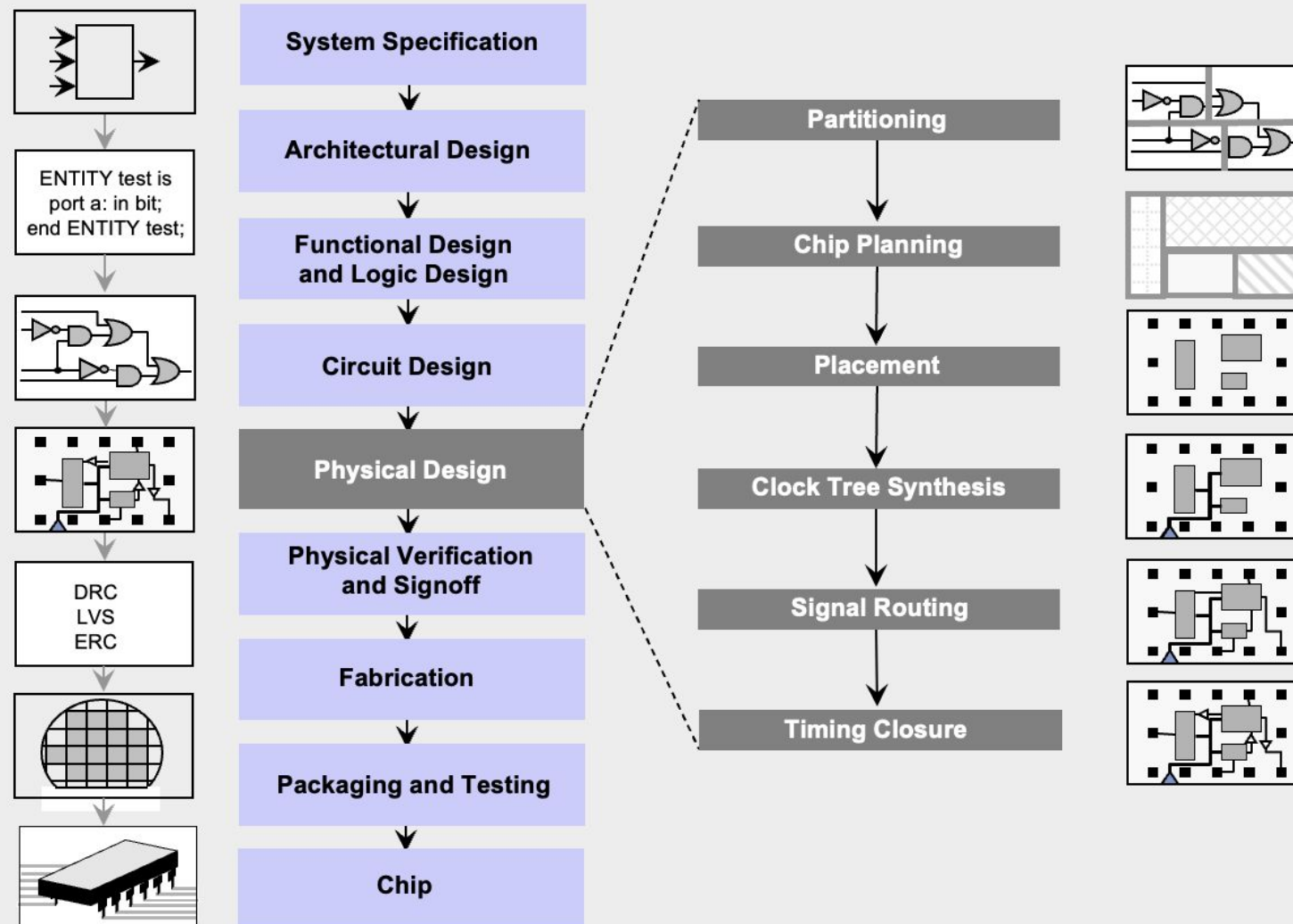


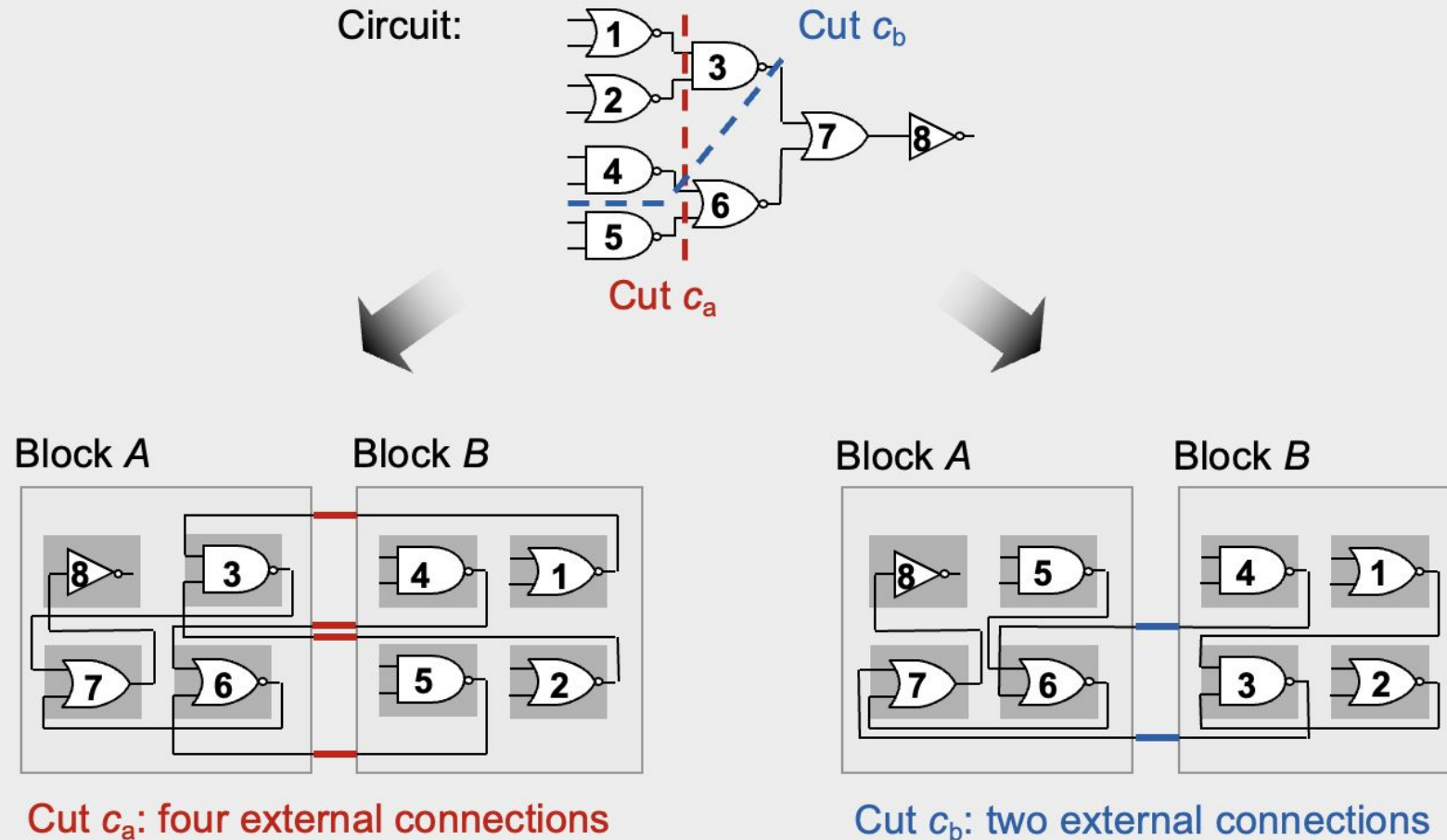
CSE 460: VLSI Design

Lecture 13+14: VLSI Physical Design

VLSI Design Flow

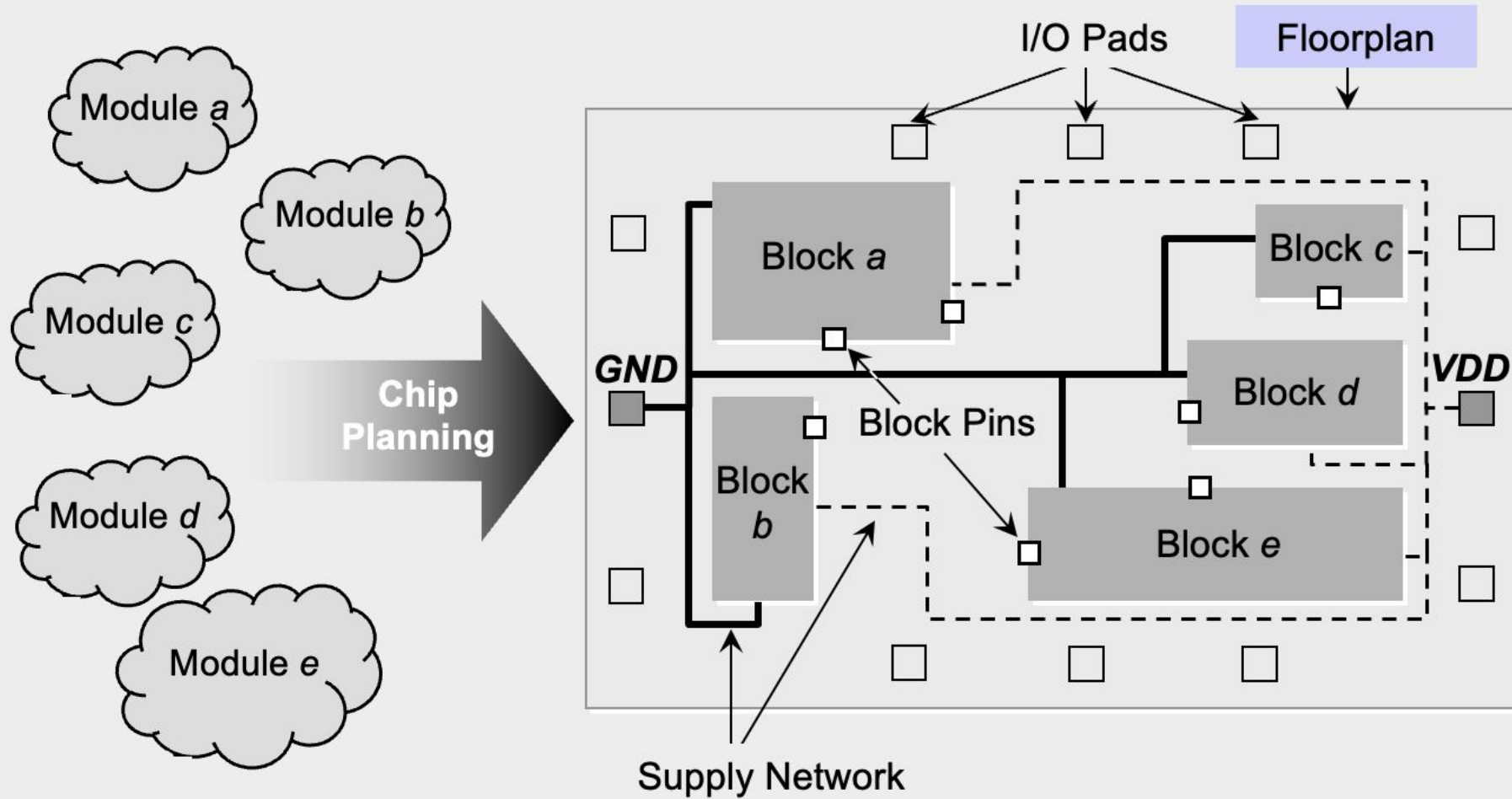


Partitioning



Algorithms Involved: Kernighan-Lin (KL) Algorithm, Fiduccia-Mattheyses (FM) Algorithm

Chip Planning



Chip Planning (Example)

