Example

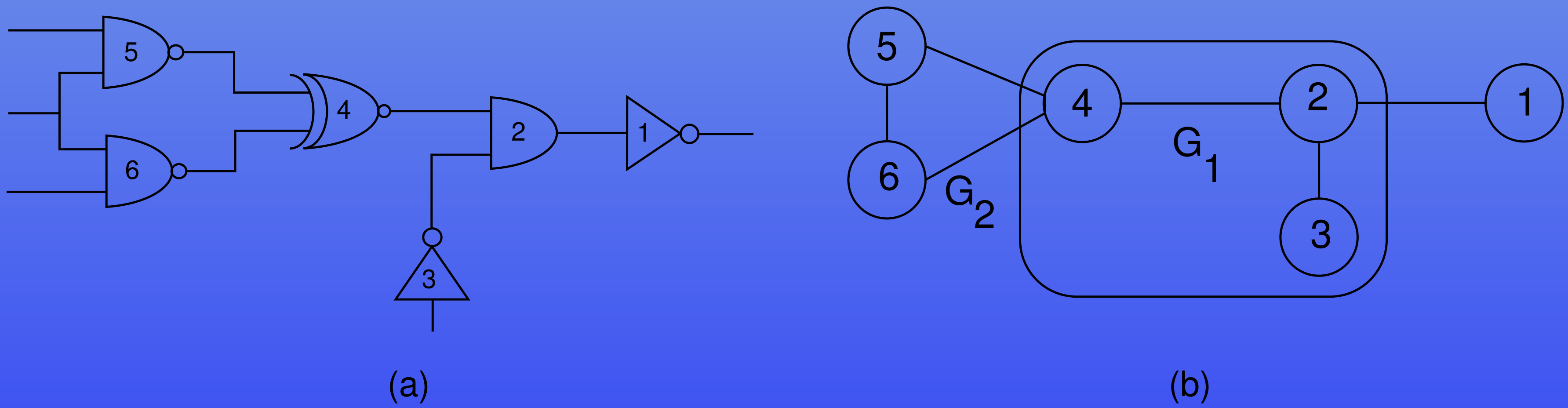

Figure 6: (a) A circuit to be partitioned (b) Its corresponding graph

Example - contd

- Step 1: Initialization.

  Let the initial partition be a random division of vertices into the partition A={2,3,4} and B={1,5,6}.

  $A' = A$ ={2,3,4}, and $B' = B$ ={1,5,6}.

- Step 2: Compute $D-$values.

$$
\begin{aligned}
D_1 &= E_1 - I_1 = 1 - 0 = +1 \\
D_2 &= E_2 - I_2 = 1 - 2 = -1 \\
D_3 &= E_3 - I_3 = 0 - 1 = -1 \\
D_4 &= E_4 - I_4 = 2 - 1 = +1 \\
D_5 &= E_5 - I_5 = 1 - 1 = +0 \\
D_6 &= E_6 - I_6 = 1 - 1 = +0
\end{aligned}
$$

Example - contd

- Step 3: Compute gains.

$$
\begin{aligned}
g_{21} &= D_2 + D_1 - 2c_{21} = (-1) + (+1) - 2(1) = -2 \\
g_{25} &= D_2 + D_5 - 2c_{25} = (-1) + (+0) - 2(0) = -1 \\
g_{26} &= D_2 + D_6 - 2c_{26} = (-1) + (+0) - 2(0) = -1 \\
g_{31} &= D_3 + D_1 - 2c_{31} = (-1) + (+1) - 2(0) = +0 \\
g_{35} &= D_3 + D_5 - 2c_{35} = (-1) + (+0) - 2(0) = -1 \\
g_{36} &= D_3 + D_6 - 2c_{36} = (-1) + (+0) - 2(0) = -1 \\
g_{41} &= D_4 + D_1 - 2c_{41} = (+1) + (+1) - 2(0) = +2 \\
g_{45} &= D_4 + D_5 - 2c_{45} = (+1) + (+0) - 2(1) = -1 \\
g_{46} &= D_4 + D_6 - 2c_{46} = (+1) + (+0) - 2(1) = -1
\end{aligned}
$$

- The largest g value is g_{41}. (a_1, b_1) is $(4, 1)$, the gain $g_{41} = g_1$= 2, and $A' = A' -\{4\}$={2,3}, $B' = B' - \{1\} = \{5, 6\}$.

Example - contd

- Both A' and B' are not empty; then we update the $D-$values in the next step and repeat the procedure from Step 3.
- Step 4: Update $D-$values of nodes connected to (4,1).

