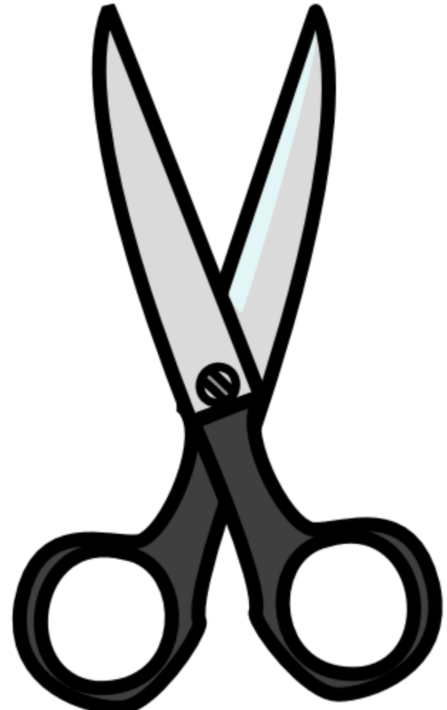
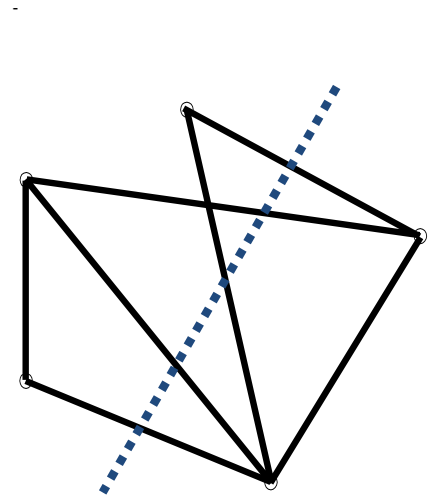
Partitioning

CPEN 513

2019/2020 Term 2





Partitioning:

Often, circuits are bigger than a single chip
-> Need to divide the circuit among chips

Also, may need to divide a circuit among two portions of a chip

Two approaches:

1. Multi-way Partitioning
2. Recursive Bi-partitioning

Constructive Algorithms:

Back in Grade School Gym class, how did you divide into two teams?

Constructive Algorithms:

```
Choose two seeds, put one in partition A and one in B
While there are still nodes left {
    Pick node "most tightly" connected to all nodes
            already in A and put it in partition A
    Pick node "most tightly" connected to all nodes
            already in B and put it in partition B
}
```

Constructive Algorithms:

Problems with this approach:

- Very sensitive to choice of seeds

- For the first few nodes we are adding in each partition, we have very little information about what will end up in each partition at the end

We would like to make "bad moves" sometimes

Simulated Annealing:

Cost Function could be $|C|$ where C is the set of cut edges (in other words, we are trying to minimize the number of cut edges).

Problem: This will create very unbalanced partitions

It has been shown that this works well:

$$|C| + \lambda B$$

where B somehow quantifies inbalance

Kernighan-Lin / Fiduccia-Matheyses:

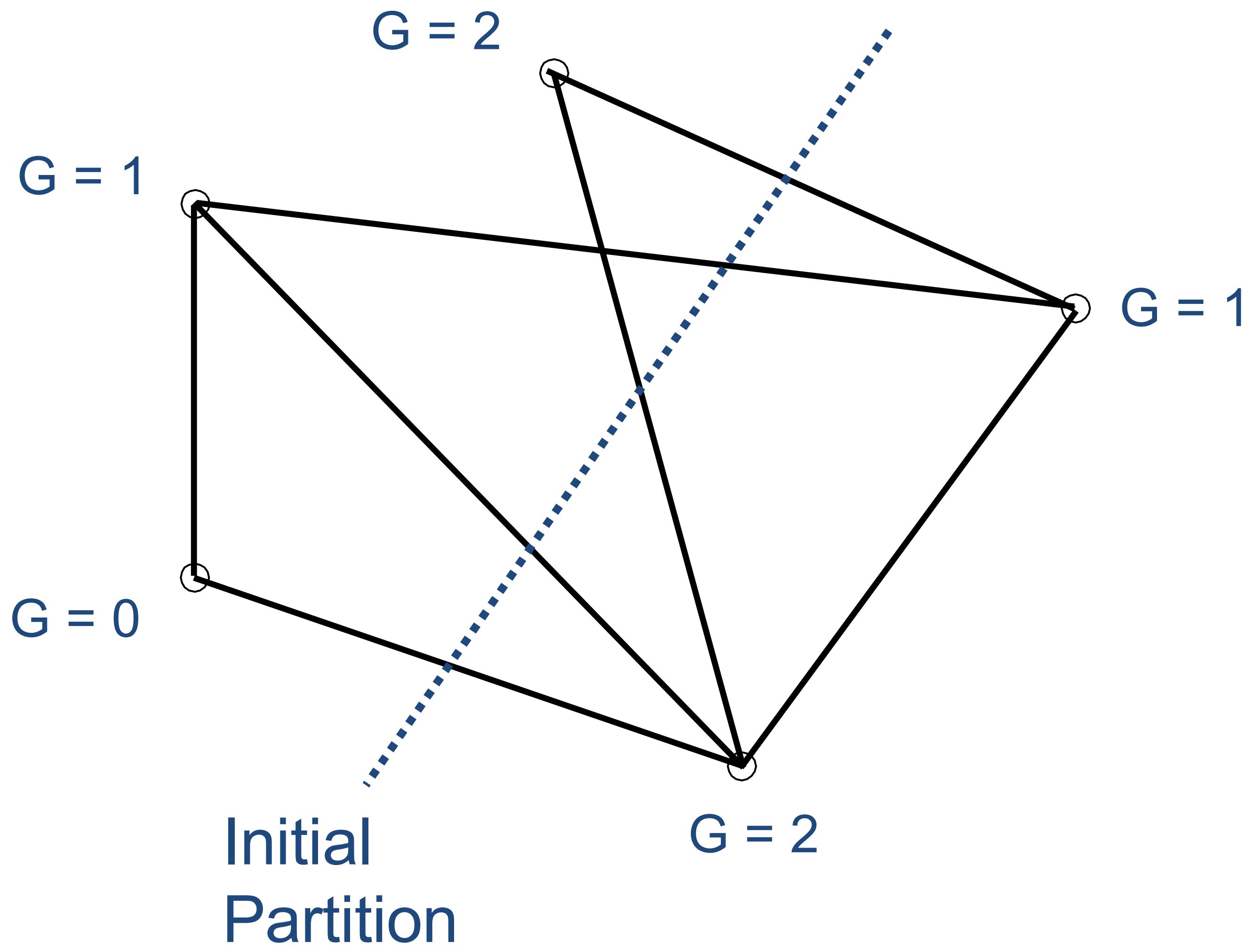
Main difference between this and SA:
Moves are not chosen randomly.