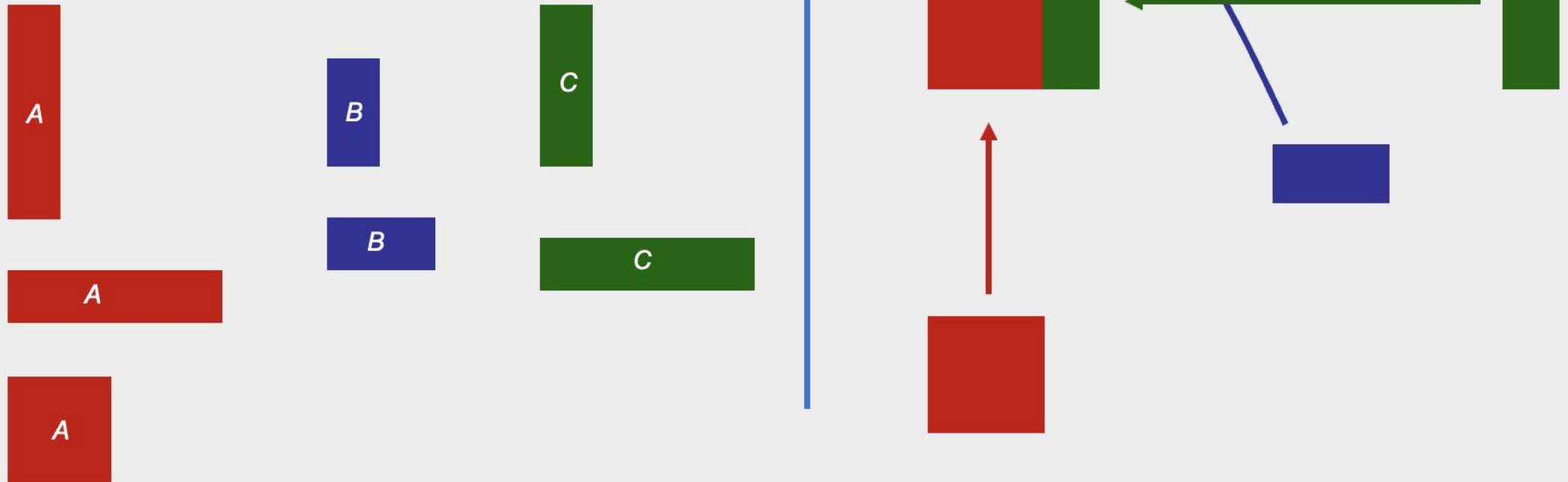
Given: Three blocks with the following potential widths and heights

Block A: $w = 1, h = 4$ or $w = 4, h = 1$ or $w = 2, h = 2$

Block B: $w = 1, h = 2$ or $w = 2, h = 1$

Block C: $w = 1, h = 3$ or $w = 3, h = 1$

Task: Floorplan with minimum total area enclosed

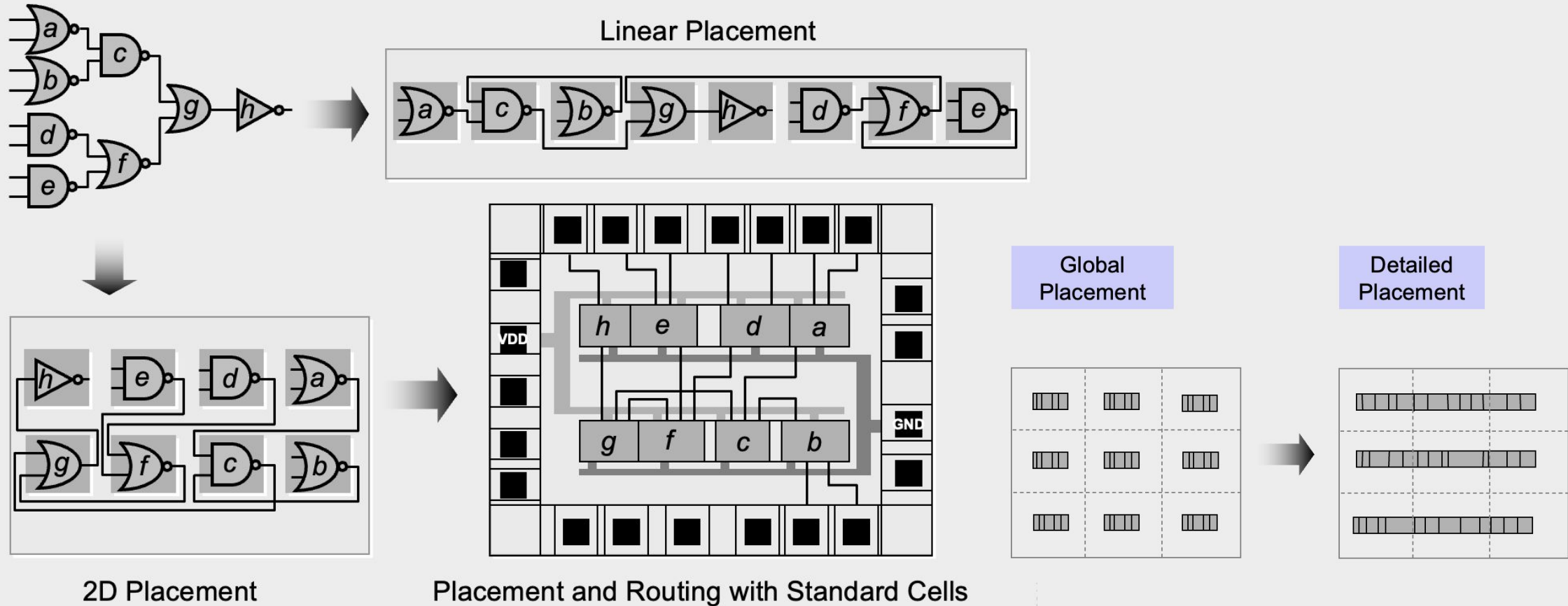


Solution:

Aspect ratios

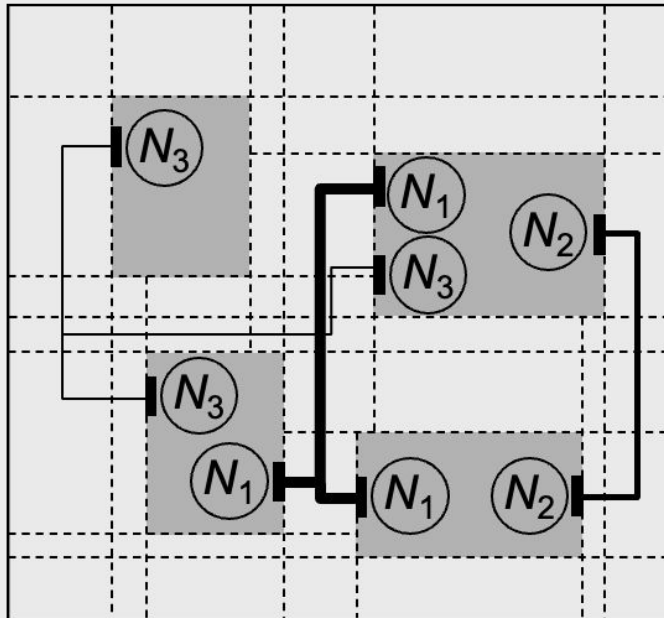
Block A with $w = 2, h = 2$; **Block B** with $w = 2, h = 1$; **Block C** with $w = 1, h = 3$

Placement

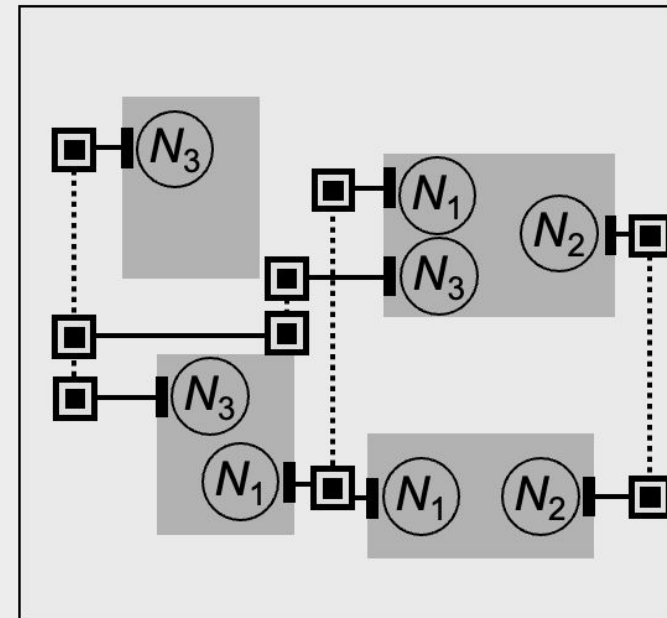


Signal Routing

Global Routing



Detailed Routing



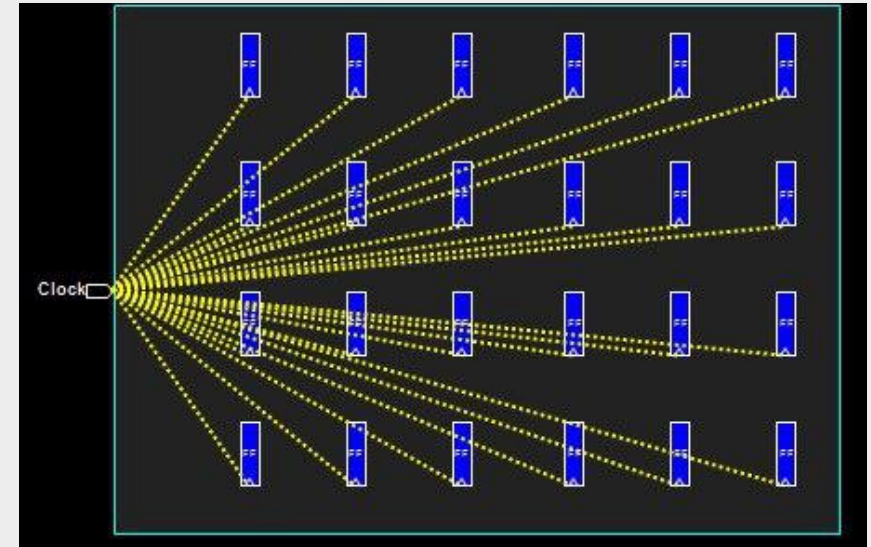
Clock Tree Synthesis (CTS)

Definition:

- ❖ Clock tree synthesis is a process which make sure that the clock gets distributed evenly to all sequential elements in a design.
- ❖ CTS is the process of insertion of buffers or inverters along the clock paths of ASIC design in order to achieve minimum skew or balanced skew.
- ❖ In ICs, clock consumes around half of the total power consumption. Here clock gating technique helps to reduce power consumption by the clocks.

Goals of CTS:

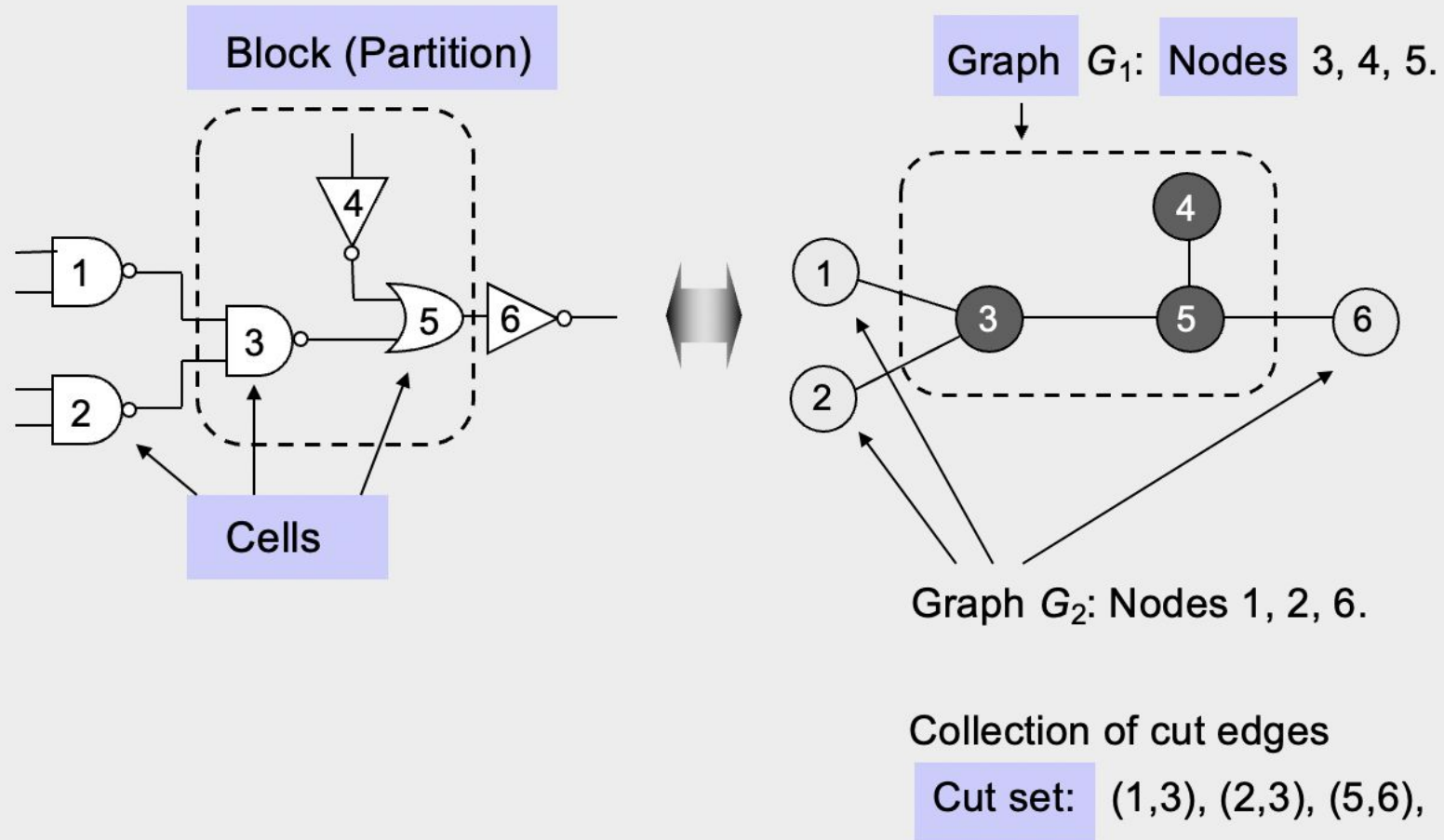
- ❖ To meet clock tree design rule constraints such as maximum transition, maximum load capacitance and maximum fanout.
- ❖ To meet clock tree targets such as minimum skew and minimum insertion delay.



Timing Closure

- ❖ Timing closure is the process by which a logic design consisting of primitive elements such as combinatorial logic gates (and , or , not , nand , nor , etc.) and sequential logic gates (flip flops, latches, memories) is modified to meet its timing requirements.
- ❖ Timing closure is done through layout optimizations and netlist modifications.

Partition Terminology



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.

Optimization Goals

□ In detail, what are the optimization goals?

