Quadratic and Linear WL Placement
Using Quadratic Programming: Gordian & Gordian-L

Shantanu Dutt
ECE Dept., Univ. of Illinois at Chicago

Acknowledgements: Adapted from the slides
“Gordian Placement Tool: Quadratic and Linear Problem Formulation “
by
Ryan Speelman
Jason Gordon
Steven Butt
UCLA, EE 201A 5-6-04
Papers Covered


Quadratic Problem Formulation

- Find approximate positions for blocks (*global placement*)
- Try to minimize the sum of squared wire length.
- Sum of squared wire length is quadratic in the cell coordinates.
- The global optimization problem is formulated as a quadratic program.
- It can be proved that the quadratic program is convex, and as such, can be solved in polynomial time
Quadratic Problem Formulation

Let \((x_i, y_i)\) = Coordinates of the center of cell \(i\)
\(w_{ij}\) = Weight of the net between cell \(i\) and cell \(j\)
\(x, y\) = Solution vectors

Cost of the net between cell \(i\) and cell \(j\)
\[\frac{1}{2}w_{ij} \left( (x_i - x_j)^2 + (y_i - y_j)^2 \right)\]

Total cost = \[\frac{1}{2}x^T Q x + d_x^T x + \frac{1}{2}y^T Q y + d_y^T y + \text{const}\]

Constants and linear components in the total cost equation are derived from information about chip constraints such as fixed modules and I/O pins.
Quadratic Problem Formulation

- Look closer at the one-dimensional problem
  - Cost = \( \frac{1}{2} x^T C x + d^T x \)

- At the \( i_{th} \) level of optimization, the placement area is divided up into at most \( q \leq 2^i \) regions

- The centers of these regions impose constraints on the global placement of the modules
  - \( A^{(i)} x = u^{(i)} \)

- The entries of the matrix \( A \) are all 0 except for one nonzero entry [e.g., \( 1/[\text{no. of cells in that region}] \)] corresponding to the region that a given module belongs to, and \( u_j \) is the center coordinate of the \( j^{th} \) region.
Quadratic Problem Formulation

• Combine the objective function and the linear constraints to obtain the linearly constrained quadratic programming problem (LQP)

\[
\text{LQP: } \min_{x \in \mathbb{R}^m} \left\{ \phi(x) = \frac{1}{2} x^T C x + d^T x \mid A^{(i)} x = u^{(i)} \right\}.
\]

• Since the terms of this function define a convex subspace of the solution space, it has a unique global minimum \((x^*)\)
Partitioning

- Gordian does not use partitioning to reduce the problem size, but to restrict the freedom of movement of the modules.
- Decisions in the partitioning steps place modules close to their final positions, so good partitioning is crucial.
- Decisions are made based on global placement constraints, but also need to take into account the number of nets crossing the new cut line.

\[ \frac{F_{p'}}{F_p} = \frac{\sum_{\mu \in \mathcal{M}_p} F_{\mu}}{\sum_{\mu \in \mathcal{M}_p} F_{\mu}} = \alpha \]

\[ c_n(\alpha) = \sum_{\nu \in \mathcal{M}_c} w_{\nu} \]

Fp, Fp’ are new partition areas. Alpha is the area ratio, usually 0.5.

Cp is the sum of the weights of the nets that cross the partition.
Improving Partitioning

• Variation of cut direction and position
  – Going through a sorted list of module coordinates, you can calculate Cp for every value of $\alpha$ by drawing the partition line after each module in sequence

• Module Interchange
  – Take a small set of modules in the partition and apply a min-cut approach

• Repartitioning
  – In the beginning steps of global optimization, modules are usually clustered around the centers of their regions
  – If regions are cut near the center, placing a module on either side of the region could be fairly arbitrary
  – Apply a heuristic, if two modules overlap near a cut then they are merged into one of the regions
Final Placement

• A final placement is the last, but possibly most important, step in the GORDIAN Algorithm.

• After the main body of the GORDIAN algorithm finishes, which is the alternating global optimization and partitioning steps, each of the blocks containing k or less modules needs to be optimized.