Definition: Gain(v)

- Given a partition, and a node v, Gain(v) is defined as the improvement in cut-set size if v is swapped to the opposite partition
- Equivalently,

Gain(v) = # incident edges that cross partition -
incident edges that do not

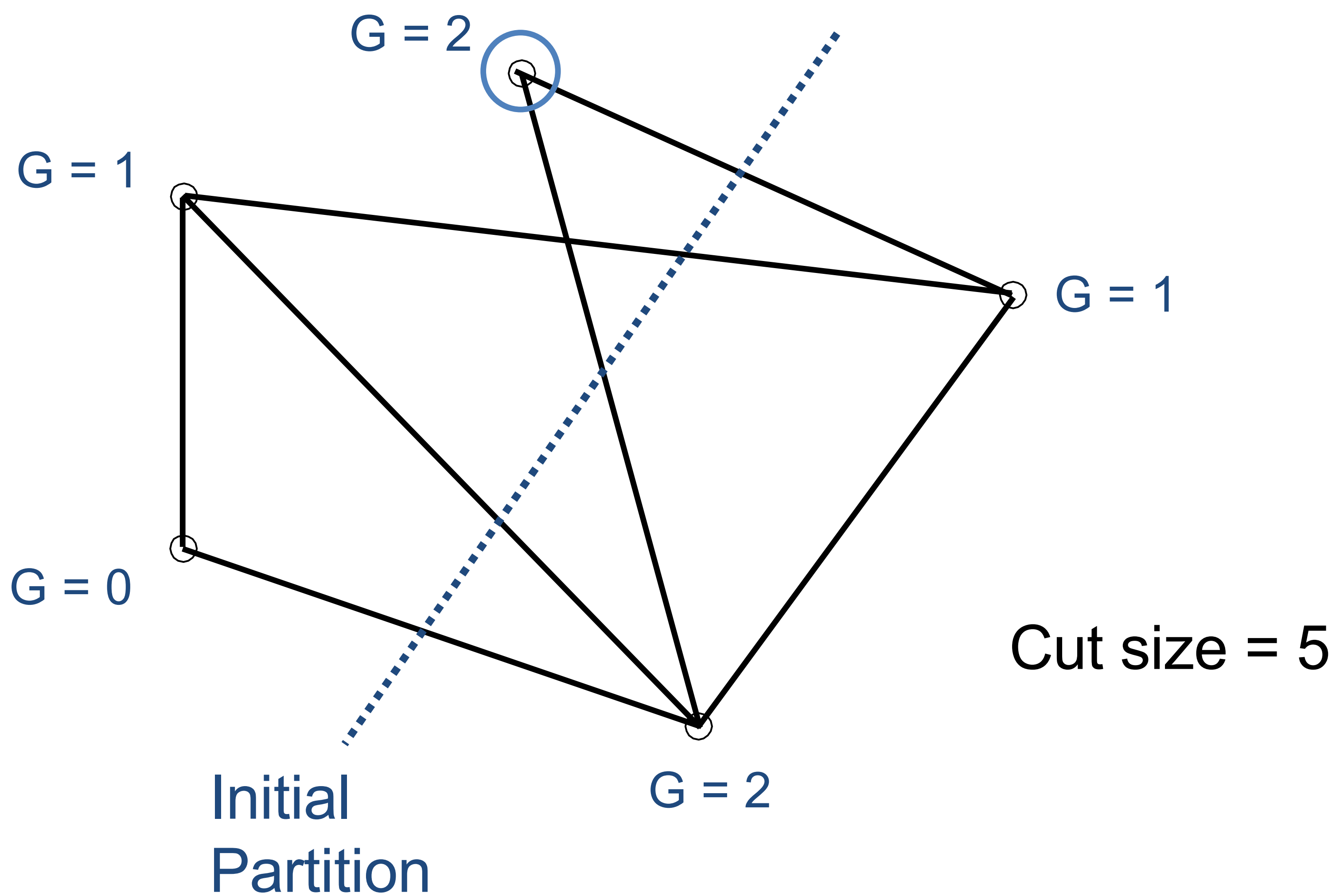
Kernighan-Lin / Fiduccia-Matheyses:



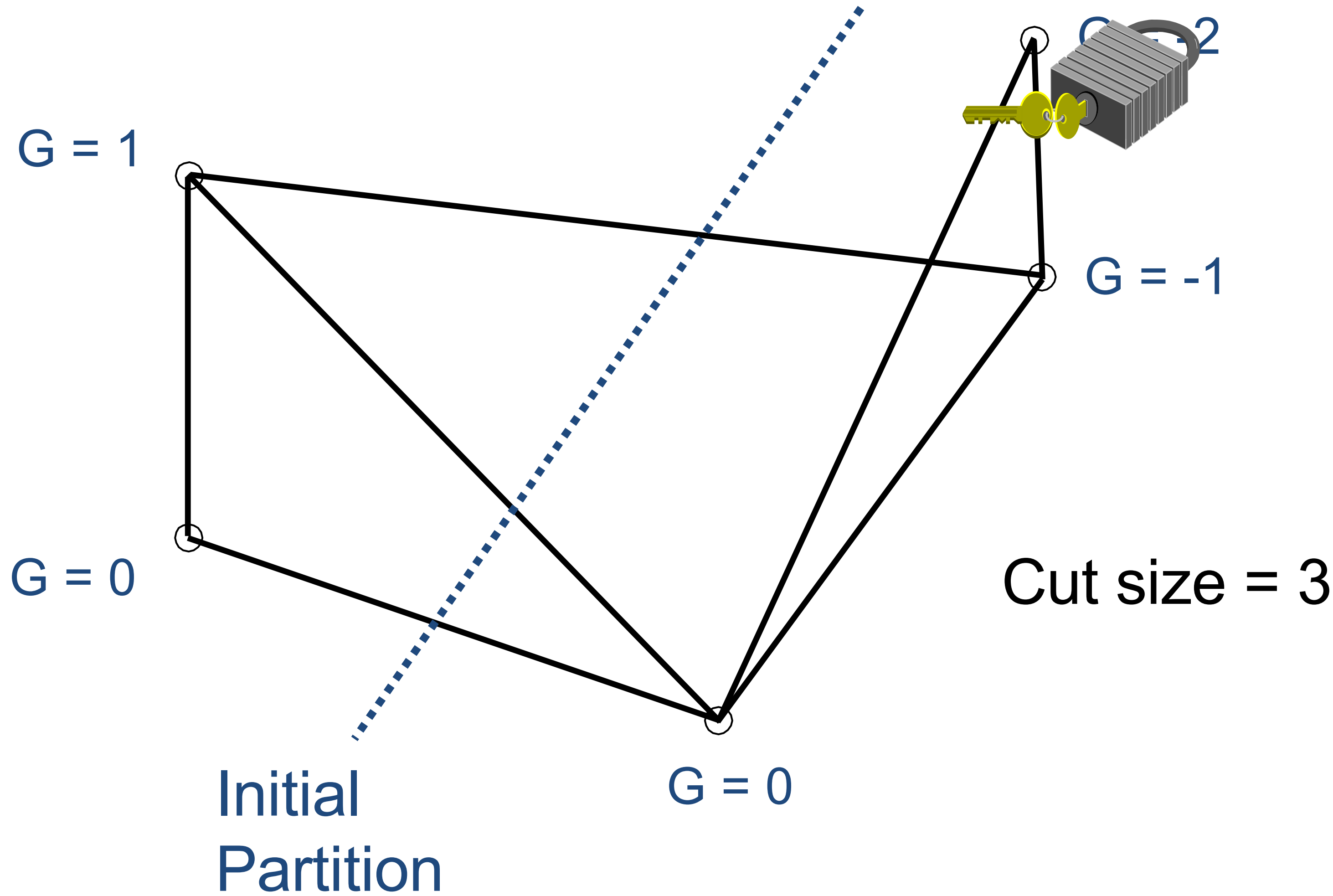
Kernighan-Lin / Fiduccia-Matheyses:

```
for a small number of passes {
        unlock all nodes
        while some nodes are unlocked {
                calculate all gains
                choose node with highest gain whose
                        movement would not cause an
                        imbalance
                move node to other block and lock it
        }
        choose best cut seen in this pass
}
```

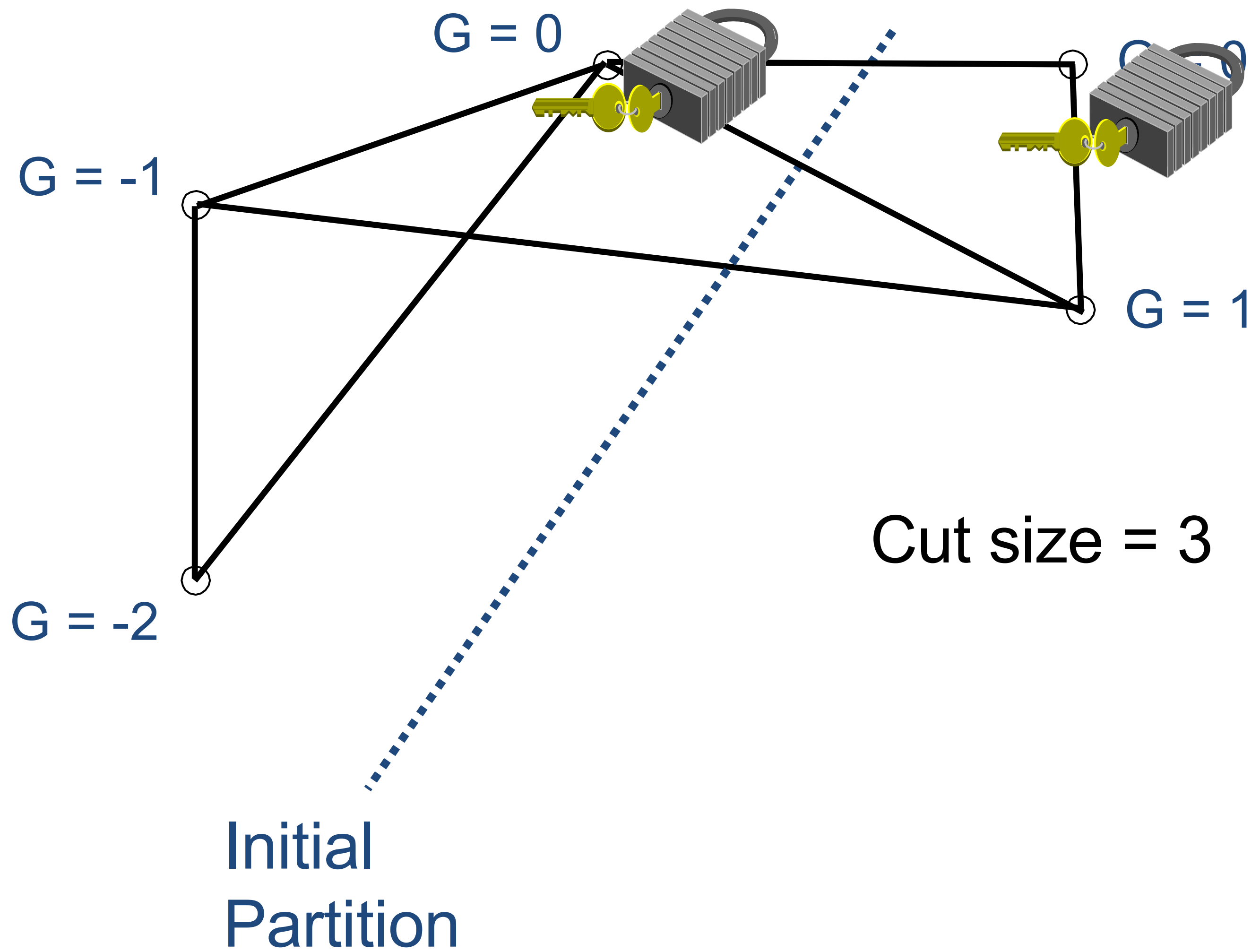
Kernighan-Lin / Fiduccia-Matheyses:



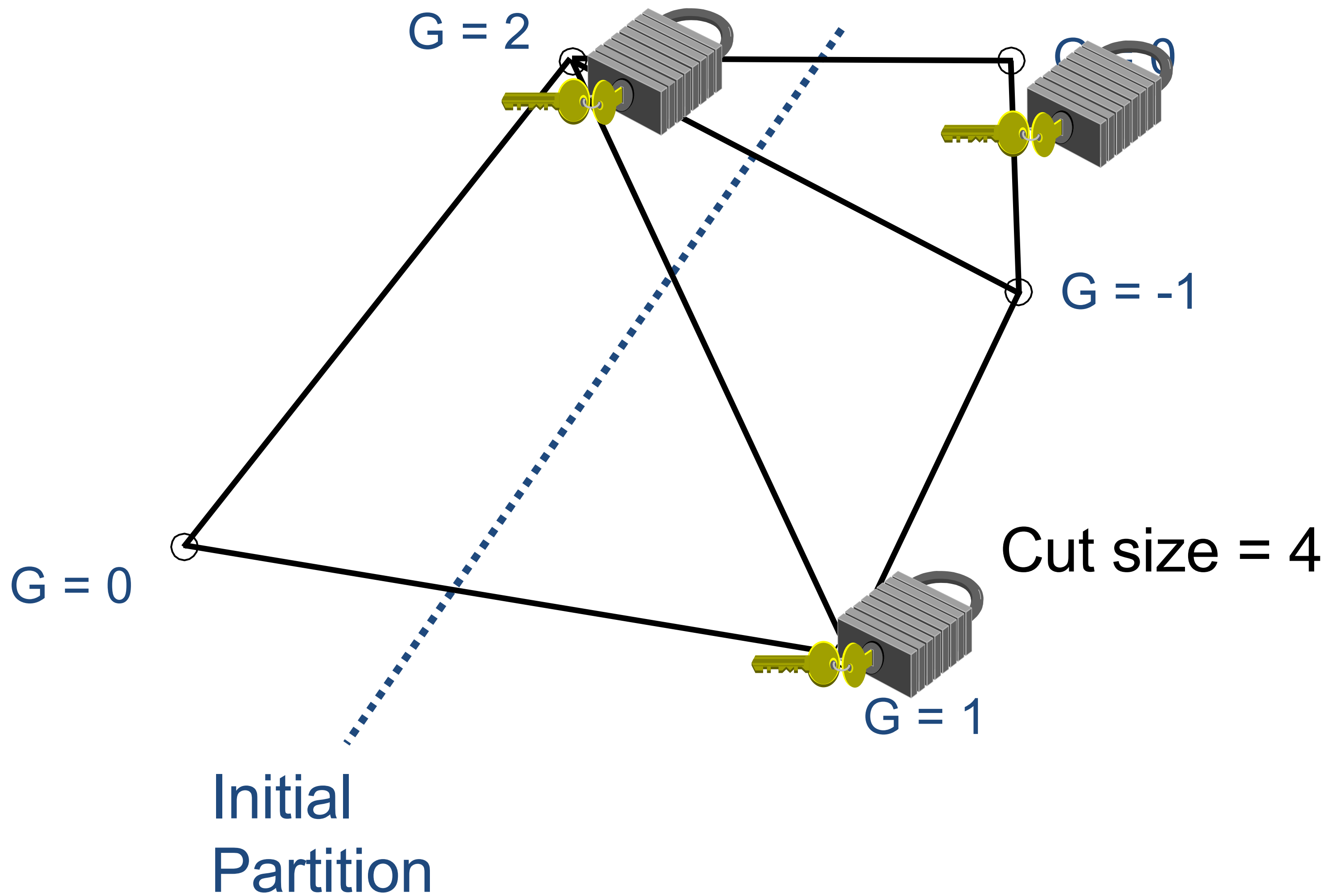
Kernighan-Lin / Fiduccia-Matheyses:



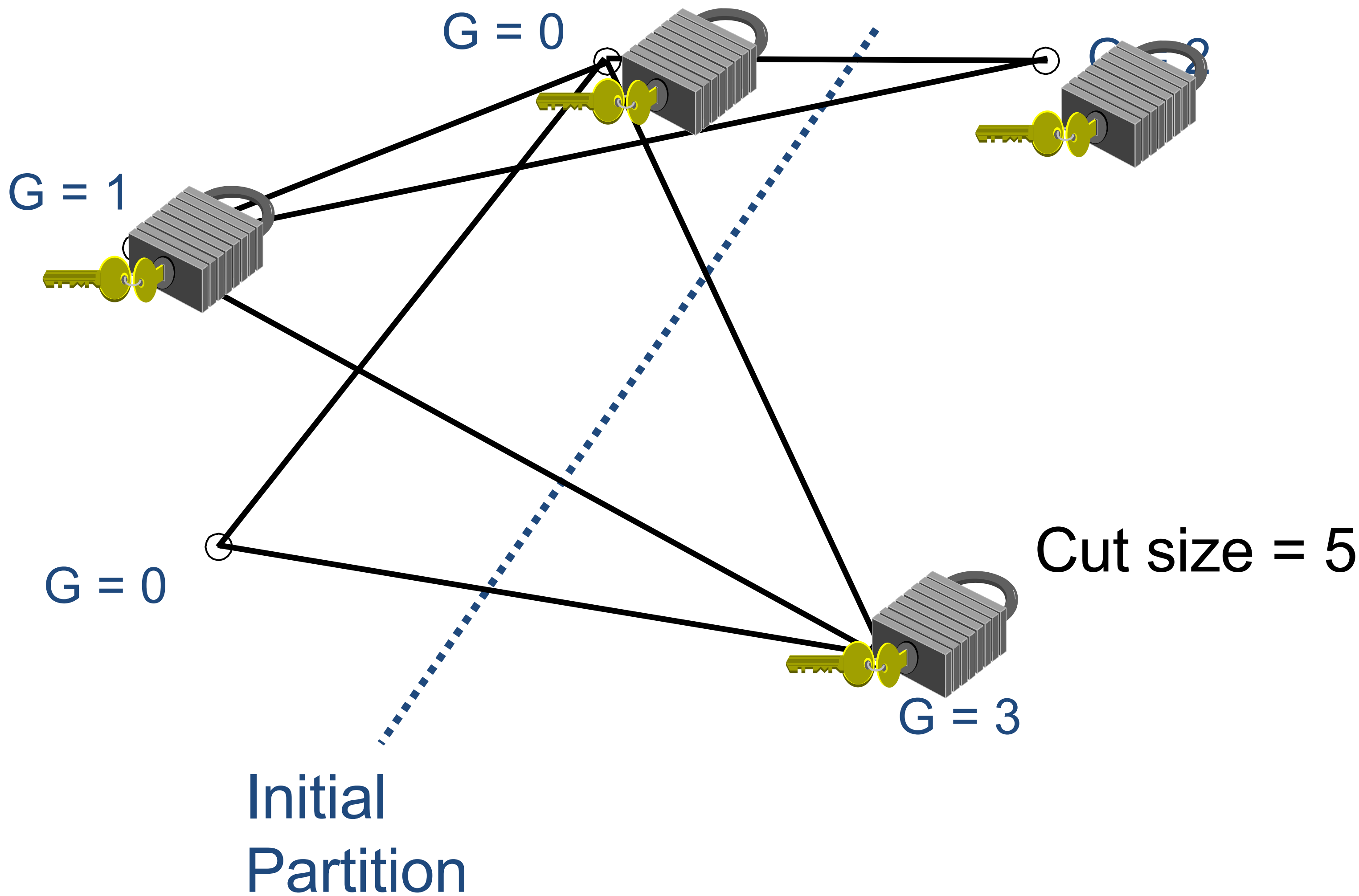
Kernighan-Lin / Fiduccia-Matheyses:



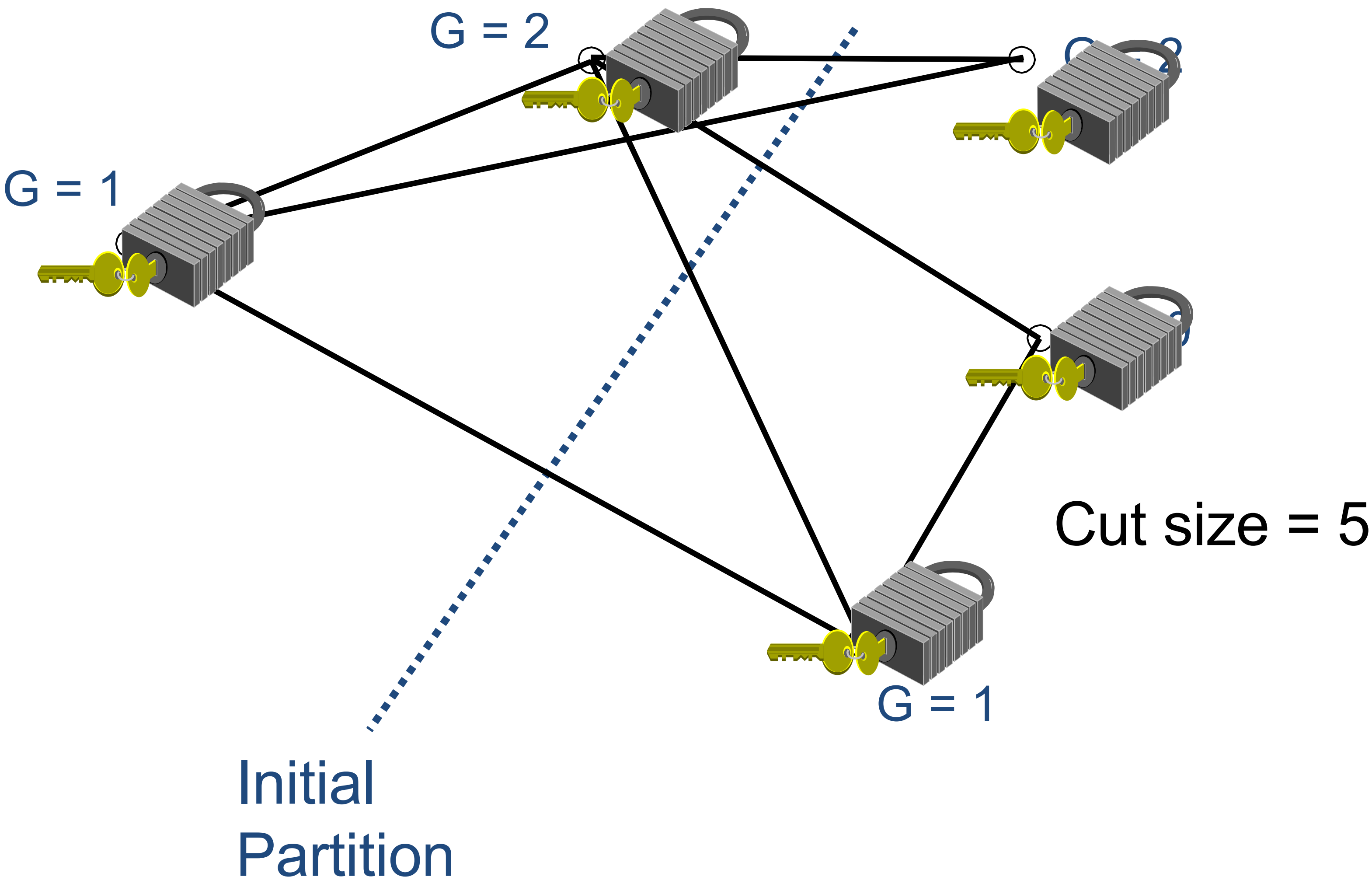
Kernighan-Lin / Fiduccia-Matheyses:



Kernighan-Lin / Fiduccia-Matheyses:



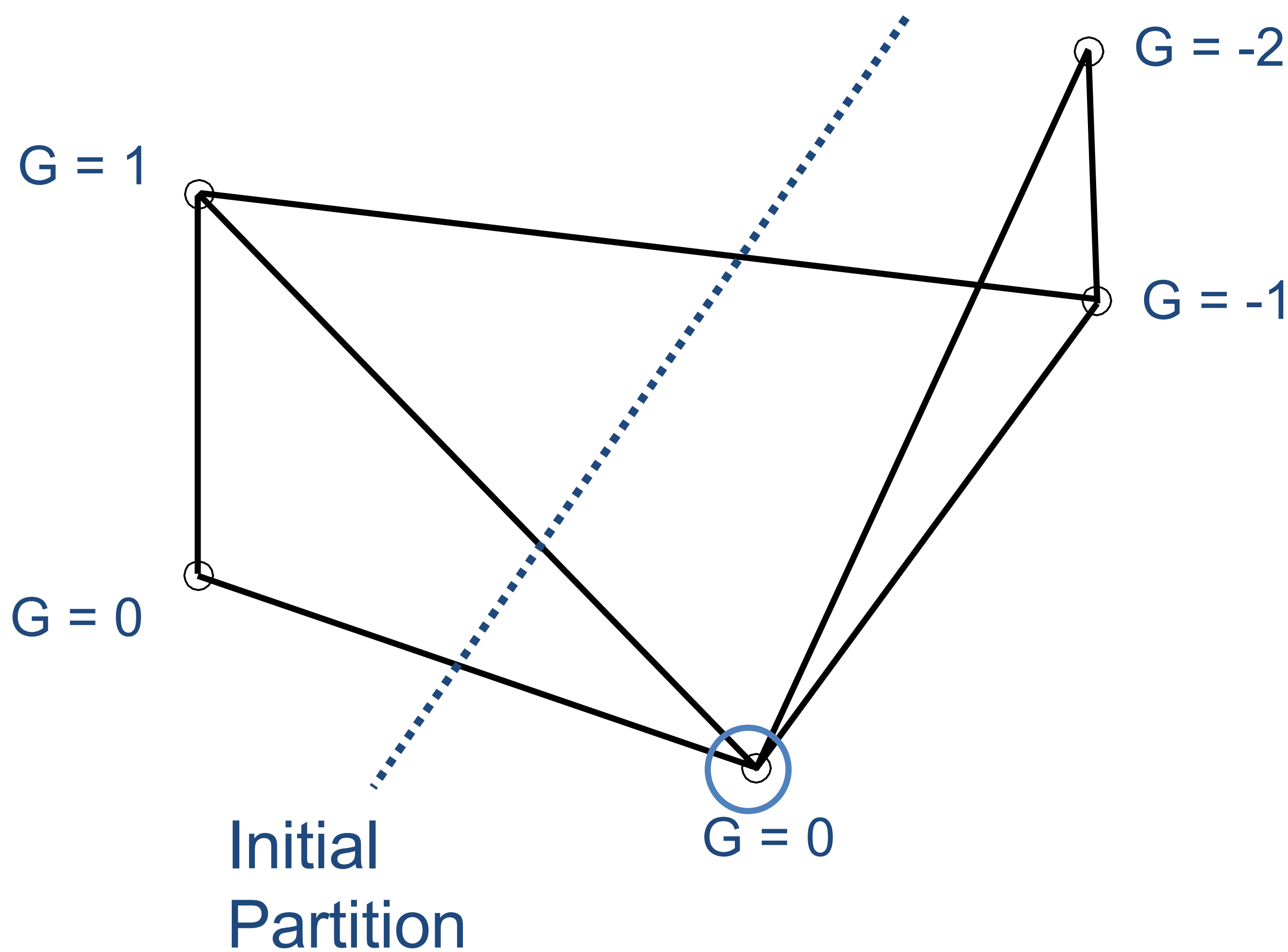
Kernighan-Lin / Fiduccia-Matheyses:



All the nodes are locked so we are done this iteration

Best cut size we saw was 3, so roll back to that and do another iteration

Kernighan-Lin / Fiduccia-Matheyses:



Repeat this for a small number of iterations
(typically 4-5 is enough)