- 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

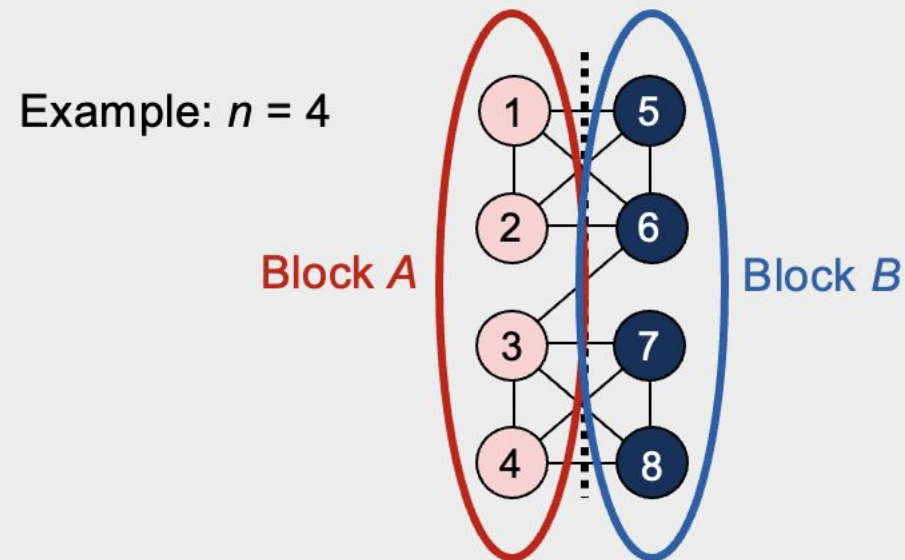
□ How can we meet these goals?

- Unfortunately, this problem is NP-hard
- Efficient heuristics are developed in the 1970s and 1980s. They are high quality and in low-order polynomial time.

Kernighan Lin (KL Algorithm)

Given: A graph with $2n$ nodes where each node has the same weight.

Goal: A partition (division) of the graph into two disjoint subsets A and B with minimum cut cost and $|A| = |B| = n$.



KL Algorithm (Terminology)

Cost $D(v)$ of moving a node v

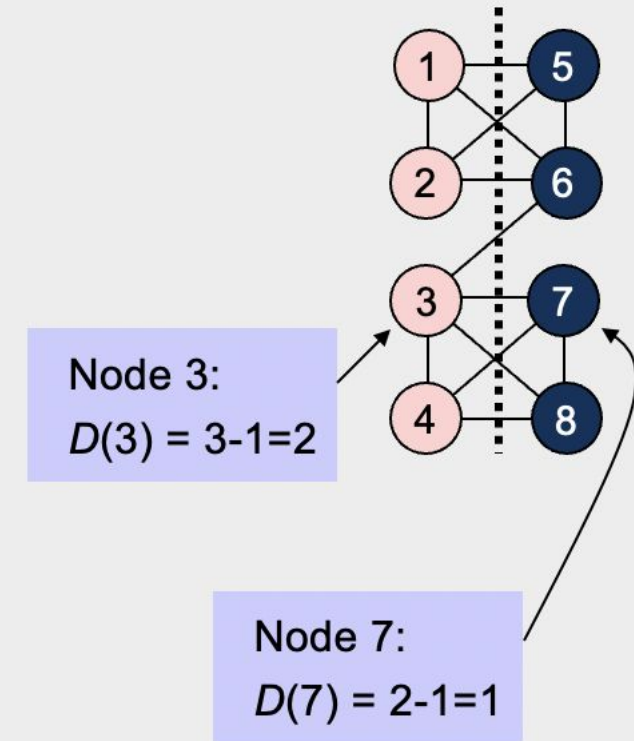
$$D(v) = |E_c(v)| - |E_{nc}(v)| ,$$

where

$E_c(v)$ is the set of v 's incident edges that are cut by the cut line, and

$E_{nc}(v)$ is the set of v 's incident edges that are not cut by the cut line.

High costs ($D > 0$) indicate that the node should move, while low costs ($D < 0$) indicate that the node should stay within the same partition.



KL Algorithm (Terminology)

Gain of swapping a pair of nodes a and b

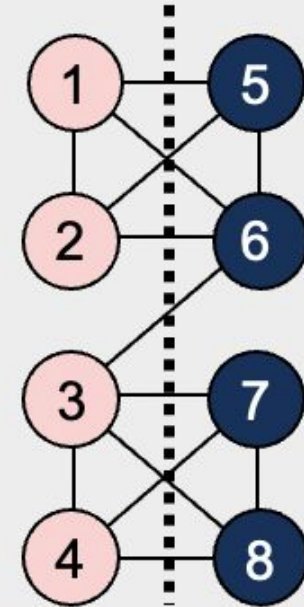
$$\Delta g = D(a) + D(b) - 2 * c(a,b),$$

where

- $D(a)$, $D(b)$ are the respective costs of nodes a , b
- $c(a,b)$ is the connection weight between a and b :
If an edge exists between a and b ,
then $c(a,b)$ = edge weight (here 1),
otherwise, $c(a,b)$ = 0.

The gain Δg indicates how useful the swap between two nodes will be

The larger Δg , the more the total cut cost will be reduced



KL Algorithm (Terminology)

Gain of swapping a pair of nodes a and b

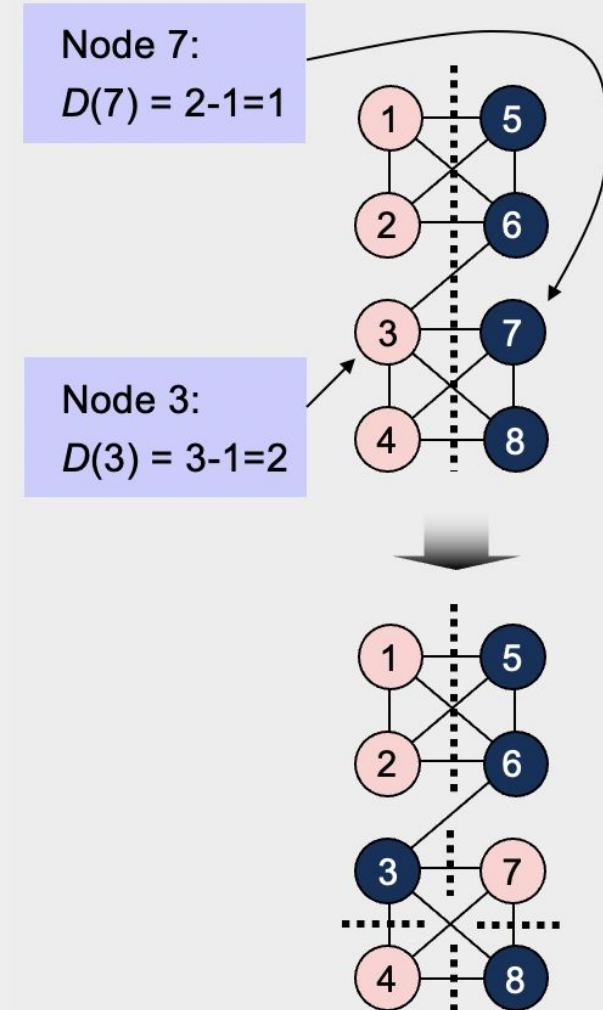
$$\Delta g = D(a) + D(b) - 2 * c(a,b),$$

where

- $D(a)$, $D(b)$ are the respective costs of nodes a , b
- $c(a,b)$ is the connection weight between a and b :
If an edge exists between a and b ,
then $c(a,b)$ = edge weight (here 1),
otherwise, $c(a,b)$ = 0.

$$\Delta g (3,7) = D(3) + D(7) - 2 * c(a,b) = 2 + 1 - 2 = 1$$

=> Swapping nodes 3 and 7 would reduce the cut size by 1



KL Algorithm (Terminology)

Gain of swapping a pair of nodes a and b

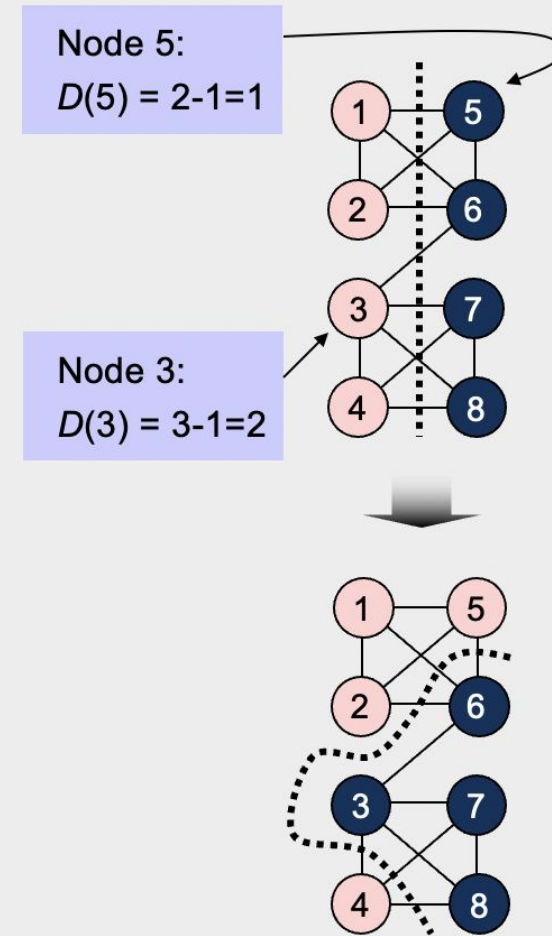
$$\Delta g = D(a) + D(b) - 2 * c(a,b),$$

where

- $D(a)$, $D(b)$ are the respective costs of nodes a , b
- $c(a,b)$ is the connection weight between a and b :
If an edge exists between a and b ,
then $c(a,b)$ = edge weight (here 1),
otherwise, $c(a,b)$ = 0.

$$\Delta g (3,5) = D(3) + D(5) - 2 * c(a,b) = 2 + 1 - 0 = 3$$

=> Swapping nodes 3 and 5 would reduce the cut size by 3



KL Algorithm (Terminology)

Gain of swapping a pair of nodes a and b

The goal is to find a pair of nodes a and b to exchange such that Δg is maximized and swap them.

