Contents

- Min-cut
  - Optimal Local Placement
- Gordian
  - QP
  - Centre of Mass
- NTUPlace3
  - QP
  - Density Mitigation
- Timberwolf
  - SA
- Kraftwerk2
- ZSA
- Timing-Driven Placement
  - Timingdragon
Placement Algorithms

- **Combinatorial methods**
  - Min-cut placement
  - Cluster growth

- **Nondeterministic methods:**
  - Simulated annealing (SA)
  - Genetic algorithm (GA)

- **Analytical methods:**
  - Force-directed placement
  - Quadratic placement (QP)
  - Non-quadratic placement

- **Mixed-size placement**
Placement Problem

- **Placement Problem**: assign cells to positions on the chip, such that no two cells overlap with each other (legalization) and some cost function (e.g., wirelength) is optimized
  - A major step in physical design that has been studied for 40+ years.
Modern Placement Challenges

- **High complexity**
  - Millions of objects to be placed

- **Placement constraints**
  - Preplaced blocks
  - Chip density, etc.

- **Mixed-size placement**
  - Hundreds/thousands of large macros with millions of small standard cells

- **Datapath placement**
  - Regular structure, functional stage alignment
  - Many more
Basic Wirelength Models

- **Half-perimeter wirelength (HPWL):** Half the perimeter of the bounding rectangle that encloses all the pins of the net to be connected
  - Most widely used approximation!
- **Squared Euclidean distance:** Squares of all pairwise terminal distances in a net using a quadratic cost function
  \[
  \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_{ij} [(x_i - x_j)^2 + (y_i - y_j)^2]
  \]
- **Steiner-tree approximation:** Computationally expensive
- **Minimum spanning tree:** Good approximation to Steiner trees
Wirelength Estimation Examples

- Half perimeter wirelength = 11
- Squared Euclidean distance = 15.1 (W = 0.1)
- Min Steiner tree wirelength = 12
- Min spanning tree wirelength = 13
Quadratic Wirelength vs. Linear Wirelength

- HPWL is convex, but not differentiable
- Quadratic wirelength is a convex, differentiable function
  - easier/faster to minimize
- Both models correlate with each other “to some degree”
  - (might incur large errors if two cells are far away)
Placement Algorithm Paradigms

- The placement problem is NP-complete.
- Popular placement algorithms:
  - **Constructive algorithms**: once the position of a cell is fixed, it is not modified anymore
    - Cluster growth, min cut, QP, etc.
  - **Iterative algorithms**: intermediate placements are modified in an attempt to improve the cost function.
    - Force-directed method, nonlinear placement, etc
  - **Nondeterministic approaches**: simulated annealing, genetic algorithm, etc.

- May combine multiple elements:
  - Constructive algorithms are used to obtain an initial placement
  - The initial placement is followed by iterative improvement
  - The results can further be improved by simulated annealing
Greedy method: Selects unplaced components and places them in available slots.

SELECT: Choose the unplaced component that is most strongly connected to all of the placed components (or most strongly connected to any single placed component)

PLACE: Place the selected component at a slot such that a certain “cost” of the partial placement is minimized
The module located closest to the rectilinear polygon is selected as seed. Module cluster boundary, tiles are labelled “2”. Outside tiles tagged as “1”. Order criterion: minimum cell distance from deposition site.
Cluster Growth Example

- # of other terminals connected: \( ca=3, cb=1, cc=1, cd=1, ce=4, cf=3, \)
  and \( cg=3; \) e has the most connectivity.
- Place e in the center, slot 4. a, b, g are connected to e, and \( cae=2, cbe=ceg=1; \) Place a next to e (say, slot 3).
- Continue until all cells are placed
- Further improve the placement by swapping gates
Partitioning-based Placement

- Rely on a good partitioner (e.g., hMetis) and partitionmetric (e.g., exact netweight modeling)

  Pros
  - Is very fast
  - Has very good scalability for handling large-scale designs

  Cons
  - Relatively harder to handle design with large white spaces
Top-Down Min-Cut Placement

- **Quadrature**: suitable for circuits with high density in the center
- **Bisection**: good for standard-cell placement
- **Slice/Bisection**: good for cells with high interconnection on the periphery

![Diagram of placement methods]
Min-Cut Placement Algorithm

- **Algorithm:** Min_Cut_Placement\((N, n, C)\)

  /* \(N\): the layout surface */
  /* \(n\): # of cells to be placed */
  /* \(n0\): # of cells in a slot */
  /* \(C\): the connectivity matrix */

  1 begin
  2 if \((n \leq n0)\) then PlaceCells\((N, n, C)\)
  3 else
  4 \((N1, N2) = \text{CutSurface}(N);\)
  5 \((n1, C1), (n2, C2) = \text{Partition}(n, C);\)
  6 Call Min_Cut_Placement\((N1, n1, C1)\); \(\)
  7 Call Min_Cut_Placement\((N2, n2, C2)\);
  8 end
Quadrature Placement Example

- Apply the F-M or K-L heuristic to partition + Quadrature Placement:
  - Cost $C_1 = 4$, $C_2L = C_2R = 2$, etc.
Min-Cut Placement with Terminal Propagation

- Drawback of the original min-cut placement: Does not consider the positions of terminal pins that enter a region.
- What happens if we swap \{1, 3, 6, 9\} and \{2, 4, 5, 7\} in the previous example?
Terminal Propagation

- Should use the fact that $s$ is in $L_1$!

- The minimum WL solution is selected to choose which of the two partitions go into $R_1$ (up) and $R_2$ (down)

The minimum WL solution is selected to choose which of the two partitions go into $R_1$ (up) and $R_2$ (down)
Partitioning must be done breadth-first, not depth first
Terminal Propagation in more Detail

- Assume that left half of the chip \( L \) is already partitioned into \( \{P_c, P_d\} \), \( y \in P_c \) is located at the center of \( P_c \)
- Now partitioning the right half \( R \) into \( \{P_a, P_b\} \)
- For each gate \( x \in R \) connected to another gate \( y \), outside \( R \), check to see if \( y \) is too closely located to the cutline
  - inside the “window”; two parallel lines to the cutline
  - If \( y \) is outside the window, “propagate” \( y \) to \( p \)
    - where \( p \) is point on the boundary of \( R \)
  - \( p \) becomes the “propagated terminal” for \( y \)
  - We then connect \( x \) and \( p \) and ignore the \( x-y \) connection
  - During the partitioning refinement, \( p \) is fixed at this initial location, and acts like an “anchor” that pulls \( x \) into \( P_a \) that \( p \) is locked in
  - This is due to the fact that if \( x \) is not partitioned together with \( p \), the \( x-p \) connection increases the cutsize by one
Min-Cut Placement Example

- Perform quadrature mincut onto $4 \times 4$ grid
- Start with vertical cut first

\[
\begin{align*}
\mathcal{N}_1 &= \{e, f\} \\
\mathcal{N}_2 &= \{a, e, i\} \\
\mathcal{N}_3 &= \{b, f, g\} \\
\mathcal{N}_4 &= \{c, g, l\} \\
\mathcal{N}_5 &= \{d, l, h\} \\
\mathcal{N}_6 &= \{e, i, j\} \\
\mathcal{N}_7 &= \{f, j\} \\
\mathcal{N}_8 &= \{g, j, k\} \\
\mathcal{N}_9 &= \{l, o, p\} \\
\mathcal{N}_{10} &= \{h, p\} \\
\mathcal{N}_{11} &= \{i, m\} \\
\mathcal{N}_{12} &= \{j, m, n\} \\
\mathcal{N}_{13} &= \{k, n, o\}
\end{align*}
\]

undirected graph model w/ k-clique weighting
thin edges = weight 0.5, thick edges = weight 1
Min-Cut Placement Example – Cuts 1, 2

- First cut has min-cutsize of 3 (not unique)
- Both cuts 1 and 2 divide the entire chip

(a) cut 1

(b) cut 2, 1st-level quadrants formed
Min-Cut Placement Example – Cuts 3, 4

- Each cut minimizes cutsize
  - Helps reduce overall wirelength

(c) cut 3
(d) cut 4
we compute the half-perimeter of the bounding box of the nets as follows: $w(n1) = 1$, $w(n2) = 2$, $w(n3) = 2$, $w(n4) = 2$, $w(n5) = 2$, $w(n6) = 3$, $w(n7) = 1$, $w(n8) = 3$, $w(n9) = 3$, $w(n10) = 1$, $w(n11) = 1$, $w(n12) = 2$, $w(n13) = 4$. Thus, the total wirelength cost is 27.
Recursive Bisection Example with Terminal Propagation – Cuts 1 and 2

- **Start with vertical cut**
  - Perform terminal propagation with middle third window

(a) cut 1  
(b) cut 2
Recursive Bisection Example with Terminal Propagation – Cut 3

- **Two terminals are propagated and are “pulling” nodes**
  - Node $k$ and $o$ connect to $n$ and $j$: $p_1$ propagated (outside window)
  - Node $g$ connect to $j$, $f$ and $b$: $p_2$ propagated (outside window)
  - Terminal $p_1$ pulls $k/o/g$ to top partition, and $p_2$ pulls $g$ to bottom
Recursive Bisection Example with Terminal Propagation – Cut 4

- **One terminal propagated**
  - Node $n$ and $j$ connect to $o/k/g$: $p_1$ propagated
  - Node $i$ and $j$ connect to $e/f/a$: no propagation (inside window)
  - Terminal $p_1$ pulls $n$ and $j$ to right partition
Recursive Bisection Example with Terminal Propagation – Cut 5

- **Three terminals propagated**
  - Node $i$ propagated to $p_1$, $j$ to $p_2$, and $g$ to $p_3$
  - Terminal $p_1$ pulls $e$ and $a$ to left partition
  - Terminal $p_2$ and $p_3$ pull $f/b/e$ to right partition
Recursive Bisection Example with Terminal Propagation – Cut 6

- One terminal propagated
  - Node $n$ and $j$ are propagated to $p_1$
  - Terminal $p_1$ pulls $o$ and $k$ to left partition

[Diagram showing the propagation process]
Recursive Bisection Example with Terminal Propagation – Cut 7

- Three terminals propagated
  - Node j/lb propagated to \( p_1 \), o/k to \( p_2 \), and h/p to \( p_3 \)
  - Terminal \( p_1 \) and \( p_2 \) pull g and l to left partition
  - Terminal \( p_3 \) pull l and d to right partition
Recursive Bisection Example with Terminal Propagation – Cuts 8 to 15

- 16 partitions generated by 15 cuts
  - HPBB wirelength = 23

![Diagram showing the bisection example with 16 partitions and 15 cuts]

we compute the half-perimeter of the bounding box of the nets as follows: $w(n_1) = 1$, $w(n_2) = 2$, $w(n_3) = 2$, $w(n_4) = 2$, $w(n_5) = 2$, $w(n_6) = 2$, $w(n_7) = 1$, $w(n_8) = 2$, $w(n_9) = 3$, $w(n_{10}) = 1$, $w(n_{11}) = 1$, $w(n_{12}) = 2$, $w(n_{13}) = 2$. Thus, the total wirelength cost is 23.
Min-Cut Strategies Comparison

- **Quadrature vs recursive bisection + terminal propagation**
  - Number of cuts: 6 vs 15
  - Wirelength: 27 vs 23
Capo Placer Overview

- Pure recursive bi-partitioning placer
  - Multi-level FM for instances with > 200 cells
  - Flat FM for instances with 35-200 cells
  - Branch-and-bound for instances with < 35 cells
- Careful handling of partitioning tolerance
  - Uncorking: Prevent large cells from blocking smaller cells to move
  - Repartitioning: Several FM calls with decreasing tolerance
  - Block splitting heuristics: Higher tolerance for vertical cut
  - Hierarchical tolerance computation: Instance with more whitespace can have a bigger partitioning tolerance
- Implementation with standard interfaces
  - LEF/DEF and GSRC

Partitioning Tolerance = size ratio

Floorplanning uses Parquet tool – takes place bottom-up, based on failure

Terminal propagation during the floorplanning of a placement bin. The shaded placement bin is being floorplanned because of large/many macros. Dotted lines depict the external connections to objects inside the bin, being propagated as terminals to the placement bin boundaries. An inessential net for floorplanning is also shown. The shown 3-pin inessential net has no consequence on the floorplanning decision for HPWL minimization and is removed when forming the floorplanning problem.