• For the Standard Cell Design the modules are collected in rows, for the macro-cell design an area optimization is performed, packing the modules in a compact slicing structure.
Standard Cell Final Placement

- In Standard Cell Designs the Modules are approximately the same height but can vary drastically in width.
- The region area is determined by the widths of the channels between the rows and by the lengths of the rows.
- The goal is to obtain narrow widths between rows by having equally distributed low wiring density and rows with equal length.
- To create rows of about equal length is necessary to have a low area design. This is done by estimating the number of feed-throughs in each row and making rows with large feed-throughs shorter than average to allow for the feed-through blocks that will be needed. In the end the row lengths should not vary from the average by more than 1-5%.
- A final row length optimization is created by interchanging select modules in nearby rows who have y-coordinates close to the cut-line.
Linear or Quadratic Objective Function?

- Gordian used a quadratic objective function as the cost function in the global optimization step

- Is a linear objective function better?

- What are the tradeoffs for each?

- What are the results of using a linear objective function compared with using a quadratic one?
Comparison of Linear and Quadratic Objective Function

**Quadratic objective function**

\[
\text{Min } \sum_{n_i} (l_{av} + \delta_i)^2
\]

- +ve and –ve deviations add up
- Thus the above formulation also minimizes the deviations \(\delta_i\) (in addition to \(l_{av}\))

**Linear objective function**

\[
\text{Min } \sum_{n_i} (l_{av} + \delta_i)
\]

- +ve and –ve deviations cancel each other
- Thus the above formulation only minimizes \(l_{av}\)
Comparison of Linear and Quadratic Objective Function

- Minimization of the quadratic objective function tends to make the average nets longer but w/ smaller variation in net lengths

- Minimization of the linear objective function results in shorter average nets overall but w/ larger variation in net lengths
Comparison cont’d

- Quadratic objective function leads to more routing in this standard cell circuit example

- This observation is the motivation to explore linear objective functions in further detail for placement
GordianL

- Retains the basic strategy of the Gordian algorithm by alternating global placement and partitioning steps

- Modifications include the objective function for global placement and the partitioning strategy
  - Linear objective function
  - Iterative partitioning
Model for the Linear Objective Function

- All modules connected by net \( v \) are in the set \( M_v \)
- The pin coordinates are \( x_{\mu v} = x_\mu + \xi_{\mu v} \)
- The module center coordinates are \( x_\mu \)
  with the relative pin coordinates being \( \xi_{\mu v} \)

- The coordinates of the net nodes are always in the center of their connected pins, meaning \( x_\nu = \frac{1}{|M_\nu|} \sum_{\mu \in M_\nu} x_\mu \)
Linear Objective Function

\[ \Phi_q = \sum_{\nu \in N} \sum_{\mu \in M_\nu} (x_{\mu \nu} - x_\nu)^2 \quad \text{Quadratic objective function} \]

\[ \Phi_l = \sum_{\nu \in N} \sum_{\mu \in M_\nu} |x_{\mu \nu} - x_\nu| \quad \text{Linear objective function} \]

- Quadratic objective functions have been used in the past because they are continuously differentiable and therefore easy to minimize by solving a linear equation system.

- Linear objective functions have been minimized by linear programming with a large number of constraints.
- This is much more expensive in terms of computation time.
- An adjustment to the function needs to be made.
Quadratic Programming for the Linear Objective Function

- We can rewrite the objective function as:

$$\Phi_1 = \sum_{\nu \in \mathcal{N}} \sum_{\mu \in \mathcal{M}_\nu} \frac{(x_{\mu \nu} - x_\nu)^2}{|x_{\mu \nu} - x_\nu|} = \sum_{\nu \in \mathcal{N}} \sum_{\mu \in \mathcal{M}_\nu} \frac{(x_{\mu \nu} - x_\nu)^2}{g_{\mu \nu}}$$

with $g_{\mu \nu} = |x_{\mu \nu} - x_\nu|$

- The above is iterated $k$ times until $|g_{\mu \nu}^k - g_{\mu \nu}^{k-1}| < \varepsilon$