Maximum positive gain G_m of a pass

The maximum positive gain G_m corresponds to the best prefix of m swaps within the swap sequence of a given pass.

These m swaps lead to the partition with the minimum cut cost encountered during the pass.

G_m is computed as the sum of Δg values over the first m swaps of the pass, with m chosen such that G_m is maximized.

$$G_m = \sum_{i=1}^m \Delta g_i$$

KL Algorithm (One-pass)

Step 0:

- $V = 2n$ nodes
- $\{A, B\}$ is an initial arbitrary partitioning

Step 1:

- $j = 1$
- Compute $D(v)$ for all nodes $v \in V$

Step 2:

- Choose a_j and b_j such that $\Delta g_j = D(a_j) + D(b_j) - 2 * c(a_j, b_j)$ is maximized
- Swap and fix a_j and b_j

Step 3:

- If all nodes are fixed, go to Step 4. Otherwise
- Compute and update D values for all nodes that are connected to a_j and b_j and are not fixed.
- $j = j + 1$
- Go to Step 2

Step 4:

- Find the move sequence $1 \dots m$ ($1 \leq m \leq l$), such that $G_m = \sum_{i=1}^m \Delta g_i$ is maximized
- If $G_m > 0$, go to Step 5. Otherwise, END

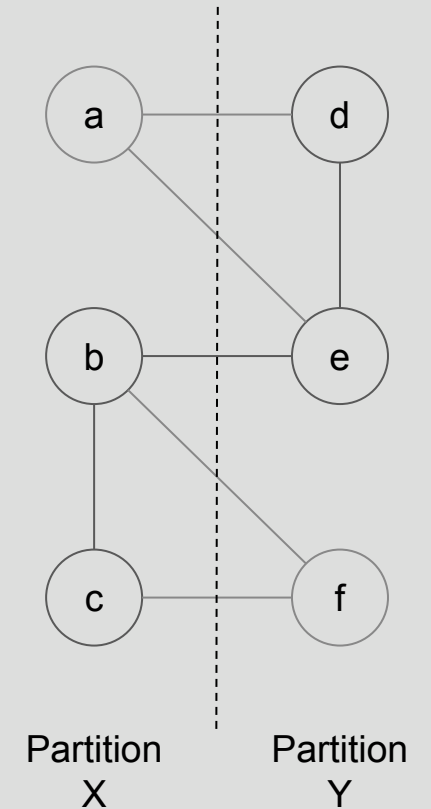
Step 5:

- Execute m swaps, reset remaining nodes
- Go to Step 1

KL Algorithm (Example 1)

The graph (nodes a-f) can be optimally partitioned using the Kernighan-Lin algorithm. The dotted line represents the initial partitioning. Assume all the edges have the same weight.

- a. What is the initial cut cost?
- b. Perform the first pass of the algorithm.
[Hint: For the “i”th iteration of the first pass, until all the nodes are swapped and fixed, do the following:
 - i. Compute/update the node costs of all unfixed nodes
 - ii. Find the maximum gain of swapping a pair of nodes (Δg_i)
 - iii. Swap the pair and draw the updated graph]
- c. Finish the first pass by computing the maximum positive gain, G_m .
Suggest how many swaps should be actually executed in the first pass.
- d. Should you perform subsequent passes of the algorithm? Why or why not?

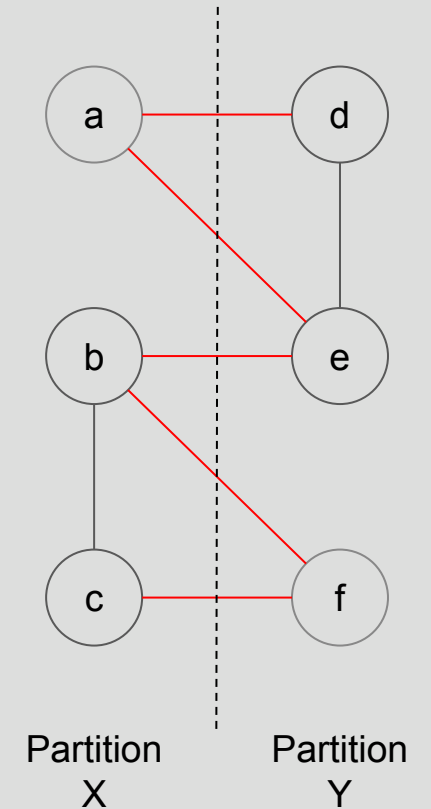


KL Algorithm (Example 1)

The graph (nodes a-f) can be optimally partitioned using the Kernighan-Lin algorithm. The dotted line represents the initial partitioning. Assume all the edges have the same weight.

a. What is the initial cut cost?

Ans: We can see that the initial partition line intersects 5 edges of the graph (shown in **red**). So the initial cut cost is **5**.



KL Algorithm (Example 1)

The graph (nodes a-f) can be optimally partitioned using the Kernighan-Lin algorithm. The dotted line represents the initial partitioning. Assume all the edges have the same weight.

b. Perform the first pass of the algorithm.

For the “i=1” iteration of the first pass:

i. Compute the node costs of all unfixed nodes

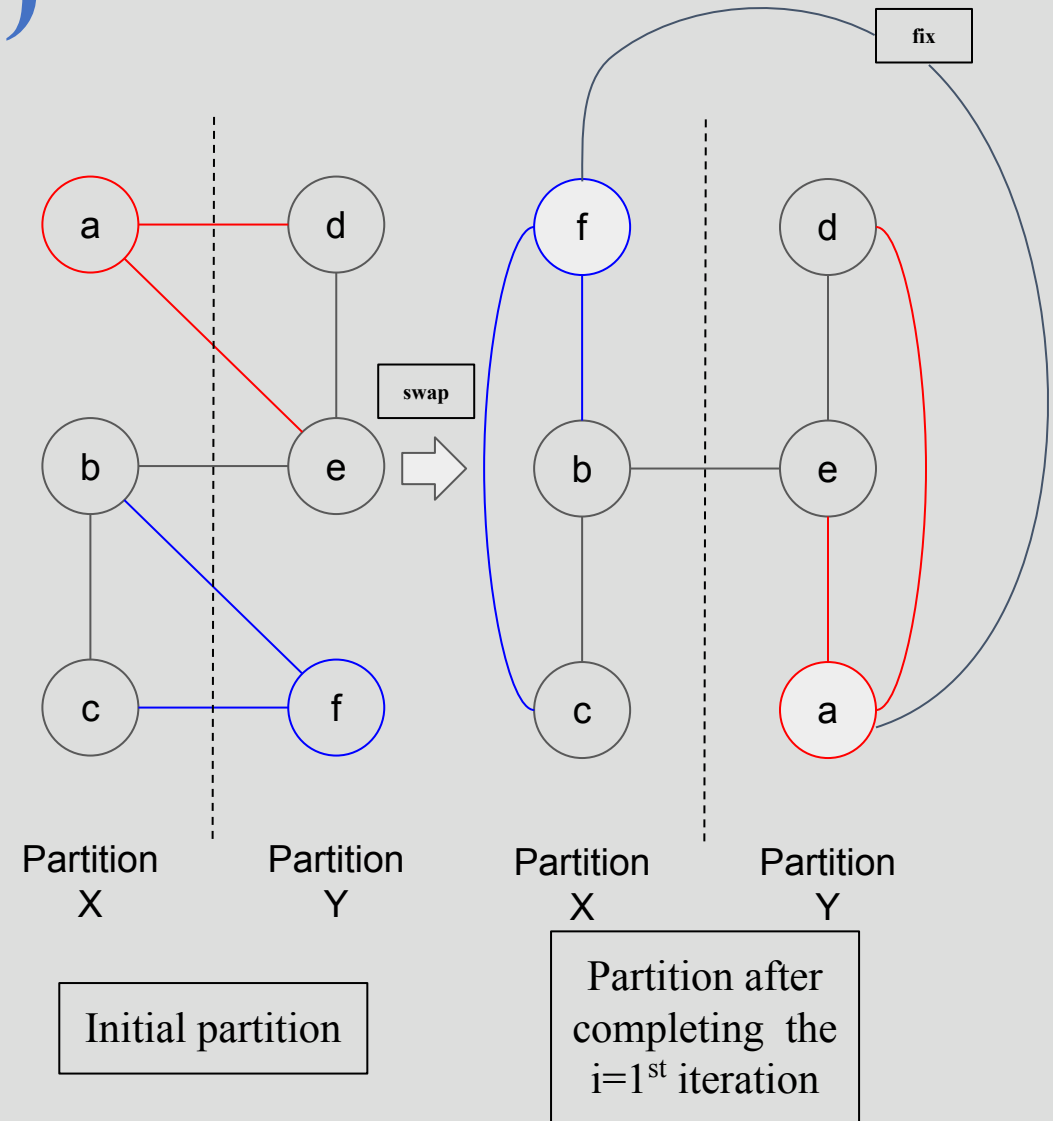
Ans: $D(a) = 2 - 0 = 2$; $D(b) = 2 - 1 = 1$; $D(c) = 1 - 1 = 0$;
 $D(d) = 1 - 1 = 0$; $D(e) = 2 - 1 = 1$; $D(f) = 2 - 0 = 2$.

ii. Find the maximum gain of swapping a pair of nodes (Δg_i)

Ans: $\Delta g_1(a, f) = 2 + 2 - 0 = 4$.

iii. Swap the pair and draw the updated graph

Ans: Swap nodes a & f, and fix them.