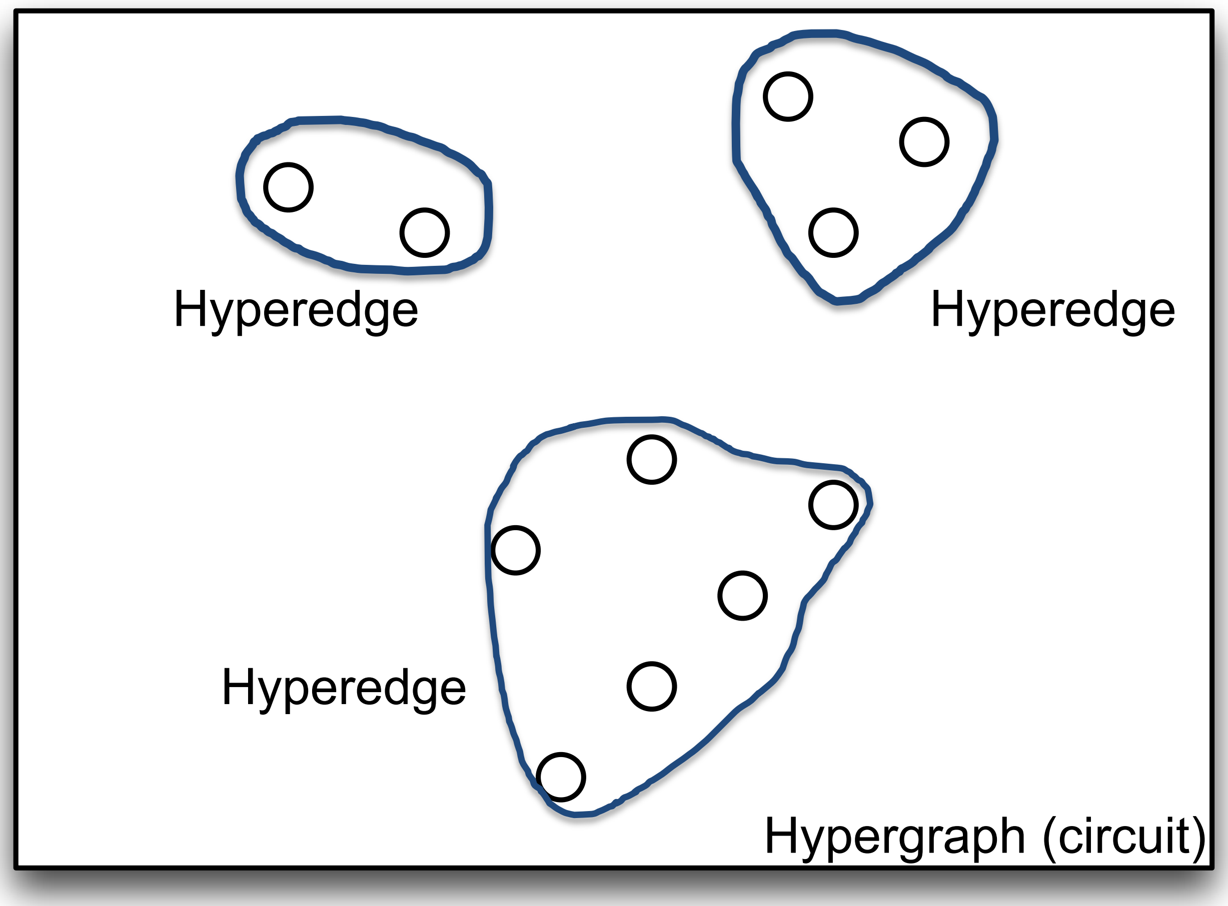
Choose the best cut-size found
and that is our answer

By using clever data structures
this algorithm is O(n)

Multi-sink nets

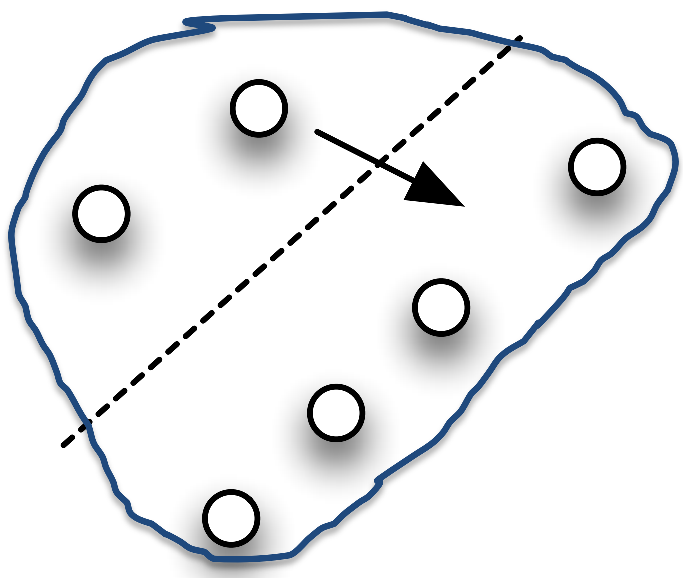
What about nets that have more than one sink?

- Each edge can have more than 2 endpoints



Dealing with Multi-sink nets

Option 1: Use FM as normal:

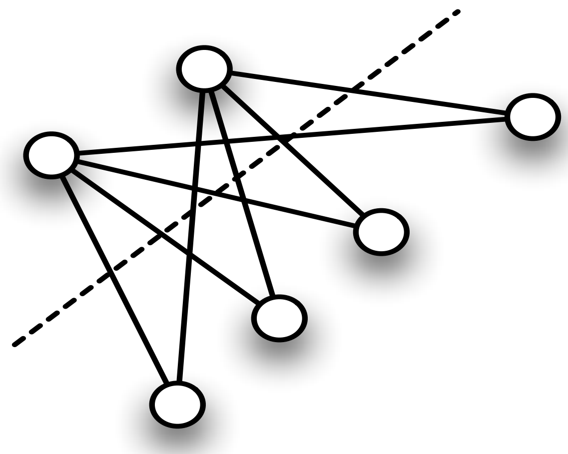


Problem: no motivation to make this move

Dealing with Multi-sink nets

Option 2: Convert hypergraph to a graph and do FM on the graph

usual technique: replace each hyperedge with a clique, and weight each edge on the clique as $\frac{1}{|e|-1}$ where e is the number of terminals