- Thus in the $k$’th iteration we are solving:

$$\Phi_1 = \sum_{\nu \in \mathcal{N}} \sum_{\mu \in \mathcal{M}_\nu} \frac{(x_{\mu \nu} - x_\nu)^2}{|x_{\mu \nu} - x_\nu|_{k-1}} \approx \sum_{\nu \in \mathcal{N}} \sum_{\mu \in \mathcal{M}_\nu} \frac{(x_{\mu \nu} - x_\nu)^2}{|x_{\mu \nu} - x_\nu|_k} \approx \sum_{\nu \in \mathcal{N}} \sum_{\mu \in \mathcal{M}_\nu} |x_{\mu \nu} - x_\nu|_k$$
Quadratic Programming for the Linear Objective Function

- Through experimentation the area after final routing is better if the factor $g_{\mu\nu}$ is replaced by a net specific factor

$$g_\nu = \sum_{\mu \in M_\nu} |x_{\mu\nu} - x_\nu|.$$
Quadratic Programming cont’d

-The advantages of this approach are:
1. The summation reduces the influence of nets with many connected modules and emphasizes the majority of nets connecting only two or three modules.
2. The force on modules close to the net node is reduced since: 
   \[ \frac{1}{g_{\nu}} \ll \frac{1}{g_{\mu \nu}} \]
   this helps in optimizing a WL metric close to HPBB in which only the coordinates of boundary cells of the BB (those that are far from the “centroid” or “net node” coordinates)

\[ \Phi_i^{(k)} = \sum_{\nu \in N} \sum_{\mu \in M_\nu} \frac{1}{g_{\nu}} (x_{\mu \nu} - x_{\nu})^2 \]

- To solve the problem an iterative solution method is constructed with iteration count \( k \) for the modified objective

The quadratic programming problem can now be solved by a conjugate gradient method with preconditioning by incomplete Cholesky factorization

```
procedure global placement (GP) 
  k = 0;
  for each \( \nu \in N \) 
    \( g_{\nu}^{(k)} = 1; \)
  endfor 
  do 
    \( \Phi_i^{(k)} \rightarrow \text{min}; \)
    \( k = k + 1; \)
    for each \( \nu \in N \) 
      \( g_{\nu}^{(k)} = \max(w_0; \sum_{\mu \in M_\nu} |x_{\mu \nu} - x_{\nu}|); \)
    endfor 
    while \( \sum_{\nu \in N} |g_{\nu}^{(k)} - g_{\nu}^{(k-1)}| > \epsilon; \)
  enddo
```

Figure 4: Global placement with linear objective
Problems: No cell non-overlap Constraints

Quadratic objective function  Linear objective function

a) Global placements with 1 region
Iterative Partitioning: Cell Overlap Removal

- Modules in a region are bipartitioned iteratively instead of in one step.

- Module set $\mathcal{M}_\rho$ is partitioned into $\mathcal{M}_\rho'$ and $\mathcal{M}_\rho''$ such that
  
  $x_{\mu'} \leq x_{\mu''}$ for $\mu' \in \mathcal{M}_\rho', \mu'' \in \mathcal{M}_\rho''$.

  and $\sum_{\mu' \in \mathcal{M}_\rho'} f_{\mu'} \approx \sum_{\mu'' \in \mathcal{M}_\rho''} f_{\mu''}$. ($f_{\mu}$ is module $\mu$'s area)

- Also, to distribute the modules better over the whole placement area, positioning constraints fix the center of gravity (CG) of modules in the set $\mathcal{M}_\rho'$ ($\mathcal{M}_\rho''$) on the center coordinate $x_{\rho'}$ ($x_{\rho''}$) of the region $\rho'$ ($\rho''$), i.e. $\sum_{\mu' \in \mathcal{M}_\rho'} x_{\mu'} f_{\mu'} = x_{\rho'} \sum_{\mu' \in \mathcal{M}_\rho'} f_{\mu'}$.

  \[
  \text{Sum} (x_i)/k = \text{centroid of the cells} = x_r \text{ (centroid of region)}
  \]
Iterative Partitioning (cont’d)

- The modified iterative partitioning forces the modules more and more away from the center of the region.