If the bounding box of any multi-pin net covers the bounding box of the entire bin, then this net will have no consequence on the HPWL minimization during floorplanning. Such a net is treated as inessential and ignored when forming the floorplanning problem.

Clustering is based on Hmetis
Placement by Simulated Annealing – TW6

- iTools: http://www.twolf.com
- TimberWolf 6.0 – allows overlaps
- TimberWolf: Stage 1
  - Modules are moved between different rows, as well as within the same row
  - Modules overlaps are allowed
  - When the temperature is reached below a certain value, stage 2 begins
- TimberWolf: Stage 2
  - Remove overlaps
  - Annealing process continues, but only interchanges adjacent modules within the same row
Timberwolf 6.0 Solution Space and Neighborhood Structure

- **Solution Space**: All possible arrangements of the modules into rows, possibly with overlaps.
- **Neighborhood Structure**: 3 types of moves
  - $M_1$: Displace a module to a new location.
  - $M_2$: Interchange two modules.
  - $M_3$: Change the orientation of a module.
Timberwolf 6.0 Solution Space and Neighborhood Structure

- TimberWolf first tries to select a move between $M_1$ and $M_2$: $\text{Prob}(M_1) = 0.8$, $\text{Prob}(M_2) = 0.2$.
- If a move of type $M_1$ is chosen and it is rejected, then a move of type $M_3$ for the same module will be chosen with probability 0.1.
- Restrictions: (1) what row for a module can be displaced? (2) what pairs of modules can be interchanged?

**Key: Range Limiter**

- At the beginning, $(W_T, H_T)$ is big enough to contain the whole chip.
- Window size shrinks as the temperature decreases. Height, width are proportional to $\log(T)$.
- Stage 2 begins when window size is so small that no inter-row module interchanges are possible.
Timberwolf 6.0 Cost Function

- **Cost Function** $C = C_1 + C_2 + C_3$

  $C_1 = \sum_{i \in \text{Nets}} (\alpha_i w_i + \beta_i h_i)$  
  $C_2 = \gamma \sum_{i \neq j} O_{ij}^2$  
  $C_3 = \delta \sum_{r \in \text{rows}} |L_r - D_r|$

  - $C_1$: total estimated wirelength
    - $\alpha_i, \beta_i$ are horizontal and vertical weights respectively
    - For $\alpha_i = 1, \beta_i = 1 \Rightarrow (1/2) \times \text{perimeter of BB of Net I}$
    - **Critical nets**: increase $\alpha_i, \beta_i$
    - If vertical wirings are “cheaper”, use smaller vertical weights

  - **C2: penalty function for module overlaps**
    - where $\gamma$ is penalty weight, and $O_{ij}$ is the amount of overlaps in x-dimension between modules $i$ and $j$

  - **C3: penalty function controlling row length**
    - where $\delta$ is the penalty weight, $D_r$ is the desired row length and $L_r$ is the sum of the widths of modules in row $r$
Timberwolf 6.0 Annealing Schedule

- $T_k = r_k T_{k-1}$
- $r_k$ increases from 0.8 to a maximum value of 0.94, and then decreases to 0.8
- At each $T$, a total number of $nP$ attempts are made
  - Where $n$ is the number of standard-cells
  - $P$ is a user specified constraint
- Termination Condition
  - $T < 0.1$
TimberWolf 7.0

- Cost Function $C = C_1 + C_s$
  - $C_1$: total estimated wirelength due to move,
  - $C_s$: additional wirelength due to inter-row cell shifting
    - Number of cells shifted per row are minimized
      - move to the left or to the right accordingly
  - Hierarchical placement approach is used
    - Netlist is clustered twice in a recursive fashion
    - The top-level (= second-level) clusters are first placed during the high annealing temperature region
    - These clusters are then decomposed to reveal the first-level clusters
    - The placement among the first-level clusters is refined during the mid annealing temperature region
    - Lastly, the first-level clusters are decomposed to reveal the original gate-level netlist
    - TimberWolf 7.0 refines this gate-level placement during the low annealing temperature region

$C_1$, $C_s$ are really Deltas here = DWL, DWs
$C_s$ is estimated to reduce computational complexity
Shift amount is + for right, - for left
Cost Function $C = C_1 + C_s$

- $C_1$: total estimated wirelength due to move,
- $C_s$: additional wirelength due to inter-row cell shifting

  - Number of cells shifted per row are minimized
  - move to the left or to the right accordingly

$C_s$ is estimated: for cell $z$ to be shifted:

- **Model A:** for each net incident to $z$, we find two “break points” $a$ and $b$ (-z), which defines range $[a, b]$, where the wirelength of the net
  - (1) does not change if $z$ is located within the range, (2) increases if $z$ is located to the right of $b$, and (3) decreases if $z$ is located to the left of $a$;
  - the superposition of break points estimates WL change

  \[\text{gradient}(z) = \sum_{i \in N_z} D_i(0)\]

- **Model B:** we first compute the “gradient” of $z$ as:
  - where $N_z$ denotes the nets incident to $z$, $D_i(0)$ denotes the “rate of wirelength change of net $i$ measured at the origin”;
  - If $z$ is at the left boundary of the bounding box of net $i$, $D_i(0) = -1$, if at the right boundary, $D_i(0) = 1$, else $D_i(0) = 0$
  - Once the gradient of all cells to be shifted is computed, $C_s$ is estimated as follows:

  \[C_s = \sum_{j \text{shifted cell}} \text{gradient}(j) \cdot \text{shift}_\text{amount}(j)\]

$C_1$, $C_s$ are really Deltas here = DWL, DWs

$C_s$ is estimated to reduce computational complexity

Shift amount is + for right, - for left
TimberWolf Placement Example

- Perform Timber-Wolf placement
- Based on the given standard cell placement
- Initial HPBB wirelength = $x$

\[
\begin{align*}
n_1 &= \{a, c, g\} \\
n_2 &= \{f, o\} \\
n_3 &= \{b, c, k, n\} \\
n_4 &= \{d, h, i\} \\
n_5 &= \{j, l, m\} \\
n_6 &= \{d, k, j\} \\
n_7 &= \{c, e, f, h, n\} \\
n_8 &= \{d, l\} \\
n_9 &= \{b, g, i, m\} \\
n_{10} &= \{a, k, o\}
\end{align*}
\]

\[
\begin{align*}
n_1 &= 12+7 = 19, \\
n_2 &= 7+7 = 14, \\
n_3 &= 12+7 = 19, \\
n_4 &= 5+14 = 19, \\
n_5 &= 4+7 = 11, \\
n_6 &= 7+14 = 21, \\
n_7 &= 14+14 = 28, \\
n_8 &= 5+7 = 12, \\
n_9 &= 12+7 = 19, \\
n_{10} &= 9+14 = 23.
\end{align*}
\]
TimberWolf Placement Example – Swap (b, e)

- **Swap node b and e**
- We shift node h: on the shorter side of the receiving row
- Node b included in nets \{n_3, n_9\}, and e in \{n_1, n_7\}
TimberWolf Placement Example – Swap (b, e)

- \( \Delta W = \text{wirelength change from swap} \)

Let \( w(x) \) and \( w'(x) \) respectively denote the wirelength before and after the swap. Then,

\[
\begin{align*}
\Delta(n_3) &= w'(n_3) - w(n_3) = 24 - 19 = 5 \\
\Delta(n_9) &= w'(n_9) - w(n_9) = 26 - 19 = 7 \\
\Delta(n_1) &= w'(n_1) - w(n_1) = 26 - 19 = 7 \\
\Delta(n_7) &= w'(n_7) - w(n_7) = 28 - 28 = 0
\end{align*}
\]

Thus,

\[ \Delta W = \Delta(n_3) + \Delta(n_9) + \Delta(n_1) + \Delta(n_7) = 19 \]
TimberWolf Placement Example – Swap (b, e)

- $$\Delta W_s = \text{wirelength change from shifting}$$
  - $$h$$ is shifted and included in $$n_4 = \{d,h,i\}$$ and $$n_7 = \{c,e,f,h,n\}$$
  - $$h$$ is on the right boundary of $$n_4$$: $$\text{gradient}(h)++$$
  - $$h$$ is not on any boundary of $$n_7$$: no further change on $$\text{gradient}(h)$$

(a) BB of $$n_4$$ with $$h$$ on RHS boundary
(b) BB of $$n_7$$ with $$h$$ not on any boundary
TimberWolf Placement Example – Swap (b, e)

Thus, $\text{gradient}(h) = 1$. Since $h$ is shifted to the right by 1

$$\text{shift}\_\text{amount}(h) = 1$$

Thus,

$$\Delta W_S = \text{gradient}(h) \cdot \text{shift}\_\text{amount}(h) = 1 \cdot 1 = 1$$

Based on the calculation of $\Delta W$ and $\Delta W_S$, we get

$$\Delta C = \Delta W + \Delta W_S = 19 + 1 = 20$$
How accurate is $\Delta W_s$ estimation?

- Node $h$ is included in $n_4 = \{d,h,i\}$ and $n_7 = \{c,e,f,h,n\}$
- Real change is also 1: accurate estimation

$$w'(n_4) - w(n_4) + w'(n_7) - w(n_7) = 20 - 19 + 28 - 28 = 1$$
TimberWolf Placement Example – Swap (b, e) – Model A

- Based on piecewise linear graph
  - Shifting \( h \) causes the wirelength of \( n_4 \) to increase by 1 (19 to 20) and no change on \( n_7 \) (stay at 28)

\[ n_7 \text{ above, } n_4 \text{ below – break points for } n_7: f, n, \text{ for } n_4: i, d \]
TimberWolf Placement Example – Swap \((m, n)\)

- **Swap node \(m\) and \(o\)**
  - We shift node \(d\) and \(g\) on the shorter side of the receiving row
  - Node \(m\) included in nets \(\{n_5, n_9\}\), and \(o\) in \(\{n_2, n_{10}\}\)

\[
\begin{align*}
\text{(a)} & \quad \text{Initial configuration} \\
\text{(b)} & \quad \text{After swap}
\end{align*}
\]
TimberWolf Placement Example – Swap (m, n)

- \( \Delta W = \text{wirelength change from swap} \)

\[
\begin{align*}
\Delta(n_5) &= w'(n_5) - w(n_5) = 12 - 11 = 1 \\
\Delta(n_9) &= w'(n_9) - w(n_9) = 22 - 26 = -4 \\
\Delta(n_2) &= w'(n_2) - w(n_2) = 7 - 14 = -7 \\
\Delta(n_{10}) &= w'(n_{10}) - w(n_{10}) = 23 - 23 = 0
\end{align*}
\]

Thus,

\[
\Delta W = \Delta(n_5) + \Delta(n_9) + \Delta(n_2) + \Delta(n_{10}) = -10
\]
TimberWolf Placement Example – Swap (m, n)

- Cell $d$ and $g$ are shifted
  - $d$ is included in $n_4 = \{d,h,i\}$, $n_6 = \{d,k,j\}$, and $n_8 = \{d,l\}$
  - $d$ is on the right boundary of $n_6$ and $n_8$
  - So, $\text{gradient}(d) = 2$

![Diagram showing cell shifts and gradient calculation](image)
TimberWolf Placement Example – Swap \((m, n)\)

- **Cell \(d\) and \(g\) are shifted**
  - \(g\) is included in \(n_1 = \{a,e,g\}\), and \(n_9 = \{b,g,i,m\}\)
  - \(g\) is on the right boundary of \(n_1\) and \(n_9\)
  - So, \(\text{gradient}(g) = 2\)
TimberWolf Placement Example – Swap (m, n)

Both cell \( d \) and \( g \) are shifted to the right by 2. Thus,

\[
\Delta W_S = \text{gradient}(d) \cdot \text{shift\_amount}(d) + \\
\text{gradient}(g) \cdot \text{shift\_amount}(g) = 2 \cdot 2 + 2 \cdot 2 = 8
\]

Based on the calculation of \( \Delta W \) and \( \Delta W_S \), we get

\[
\Delta C = \Delta W + \Delta W_S = -10 + 8 = -2
\]
TimberWolf Placement Example – Swap (k, m)

