The problem formulation begins with the concept of a partition. Given a graph, the goal is to divide the set of vertices into two or more subsets such that each subset is of manageable size. The size of each subset is bounded by certain constraints. The objective is to minimize the interconnections between the subsets, which simplifies the design process by allowing each subsystem to be designed independently, speeding up the design process.

Decomposition scheme has the following attributes:

- Each subsystem can be designed independently, speeding up the design process.
- The design process is efficient and can handle large systems.
- The subsystems are of manageable size.
- Decomposition is carried out hierarchically until each module.

Levels of Partitioning:

- System, Board, Chip.
Kernighan-Lin Algorithm: A Weighted Example

Properties

Kernighan-Lin Algorithm: A Simple Example

- Each edge has a unit weight.
- The cut cost is the sum of the weights of the edges cut by the cut.
- The vertex removal cost is the sum of the weights of the edges incident to the vertex.
- The vertex insertion cost is the sum of the weights of the edges that will be added when the vertex is inserted.

Question: How to compute cost reduction? What pairs to be swapped?

Step # Vertex Pair Cost Reduction Cut Cost

1
2
3
4
5
6
7
8
9
10

This process continues until all the vertices are locked.

- The process converges in any further exchange.
- The vertices are then locked (and thus are prohibited from increasing in cut size).
- Consider the pair which gives the largest decrease of the smallest.
- In the case of multiple exchange, keep the vertices in decreasing order.

Initial cut cost = \( (1+3+2)+(1+3+2)+(1+3+2) = 18 \) (22−4)
Time Complexity

1. Time complexity?
2. Apply the K I partition algorithm for each pair of subsets.
3. Partition the edges into a 2 ranging sets.
4. Apply the K I partition algorithm.
5. Apply the K I partition algorithm.
6. Apply the K I partition algorithm.
7. Apply the K I partition algorithm.
8. Apply the K I partition algorithm.
9. Apply the K I partition algorithm.
10. Apply the K I partition algorithm.
11. Apply the K I partition algorithm.
12. Apply the K I partition algorithm.
13. End

Algorithm: K I partition

Input: G (A, E)
Output: G (A, E)

Algorithm: K I partition

Input: G (A, E)
Output: G (A, E)
**Net-Cut Model**

- Easy modification of the K-L heuristic.
  - Edge weights for an edge connecting cells x and y.

**Net-Cut Graph**

- Should not assign the same weight for all edges.

**Copied with Hypergraph**

- Two conditions a pass is high, O(n).
  - Need to handle multi-terrestrial nets directly.
  - The K-L heuristic can handle hypergraphs.
  - Need dummy vertices to handle the unbalanced problem.
  - The K-L heuristic handles only exact positions.

**Drawbacks of the Kernighan-Lin Heuristic**

- Graph subdivision.
  - Vertices and edges with a high cost increase the size of the tree.
  - Producing a vertex with weight w(a) into a clique with w(b).

- Vertices w(a) might represent block sizes different from blocks.
  - The K-L heuristic handles only vertex weights.

**Note that the overall time complexity remains O(n^2).**

- Since D' + D = 0, the need to compute (\text{cut}(a))
  \begin{cases} 1 & a = b \ \text{if} \ b \\ 0 & a \\ \text{null} \end{cases}

- Compute \text{null}.

- Partition A = \{a \in V_r | a \neq a_r \}.

- Start with \{a \}.

- Sort the \text{values} in non-increasing order.

A "Better" Implementation of K-L Algorithm.
Construction of the two arrays takes $O(d)\cdot t_{\max}$.

Size of the network: $p = \sum_{i=1}^{n} 1 = n$.

<table>
<thead>
<tr>
<th>Net</th>
<th>Net 1, 2, 3, 4, 5, 6</th>
<th>Net 1, 3, 4, 5, 6</th>
<th>Net 1, 2, 3, 4, 5, 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Input Data Structures

Definitions: Cut

Time Complexity: $O(\prod_{i} d(t))$, where $p$ is the total # of terminals.

A special data structure is used to keep track of the cuts that are moved.

- Adjacent to a terminal, a single vertex is moved across the cut in a single move.
- Only a single vertex is moved across the cut in a single move.
- New results to the F-M heuristic.
Pass: Only need O(d) time to maintain all cell gains in one

\[ B(i) = \begin{cases} 1 & \text{if } \text{cell } i \text{ is critical} \\ 0 & \text{otherwise} \end{cases} \]

4 cases: \( G(i) = 0 \) or \( T(i) = 0 \)

Net: A net is critical if its a cell which is moved will change its context.

Critical nets: a net is critical if it has 2 cells that are balanced.

Distribution of nets: \( V(i) \Rightarrow (2, 3) \)

Basic ideas: balance and movement

Cell gains and data structure manipulation

Computing cell gains

Gain of a cell depends only on its critical cells.
Algorithm for Updating Cell Gains

Case 3:

- Consider only the case when the base cell is in the left partition. The other
  - base cell: The cell selected for movement from one set to the other.

To update the gains, we only need to look at those nets connected to the base
cell which are critical before or after the move.
Repeat the whole process until new $G < 0$.

- Since $G = 0$, results in a better balanced move $c_{12} c_{0}$.

Maximum partial sum $G = 2$, $G = 2$.

<table>
<thead>
<tr>
<th>Step</th>
<th>$G$</th>
<th>$G_{1}$</th>
<th>$G_{2}$</th>
<th>$G_{3}$</th>
<th>$G_{4}$</th>
<th>$G_{5}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

**Summary of the Example**

- Total time $= \sum (\text{in})$.
- To update the cell gains, it takes $O(n^3)$ work for Net $I$.
- Updating cell gain from right to left $\Rightarrow$ move $a_1 a_2 a_3 a_4 a_5$.
- Once a net has "locked" cells at both sides, the net will remain updated.

**Complexity of Updating Cell Gains**
Simulated Annealing