KL Algorithm (Example 1)

b. Perform the first pass of the algorithm.

For the “ $i=2$ ” iteration of the first pass:

i. Update the node costs of all unfixed nodes

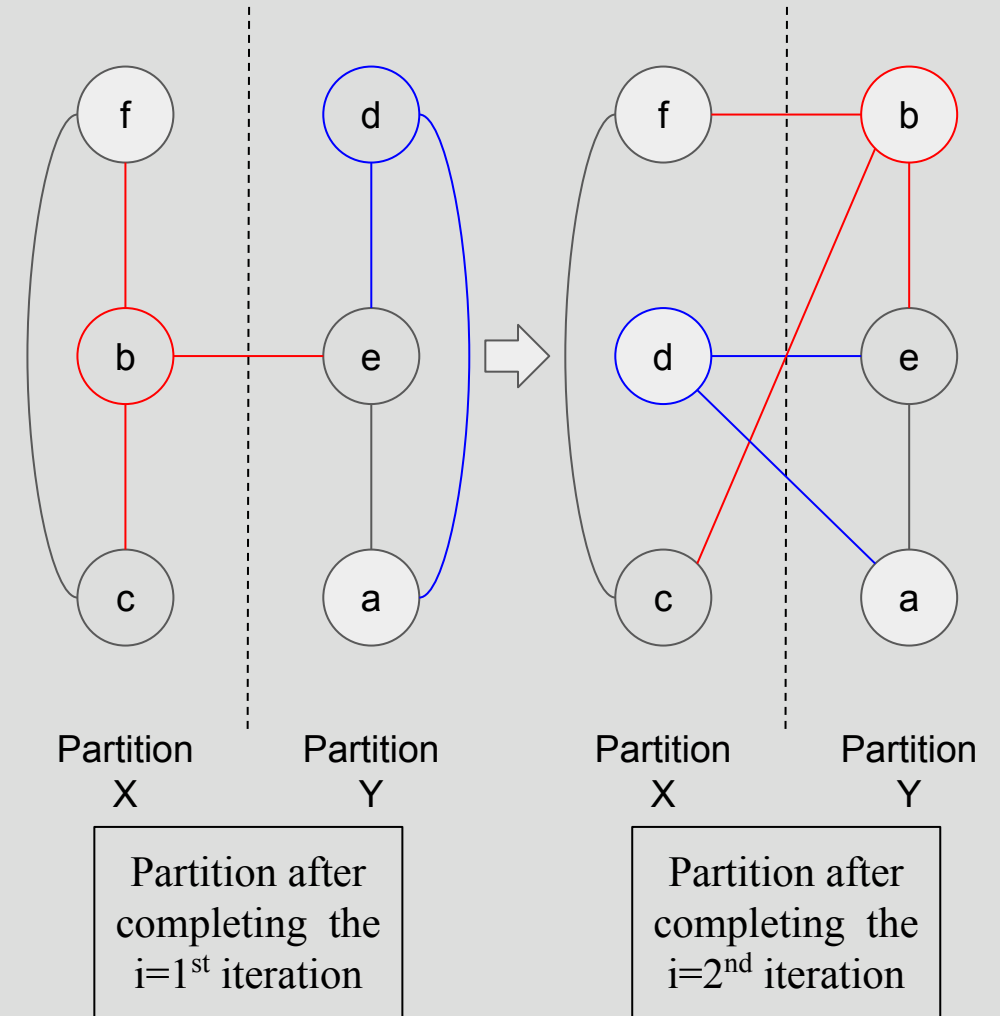
Ans: ~~$D(a) = 2 - 0 = 2$~~ ; $D(b) = 1 - 2 = -1$; $D(c) = 0 - 2 = -2$;
 $D(d) = 0 - 2 = -2$; $D(e) = 1 - 2 = -1$; ~~$D(f) = 2 - 0 = 2$~~ .

ii. Find the maximum gain of swapping a pair of nodes (Δg_i)

Ans: $\Delta g_2(b, d) = -1 - 2 - 0 = -3$ or $\Delta g_2(c, e) = -2 - 1 - 0 = -3$

iii. Swap the pair and draw the updated graph

Ans: Swap nodes b & d, and fix them or c & e, and fix them. [Both approaches are valid and will end in the same result.]



KL Algorithm (Example 1)

b. Perform the first pass of the algorithm.

For the “ $i=3$ ” iteration of the first pass:

i. Update the node costs of all unfixed nodes

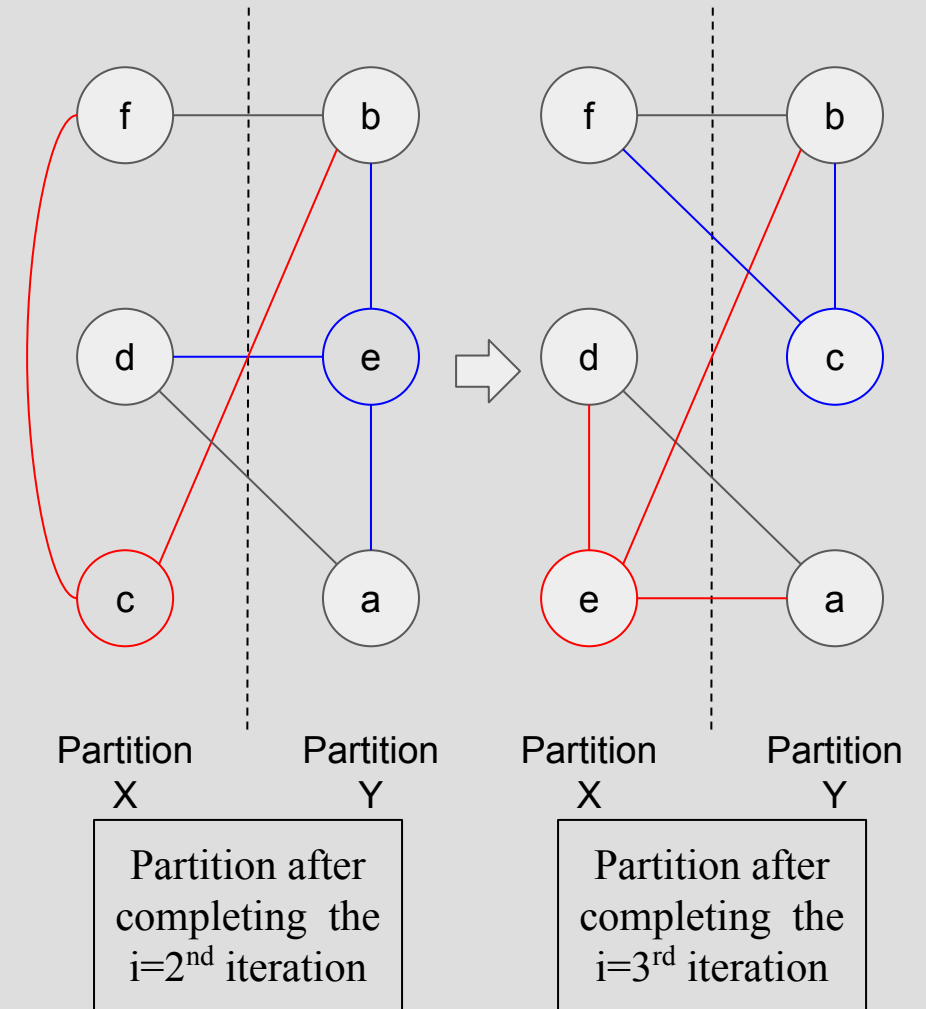
Ans: ~~$D(a) = 2 - 0 = 2$~~ ; ~~$D(b) = 1 - 2 = -1$~~ ; $D(c) = 1 - 1 = 0$;
 ~~$D(d) = 1 - 1 = 0$~~ ; $D(e) = 1 - 2 = -1$; ~~$D(f) = 2 - 0 = 2$~~ .

ii. Find the maximum gain of swapping a pair of nodes (Δg_i)