- **Swap node k and m**
  - We shift node c: on the shorter side of the receiving row
  - Node k included in nets \{n_3, n_6, n_{10}\}, and m in \{n_5, n_9\}

```
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>j</td>
<td>l</td>
<td>b</td>
<td>h</td>
</tr>
<tr>
<td>o</td>
<td></td>
<td>c</td>
<td>m</td>
<td>d</td>
</tr>
<tr>
<td>f</td>
<td>e</td>
<td>i</td>
<td>k</td>
<td>n</td>
</tr>
</tbody>
</table>

(a)

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>j</td>
<td>l</td>
<td>b</td>
<td>h</td>
</tr>
<tr>
<td>o</td>
<td></td>
<td>c</td>
<td>k</td>
<td>d</td>
</tr>
<tr>
<td>f</td>
<td>e</td>
<td>i</td>
<td>m</td>
<td>n</td>
</tr>
</tbody>
</table>

(b)
```
TimberWolf Placement Example – Swap (k, m)

- $\Delta W = \text{wirelength change from swap}$

  \[
  \begin{align*}
  \Delta(n_3) &= w'(n_3) - w(n_3) = 25 - 24 = 1 \\
  \Delta(n_6) &= w'(n_6) - w(n_6) = 16 - 23 = -7 \\
  \Delta(n_{10}) &= w'(n_{10}) - w(n_{10}) = 13 - 23 = -10 \\
  \Delta(n_5) &= w'(n_5) - w(n_5) = 21 - 12 = 9 \\
  \Delta(n_9) &= w'(n_9) - w(n_9) = 22 - 22 = 0
  \end{align*}
  \]

  Thus,

  \[
  \Delta W = \Delta(n_3) + \Delta(n_6) + \Delta(n_{10}) + \Delta(n_5) + \Delta(n_9) = -7
  \]
Cell $c$ is shifted

- $c$ is included in $n_3 = \{b,c,k,n\}$ and $n_7 = \{c,e,f,h,n\}$
- $c$ is on the left boundary of $n_3$
- So, $\text{gradient}(c) = -1$
TimberWolf Placement Example – Swap (k, m)

Since $c$ is shifted to the left by 1,

$$shift\_amount(c) = -1$$

Lastly,

$$\Delta W_s = \text{gradient}(c) \cdot shift\_amount(c) = -1 \cdot -1 = 1$$

Based on the calculation of $\Delta W$ and $\Delta W_s$, we get

$$\Delta C = \Delta W + \Delta W_s = -7 + 1 = -6$$
Dragon Placer

- Dragon takes a hybrid approach that combines SA and partitioning
  - Recursive partitioning to reduce the problem size
  - Annealing to refine the solution generated by partitioning

- Top-down hierarchical placement
  - hMetis to recursively quadrisect into $4^h$ bins at level $h$
  - Swapping of bins at each level by SA to minimize WL
  - Terminates when each bin contains < 7 cells
  - At final stage of GP, switch single cells locally among bins by SA to further minimize WL

- Detailed placement is done by greedy algorithm
Dragon Placer

- **GP and DP phases**
- **GP combines WL with min-cut based on 4 approaches**

Dragon2000 paper claims approach B is best for the IBM benchmarks
Analytical Placement

- **Key Idea:** Solve a relaxed placement problem "optimally"
  - Ignore overlaps (fixed later)
  - Adopt linear or non-linear WL estimation
  - Need I/O Pads/Pins to pull cells outwards

- **Approaches for overlap removal**
  - Cell spreading
  - Legalization

- **Pros:** very good quality for existing benchmarks
- **Cons:** hard to handle big macros rotation constraints, region constraints, blockages, etc.
Force-Directed Methods

- Similar to Graph Drawing Algorithms
- Reducing the placement problem to solving a set of simultaneous linear equations to determine equilibrium locations for cells
- Goal: Map cells to the layout surface.
Force-Directed Approaches: Zero-Force Target Location

- Cell $i$ connects to several cells $j$'s at distances $d_{ij}$, by wires of weights $w_{ij}$'s
- Total Force: $F_i = \sum w_{ij}d_{ij}$
- The zero-Force target location can be determined by equating the $x$- and $y$- components of forces to zero:

$$\sum w_{ij}(x_j - \bar{x}_i) = 0 \implies \bar{x}_i = \frac{\sum w_{ij}x_j}{\sum w_{ij}}$$

- In the example, $\bar{x}_i = \frac{8x_0 + 10x_2 + 3x_0 + 3x_2}{8 + 10 + 3 + 3} = 1.083, \bar{y}_i = 1.5$
Force-Directed Placement

- **Approach I (constructive):**
  - Start with an initial placement
  - Compute the zero-force locations for all cells.
  - Apply linear assignment (matching) to determine the “ideal” locations for the cells

- **Approach II (can be constructive or iterative):**
  - Start with an initial placement.
  - Elect a “most profitable” cell $p$ (e.g., maximum $F$, critical cells) and place it in its zero-force location
  - “Fix” placement if the zero-force location has been occupied by another cell $q$.

  **Popular options to fix:**
  - **Ripple move:** place $p$ in the occupied location, compute a new zero-force location for $q$, …
  - **Chain move:** place $p$ in the occupied location, move $q$ to an adjacent location, …
  - Move $p$ to a free location close to $q$.

Linear matching probably means move its cell to its closest zero-force location
Example of Force-Directed Placement Algorithm

Algorithm: Force-Directed Placement

begin
1 Compute the connectivity for each cell;
2 Sort the cells in decreasing order of their connectivities into list L;
4 while (IterationCount < IterationLimit) do
5   Seed ← next module from L;
6   Declare the position of the seed vacant;
7     while (EndRipple = FALSE) do
8       Compute target location of the seed;
9       case the target location
10     VACANT:
11       Move seed to the target location and lock;
12       EndRipple ← TRUE; AbortCount ← 0;
13     SAME AS PRESENT LOCATION:
14       EndRipple ← TRUE; AbortCount ← 0;
15     LOCKED:
16       Move selected cell to the nearest vacant location;
17       EndRipple ← TRUE; AbortCount ← AbortCount + 1;
18       if (AbortCount > AbortLimit) then
19         Unlock all cell locations;
20         IterationCount ← IterationCount + 1;
21     OCCUPIED AND NOT LOCKED:
22       Select cell as the target location for next move;
23       Move seed cell to target location and lock the target location;
24       EndRipple ← FALSE; AbortCount ← 0;
26 end
Gordian Placement Algorithm

- Kleinhans, Sigl and Johannes, “GORDIAN: VLSI placement by quadratic programming and slicing optimization,” TCAD, October 91 (ICCAD-90).
- Global placement (analytical scheme)
- Detailed placement (combinatorial search)

Data flow in the placement procedure GORDIAN.
Gordian Global Placement Optimization

- Apply QP (Quadratic Programming)
  - Objective Function: WL (Wire Length)
  - Squared Distance between Modules and Top-Level I/O Pins
- Constraints:
  - Centers of the regions of a partitioning level
  - Cell Spreading Scheme
QP Objective Function - 1

- General QP - Minimize: \( \frac{1}{2} x^T A x + b^T x + c \)
- This is equivalent to solving the system: \( Ax = b \)
- QP can be formulated independently in x, y directions
- For WL minimization:
  - For any connected cells \( i, j \), let \( c_{ij} \) be the connection weight
  - Cost of a 2-point net between cells \( i \) and \( j \):
    \[ c_{ij} ( (x_i - x_j)^2 + (y_i - y_j)^2 ) \]
  - Ignoring cell overlapping
  - Total Cost:
    \[ \frac{1}{2} x^T A x + d_x^T x + \frac{1}{2} y^T A y + d_y^T y + c \]

We may also put \( \frac{1}{2} \) in front of the 2-point net cost
QP Objective Function - 1

- For WL minimization: \( \frac{1}{2} x^T A x + b^T x + c \)
- For any connected cells \( i, j \), let \( c_{ij} \) be the connection weight
- Cost of a 2-point net between cells \( i \) and \( j \):
  \[ c_{ij}(x_i - x_j)^2 + (y_i - y_j)^2 \]
- Top-Level I/Os possess fixed coordinates
- Necessary for Analytical Placement Algorithms

Horizontal cost
\[
= \frac{1}{2} c_{12} (x_1 - x_2)^2 + \frac{1}{2} c_{2p} (x_2 - x_p)^2 \\
= \frac{1}{2} \begin{pmatrix} x_1 & x_2 \end{pmatrix} \begin{pmatrix} c_{12} & -c_{12} \\ -c_{12} & c_{12} + c_{2p} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ -c_{2p} x_p \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \frac{1}{2} c_{2p} x_p^2
\]

\( x_p \): a constant

We may also put \( \frac{1}{2} \) in front of the 2-point net cost
Simple QP Formulation Example

- Given a placement P with two fixed I/O pins
  - \( p_1 @ (100, 175) \)
  - \( p_2 @ (200, 225) \), and
- And three standard-cells \( a, b, c \) and 4 nets \( N_1-N_4 \):
  - \( N_1(p_1, a), N_2(a, b), N_3(b, c) \) and \( N_4(c, p_2) \)
- Find the Optimal coordinates of blocks \( a, b, c \)
Simple QP Formulation Example

- Formulate Cost in x direction:
  \[
  L_x(P) = (100 - x_a)^2 + (x_a - x_b)^2 + (x_b - x_c)^2 + (x_c - 200)^2 \\
  \frac{\partial L_x(P)}{x_a} = -2(100 - x_a) + 2(x_a - x_b) = 4x_a - 2x_b - 200 = 0 \\
  \frac{\partial L_x(P)}{x_b} = -2(x_a - x_b) + 2(x_b - x_c) = -2x_a + 4x_b - 2x_c = 0 \\
  \frac{\partial L_x(P)}{x_c} = -2(x_b - x_c) + 2(x_c - 200) = -2x_b + 4x_c - 400 = 0 
  \]

- Put it in Matrix Form $Ax = b$

  \[
  \begin{bmatrix}
  -2 & 0 & 200 \\
  -2 & -2 & 0 \\
  0 & -2 & 400
  \end{bmatrix} \begin{bmatrix}
  x_a \\
  x_b \\
  x_c 
  \end{bmatrix} = \begin{bmatrix}
  100 \\
  0 \\
  200
  \end{bmatrix}
  \]

- Solution: $x_a = 125, x_b = 150, x_c = 175$

Consider p1 -> (a, b), a->c, b->d, c->d, d->p2 with and without clique model
Simple QP Formulation Example

- **Formulate Cost in y direction:**
  \[ L_y(P) = (175 - y_a)^2 + (y_a - y_b)^2 + (y_b - y_c)^2 + (y_c - 225)^2 \]
  \[ \frac{\partial L_y(P)}{y_a} = -2(175 - y_a) + 2(y_a - y_b) = 4y_a - 2y_b - 350 = 0 \]
  \[ \frac{\partial L_y(P)}{y_b} = -2(y_a - y_b) + 2(y_b - y_c) = -2y_a + 4y_b - 2y_c = 0 \]
  \[ \frac{\partial L_y(P)}{y_c} = -2(y_b - y_c) + 2(y_c - 225) = -2y_b + 4y_c - 450 = 0 \]

- **Put it in Matrix Form Ax = b**

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y_a</th>
<th>y_b</th>
<th>y_c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>350</td>
<td>0</td>
<td>450</td>
</tr>
<tr>
<td>2</td>
<td>-2</td>
<td>0</td>
<td>-1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>450</td>
<td>-1</td>
<td>-2</td>
</tr>
</tbody>
</table>

\[\begin{align*}
x  & = 175 \\
y_a & = 0 \\
y_b & = 0 \\
y_c & = 0
\end{align*}\]

- **Solution:** \( y_a = 187.5, y_b = 200, y_c = 212.5 \)
Simple QP Formulation Example

- Given a placement P with two fixed I/O pins
  - p1 @ (100, 175)
  - p2 @ (200, 225), and
- And three standard-cells a, b, c and 4 nets N1-N4:
  - N1(P1, a), N2(a, b), N3(b, c) and N4(c, P2)
- Find the Optimal coordinates of blocks a, b, c
Another Simple QP Formulation Example

- Given a placement P with two fixed I/O pins
  - p1 @ (100, 170)
  - p2 @ (200, 300), and

- And four standard-cells a, b, c, d and 5 nets N1-N5:
  - N1(p1, a, b), N2(a, c), N3(b, d), N4(c, d), N5(d, p2)

