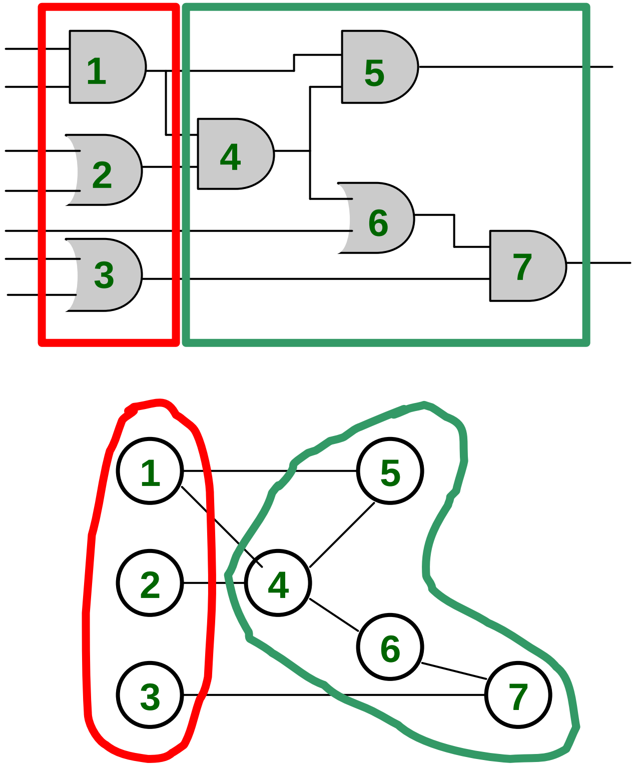
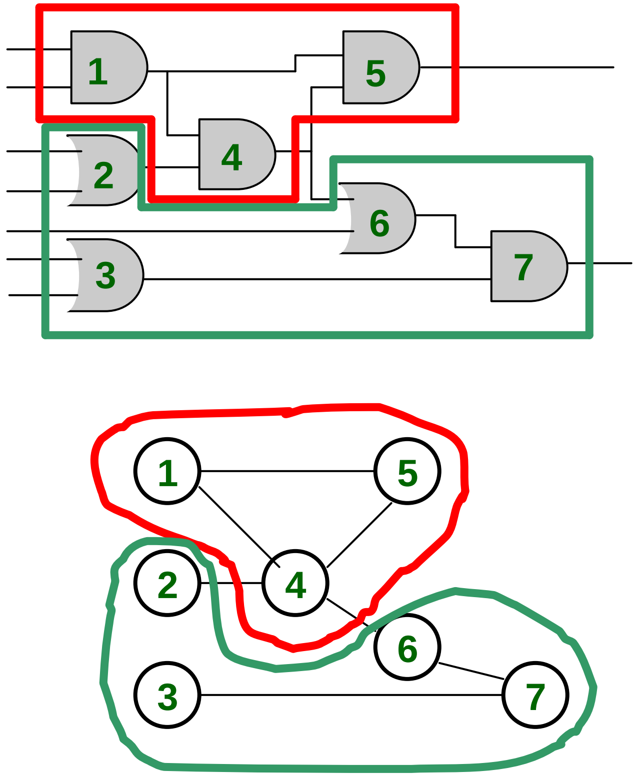
Kernighan-Lin Heuristic

- **Objective:** Partition a circuit in such a way that the numbers of connections among the subsets is minimized
  - Circuit is represented as a graph



Cut size: 4



Cut size: 2

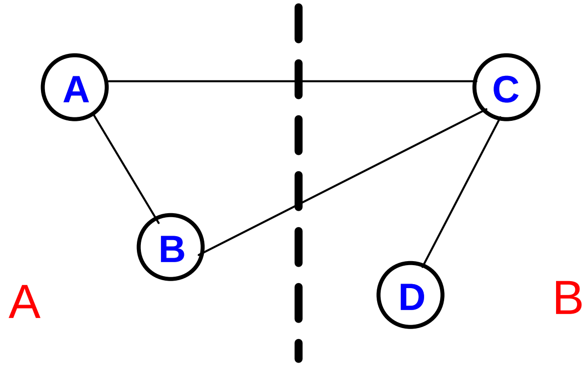
Kernighan-Lin Heuristic

- **Problem:**
  - given:
    - an edge-weighted undirected graph $G(V, E)$ with $2n$ vertices ($|V| = 2n$).
    - an edge $e(a,b) \in E$ has a weight $\omega(a,b)$
  - goal
    - Find to sets A and B, subject to $A \cup B = V$, $A \cap B = 0$ and $|A| = |B| = n$

      by minimizing the cost function $\sum_{(a,b) \in AxB} w(a,b)$
  - problem
    - 2 way, balanced partitioning problem (NP-complete)
  - idea
    - start with an initial random partition
    - exchange two vertices which are connected and see if the cut size is reduced.
    - select the best pair of vertices, look them and start again

Kernighan-Lin Heuristic

- **Definitions**



- External costs
  - the external costs of a vertex in A ($v_i \in$ A) are

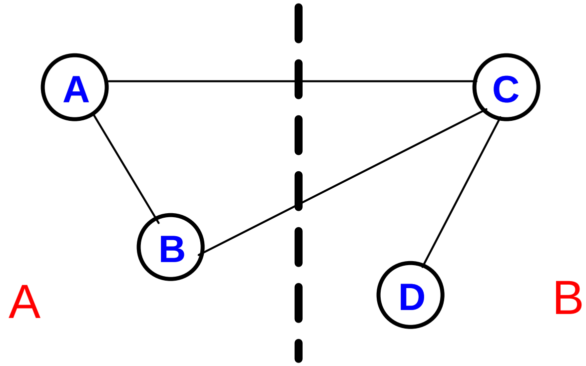
$$E(i) = \sum_{e=\{v_i, v_j\} \in E, v_j \in B} w(v_i, v_j)$$