  The vertices connected to (4,1) are vertex (2) in set A' and vertices (5,6) in set B'. The new $D-$values for vertices of A' and B' are given by

$$D_2^{'} = D_2 + 2c_{24} - 2c_{21} = -1 + 2(1-1) = -1$$
$$D_5^{'} = D_5 + 2c_{51} - 2c_{54} = +0 + 2(0-1) = -2$$
$$D_6^{'} = D_6 + 2c_{61} - 2c_{64} = +0 + 2(0-1) = -2$$

Example - contd

- To repeat Step 3, we assign $D_i = D_i^{'}$ and then recompute the gains:

$$g_{25} = D_2 + D_5 - 2c_{25} = (-1) + (-2) - 2(0) = -3$$
$$g_{26} = D_2 + D_6 - 2c_{26} = (-1) + (-2) - 2(0) = -3$$
$$g_{35} = D_3 + D_5 - 2c_{35} = (-1) + (-2) - 2(0) = -3$$
$$g_{36} = D_3 + D_6 - 2c_{36} = (-1) + (-2) - 2(0) = -3$$

- All the g values are equal, so we arbitrarily choose g_{36}, and hence the pair (a_2, b_2) is $(3, 6)$,

$$g_{36} = g_2 = -3,$$
$$A' = A' - \{3\} = \{2\},$$
$$B' = B' - \{6\} = \{5\}.$$

Example - contd

- The new $D-$values are:

$$D_2' = D_2 + 2c_{23} - 2c_{26} = -1 + 2(1 - 0) = 1$$
$$D_5' = D_5 + 2c_{56} - 2c_{53} = -2 + 2(1 - 0) = 0$$

- The corresponding new gain is:
  $g_{25} = D_2 + D_5 - 2c_{52} = (+1) + (0) - 2(0) = +1$
- Therefore the last pair (a_3, b_3) is (2,5) and the corresponding gain is $g_{25} = g_3 = +1$.

Example - contd

- Step 5: Determine k.
- We see that $g_1 = +2$, $g_1 + g_2 = -1$, and $g_1 + g_2 + g_3 = 0$.
- The value of k that results in maximum G is 1.
- Therefore, $X = \{a_1\} = \{4\}$ and $Y = \{b_1\} = \{1\}$.
- The new partition that results from moving X to B and Y to A is, $A = \{1, 2, 3\}$ and $B = \{4, 5, 6\}$.
- The entire procedure is repeated again with this new partition as the initial partition.
- Verify that the second iteration of the algorithm is also the last, and that the best solution obtained is $A = \{1, 2, 3\}$ and $B = \{4, 5, 6\}$.

Time Complexity

- Computation of the $D-$values requires $O(n^2)$ time $((O(n)$ for each node).
- It takes constant time to update any $D-$value. We update as many as $(2n - 2i)$ $D-$values after swapping the pair (a_i, b_i).
- Therefore the total time spent in updating the $D-$values can be

$$\sum_{i=1}^{n}(2n - 2i) = O(n^2)$$

- The pair selection procedure is the most expensive step in the Kernighan-Lin algorithm. If we want to pick (a_i, b_i), there are as many as $(n - i + 1)^2$ pairs to choose from leading to an overall complexity of $O(n^3)$.

Time Complexity - contd

- To avoid looking at all pairs, one can proceed as follows.
- Recall that, while selecting (a_i, b_i), we want to maximize $g_i = D_{a_i} + D_{b_i} - 2c_{a_i b_i}$.
- Suppose that we sort the $D-$values in a decreasing order of their magnitudes. Thus, for elements of Block A,

$$D_{a_1} \geq D_{a_2} \geq \cdots \geq D_{a_{(n-i+1)}}$$

- Similarly, for elements of Block B,

$$D_{b_1} \geq D_{b_2} \geq \cdots \geq D_{b_{(n-i+1)}}$$

Time Complexity - contd

- Sorting requires $O(n \log n)$.
- Next, we begin examining D_{a_i} and D_{b_j} pairwise.
- If we come across a pair (D_{a_k}, D_{b_l}) such that $(D_{a_k} + D_{b_l})$ is less than the gain seen so far in this improvement phase, then we do not have to examine any more pairs.
- Hence, if $D_{a_k} + D_{b_l} < g_{ij}$ for some i, j then $g_{kl} < g_{ij}$.
- Since it is almost never required to examine all the pairs (D_{a_i}, D_{b_j}), the overall complexity of selecting a pair (a_i, b_i) is $O(n \log n)$.
- Since n exchange pairs are selected in one pass, the complexity of Step 3 is $O(n^2 \log n)$.

Time Complexity - contd

- Step 5 takes only linear time.
- The complexity of the Kernighan-Lin algorithm is $O(pn^2 \log n)$, where p is the number of iterations of the improvement procedure.
- Experiments on large practical circuits have indicated that p does not increase with n.
- The time complexity of the pair selection step can be improved by scanning the unsorted list of $D-$values and selecting a and b which maximize D_a and D_b. Since this can be done in linear time, the algorithm's time complexity reduces to $O(n^2)$.
- This scheme is suited for sparse matrices where the probability of $c_{ab} > 0$ is small. Of course, this is an approximation of the greedy selection procedure, and may generate a different solution as compared to greedy selection.