- Find the Optimal coordinates of blocks a, b, c, d

Consider p1 -> (a, b), a->c, b->d, c->d, d->p2 with and without clique model
Nets list (output -> input, i/o -> input/output)

p1: a b
a: c
b: d
p2: d

Dx Dy

100 170
100 170
0 0
200 300
Solution for x

\[
\begin{align*}
p1 & \quad p2 & \quad \begin{bmatrix} a & b & c & d \end{bmatrix} & \quad \begin{bmatrix} Dx \quad Dy \end{bmatrix} \\
\text{a} & \text{i} & \text{0} & \text{2} & \text{0} & \text{-1} & \text{0} & \begin{bmatrix} 100 \quad 170 \end{bmatrix} \\
\text{b} & \text{i} & \text{0} & \text{0} & \text{2} & \text{0} & \text{-1} & \begin{bmatrix} 100 \quad 170 \end{bmatrix} \\
\text{c} & \text{0} & \text{0} & \text{-1} & \text{0} & \text{2} & \text{-1} & \begin{bmatrix} 0 \quad 0 \end{bmatrix} \\
\text{d} & \text{0} & \text{1} & \text{0} & \text{-1} & \text{-1} & \text{3} & \begin{bmatrix} 200 \quad 300 \end{bmatrix}
\end{align*}
\]

http://wims.unice.fr/wims/wims.cgi

Hint: Want to enter \( x^2 \) Type \( x^2 \) or \( x**2 \).
You have entered the system

\[
\begin{align*}
2x_1 - x_2 & = 100 \\
-x_1 + 2x_2 - x_3 & = 0 \\
-x_3 & = 200
\end{align*}
\]

This system has a unique solution, which is

\[
\begin{align*}
x_1 = 1300/11, x_2 = 1400/11, x_3 = 1500/11, x_4 = 1700/11
\end{align*}
\]

\[\text{ax} = 118, \text{bx} = 127, \text{cx} = 136, \text{dx} = 154\]
Solution for y

\[
\begin{array}{c|c|c|c|c}
\text{p1} & \text{p2} & \text{a} & \text{b} & \text{c} & \text{d} & \text{Dx} & \text{Dy} \\
\hline
\text{a} & 1 & 0 & 0 & 2 & 0 & -1 & 100 \\
\text{b} & 1 & 0 & 0 & 0 & 2 & 0 & -1 \\
\text{c} & 0 & 0 & 0 & 0 & 2 & 0 & -1 \\
\text{d} & 0 & 1 & 0 & 0 & -1 & 3 & 200 \\
\end{array}
\]

\[\begin{align*}
\text{Linear solver} \\
\text{Hint: Want to enter x^2? Type x^2 or x*2.} \\
\text{You have entered the system} \\
\begin{align*}
2x_1 - x_2 &= 170 \\
2x_2 - x_3 &= 170 \\
-x_1 + 2x_2 &= 0 \\
-x_3 + 3x_2 &= 300
\end{align*}
\text{This system has a unique solution, which is} \\
\{x_1 = 2130/11, x_2 = 2260/11, x_3 = 2390/11, x_4 = 2650/11\}
\end{align*}\]

\[\text{ay = 193, by = 205, cy = 217, dy = 240}\]
Another Simple QP Formulation Example

- Given a placement $P$ with two fixed I/O pins
  - $p1$ @ (100, 170)
  - $p2$ @ (200, 300), and
- And four standard-cells $a$, $b$, $c$, $d$ and 5 nets $N1$-$N5$:
  - $N1(p1, a, b)$, $N2(a, c)$, $N3(b, d)$, $N4(c, d)$, $N5(d, P2)$
- Find the Optimal coordinates of blocks $a$, $b$, $c$, $d$

Consider $p1 -> (a, b)$, $a -> c$, $b -> d$, $c -> d$, $d -> p2$ with and without clique model

Nets list (output -> input, i/o -> input/output)

$p1$: $a$, $b$ (size = 3) ⇔ $p1$-$a$ (+2/3), $p1$-$b$ (+2/3), $a$-$a$ (-2/3), $a$-$b$ (-2/3), $b$-$a$ (-2/3), $b$-$b$ (+2/3)

$p2$: $d$ (size = 2) ⇔ $p2$-$d$ (+1)

Sum of $a$ weights: $2/3 + 2/3 + 1 = 7/3$
Sum of $b$ weights: $2/3 + 2/3 + 1 = 7/3$
Sum of $c$ weights: $1 + 1 = 2$
Sum of $d$ weights: $1 + 1 + 1 = 3$
Solution for $x$

\[
\begin{array}{cccc|cc}
\text{pl} & \text{p2} & a & b & c & d & Dx & Dy \\
\hline
a & 2/3 & 0 & 7/3 & -2/3 & -1 & 0 & 200/3 & 340/3 \\
b & 2/3 & 0 & b & -2/3 & 7/3 & 0 & -1 & 200/3 & 340/3 \\
c & 0 & 0 & c & -1 & 0 & 2 & -1 & 0 & 0 \\
d & 0 & 1 & d & 0 & -1 & -1 & 3 & 200 & 300 \\
\end{array}
\]

**Linear solver**

Hint: Want to enter $x^2$, type $x^2$ or $x=x^2$.

You have entered the system

\[
\begin{align*}
7/3 \ x_1 - 2/3 \ x_2 - x_3 &= 200/3 \\
-2/3 \ x_1 + 7/3 \ x_2 - x_4 &= 200/3 \\
-x_1 + 2 \ x_3 - x_4 &= 0 \\
-x_2 - x_3 + 3 \ x_4 &= 200
\end{align*}
\]

This system has a unique solution, which is

\[
\begin{align*}
x_1 &= 5000/39 \\
x_2 &= 4000/3 \\
x_3 &= 5600/39 \\
x_4 &= 6200/39
\end{align*}
\]

- $ax = 128$, $bx = 133$, $cx = 144$, $dx = 158$
Solution for $y$

<table>
<thead>
<tr>
<th></th>
<th>p1</th>
<th>p2</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>Dx</th>
<th>Dy</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>2/3</td>
<td>0</td>
<td>7/3</td>
<td>-2/3</td>
<td>-1</td>
<td>0</td>
<td>200/3</td>
<td>340/3</td>
</tr>
<tr>
<td>b</td>
<td>2/3</td>
<td>0</td>
<td>-2/3</td>
<td>7/3</td>
<td>0</td>
<td>-1</td>
<td>200/3</td>
<td>340/3</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>2</td>
<td>-1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>3</td>
<td>200</td>
<td>300</td>
</tr>
</tbody>
</table>

Linear solver

Hint: Want to enter x? Type x or x.x.  
You have entered the system

\[
\begin{align*}
7/3 x_1 + 2/3 x_2 - x_3 &= 340/3 \\
2/3 x_1 + 7/3 x_2 - x_4 &= 340/3 \\
-x_1 + 2 x_3 - x_4 &= 0 \\
-x_2 - 3 x_3 + 5 x_4 &= 300
\end{align*}
\]

This system has a unique solution, which is

\[
\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 620/3 \\ 640/3 \\ 680/3 \\ 740/3 \end{bmatrix}
\]

\[ ay = 206, by = 213, cy = 226, dy = 246 \]
Octave Linear Solver
Another Simple QP Formulation Example

- Given a placement $P$ with two fixed I/O pins
  - $p_1 @ (100, 170)$
  - $p_2 @ (200, 300)$, and
- And four standard-cells $a$, $b$, $c$, $d$ and 5 nets $N1$-$N5$:
  - $N1(P_1, a, b)$, $N2(a, c)$, $N3(b, d)$, $N4(c, d)$, $N5(d, P_2)$
- Find the Optimal coordinates of blocks $a$, $b$, $c$, $d$

Consider $p_1 \rightarrow (a, b)$, $a \rightarrow c$, $b \rightarrow d$, $c \rightarrow d$, $d \rightarrow p_2$ with and without star model
Nets list (output -> input, i/o -> input/output)
Must incorporate Star pins as well, when computing net connectivity:
- $p_1$: $a$ $b$ $s_1$ (size = 4)
- $a$: $c$ $s_2$ (size = 3)
- $b$: $d$ $s_3$ (size = 3)
- $c$: $d$ $s_4$ (size = 3)
- $p_2$: $d$ $s_5$ (size = 3)

$Dx(a) = 100 + 120 + 140 = 360$, $Dy(a) = 170 + 210 + 280 = 690$
$Dx(b) = 100 + 120 + 160 = 380$, $Dy(b) = 200 + 210 + 240 = 650$
$Dx(c) = 140 + 180 = 320$, $Dy(c) = 280 + 260 = 540$
$Dx(d) = 200 + 160 + 180 + 190 = 730$, $Dy(d) = 300 + 240 + 260 + 290 = 1090$
Solution for x

<table>
<thead>
<tr>
<th>p1</th>
<th>p2</th>
<th>s1</th>
<th>s2</th>
<th>s3</th>
<th>s4</th>
<th>s5</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>Dx</th>
<th>Dy</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>360</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>-1</td>
<td>380</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>4</td>
<td>-1</td>
<td>320</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>6</td>
<td>730</td>
</tr>
</tbody>
</table>

```
Dx  Dy  
360 690
380 650
320 540
730 1090
```

**Linear solver**

Hint: Want to enter x^2? Type x**2**.

You have entered the system:

\[
\begin{align*}
4x_1 - x_2 &= 360 \\
-4x_1 + 4x_2 - x_3 &= 380 \\
-x_1 + 4x_2 - x_4 &= 320 \\
-x_3 - x_4 + 6x_5 &= 730
\end{align*}
\]

This system has a unique solution, which is:

\[
\begin{align*}
(x_1 &= 42340/329, x_2 = 45270/329, x_3 = 56920/329, x_4 = 56630/329)
\end{align*}
\]

- \( ax = 128.7 \), \( bx = 137.6 \), \( cx = 154.7 \), \( dx = 170.4 \)
Solution for \( y \)

<table>
<thead>
<tr>
<th>( p_1 )</th>
<th>( p_2 )</th>
<th>( s_1 )</th>
<th>( s_2 )</th>
<th>( s_3 )</th>
<th>( s_4 )</th>
<th>( s_5 )</th>
<th>( a )</th>
<th>( b )</th>
<th>( c )</th>
<th>( d )</th>
<th>( D_x )</th>
<th>( D_y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>360</td>
<td>690</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>-1</td>
<td>380</td>
<td>650</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>320</td>
<td>540</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>6</td>
<td>730</td>
</tr>
</tbody>
</table>

Linear solver

Hint: Want to enter x? Type x=2 or x*2.
You have entered the system

\[
\begin{align*}
4x_1 - x_2 &= -690 \\
4x_2 - x_3 &= 650 \\
x_1 + 4x_3 &= 540 \\
x_3 - x_4 &= 1090
\end{align*}
\]

This system has a unique solution, which is

\[
\{ x_1 = 78100/329, x_2 = 78100/329, x_3 = 86090/329, x_4 = 86500/329 \}
\]

\[ ay = 237, by = 228.27, cy = 260.1, dy = 263.1 \]
Another Simple QP Formulation Example

- Given a placement P with two fixed I/O pins
  - p1 @ (100, 170)
  - p2 @ (200, 300), and

- And four standard-cells a, b, c, d and 5 nets N1-N5:
  - N1(P1,a,b), N2(a,c), N3(b,d), N4(c,d), N5(d,P2)

- Find the Optimal coordinates of blocks a, b, c, d

```
<table>
<thead>
<tr>
<th>Star Model (1/k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>7/6</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>-1/3</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>
```

- s1 @ (120, 210)
- s2 @ (140, 280)
- s3 @ (160, 240)
- s4 @ (180, 260)
- s5 @ (190, 290)

Consider p1 -> (a, b), a->c, b->d, c->d, d->p2 with and without star model

Nets list (output -> input, i/o -> input/output)

Must incorporate Star pins as well, when computing net connectivity:

- p1: a b s1 (size = 4), p1->a = 1/4, p1->b = 1/4, p1->s1 = 1/4, s1->a = +1/4, s1->b = +1/4
- a: c s2 (size = 3), a->a = +1/3, a->c = -1/3, c->a = -1/3, c->c = +1/3, s2->a = +1/3, s2->c = +1/3
- b: d s3 (size = 3), b->b = +1/3, b->d = -1/3, d->b = -1/3, d->d = +1/3, s3->b = +1/3, s3->d = +1/3
- c: d s4 (size = 3), c->c = +1/3, c->d = -1/3, d->c = -1/3, d->d = +1/3, s4->c = +1/3, s4->d = +1/3
- p2: d s5 (size = 3), p2->d = 1/3, s5->d = 1/3

Sum of a weights = 1/4 + 1/4 + 1/3 + 1/3 = 7/6
Sum of b weights = 1/4 + 1/4 + 1/3 + 1/3 = 7/6
Sum of c weights = 1/3 + 1/3 + 1/3 + 1/3 = 4/3
Sum of d weights = 1/3 + 1/3 + 1/3 + 1/3 + 1/3 = 6/3 = 2