The first iteration partitions the set $M_\rho$ into three subsets $M_1^i = \{A, B\}$, $M_\rho^1 = \{C, D, E, F\}$, $M_r^i = \{G, H\}$ according to the module coordinates $x_\mu$.

The second iteration step partitions the set $M_\rho^i$ into the sets

$$M_1^2 = \{C\}, M_\rho^2 = \{E, D\}, M_r^2 = \{F\}.$$

- The iterative process finishes when the set $M_\rho^i$ becomes empty.
- In each iteration $i$, $\Phi_i^{(k)}$ of the entire ckt is opt. using the GP algorithm w/ the CG constraints for each region.
- The number of modules assigned to the sets $M_1^i$ and $M_r^i$ is determined by the area constraint:

$$\sum_{\mu \in M_1^i} f_\mu \approx \sum_{\mu \in M_\rho^i} f_\mu \leq \delta \cdot \sum_{\mu \in M_\rho} f_\mu, \ 0 < \delta \leq 0.5$$

Finally, the bipartitioning of the set $M_\rho$ is obtained by

$$M_\rho' = \bigcup M_1^i \text{ and } M_\rho'' = \bigcup M_r^i.$$

- Presumably, a similar partitioning process along the y dim. follows (not explicitly mentioned in paper).
Iterative Partitioning (cont’d)

• Cell overlaps mostly removed after the iterative partitioning process (an optimization step occurs in each partitioning step).
• Remaining overlaps are small and can be removed in a detailed placement process (e.g., [Dutt et al., ICCAD’06])
Figure 7: Placement refinement with quadratic and linear objective function
Results

Figure 6: Sum of wire lengths versus #pins

<table>
<thead>
<tr>
<th>circuit</th>
<th>#modules</th>
<th>#nets</th>
<th>GORDIAN</th>
<th>GORDIANL</th>
<th>VNR (cplrt)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>area</td>
<td>cpu</td>
<td>area</td>
</tr>
<tr>
<td>primary1</td>
<td>752</td>
<td>904</td>
<td>23.4</td>
<td>40</td>
<td>22.7</td>
</tr>
<tr>
<td>struct</td>
<td>1888</td>
<td>1920</td>
<td>9.2</td>
<td>113</td>
<td>6.7</td>
</tr>
<tr>
<td>primary2</td>
<td>2907</td>
<td>3028</td>
<td>97.3</td>
<td>260</td>
<td>85.5</td>
</tr>
<tr>
<td>biomed</td>
<td>6417</td>
<td>5742</td>
<td>62.2</td>
<td>804</td>
<td>50.5</td>
</tr>
<tr>
<td>c1355</td>
<td>554</td>
<td>595</td>
<td>2.8</td>
<td>20</td>
<td>2.2</td>
</tr>
<tr>
<td>c5313</td>
<td>2330</td>
<td>2508</td>
<td>18.1</td>
<td>162</td>
<td>15.4</td>
</tr>
<tr>
<td>s9234</td>
<td>5597</td>
<td>5844</td>
<td>46.6</td>
<td>250</td>
<td>38.0</td>
</tr>
<tr>
<td>20x20</td>
<td>6419</td>
<td>6464</td>
<td>16.5</td>
<td>839</td>
<td>15.1</td>
</tr>
<tr>
<td>AVQ</td>
<td>21046</td>
<td>21316</td>
<td>5067</td>
<td>86.9</td>
<td>22989</td>
</tr>
</tbody>
</table>

Table 1: Results (area in $mm^2$, cpu-time in seconds on VAX 8650)
Conclusion

- Gordian algorithm does well with large amounts of modules
  - Global optimization combined with partitioning schemes

- The choice of the objective function is crucial to an analytical placement method.

- GordianL yields area improvements of up to 20% after final routing.

- The main reason for this improvement was the length reduction of nets connecting only two and three pins.