Floorplan using Simulated Annealing

Pramod Nataraja
Praveen Kumar C P
Overview

- Normalized Polish Expression
- Problem Formulation
- Algorithm & Implementation
- Results Analysis
- Video
- Conclusion & Possible Improvements
Skewed Slicing Structure

There is a 1-1 correspondence between Slicing Floorplan, Skewed Slicing Tree, and Normalized Polish Expression.

An expression $E=e_1 e_2 \ldots e_{2n-1}$, where each $e_i \in \{1, 2, \ldots, n, H, V\}$, $1 \leq i \leq 2n-1$, is a Polish Expression of length $2n-1$ iff (1) every operand appears exactly once (2) Expression $E$ has balloting property. No of Operands $> \text{No of Operators}$.

A Normalized Polish Expression is one in which there are no consecutive operators of the same type ($H$ or $V$ respectively).
Problem Formulation

- **Objective:**
  - A feasible Floorplan optimizing the desired cost function.

- **Input:**
  - n Blocks with areas $A_1, \ldots, A_n$ and initial x,y co-ordinates.
  - Initial Polish Expression is also provided.

- **To Do**
  - An iterative process to modify the initial Polish expression by making moves and arriving at a final Polish expression that minimizes the cost function using a process that is analogous to annealing.

- **Output:**
  - Coordinates $(x_i, y_i)$ for each block.
Problem Formulation II

- **Cost Function**
  - \( \text{Cost}(F) = \alpha A + \lambda W \)
  - \( A \): area of the smallest rectangle
  - \( W \): overall wiring length
  - \( \lambda \): user-specified parameter (in our case it’s 0)

- **Constraints**
  - Move(M1, M2, M3) are allowed
  - No Move should violate the balloting property.

- **Assumptions/Notes**
  - Aspect ratio bound is not considered.
  - Initially reducing the cost function was the main factor and the time taken for annealing algorithm was ignored
Implementation & Area Computation

- **Usage of STL::Vector**
  - Vector data structure was used to store the Polish Expression, height and width of individual blocks
  - Reduced the overhead of having to manage data with Structs
  - Reduced the overhead of implementation of methods to insert, delete and modify nodes

- **Input Parsing**
  - Input files containing Polish Expression are parsed into 3 Vectors
    - One containing the Nodes and Operators
    - 2 containing width and height of each node
    - H and V are represented by -2 and -3 respectively
Area Computation

- Parse the expression to find the first operator (H or V)
- Combine the operator along with the 2 previous nodes in the expression
- Calculate the height and width of the new block obtained as shown below

- Remove the operator and the previous 2 operands from the Polish Expression and replace them with a single new block obtained
- Insert the height and width of the new block at the end of Height and width vectors of the Polish Expression
- Repeat the process until you obtain a single large node. The area of this node gives the overall area of the floorplan.
Area Computation

Example Polish Expression

2-1-0-H-V-3-V-4-V

2-5-V-3-V-4-V

6-3-V-4-V

7-4-V

8

Width and Height Vectors

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
<td>H</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
<td>H</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

The total area of the floorplan can be calculated from the dimensions of block 8.
**X and Y Co-ordinate computation**

During Area Calculation, maintain 3 more Vectors containing left node, right node and the operator each time 2 nodes are combined into a bigger block.

\[
\begin{align*}
4,2,0,V,H,3,V,1,H \\
4,5,H,3,V,1,H \\
6-3-V-1-H \\
7-1-H \\
8
\end{align*}
\]

On reaching the final Block, we assign the 0,0 as the x and y co-ordinates of the Final block which is formed by combining all blocks in the floorplan.
X and Y Co-ordinate computation

- Iterating backwards, from the Final block, we assign the x and y co-ordinates to each block formed by combining 2 blocks depending on the operator used to combine the 2 blocks to form the bigger block.

- IF the operator is H
  - X co-ordinate of the Second Block = X Co-ordinate of the block
  - Y co-ordinate of the Second Block = Y Co-ordinate of the block
  - X co-ordinate of the First Block = X Co-ordinate of the block
  - Y co-ordinate of the First Block = Y Co-ordinate of the block + Height of Second Block

- IF the operator is V
  - X co-ordinate of the Second Block = X Co-ordinate of the block + width of First Block
  - Y co-ordinate of the Second Block = Y Co-ordinate of the block
  - X co-ordinate of the First Block = X Co-ordinate of the block
  - Y co-ordinate of the First Block = Y Co-ordinate of the block
X and Y Co-ordinate computation