Dx(a) = 100 * 1/4 + 120 * 1/4 + 140 * 1/3 = 305/3, Dy(a) = 200 * 1/4 + 210 * 1/4 + 280 * 1/3 = 2350/12
Dx(b) = 100 * 1/4 + 120 * 1/4 + 160 * 1/3 = 325/3, Dy(b) = 200 * 1/4 + 210 * 1/4 + 240 * 1/3 = 365/2
Dx(c) = 140 * 1/3 + 180 * 1/3 = 320/3, Dy(c) = 280 * 1/3 + 260 * 1/3 = 540/3
\[ D_x(d) = \frac{1}{3} (200 + 160 + 180 + 190) = \frac{730}{3}, \quad D_y(d) = \frac{1}{3} (300 + 240 + 260 + 290) = \frac{1090}{3} \]
Solution for x

<table>
<thead>
<tr>
<th>p1</th>
<th>p2</th>
<th>s1</th>
<th>s2</th>
<th>s3</th>
<th>s4</th>
<th>s5</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>Dx</th>
<th>Dy</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1/4</td>
<td>0</td>
<td>1/4</td>
<td>1/3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7/6</td>
<td>0</td>
<td>-1/3</td>
<td>0</td>
<td>305/3</td>
</tr>
<tr>
<td>b</td>
<td>1/4</td>
<td>0</td>
<td>1/4</td>
<td>0</td>
<td>1/3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7/6</td>
<td>0</td>
<td>-1/3</td>
<td>325/3</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1/3</td>
<td>0</td>
<td>1/3</td>
<td>0</td>
<td>-1/3</td>
<td>0</td>
<td>4/3</td>
<td>-1/3</td>
<td>220/3</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>1/3</td>
<td>0</td>
<td>0</td>
<td>1/3</td>
<td>1/3</td>
<td>1/3</td>
<td>0</td>
<td>-1/3</td>
<td>-1/3</td>
<td>2</td>
<td>730/3</td>
</tr>
</tbody>
</table>

**Linear solver**

Hint: Want to enter $x^2$? Type $x=x^2$.

You have entered the system

\[
\begin{align*}
7/6 x_1 - 1/3 x_3 &= 305/3 \\
7/6 x_2 - 1/3 x_3 &= 325/3 \\
-1/3 x_1 + 4/3 x_3 &= 365/2 \\
-1/3 x_2 - 1/3 x_3 &= 320/3 \\
1/3 x_2 - 1/3 x_3 &= 2 x_1 = 730/3
\end{align*}
\]

This system has a unique solution, which is

\[
\{ x_1 = 130450/991, x_2 = 140510/991, x_3 = 154320/991, x_4 = 169710/991 \}.
\]

- $ax = 131.6$, $bx = 141.7$, $cx = 155.7$, $dx = 171.2$
Solution for y

<table>
<thead>
<tr>
<th>p1</th>
<th>p2</th>
<th>s1</th>
<th>s2</th>
<th>s3</th>
<th>s4</th>
<th>s5</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>Dx</th>
<th>Dy</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1/4</td>
<td>0</td>
<td>1/4</td>
<td>1/3</td>
<td>0</td>
<td>0</td>
<td>a</td>
<td>7/6</td>
<td>0</td>
<td>-1/3</td>
<td>0</td>
<td>305/3</td>
</tr>
<tr>
<td>b</td>
<td>1/4</td>
<td>0</td>
<td>1/4</td>
<td>0</td>
<td>1/3</td>
<td>0</td>
<td>0</td>
<td>b</td>
<td>0</td>
<td>7/6</td>
<td>0</td>
<td>-1/3</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1/3</td>
<td>0</td>
<td>1/3</td>
<td>0</td>
<td>c</td>
<td>-1/3</td>
<td>0</td>
<td>4/3</td>
<td>-1/3</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>1/3</td>
<td>0</td>
<td>0</td>
<td>1/3</td>
<td>1/3</td>
<td>1/3</td>
<td>d</td>
<td>0</td>
<td>-1/3</td>
<td>-1/3</td>
<td>2</td>
</tr>
</tbody>
</table>

**Linear solver**

Hint: Want to enter x? Type x-1 or x^2.

You have entered the system

\[
\begin{align*}
7/6 x_1 - 1/3 x_3 &= 1175/6 \\
7/6 x_2 - 1/3 x_3 &= 365/2 \\
-1/3 x_1 + 4/3 x_3 &= 100 \\
-1/3 x_2 - 1/3 x_3 &= 2 x_4 = 1000/3
\end{align*}
\]

This system has a unique solution, which is

\[
\{ x_1 = 240425/901, x_2 = 229745/901, x_3 = 259275/901, x_4 = 261535/901 \}
\]

\[ ay = 242.6, \ by = 231.83, \ cy = 261.6, \ dy = 263.9 \]
<table>
<thead>
<tr>
<th>Model</th>
<th>X Coordinate</th>
<th>Y Coordinate</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>P2P (no weights)</strong></td>
<td>$a_x = 118$, $b_x = 127$, $c_x = 136$, $d_x = 154$</td>
<td>$a_y = 193$, $b_y = 205$, $c_y = 217$, $d_y = 240$</td>
</tr>
<tr>
<td><strong>CLIQUE (2/n)</strong></td>
<td>$a_x = 128$, $b_x = 133$, $c_x = 144$, $d_x = 158$</td>
<td>$a_y = 206$, $b_y = 213$, $c_y = 226$, $d_y = 246$</td>
</tr>
<tr>
<td><strong>STAR (no weights)</strong></td>
<td>$a_x = 128.7$, $b_x = 137.6$, $c_x = 154.7$, $d_x = 170.4$</td>
<td>$a_y = 237$, $b_y = 228.27$, $c_y = 260.1$, $d_y = 263.1$</td>
</tr>
<tr>
<td><strong>STAR (1/n)</strong></td>
<td>$a_x = 131.6$, $b_x = 141.7$, $c_x = 155.7$, $d_x = 171.2$</td>
<td>$a_y = 242.6$, $b_y = 231.83$, $c_y = 261.6$, $d_y = 263.9$</td>
</tr>
</tbody>
</table>
Convexity

- The QP objective function is a **convex** function
  - If $x$ is a single variable $f''(x) \geq 0$
  - If $x = [x_1 \ x_2 \ \ldots \ x_n]^T$, $x^T H x \geq 0$

$$H = \begin{bmatrix}
\frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\
\frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2}
\end{bmatrix}$$

- $A$: $(x, f(x))$
- $B$: $(x', f(x'))$
- $C$: $(cx + (1-c) \ x', f(cx + (1-c) \ x'))$
- $D$: $(cx + (1-c) \ x', cf(x) + (1-c) \ f(x'))$
- **Convex**: $cf(x) + (1-c) \ f(x') \geq f(cx + (1-c) \ x')$

$H$ is the Hessian Matrix, and is a square matrix of the partial derivatives of $f$.
Gordian Partitioning Scheme

- After global optimization, modules are divided into two parts such that the module areas of both subsets are about the same
- Sort the x- (y-) coordinates of the modules and cut the region vertically (horizontally).
Gordian Partitioning Constraints

- **Constraints:** make the center of module gravity correspond to the center of the region, \( A^{(l)} x = u^{(l)} \)
- Each entry in \( A \) gives the area ratio of the module in the corresponding region.
- \( f_m \): area of module \( m \), \( u_r \): center of a region \( r \),
- \( x_m \): module coordinate

\[
\begin{align*}
\sum_{l} x_{f_a} + x_{f_b} + x_{f_c} &= u_r (f_a + f_b + f_c) \\
\sum_{l} x_{f_d} + x_{f_e} + x_{f_f} &= u_r' (f_d + f_e + f_f)
\end{align*}
\]
Gordian QP Formulation

- The objective function is a convex quadratic function.
  - $C$ is positive definite.
- Constraints are linear (and thus convex)
- QP has a unique global minimum $x^*$
- QP can be solved optimally in polynomial time
- Ignore the non-overlapping constraint at the global placement stage
- The problem in $x$- and $y$-directions can be separated and solved independently

$$QP: \min_x ( F(x) = \frac{1}{2} x^T A x + d_x^T x; \ A(i)x = u(i) )$$
Gordian Placement Example

- Perform GORDIAN placement
  - Uniform area and net weight, area balance factor = 0.5
  - Undirected graph model: each edge in k-clique gets weight 2/k

Thick edges have a weight of 2/3, dotted of ½
Gate a is connected to a 6 clique
Gate a sum of net weights (2/3 + 2/3 + 2/3 + 7/6 +1/2 + 1/2) = (19/6 + 1) = 25/6
Gate e is a member of two nets: (a, e, f, z1), (w2, a, e) = 2/4 + 2/3 = 7/6
It is important to note that the 7/6 factor attempts to account for the two different net connections to e’s inputs, one from the w2 input net, the other from the a gate output
Gates with multiple outputs? We discuss this issue later!
Top-Level I/O Placement

- Necessary for any Analytical Method to work
The Adjacency Matrix is created based on the I/O, output to input and input to output nets.
Gate connections are counted twice, e.g. a->e (output->input), and e->a (input to output).

Similar to Graph adjacency matrix, or Component CCs for both input and output connections - in terms of # of connections!
To apply net weight, the output->input connections of gate i must be explored and entries (i, j1), (i, j2), ... (i, jn) be stored
by symmetry matrix entries (jn, i) ... (j2, i), (j1, i) may be copied over

Gates with multiple outputs? We discuss this issue later!
Matrix A Build-up – Pin Connection Matrix

- Shows connections between movable nodes and IO
  - Rows = movable nodes, columns = IO (fixed)

Columns are: w1 w2 w3 ...
Matrix A Build-up – **Degree Matrix**

- Based on **both adjacency and pin connection matrices**
- Sum of entries in the same row (= node degree)

Forming the weighted diagonal requires adding each node’s weighted connections, e.g. for node e we must add a→e to both a and e’s diagonal entries; Each net contributes to all its constituent node diagonals!

Clique diagonal calculation: for each net, each pair is connected by net weight. For node a:

(w1, a, b), (w2, a, e), (a, e, f, z1) => \( \frac{2}{3} (w1→a) + \frac{2}{3} (a→b) + \frac{2}{3} (w2→a) + \frac{2}{3} (a→e) + \frac{2}{4} (a→e) + \frac{2}{4} (a→f) + \frac{2}{4} (a→z1) = \)

\(\frac{8}{3} + \frac{3}{2} = \frac{16}{6} + \frac{9}{6} = \frac{25}{6}\)
L, the Laplacian Matrix is defined as: \( L = D - A \), where \( D \) is the degree matrix, and \( A \) is the adjacency matrix of the graph.
Fixed Pin Vectors Build-up

- Based on pin connection matrix and IO location

Each entry $i$ in $d_x$, denoted $d_{x,i}$, is computed as follows:

$$d_{x,i} = - \sum_j p_{ij} \cdot x(p_j)$$

where $p_{ij}$ denotes the entry of the pin connection matrix, and $x(p_j)$ is the $x$-coordinate of the corresponding IO pin $j$.

