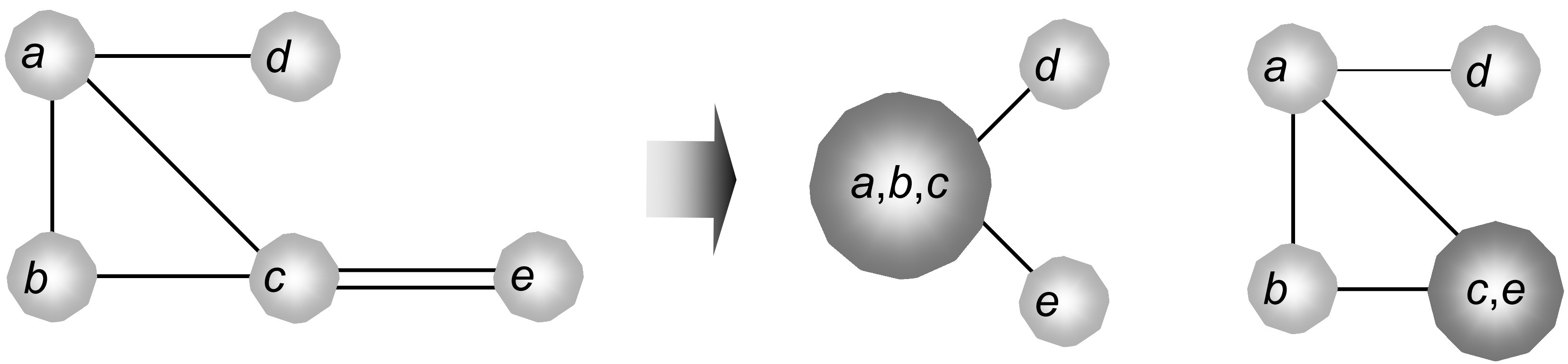
From D. Pan, EE382V Fall 2008, UT Austin

Time Complexity of FM

- For each pass:
  - Constant time to find the best vertex to move (gain bucket data structure!)
  - 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
- In practice
  - Force #passes = 2 or less
  - Cut off the pass very early (after only 5% finished)
  - Together, these two heuristic modifications result in ~50X speedup!

Clustering/Coarsening

- To make things easy, groups of tightly-connected nodes can be clustered, absorbing connections between these nodes



Initital graph

Possible clustering hierarchies of the graph

Multilevel Partitioning

(N+,N-) = Multilevel_Partition(N, E)

… recursive partitioning routine returns N+ and N- where N = N+ U N-

if |N| is small

(1) Partition G = (N,E) directly to get N = N+ U N-

Return (N+, N-)

else

(2) Coarsen G to get an approximation $G_C = (N_C, E_C)$

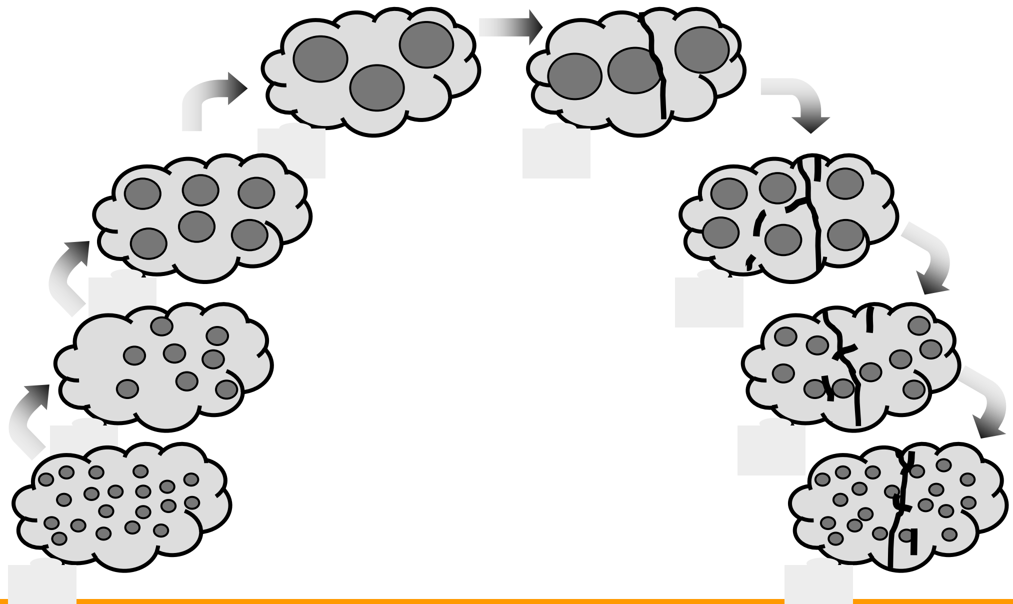
(3) (N_C+ , N_C-) = Multilevel_Partition(N_C, E_C)