- Internal costs
  - The internal costs of a vertex in A ($v_i \in$ A) are

$$I(i) = \sum_{e=\{v_i, v_j\} \in E, v_j \in A} w(v_i, v_j)$$

Kernighan-Lin Heuristic

- **Definitions**

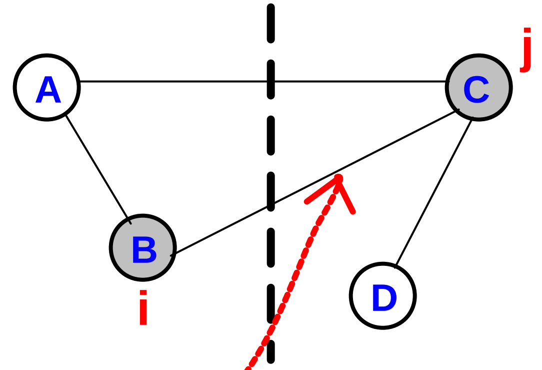


  - Gain of vertex
    - The difference between the internal cost I(i) and external costs E(i) gives an indication about the desirability to move a vertex

$$\boxed{D(i) = E(i) - I(i)}$$

Kernighan-Lin Heuristic

- **Definitions**



C = old cut cost
C‘ = new cut cost
DC = change in cut costs

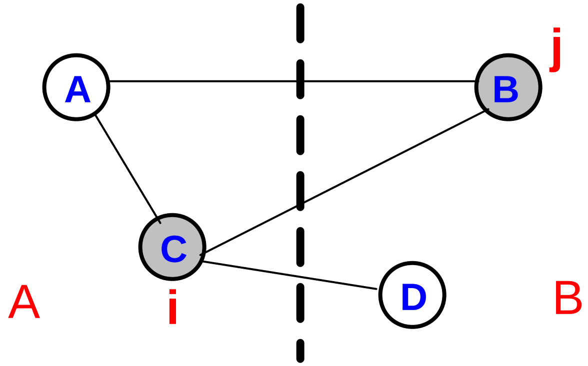
E(i) = old external costs of vertex i
E(i)‘ = new external costs of vertex i
I(i) = old internal costs of vertex i
I(i)‘ = new internal costs of vertex i

- Cut costs

$$
\begin{aligned}
D(i) &= E(i) - I(i)\\
C' &= C + I(i) + I(j) - E'(i) - E'(j)\\
E'(i) &= E(i) - w(i, j)\\
E'(j) &= E(j) - w(i, j)\\
C' - C &= DC\\
&= I(i) + I(j) - E(i) + w(i, j) - E(j) + w(i, j)\\
&= 2\, w(i, j) - D(i) - D(j)
\end{aligned}
$$

Kernighan-Lin Heuristic

- **Definitions**



C = old cut cost
C' = new cut cost
D**C** = change in cut costs

E(**i**) = old external costs of vertex **i**
E(**i**)' = new external costs of vertex **i**
I(**i**) = old internal costs of vertex **i**
I(**i**)' = new internal costs of vertex **i**

- Update of the external and internal costs

$$
\begin{aligned}
D(i) &= E(i) - I(i)\\
E'(i) &= E(i) - w(i, j_{min}) + w(i, i_{min})\\
I'(i) &= I(i) + w(i, j_{min}) - w(i, i_{min})\\
D'(i) &= E'(i) - I'(i)\\
D'(i) &= E(i) - I(i) - w(i, j_{min}) - w(i, j_{min}) + w(i, i_{min}) + w(i, i_{min})\\
&= D(i) - 2w(i, j_{min}) + 2w(i, i_{min})
\end{aligned}
$$

Kernighan-Lin Heuristic

```
Kernighan_Lin (A, B)

compute D values
for i from 1 to n
    Locked[ i ]:= false

BestCost := Cost[ 0 ] := cutsize(A, B)
BestChange := 0
for s from 1 to n/2 do
    Cost[ s ] = ¥
    for i, j from 1 to n such that vi Î A and Locked[ I ] = false and vj Î B and Locked[ j ] = false do
        if 2 w[i, j] – D[ i ] – D[ j ] < Cost[ s ] then
                Pair[ s ]    := ( i, j )
                Costs[ s ] := 2w[ i, j ] – D[ i ] – D[ j ]
    (imin, jmim) := Pair[ s ]
    Locked[ imin ] = Locked[ jmin ] = true
    for i from 1 to n such that Locked[ i ] = false do
        if vi Î A then
                D[ i ] = D[ i ] – 2w[ i, jmin ] + 2w[ i, imin ]
        else
                D[ i ] = D[ i ] – 2w[ i, imin ] + 2w[ i, jmin ]
    Cost[ s ] := Cost[ s – 1 ] + Cost[ s ]
    if Cost[ s ] < BestCost then
        BestChange := s
        BestCost := Cost[ s ]

    for s from 1 to BestChange do
        exchangePair[ s ]
```

Kernighan-Lin Heuristic

- **Properties:**
  - Time complexity is $O(n^3)$
  - The result of the algorithm is used as the initial solution for a new call of the algorithm. This is done till two calls results in the same solution. A typical number of calls is about 4-6
  - Kerinighan-Lin is an iterative heuristic. The solution depends on the start values.