- Y-direction is defined similarly

The minus is because the pin connection matrix entry $p_{ij}$ is negative
The d vector must be positive
Fixed Pin Vectors Build-up

\[ d_{x,1} = -\left( \frac{2}{3} \cdot 0 + \frac{2}{3} \cdot 0 + 0 \cdot 0 + 0 \cdot 1 + \frac{1}{2} \cdot 2 + 0 \cdot 3 + 0 \cdot 4 + 0 \cdot 4 \right) = -1 \]

By examining the remaining 9 movable cells, we get

\[ d_x = \begin{pmatrix} -1 & 0 & -2/3 & -2/3 & -1 & -1 & 0 & -3 & -4 & -4 \end{pmatrix} \]

\[
\begin{pmatrix}
\frac{2}{3} & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\
0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{pmatrix}
\]
Fixed Pin Vectors Build-up

\[ d_{y,1} = -\left( \frac{2}{3} \cdot 1 + \frac{2}{3} \cdot 2 + 0 \cdot 3 + 0 \cdot 4 + \frac{1}{2} \cdot 0 + 0 \cdot 0 + 0 \cdot 1 + 0 \cdot 2 \right) = -2 \]

By examining the remaining 9 movable cells, we get

\[ d_y^T = \begin{pmatrix} -2 & -\frac{13}{6} & -\frac{25}{6} & -\frac{4}{3} & 0 & 0 & 0 & -1 & -2 \end{pmatrix} \]

\[
\begin{pmatrix}
\frac{2}{3} & \frac{2}{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{2}{3} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{2} & \frac{2}{3} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{2} & \frac{2}{3} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{2}{3} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{pmatrix}
\]
Gordian Level 0 QP Formulation

No constraints necessary

Minimize

\[ \phi(x) = \frac{1}{2} x^T C x + d_x^T x \]

and

\[ \phi(y) = \frac{1}{2} y^T C y + d_y^T y \]

We use MOSEK and obtain the following solution:

\[ x^T = (0.95 \ 0.92 \ 1.21 \ 1.32 \ 1.32 \ 1.61 \ 1.98 \ 2.13 \ 2.59 \ 2.51) \]

\[ y^T = (1.27 \ 1.83 \ 2.48 \ 2.61 \ 1.16 \ 1.45 \ 1.84 \ 0.92 \ 1.41 \ 2.03) \]

MOSEK Alternatives:

- GSL, Intel’s MKL
Gordian Level 0 Placement

- Cells with Actual Dimensions will Overlap!
Gordian Level 1 Partitioning

- Perform level 1 partitioning
  - Obtain center locations for center-of-gravity constraints

\[ u_x^{(1)} = [1 \ 3]^T, u_y^{(1)} = [2 \ 2]^T \]
Gordian Level 1 Constraint

We first sort the nodes based on their $x$-coordinates:

$$\{b, a, c, e, d, f, g, h, j, i\}$$

We perform partitioning under $\alpha = 0.5$:

$$S_{\rho'} = \{b, a, c, e, d\}, \quad S_{\rho''} = \{f, g, h, j, i\}$$

The center location vectors are:

$$u_x^{(1)} = \left(\frac{1}{3}\right), \quad u_y^{(1)} = \left(\frac{2}{2}\right)$$

We build the matrix $A^{(1)}$ for the center-of-gravity constraint at level $l = 1$:

$$A^{(1)} = \begin{pmatrix}
\frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3}
\end{pmatrix}$$

Alpha is the area ratio constraint - check
Gordian Level 1 LQP Formulation

We now solve the following Linearly constrained QP (LQP) to obtain the new placement for the movable nodes:

\[
\begin{align*}
\text{Minimize } & \phi(x) = \frac{1}{2} x^T C x + d_x^T x, \text{ subject to } A^{(1)} \cdot x = u_x^{(1)} \\
\text{Minimize } & \phi(y) = \frac{1}{2} y^T C y + d_y^T y, \text{ subject to } A^{(1)} \cdot y = u_y^{(1)}
\end{align*}
\]

The solutions are as follows:

\[
\begin{align*}
x^T &= (0.70 \ 0.71 \ 1.17 \ 1.21 \ 1.22 \ 2.17 \ 3.10 \ 2.84 \ 3.56 \ 3.33) \\
y^T &= (1.34 \ 1.94 \ 2.66 \ 2.76 \ 1.30 \ 1.83 \ 2.45 \ 1.32 \ 1.91 \ 2.49)
\end{align*}
\]
Gordian Level 1 Placement
Gordian Verification

- Verify that the LQP constraints are satisfied in the LHS partition

The following cells are partitioned to the left: \( a(0.70, 1.34), b(0.71, 1.94), c(1.17, 2.66), d(1.21, 2.76), \) and \( e(1.22, 1.30) \). Thus, the center of gravity is located at:

\[
\frac{0.70 + 0.71 + 1.17 + 1.21 + 1.22}{5} = 1.00
\]
\[
\frac{1.34 + 1.94 + 2.66 + 2.76 + 1.30}{5} = 2.00
\]

This agrees with the center location \((1, 2)\).
Add two more cut-lines
This results in $p_1 = \{c, d\}, p_2 = \{a, b, e\}, p_3 = \{g, i\}, p_4 = \{f, h, l\}$
Gordian Level 2 Constraint

The center location vectors are:

\[ u_x^{(2)} = \begin{pmatrix} 1 \\ 1 \\ 3 \end{pmatrix}, \quad u_y^{(2)} = \begin{pmatrix} 3.2 \\ 1.2 \\ 1.2 \end{pmatrix} \]

Next, we build the matrix \( A^{(2)} \) for the center-of-gravity constraint at level \( l = 2 \). Recall that \( p_1 = \{c, d\} \), \( p_2 = \{a, b, e\} \), \( p_3 = \{g, j\} \), \( p_4 = \{f, h, i\} \). Thus,

\[ A^{(2)} = \begin{pmatrix} 0 & 0 & 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 0 & 0 & 1/3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 1/3 & 0 & 1/3 & 1/3 & 0 & 0 \end{pmatrix} \]

where the rows denote the partitions \( p_1 \) through \( p_4 \), and the columns denote the cells \( a \) through \( j \).
Gordian Level 2 LQP Formulation

We now solve the following LQP to obtain the placement of the movable nodes:

Minimize $\phi(x) = \frac{1}{2} x^T C x + d_x^T x$, subject to $A^{(2)} \cdot x = u^{(2)}_x$

Minimize $\phi(y) = \frac{1}{2} y^T C y + d_y^T y$, subject to $A^{(2)} \cdot y = u^{(2)}_y$

The solutions are as follows:

$x^T = \begin{pmatrix} 0.83 & 0.78 & 1.00 & 1.00 & 1.39 & 2.28 & 2.89 & 3.06 & 3.66 & 3.11 \end{pmatrix}$

$y^T = \begin{pmatrix} 1.01 & 1.78 & 3.08 & 3.32 & 0.82 & 1.44 & 3.18 & 0.59 & 1.57 & 3.22 \end{pmatrix}$
Gordian Level 2 Placement

- Clique-based wiring is shown
Gordian Example Summary

- **Center-of-gravity constraint**
  - Helps spread the cells evenly while monitoring WL
  - Removes overlaps among the cells
    - Using actual cell dimension
Clustered Placement QP Formulation

- each cluster is composed of arbitrary set of components
- N Components, M Clusters, T I/Os
  - A Matrix Size is \((M + N) \times (M + N)\)
  - Pin Connection Matrix is \((M \times N) \times T\)
- each (Component \(\rightarrow\) Cluster) Connection gets -1, or -weight in (component, cluster) cell of Laplacian Matrix
- each (Cluster \(\rightarrow\) Component) Connection gets -1, or -weight in (cluster, component) cell of Laplacian Matrix
- Thus, diagonal \(A\) entries \((i, i)\) should **sum the total of relevant** (Component \(\rightarrow\) Cluster) or (Cluster \(\rightarrow\) Component) connections

Magnitude of diagonal entries \((i, i)\), per row, must be > sum of \((i, j)\) connections

![Diagram of cluster placement QP formulation](image)
The issue with this is that it creates an asymmetrical Laplacian Matrix representation, with component to multiple clustered components, and vice versa, being assigned very large weights in their respective Degree Matrix (diagonal) entry, as well as large edge weights in their Adjacency Matrix entries.

For example, in the counter-MAPPED7.v example, the Adjacency Matrix entries for the connection between counter/U4, a reset buffer, and Cluster 0, (3, 9), and (9, 3) are -8, and the Degree Matrix entry for Cluster 0, (9, 9) is 26.

To correct this asymmetry it is necessary to also collapse component to cluster and cluster to component multiple connections to single ones.
Clustering Placement QP Formulation

- Each cluster is composed of an arbitrary set of components.
- \( N \) Components, \( M \) Clusters, \( T \) I/Os.
  - A matrix size is \((M + N) \times (M + N)\).
  - Pin connection matrix is \((M \times N) \times T\).
- Multiple (Component \( \rightarrow \) Cluster), (Cluster \( \rightarrow \) Component) connections are collapsed to a single (Component \( \rightarrow \) Cluster), (Cluster \( \rightarrow \) Component) connections.
  - \(-1\) or \(-\text{weight}\).
- Thus, diagonal A entries \((i, i)\) should sum the total of relevant (Component \( \rightarrow \) Cluster) or (Cluster \( \rightarrow \) Component) connections.
  - Magnitude of diagonal entries \((i, i)\), per row, must be > sum of \((i, j)\) connections.
Clustered Placement QP Formulation

- each pin may be connected to multiple cluster components
- multiple (Pin → Cluster), (Cluster → Pin) connections are collapsed to a single (Pin → Cluster), (Cluster → Pin) connections
- each cluster is represented by a single pin connection column
- Cluster cells may be connected to multiple cells of another cluster
- multiple (Cluster → Cluster) connections must also be collapsed to single (i, j) connections in Laplacian Matrix
  - -1 or –weight
- Thus, diagonal A entries (i, i) should **consider collapsed** (Cluster → Cluster) connections
  - Magnitude of diagonal entries (i, i), per row, must be > sum of (i, j) connections

![Diagram showing clustered placement QP formulation](image)
The Adjacency Matrix is created based on the I/O, output to input and input to output nets. Gate connections are counted twice, e.g. a->e (output->input), and e->a (input to output).

Similar to Graph adjacency matrix, or Component CCs for both input and output connections

Gates with multiple outputs? We discuss this issue later!

For the purple cluster, what are the changes to the Matrix?
Laplacian matrix HPWL does not see nets net1, net2 as individual nets
Lumped HPWL is effectively used, except for individual net weights, which may be used, to compensate
Analytical Net Models

- Weights Applied Directly on the Laplacian Matrix

- LHS: (a) Net, (b) Hyperedge, (c) Clique, (d) Star
- RHS: (a) routed Net, BB Model (for HPWL), MST, Steiner Tree
In the P2P model, only output to input connections are modelled. Thus, it is possible for the solution to be uneven lengths to output CCs. Adding Clique or Star Points can alleviate this issue. Linearisation can also be achieved by appropriate factoring of Quadratic lengths.
Star Model requires an Initial Placement!
Analytical Net Models

Typical net models include clique or star models, as shown in Figure 18.2 for a five-pin net. In the clique model, each \( k \)-pin net is replaced by \( k(k-1)/2 \) two-pin nets. In the star model, a star node is added for each net to which all pins of the nets are connected. If the weight of a \( k \)-pin hyperedge is \( W \), it is common to weight the set of two-pin nets using a weight such as \( W/(k-1) \).
Analytical Net Models

### TABLE 17.1

<table>
<thead>
<tr>
<th>Model</th>
<th>Steiner(V)</th>
<th>Clique(V)</th>
<th>Star(V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BB(V)</td>
<td>$\frac{k}{2}$</td>
<td>$\frac{k}{2}$</td>
<td>$\frac{k}{2}$</td>
</tr>
<tr>
<td>Steiner(V)</td>
<td>$\frac{k}{2}$</td>
<td>$\frac{k}{2}$</td>
<td>$\frac{k}{2}$</td>
</tr>
<tr>
<td>Clique(V)</td>
<td>$\frac{k}{2}$</td>
<td>$\frac{k}{2}$</td>
<td>$\frac{k}{2}$</td>
</tr>
<tr>
<td>Star(V)</td>
<td>$\frac{k}{2}$</td>
<td>$\frac{k}{2}$</td>
<td>$\frac{k}{2}$</td>
</tr>
</tbody>
</table>

The following result tells how well the other three net models approximate the length of an optimum minimal Steiner tree. For two-terminal nets, all the net models are identical.

**Theorem 1.** Let $V$ be a finite set of points in $\mathbb{R}^d$ and $n = |V| \geq 3$. Then Table 17.1 shows an upper bound on $\frac{C}{BB}$ for net models $M_1$ (rows) and $M_2$ (columns) for $BB$, Steiner, Clique, and Star.

As an example how to read Table 17.1, the entry in the second row and third column says that $Steiner(V) \leq \frac{3}{2} Clique(V)$ for all $V$ and $Steiner(V) \leq \frac{3}{2} Clique(V)$ if $n \neq 4$. All inequalities are essentially tight for all $n$. This result is due to Bremner and Vicsek (2001).