Final Polish Expression: 4,2,0,V,H,3,V,1,H
Simulated Annealing

- Locating a good approximation to the global optimum of a given function in a large search space.

```
Algorithm: Simulated_Annealing_Floorplanning(P, ε, r, k)
begin
  E ← 12V3V4V...nV; /* initial solution */
  Best ← E; \( T_0 \leftarrow \frac{\Delta_{\text{in}}}{\ln(P)} \); M ← MT ← uphill ← 0; N = kn;
repeat
  MT ← uphill ← reject ← 0;
repeat
  SelectMove(M);
  Case M of
  \( M_1 \): Select two adjacent operands \( e_i \) and \( e_j \); \( NE \leftarrow \text{Swap}(E, e_i, e_j) \);
  \( M_2 \): Select a nonzero length chain \( C \); \( NE \leftarrow \text{Complement}(E, C) \);
  \( M_3 \): done ← FALSE;
  while not (done) do
    Select two adjacent operand \( e_i \) and operator \( e_{i+1} \);
    if \((e_{i-1} \neq e_{i+1}) \) and \((2N_{i+1} < i)\) then done ← TRUE;
    \( NE \leftarrow \text{Swap}(E, e_i, e_{i+1}) \);
  MT ← MT + 1; \( \Delta \text{cost} \leftarrow \text{cost}(NE) - \text{cost}(E) \);
  if \( (\Delta \text{cost} \leq 0) \) or \( \text{(Random < } e^{-\frac{\Delta \text{cost}}{kT}}) \) \( \) then
    if \( (\Delta \text{cost} > 0) \) then uphill ← uphill + 1;
    E ← NE;
    if \( \text{cost}(E) < \text{cost}(\text{best}) \) then \text{best} ← E;
  else reject ← reject + 1;
  until \((\text{uphill} > N) \) or \((MT > 2N)\);
  T ← rT; /* reduce temperature */
  until \((\frac{\text{reject}}{MT} > 0.95) \) or \((T < \epsilon) \) or \( \text{OutOfTime} \);
end
```

Figure taken from Prof. Sung Kyu Lim Lecture
<table>
<thead>
<tr>
<th></th>
<th>5_block.ple</th>
<th>10_block.ple</th>
<th>30_block.ple</th>
<th>100_block.ple</th>
<th>150_block.ple</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Area</td>
<td>65</td>
<td>147</td>
<td>1075</td>
<td>7199</td>
<td>14104</td>
</tr>
<tr>
<td>Final Area</td>
<td>40</td>
<td>70</td>
<td>315</td>
<td>1044</td>
<td>1540</td>
</tr>
<tr>
<td>Total Block Area</td>
<td>38</td>
<td>68</td>
<td>290</td>
<td>920</td>
<td>1342</td>
</tr>
<tr>
<td>% White Space</td>
<td>5</td>
<td>2.8</td>
<td>7.9</td>
<td>11.8</td>
<td>12.8</td>
</tr>
</tbody>
</table>
### Tuning Parameters

- The runtime and quality of solution depends on various parameters.
- \( k \): Number of moves allowed at each temperature level.
- \( P \): Initial Probability for deciding the starting temperature.
- \( \epsilon \): Lowest Temperature until which annealing is performed.
- \( r \): Temperature reducing factor.

<table>
<thead>
<tr>
<th>Nodes</th>
<th>( k )</th>
<th>( P )</th>
<th>( r )</th>
<th>( \epsilon )</th>
<th>Avg Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>10</td>
<td>0.85(( T_0=144 ))</td>
<td>0.85</td>
<td>0.1</td>
<td>0.4 sec</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>0.95(( T_0=539 ))</td>
<td>0.95</td>
<td>0.01</td>
<td>4.2 sec</td>
</tr>
<tr>
<td>30</td>
<td>50</td>
<td>0.95(( T_0=1764 ))</td>
<td>0.95</td>
<td>0.00001</td>
<td>153 sec</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>0.95(( T_0=2000 ))</td>
<td>0.85</td>
<td>0.0000001</td>
<td>37.8 min</td>
</tr>
<tr>
<td>150</td>
<td>50</td>
<td>0.9(( T_0=4187 ))</td>
<td>0.95</td>
<td>0.00000001</td>
<td>61.6 min</td>
</tr>
</tbody>
</table>

### Tradeoff between quality and runtime (100 Nodes)

<table>
<thead>
<tr>
<th>( K )</th>
<th>( P )</th>
<th>( R )</th>
<th>( \epsilon )</th>
<th>Final Area</th>
<th>Avg Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.9</td>
<td>0.85</td>
<td>0.0001</td>
<td>1462</td>
<td>322 sec</td>
</tr>
<tr>
<td>50</td>
<td>0.9</td>
<td>0.95</td>
<td>0.00000001</td>
<td>1100</td>
<td>7998 sec</td>
</tr>
</tbody>
</table>
Transformation of 5 Blocks
Transformation of 30 Blocks
100 Block Video
150 Block Video
Allowing rotation of Block (M4 move)

Annealing Results

<table>
<thead>
<tr>
<th></th>
<th>5_block</th>
<th>10_block</th>
<th>30_block</th>
<th>100_block</th>
<th>150_block</th>
<th>150_block</th>
</tr>
</thead>
<tbody>
<tr>
<td>Final Area (M1,M2,M3)</td>
<td>40</td>
<td>70</td>
<td>315</td>
<td>1044</td>
<td>1540</td>
<td></td>
</tr>
<tr>
<td>Final Area (M1,M2,M3,M4)</td>
<td>40</td>
<td>70</td>
<td>312</td>
<td>1020</td>
<td>1537</td>
<td></td>
</tr>
<tr>
<td>Block Area</td>
<td>38</td>
<td>68</td>
<td>290</td>
<td>920</td>
<td>1342</td>
<td></td>
</tr>
</tbody>
</table>
Conclusion

- Simulated Annealing is an effective method for floorplanning as this method helps us explore more solution space and increase the chance of finding good quality solution.
- There is a tradeoff between quality of solution and execution time.
- The execution time depends on the tuning parameters, so arriving at optimal set of values these parameters is time consuming.

Possible Improvements

- Aspect Ratio may also be considered as another factor for the cost function.
- Wirelength also can be added to the cost function to minimize the wirelength as well.
- An efficient method to compute the tuning parameters needs to be devised.