(4) Expand (N_C+ , N_C-) to a partition (N+ , N-) of N

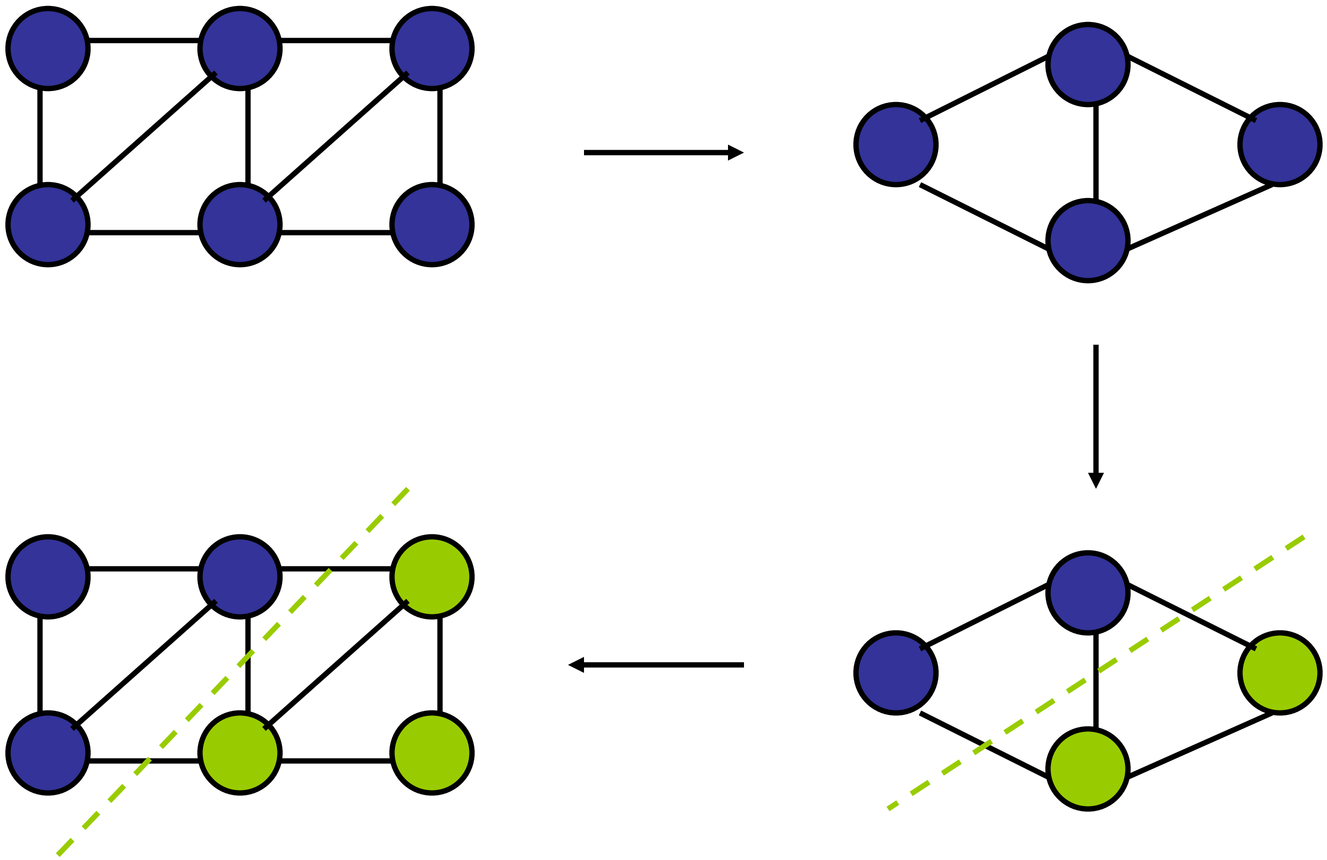
(5) Improve the partition (N+ , N-)