- **Steiner(V) ≤ Clique(V) ≤ Star(V) for n ≠ 4**
- **BUT:** Clique is very compute intensive as Matrix is dense
- **Hybrid Net Model**
  - For net size < 4 use Clique, ≥ 4 use Star

Hybrid Model – see Fastplace placer – both models are proven equivalent for scaled clique weight

Typical net models include clique or star models, as shown in Figure 18.2 for a five-pin net. In the clique model, each $k$-pin net is replaced by $k(k - 1)/2$ two-pin nets. In the star model, a star node is added for each net to which all pins of the nets are connected. If the weight of a $k$-pin hyperedge is $W$, it is common to weight the set of two-pin nets using a weight such as $W/(k - 1)$

Clique introduces $k^2$ non-zero additional entries in Laplacian Matrix
Star introduces $k$ non-zero additional entries in Laplacian Matrix
Hybrid Model – see Fastplace placer – both models are proven equivalent for scaled clique weight
Analytical Net Models

- **Weights Applied Directly on the Laplacian Matrix**
- **Clique Model**
  - Utilizes all possible two-pin connections of a net
  - For a net with \( p \) vertices a complete graph is considered, where the weight of each edge is \( \frac{1}{|p| - 1} \) or \( \frac{2}{|p|} \)
  - Number of tree edges is \((|p| - 1)\)
  - Number of clique graph edges is \(\sum \text{Clique edges is model}\)
  - Thus, under these models, for a clique of size \( p \) we have
    \[
    \frac{2}{|p|} \frac{1}{2} |p| (|p| - 1) = |p| - 1 \quad \text{or} \quad \frac{1}{(|p| - 1) \frac{1}{2} |p|} (|p| - 1) = \frac{1}{2} |p|
    \]

The clique model stems from the Analytical Placement Cost Function, which computes all point to point WL
Analytical Net Models

- **Weights Applied Directly on the Laplacian Matrix**
  - **Clique Model**
    - Utilizes all possible two-pin connections of a net
    - For a net with \( p \) vertices a complete graph is considered, where the weight of each edge is \( \frac{1}{|p| - 1} \)
    - Number of tree edges is \((|p| - 1)\)
    - Number of clique graph edges is:
      - Quadratic Complexity on Net Size

- **Star Model**
  - Introduces an additional star pin per net (median)
  - Connects each net pin to the star pin
  - A net consisting of \( p \) vertices will be represented by \(|p| - 1\) edges

The notion of net models was introduced by Partitioning Algorithms. If a net on \(|e|\) vertices is represented as a complete graph on \(|e|\) vertices and if \(|e|\) is large, the \(|e|\) vertices will likely be placed in the same block after bipartitioning. The result is that nets with small numbers of vertices may be cut because of the predominance of large nets [SK72]. The solution to this problem is to weight the graph edges of the net so that regardless of how vertices in the graph are partitioned, the sum of the weights of the edges cut should be as close to 1 as possible. Generally, it will not be possible for this sum to be exactly 1 as the theorem below states.
Bound2Bound Net Model

- Linear complexity – Matrix modification
- Net consists of one or more two-pin connections
- For each connection, weight is determined by:

\[
\omega_{B2B}^{x,pq} = \begin{cases} 
0, & \text{if pin } p \text{ and pin } q \text{ are inner pins} \\
\frac{2}{P-1} \frac{1}{x_p^{pin} - x_q^{pin}}, & \text{else.} 
\end{cases}
\]

Clique Model adds internal net lengths
Bound2Bound does not
Furthermore, all possible two pin connections are separated into the following three categories:
1. connections between the two boundary pins \((p = 1, q = 2)\),
2. connections between the “left” boundary pin 1 and the inner pins \((p = 1, q \geq 3)\),
3. and connections between the “right” boundary pin 2 and the inner pins \((p = 2, q \geq 3)\).

The inner two-pin connections \((p \geq 3, q > 3)\) are not considered, as they have a connection weight of zero.
For minor HPWL changes, it is best to incrementally update HPWL.
Incremental HPWL Computation

- For minor HPWL changes, it is best to incrementally update HPWL

Set of moved components
For minor HPWL changes, it is best to incrementally update HPWL.
Incremental HPWL Computation

For minor HPWL changes, it is best to incrementally update HPWL.
For minor HPWL changes, it is best to incrementally update HPWL.
Placement Density Map as a Analytical Force Directed Method

Fig. 6. Illustration of one placement iteration. The numbers in the big arrows represent the sequence of the steps executed in each placement iteration. (d), (e) Density plots, with white and black colors representing low and high densities, respectively. (a) Starting placement, (b) Hold force, (c) Resulting placement. (d) Supply and demand system $U$. (e) Potential $\Phi$. (f) Target points and move force.
Understanding Density Map Properties

1. Density(Bin Area) = Density(Bin Util.) x (Binw * Binh)

2. Gradient(Bin Area) = Gradient(Bin Util.) x (Binw * Binh)

Density Map (D-S) must be balanced:
To balance, Component Area must be multiplied by (1/utilisation)
Default (empty) Supply, per bin is -1

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Density</th>
<th>Gradient</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>3</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>-1</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

Density(Bin Area) = Density(Bin Util.) x (Binw * Binh)
Gradient(Bin Area) = Gradient(Bin Util.) x (Binw * Binh)
The distribution of the cells is modeled by a Demand-Supply System D.

\[ D(x, y) = D^{dem}(x, y) - D^{sup}(x, y) \]

- The \( D^{dem}(x, y) \) refers to the cells and the \( D^{sup}(x, y) \) refers to the placement area (usually the core).
- Cells are moved away from high density regions (high DEMAND) to low-density regions (high SUPPLY).
Density Map - 2

- Due to the fact that $D_{dem} < D_{sup}$, the system $D$ must be balanced and adapted as it is presented, below:

$$\int_{-\infty}^{+\infty} D_{dem}(x, y) dx dy = \int_{-\infty}^{+\infty} D_{sup}(x, y) dx dy$$

- To formulate the demand, a rectangle function $R$ is used:

$$R(x, y, x_{ll}, y_{ll}, w, h) = \begin{cases} 
1, & \text{if } 0 \leq x - x_{ll} \leq w \\
& \land \ 0 \leq y - y_{ll} \leq h \\
0, & \text{elsewhere}
\end{cases}$$

- where,
  - $x, y$ represent all the points inside the rectangle
  - $x_{ll}, y_{ll}$ represent either the center or the left corner of a module/cell
  - $w, h$ represent the width and height of the cell, accordingly.
Density Map - 3

- **Cell \(i\) demand**
  \[
  D_{cell}^{dem}(x, y) = d_{cell, i} \ast R(x, y, x'_{i} - \frac{w_{i}}{2}, y'_{i} - \frac{h_{i}}{2}, w_{i}, h_{i})
  \]

- The individual module density \(d_{cell, i}\) is usually set to 1.

- **Cell \(i\) supply**
  \[
  D_{cell}^{sup}(x, y) = d_{sup} \ast R(x, y, x_{chip}, y_{chip}, w_{chip}, h_{chip})
  \]

- **Supply density:**
  \[
  d_{sup} = \sum_{i=1}^{M+F} \frac{(d_{cell, i} \ast A_{m})}{A_{chip}}
  \]
Density Map - 4

- \( D \) is interpreted as a charge distribution and creates an electrostatic potential \( \varphi \), based on Poisson’s equation:

\[
\left( \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) \varphi(x, y) = -D(x, y)
\]

- Solving the equation determines target points:

\[
\dot{x}_i = x'_i - \frac{\partial \varphi(x, y)}{\partial x}\bigg|_{(x'_i, y'_i)}
\]

- Potential \( \varphi \) Gradients are collected in a vector:

\[
\Phi = \left( \frac{\partial \varphi}{\partial x}\bigg|_{(x'_i, y'_i)}, \frac{\partial \varphi}{\partial x}\bigg|_{(x'_2, y'_2)}, ..., \frac{\partial \varphi}{\partial x}\bigg|_{(x'_M, y'_M)} \right)^T
\]
Density Map - 5

- correct preset supply value is indeed - I
  - i.e. negative bin unit supply
- Demand was NOT originally adapted to the Supply
  - When this is the case, the two surface integrals are not identical, i.e. there is more Supply than Demand
  - This will lead to an uneven poisson solution, particularly for Dirichlet boundary conditions, where the "more empty space" will create deep troughs (double integral of space produces potential, so maximises at its centre)
    - as a consequence causes spreading saturation.
  - This behaviour was indeed observed when using Dirichlet boundary constraints
  - Neumann boundary constraints do not exhibit this behaviour, due to the explicit gradient (zero) constraint.
Density Map - 6

- to adapt demand to supply, \( \text{total supply} = \text{total demand} \)
  - component area must be inflated by \( (1/\text{utilisation}) \)
  - as \( \text{utilisation} = \text{demand/supply} \)
  - \( \text{demand} = \text{supply} \times (1/\text{utilisation}) \)
- The \( (1/\text{utilisation}) \) area scaling factor can then be converted to
  - a dimensions factor for components, as \( 1/\sqrt{\text{utilisation}} \)
  - \( \text{utilisation\_wh\_scaling\_factor} = 1/\sqrt{\text{utilisation}} \);
  // dimensions, i.e. width, height scaling factor for demand supply balancing //
  - \( \text{cw} = \text{cw} \times \text{utilisation\_wh\_scaling\_factor} \);
  - \( \text{ch} = \text{ch} \times \text{utilisation\_wh\_scaling\_factor} \);
  // scale component, cluster BB width, height by utilisation\_wh\_scaling\_factor, to adapt demand to supply //

// demand must be adapted to supply, so that their two surface integrals are identical, and Demand - Supply is balanced //
// Demand - Supply is interpreted as charge density, and is integrated twice to yield (1) force and (2) potential (work/unit charge) //
Density Map – 7

- Boundary Conditions
  - Dirichlet vs. Neumann

Original

Dirichlet
Quality Control - Acceleration

4.2 Quality Control

In order to control the important trade-off between the quality of placement and the CPU time, a quality control procedure is applied to the use of each version in global placement.

The simulation procedure is based on the following parameters. Typically, each circuit element is placed in a position that minimizes the total line length. In this case, the trade-off between quality of placement and CPU time is significant. The solution is obtained by using a combination of two algorithms: one for the initial placement and another for the optimization process. The initial placement is done by using a simple iterative algorithm, while the optimization process is based on a more complex algorithm that considers the placement of each circuit element as a separate entity.

4.3.2 Spring Stiffness of the Target Points

To control the number of iterations, a target point's spring constant $k$ is used. This constant is set based on the quality control and the available computational resources. A strategy is employed to continuously adjust $k$ during the optimization process. 

Figure 4.3.2 shows the iteration count $N_{iter}$ vs. the number of iterations $n$. The graph demonstrates the effectiveness of the optimization method in controlling the number of iterations. The results indicate a significant reduction in the number of iterations required for convergence, which is crucial for reducing the overall CPU time.

The improvement in quality is achieved by increasing the spring constant $k$, which forces the target points to converge faster and with higher accuracy. This approach allows for a better trade-off between quality of placement and CPU time by using a strategy that balances the need for accuracy with the limitations of computational resources.
Quality Control - Acceleration

- **GP Algorithm Acceleration**
  - Iteratively Calibrates Move Force to achieve a target movement
  - In addition, uses heuristic move weight scaling in first iteration
The interaction between the two, provided the bin size is set to the average cell area and combined with Neumann boundary conditions for the Density Map, achieve a good HPWL result with a small number of GP iterations. The iterative QC weight scaling, combined with the initial weight scaling, manage to accelerate the GP process around a larger value of target movements, without a significant loss in HPWL QOR. Neumann boundary is necessary, as gradient values will scale gracefully with design size, as Density Map height (initial) does not confine gradient values to smaller Density Map area.
The weight multiplier $1/(\text{total number of placeable elements})/(\text{average of bin width, bin height})$ is used to normalise, i.e. scale down the bin utilisation gradient value for the initial GP iteration; scaling avoids excess weight values being produced from the large initial bin utilisation gradient values; the latter are large due to placeable element stacking at the center of the Core Area, particularly for the B2B/B2B Net Model; no scaling would result in many upper movement bound violations in early iterations.

In scaling factor $(\text{total number of placeable elements})/(\text{average of bin width, bin height})$, the numerator corresponds to the maximum bin demand at the center of the core area for the B2B/B2B Net Model; this is identical to bin utilisation, if bin area corresponds to the average cell area, i.e. $\text{MAX}(\text{bin demand})$.