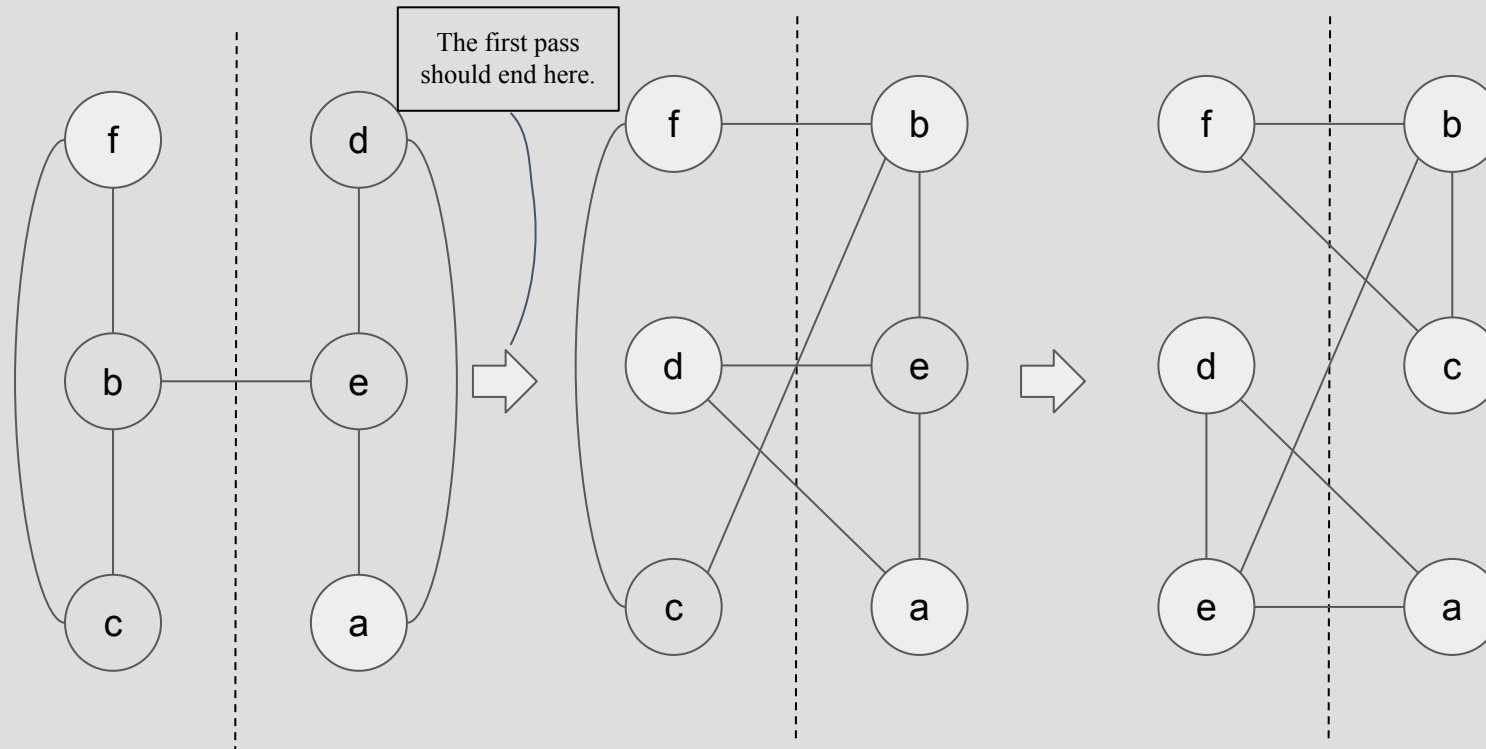
Ans: $\Delta g_3(c, e) = 0 - 1 - 0 = -1$

iii. Swap the pair and draw the updated graph

Ans: Swap nodes c & e, and fix them.



KL Algorithm (Example 1)



- c. Finish the first pass by computing the maximum positive gain, G_m . Suggest how many swaps should be actually executed in the first pass.

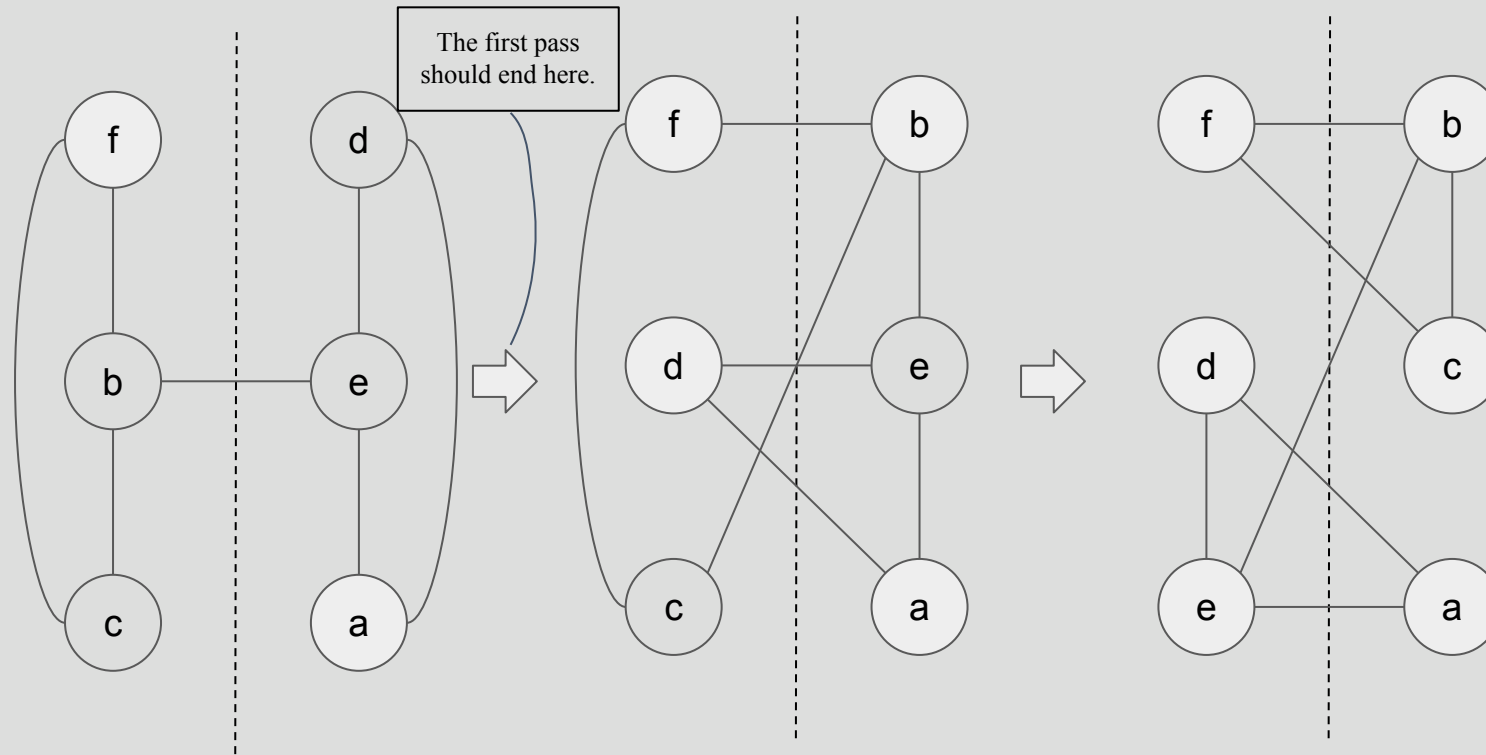
$$G_1 = \Delta g_1 = 4 = 4$$

$$G_2 = \Delta g_1 + \Delta g_2 = 4 - 3 = 1$$

$$G_3 = \Delta g_1 + \Delta g_2 + \Delta g_3 = 4 - 3 - 1 = 0$$

Hence, maximum positive gain is achieved for $m = 1$. Only the swap for Δg_1 , or (a, f) should be executed.

KL Algorithm (Example 1)



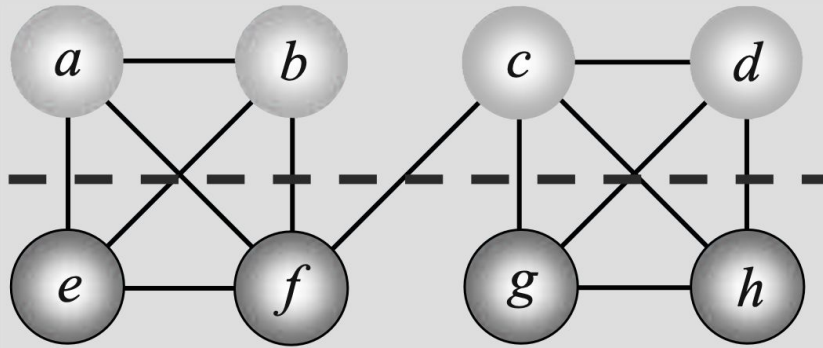
d. Should you perform subsequent passes of the algorithm? Why or why not?

Ans: Since $G_m > 0$, subsequent passes should be performed (until $G_m \leq 0$).

KL Algorithm (Example 2)

Given: initial partition of nodes $a-h$ (right).

Task: perform the first pass of the KL algorithm.



Solution:

Initial cut cost = 9.