Problem: 1. Wrong Optimization Goal

2. Reduces “sparsity” of graph (increased runtime)

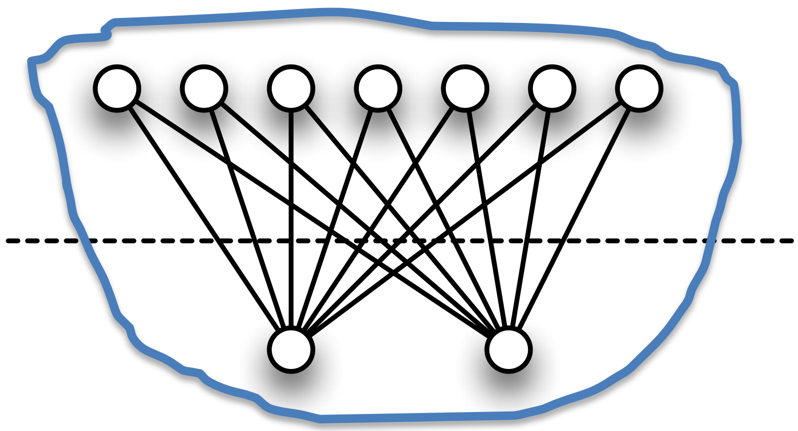
Other approaches: delete edges of the clique randomly

Assignment 3: Due March 2

You will write a Kernighan-Lin / Fiduccia-Matheyses partitioner

Important: Cost function is # of nets that cross the partition

- Each net contributes either 0 or 1 to the sum
- Be careful for multi-terminal nets



This net contributes 1 to the sum.

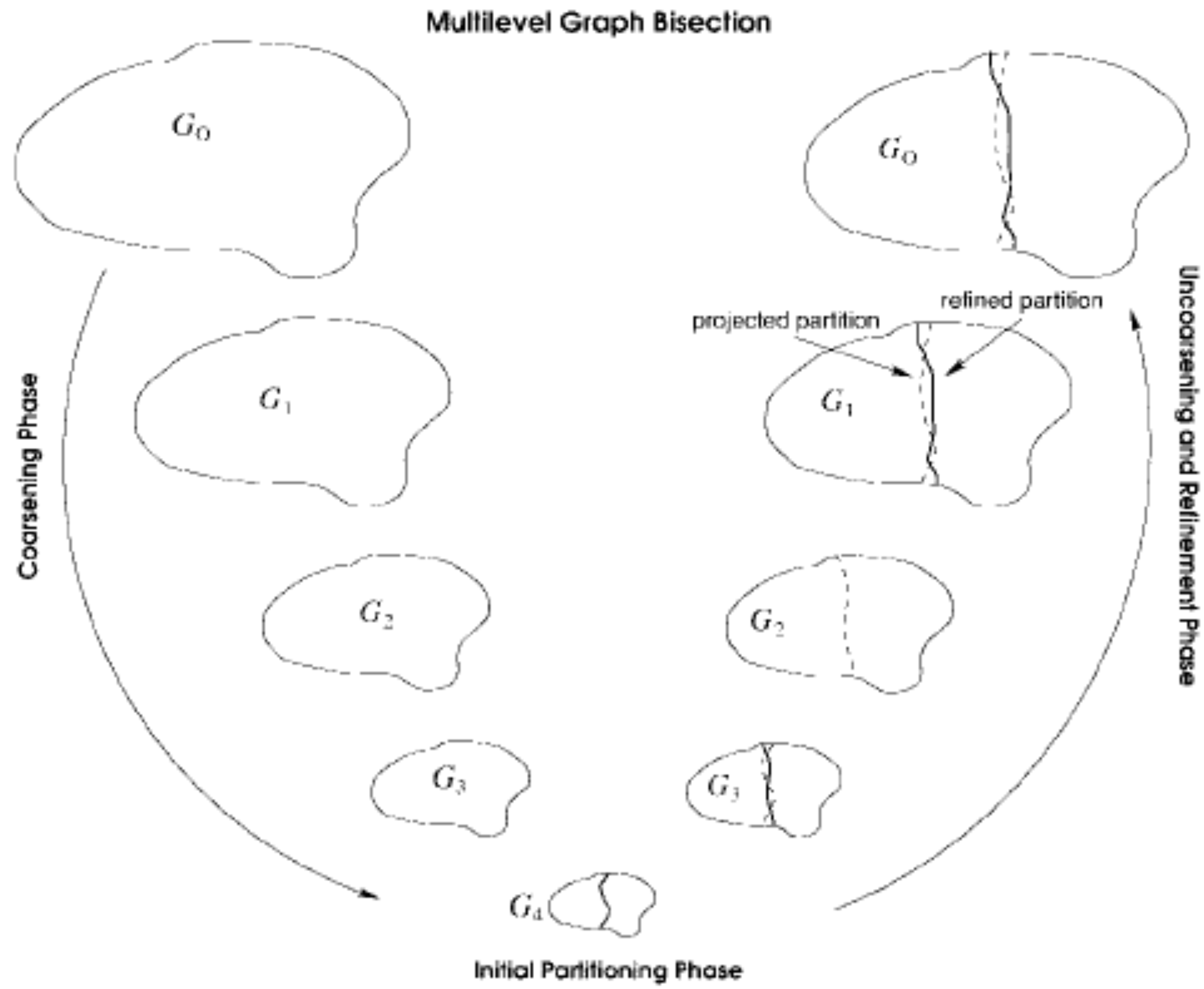
[redacted]

Board notes: Branch and Bound

Board notes: Genetic Algorithms

See paper by Bui on the website – no paper review reqd

Combining with bottom-up Clustering: Hmetis



See paper by Karypis on the website – no paper review reqd

Summary of Partitioning

1. Constructive Algorithms
2. Simulated Annealing
3. Kernighan-Lin / Fiduccia-Matheyses
4. Branch and Bound (Assignment 3!)
5. Genetic Algorithms

Clearly, partitioning algorithms have applications outside of CAD.