As for the gradient value computation, neighbouring bins are used, thus $dx$ used in the bin utilisation gradient $dy/dx$ is bin width or height; therefore, the gradient scaling factor value is $\text{MAX}(\text{bin demand})/\text{average } dx$.
Density @ centre = (D – 0)/Dx = (N/(N/Dx) – 0)/Dx = ~1
Quality Control - Acceleration

- **Original Acceleration Function**
  - $1 + \tanh(\log(1/x))$

- **Introduced Polynomial Acceleration which is more versatile**
  - $n/(x^2+1)$, where $n$ is a weight factor
  - Example on RHS is with $n=3$
  - $1 + \tanh(\log(1/x))$ is equivalent to $2/(x^2+1)$

- **Default is $2/(x^2+1)$**
  - User may specify weight, if needed
Modern Placement Challenges

- **High complexity**
  - Millions of objects to be placed

- **Placement constraints**
  - Preplaced blocks
  - Chip density, etc.

- **Mixed-size placement**
  - Hundreds/thousands of large macros with millions of small standard cells

- **Many more**
  - Datapath
  - 3D IC
  - Analog, etc.

- **2.5M placeable objects**
  - Mixed-size design

- **Macros in SoC Design**

- **Datapath Placement**

- **3DIC**

- **CE439 - CAD Algorithms II**

- **13/3/2018**
Chen, et al., "A high quality analytical placer considering preplaced blocks and density constraint," ICCAD-06 (TCAD-08)

- **Global Placement (GP)**: Computes the best position for each object to minimize the cost, e.g. WL, ignoring overlaps.
- **Legalization (LG)**: Removes all overlaps between objects.
- **Detailed Placement (DP)**: Improves the Legalised solution cost further.
HPBB Wirelength Models

- Linear (Golden) HPBB Model:

\[ W(x, y) = \sum_{e \in E} \left( \max_{v_i, v_j \in E} |x_i - x_j| + \max_{v_i, v_j \in E} |y_i - y_j| \right) \]

- Is neither smooth or differentiable
- Approximations
  - Quadratic (Gordian)
  - Lp-Norm
  - LSE (Log-Sum-Exponential), i.e. Soft Max Approximation
  - CHKS, etc.
Log-Sum-Exp (LSE) Wirelength Model

- **Soft Max Definition**
  - The soft maximum of two variables $x, y$ is the function:
    $$ g(x, y) = \ln(e^x + e^y) $$
  - The soft maximum approximates the hard maximum and is a convex function just like the hard maximum ($\text{max}$)
    - the accuracy of the soft maximum depends on scale, e.g. $g(1, 2) = 2.31$, but $g(10, 20) = 20.00004$
    - The *hardness* of the soft maximum can be controlled by a parameter $k$:
      $$ g(x, y) = \ln\left(e^{(kx)} + e^{(ky)}\right)/k $$

To see that the soft maximum approximates the hard maximum, note that if $x$ is a little bigger than $y$, $\exp(x)$ will be a lot bigger than $\exp(y)$. That is, exponentiation exaggerates the differences between $x$ and $y$. If $x$ is significantly bigger than $y$, $\exp(x)$ will be so much bigger than $\exp(y)$ that $\exp(x) + \exp(y)$ will essentially equal $\exp(x)$ and the soft maximum will be approximately $\log(\exp(x)) = x$, the hard maximum.
Log-Sum-Exp (LSE) Wirelength Model

- Introduced by Naylor in 2001

\[ W_{LSE}(x, y) = \gamma \sum_{e \in E} (\ln \sum_{v_i \in E} e^{\frac{x_i}{\gamma}} + \ln \sum_{v_i \in E} e^{-\frac{x_i}{\gamma}} + \ln \sum_{v_i \in E} e^{\frac{y_i}{\gamma}} + \ln \sum_{v_i \in E} e^{-\frac{y_i}{\gamma}}) \]

- whereby:

\[ \max(x) \equiv \gamma \ln \sum_{v_i \in E} e^{\frac{x_i}{\gamma}} - \min(x) = \max(-x) \]

- effective smooth, differentiable approximation for HPWL
- Approaches exact HPWL when \( \gamma \to 0 \)
- Has dominated modern placement for 10+ years!
Weighted-Average (WA) Model

- Hsu, Chang, Balabanov, DAC 2011

\[ W_{WA}(x, y) = \sum_{e \in E} \frac{\sum_{v_i \in e} x_i e^{\frac{x_i}{\gamma}}}{\sum_{v_i \in e} e^{\frac{x_i}{\gamma}}} - \frac{\sum_{v_i \in e} x_i e^{-\frac{x_i}{\gamma}}}{\sum_{v_i \in e} e^{-\frac{x_i}{\gamma}}} + \frac{\sum_{v_i \in e} y_i e^{\frac{y_i}{\gamma}}}{\sum_{v_i \in e} e^{\frac{y_i}{\gamma}}} - \frac{\sum_{v_i \in e} y_i e^{-\frac{y_i}{\gamma}}}{\sum_{v_i \in e} e^{-\frac{y_i}{\gamma}}} \]

- Ratio-ed LSE
- Weighted average of a set of x coordinates, xi, of a net i:
- X(xi) approximates the maximum value of xi:

\[ X(x_i) = \frac{\sum_{v_i \in E} x_i F(x_i)}{\sum_{v_i \in E} F(x_i)} = \frac{\sum_{v_i \in E} x_i e^{\frac{x_i}{\gamma}}}{\sum_{v_i \in E} e^{\frac{x_i}{\gamma}}} \]

- Effective smooth, differentiable approximation for HPWL
- Approaches exact HPWL when \( \gamma \to 0 \)
Other Smooth Wire Models

- **Quadratic Model:** BonnPlace, FastPlace, Kraftwerk, mFAR
  
  Not exact Wirelength Modelling

  \[ W_{QP}(x, y) = \sum_{e \in E} \left( \sum_{v_i, v_j \in e, i < j} w_{ij} (x_i - x_j)^2 + \sum_{v_i, v_j \in e, i < j} w_{ij} (y_i - y_j)^2 \right) \]

- **Lp-Norm Model**
  
  Approaches exact HPWL when \( p \to \infty \)

  Is compute intensive

  \[ W_{LPN}(x, y) = \sum_{e \in E} \left( \sum_{v_i \in E} x_i^p - \sum_{v_i \in E} x_i^{p-1} + \sum_{v_i \in E} y_i^p - \sum_{v_i \in E} y_i^{p-1} \right) \]
Wire Models Ranking

- Quadratic
- LSE
- LPN
- Quadratic
- LSE
- WA
- LPN
Y-axis is Wirelength – smooth 2 variable functions
Placement with Density Constraint

- Given the chip region and block dimensions, divide placement region into bins
- Determine \((x, y)\) for all movable blocks
  - \(\min W(x, y)\) -- wirelength function s.t.
  - 1. \(\text{Density}_{b}(x, y) \leq \text{MaximumDensity}_{b}\), for each bin \(b\)
  - 2. No overlap between blocks

Density Definitions:

\[
\text{Density(\text{Global})} = \frac{\sum \text{Area}_{\text{stdcells}}}{\sum \text{Area}_{\text{bins}}}
\]

\[
\text{Density(\text{bin } b)} = \frac{\sum_{\text{bin } b} \text{Area}_{\text{stdcells}}}{\text{Area}_{\text{bin } b}}
\]
Density Model

- Compute the block area of each bin to obtain the bin density

\[ D_b(x, y) = \sum_{v \in V} P_x(b, v) P_y(b, v) \]
Density Smoothing

- Apply bell-shaped function to make bin density function smooth (Kahng & Wang)

\[
D_b(x, y) = \sum_{v \in V} p_b(b, v) p_v(x, v) \\
D'_b(x, y) = \sum_{v \in V} c_v p_b(b, v) p_v(x, v)
\]

- Bell-shaped smooth function

- Continuous & differentiable

\[
p_b(b, v) = \begin{cases} 
1 - \frac{ad^2}{4b}, & 0 \leq d_x^2 + w_b, \\
b(b_x^2 - \frac{w_b^2}{2} - 2w_k)^2, & \frac{w_b^2}{2} + w_k \leq d_x^2 \leq \frac{w_b^2}{2} + 2w_k, \\
0, & \text{otherwise}
\end{cases}
\]

where

\[
a = 5/(4w_b + 2w_k), \\
b = 2/(w_b(w_b + 4w_k))
\]

- Bin density

- Continuous & differentiable

wb bin width, wv block width, cv normalization factor
Density Smoothing - Sigmoid

- \( f(l, x, u) = 1, \text{if } l < x < u \)

- \( p(t) = \frac{1}{1 + e^{-\alpha t}} \)

- \( f(l, x, u) \sim p(x - l)p(u - x) \)

- effective smooth, differentiable approximation for the bin density function

- Approximates exact 0-1 logic function when \( \alpha \to 0 \)
Density-based Placement Process

Increase density weight

Increase density weight

STOP! Spreading enough!
Analytical Placement Model

- **Global Placement Problem with Density Constraint**
  \[
  \min W(x, y) \\
  \text{s.t. } D_b(x, y) \leq M_b
  \]

- Minimise Wirelength, Db: density, Mb: max density of bin b

- **Constraints can be relaxed into the Objective Function**

  \[
  \min W(x, y) + \lambda \sum_{\text{bins } b} (D_b(x,y) - M_b)^2
  \]

- Use gradient methods to solve it

- Increase lambda gradually to reduce density penalty and find optimal (x, y) under density constraint
Gradient Solvers

- \( \min f(x) \)
- \( x_0 \leftarrow \) initial value
- Repeat until convergence
  - \( x_{i+1} = x_i - f'(x)_{x=x_i} \times \) stepsize

The function \( f \) must be differentiable and smooth to apply the gradient solver.
Dynamic Step-Size Control

- Step size is too large
  - May not converge to a good solution
- Step size is too small
  - Incur long running time
- Adjust the step size s.t. the average Euclidean movement of all blocks is a fixed value

\[ \text{stepsize } \alpha_k = \frac{s}{\|d_k\|_2} \]

- \(d_k\) is conjugate direction(s)
- \(s\) is a user-specified factor

Norm 2 is Euclidean Distance
Look-ahead Legalisation
<table>
<thead>
<tr>
<th>Placer</th>
<th>Wirelength Model</th>
<th>Overlap Model</th>
<th>Integration</th>
<th>Optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>APlace</td>
<td>LSE</td>
<td>Density</td>
<td>Penalty Method</td>
<td>Nonlinear</td>
</tr>
<tr>
<td>BonnPlace</td>
<td>Quadratic</td>
<td>Partitioning</td>
<td>Region Constraints</td>
<td>Quadratic</td>
</tr>
<tr>
<td>DPlace</td>
<td>Quadratic</td>
<td>Diffusion</td>
<td>Fixed Point</td>
<td>Quadratic</td>
</tr>
<tr>
<td>FastPlace</td>
<td>Quadratic</td>
<td>Cell Shifting</td>
<td>Fixed Point</td>
<td>Quadratic</td>
</tr>
<tr>
<td>FDP</td>
<td>Quadratic</td>
<td>Density</td>
<td>Fixed Point</td>
<td>Quadratic</td>
</tr>
<tr>
<td>Gordian</td>
<td>Quadratic</td>
<td>Partitioning</td>
<td>Region Constraints</td>
<td>Quadratic</td>
</tr>
<tr>
<td>hATF</td>
<td>Quadratic</td>
<td>Partitioning</td>
<td>Region Constraints</td>
<td>Quadratic</td>
</tr>
<tr>
<td>Kraftwerk2</td>
<td>Bound2Bound</td>
<td>Density</td>
<td>Fixed Point</td>
<td>Quadratic</td>
</tr>
<tr>
<td>mFAR</td>
<td>Quadratic</td>
<td>Density</td>
<td>Fixed Point</td>
<td>Quadratic</td>
</tr>
<tr>
<td>mPL6</td>
<td>LSE</td>
<td>Density</td>
<td>Penalty Method</td>
<td>Nonlinear</td>
</tr>
<tr>
<td>NTUPlace3</td>
<td>LSE/WA</td>
<td>Density</td>
<td>Penalty Method</td>
<td>Nonlinear</td>
</tr>
<tr>
<td>RQL</td>
<td>Quadratic</td>
<td>Cell Shifting</td>
<td>Fixed Point</td>
<td>Quadratic</td>
</tr>
<tr>
<td>SimPL</td>
<td>Bound2Bound</td>
<td>Partitioning</td>
<td>Region Constraints</td>
<td>Quadratic</td>
</tr>
<tr>
<td>Vassttu</td>
<td>LSE</td>
<td>Fixed Point</td>
<td>Fixed Point</td>
<td>Nonlinear</td>
</tr>
</tbody>
</table>