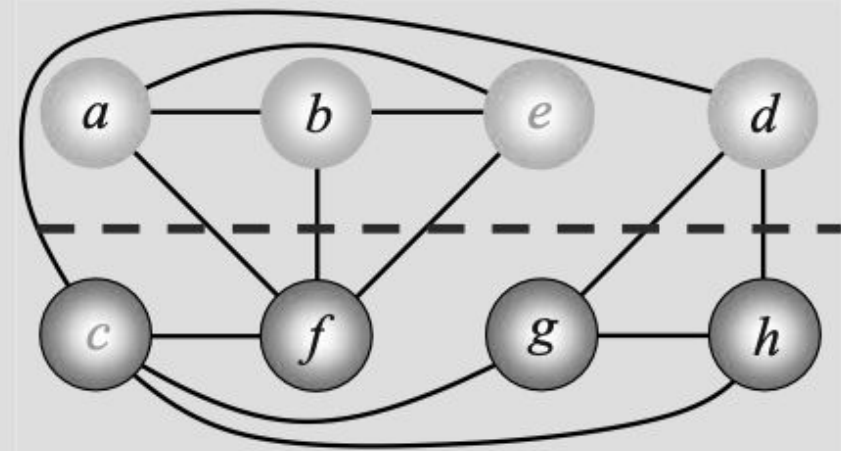
Compute $D(v)$ costs for all free nodes $a-h$.

$D(a) = 1, D(b) = 1, D(c) = 2, D(d) = 1,$

$D(e) = 1, D(f) = 2, D(g) = 1, D(h) = 1$

$\Delta g_1 = D(c) + D(e) - 2c(c,e) = 2 + 1 - 0 = 3$

Swap and fix nodes c and e .



Update $D(v)$ costs for all free nodes connected to newly swapped nodes c and e : a, b, d, f, g and h .

$D(a) = -1, D(b) = -1, D(d) = 3,$

$D(f) = 2, D(g) = -1, D(h) = -1$

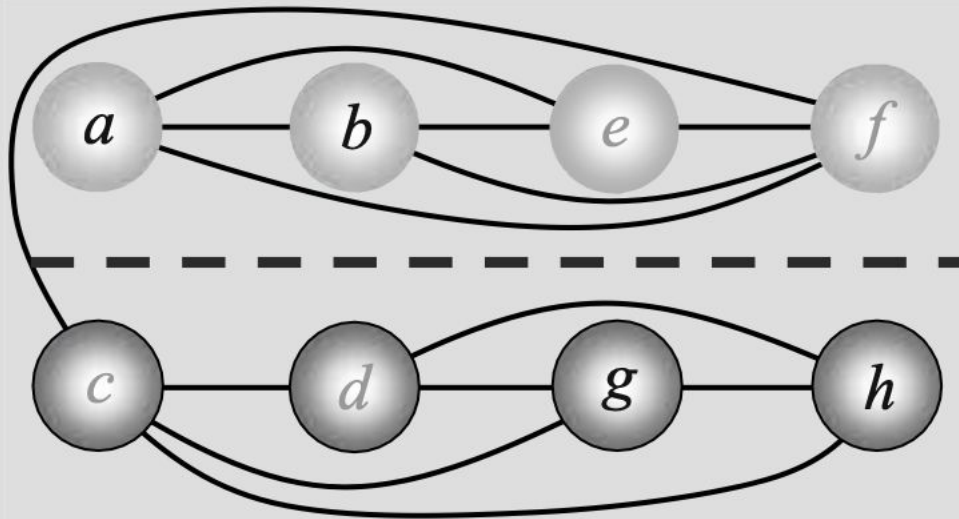
$\Delta g_2 = D(d) + D(f) - 2c(d,f) = 3 + 2 - 0 = 5$

Swap and fix nodes d and f .

KL Algorithm (Example 2)

Given: initial partition of nodes $a-h$ (right).

Task: perform the first pass of the KL algorithm.

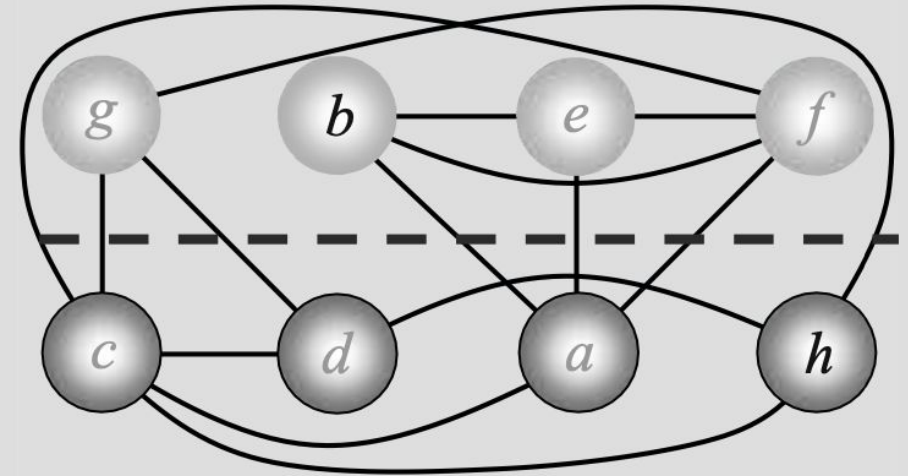


Update $D(v)$ costs for all free nodes connected to newly swapped nodes d and f : a, b, g and h .

$$D(a) = -3, D(b) = -3, D(g) = -3, D(h) = -3$$

$$\Delta g_3 = D(a) + D(g) - 2c(a,g) = -3 + -3 - 0 = -6$$

Swap and fix nodes a and g .



Update $D(v)$ costs for all free nodes connected to newly swapped nodes a and g : b and h .

$$D(b) = -1, D(h) = -1$$

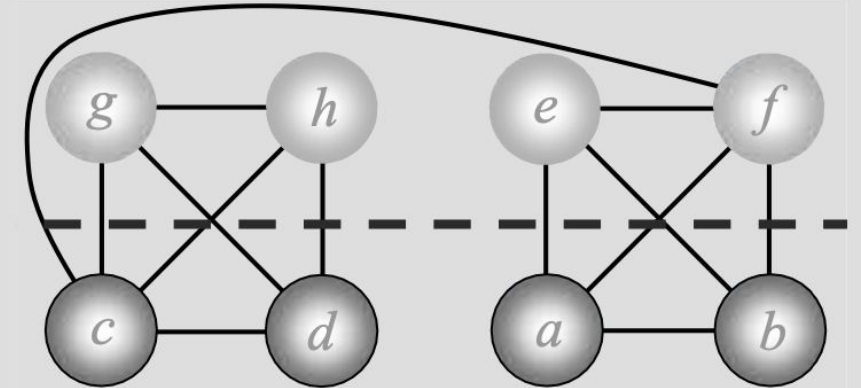
$$\Delta g_4 = D(b) + D(h) - 2c(b,h) = -1 + -1 - 0 = -2$$

Swap and fix nodes b and h .

KL Algorithm (Example 2)

Given: initial partition of nodes $a-h$ (right).

Task: perform the first pass of the KL algorithm.



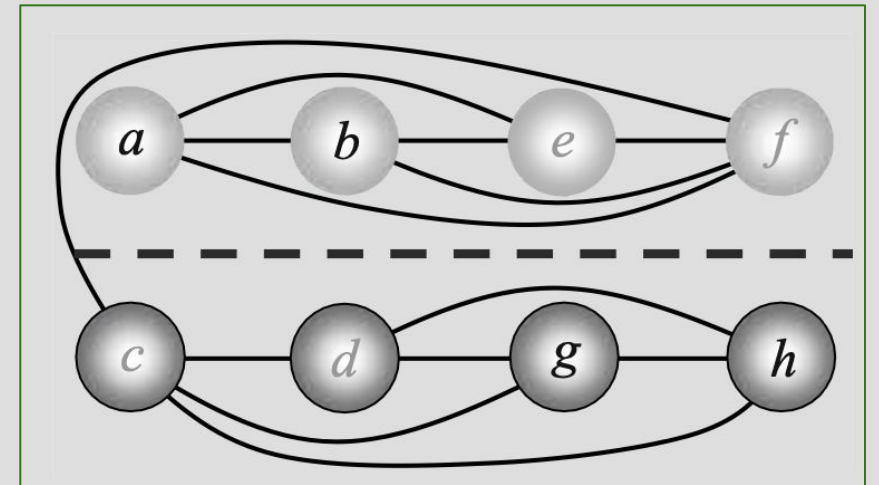
Compute maximum positive gain G_m .

$$G_1 = \Delta g_1 = 3$$

$$G_2 = \Delta g_1 + \Delta g_2 = 3 + 5 = 8$$

$$G_3 = \Delta g_1 + \Delta g_2 + \Delta g_3 = 3 + 5 + -6 = 2$$

$$G_4 = \Delta g_1 + \Delta g_2 + \Delta g_3 + \Delta g_4 = 3 + 5 + -6 + -2 = 0$$



$G_m = 8$ with $m = 2$. Since $G_m > 0$, the first $m = 2$ swaps are executed: (c, e) and (d, f) . Additional passes are performed until $G_m \leq 0$.

Thank you.