Return (N+ , N-)

endif

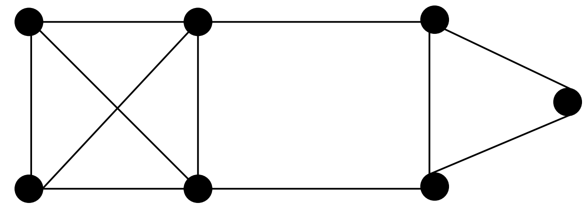


An Example

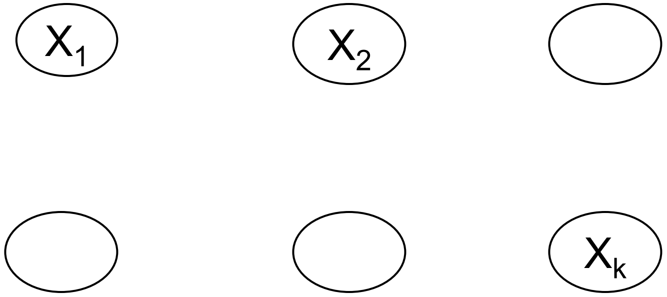


Variants of Partitioning

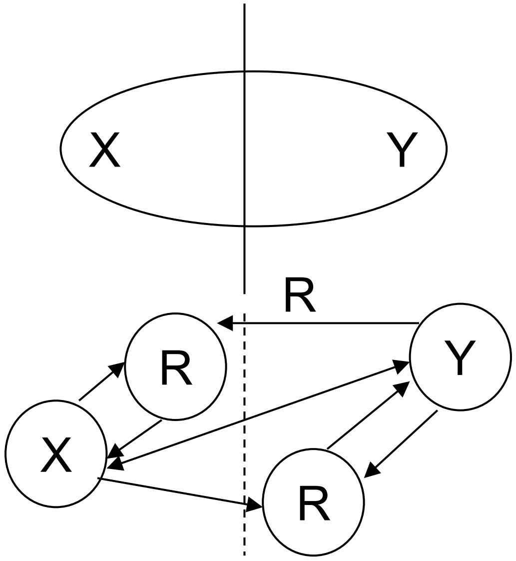
- Ratio Cut: $Minimize \dfrac{C(X,\bar{X})}{|X||\bar{X}|}$



- Multi-Way Partitioning

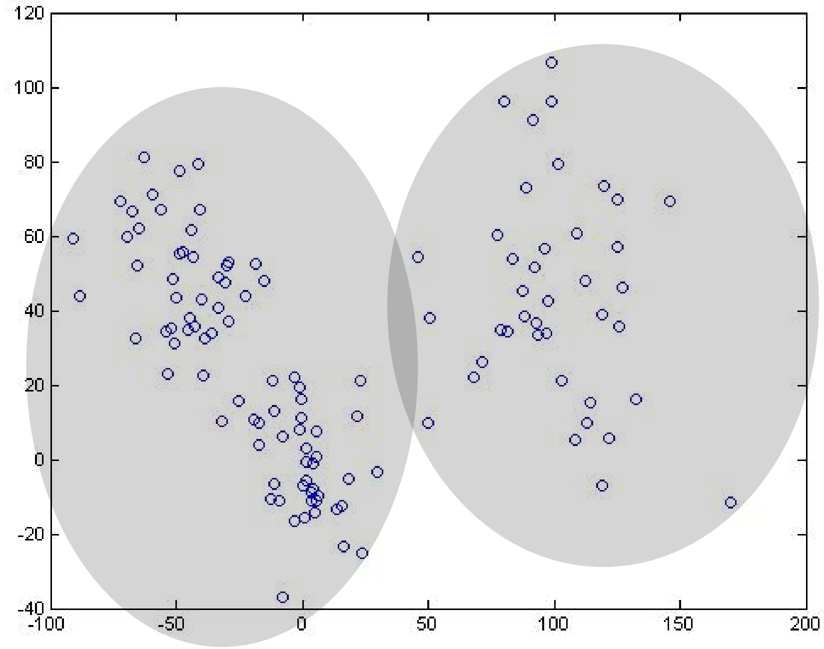


- Replication cut partitioning



Related Problem: Clustering

- A way of grouping together data samples that are ***similar*** in some way - according to some criteria that you pick



K-means Clustering

- Choose a number of clusters *k*
- Initialize cluster centers $\mu_1,\ldots\ \mu_k$
  - Could pick *k* data points and set cluster centers to these points
  - Or could randomly assign points to clusters and take means of clusters
- For each data point, compute the cluster center it is closest to (using some distance measure) and assign the data point to this cluster
- Re-compute cluster centers (mean of data points in cluster)
- Stop when there are no new re-assignments

K-means Clustering Issues

- Random initialization means that you may get different clusters each time
  - Common approach: pick best of few random initializations
- You have to pick the number of clusters…