Lecture 12: Placement – II

Tsung-Wei (TW) Huang
Department of Electrical and Computer Engineering
University of Utah, Salt Lake City, UT

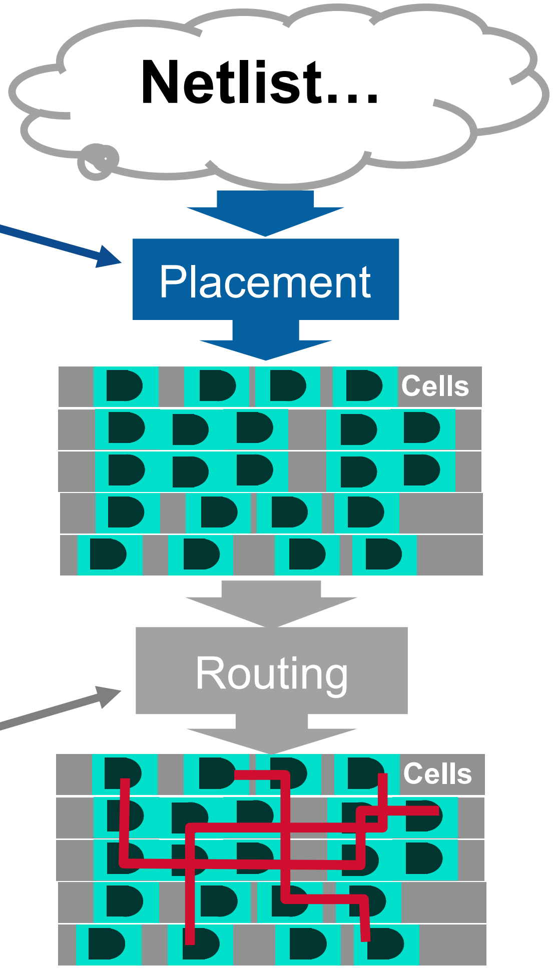
In-class Presentation: 10/19

- **Floorplan research presentation on 10/19 (in class)**
  - Xiaoping Tang and D. F. Wong, "FAST-SP: a fast algorithm for block placement based on sequence pair," *IEEE/ACM Asia and South Pacific Design Automation Conference,* 2001
  - Jackey Z. Yan and Chris Chu, “DeFer: Deferred Decision Making Enabled Fixed-Outline Floorplanning Algorithm,” *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 29(3): 367-381, 2010
- Upload your pptx to https://github.com/tsung-wei-huang/ece5960-physical-design/issues/10 before presentation

Recap: Placement Problem

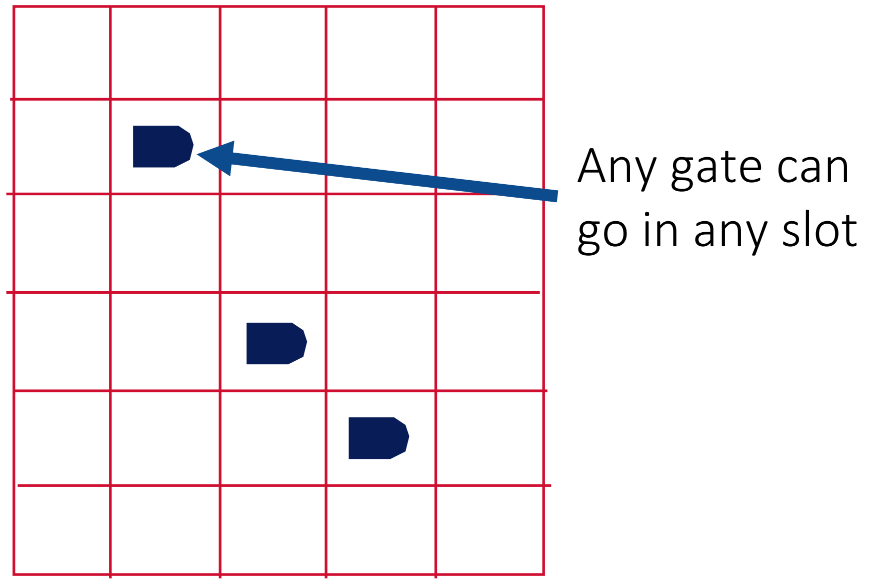
- **What does a placer do?**
  - **Input:** Netlist of gates & nets
  - **Output:** Exact location of each gate
  - **Goal:** Able to **route** (connect) all wires

- **Placement is VERY HARD!**
  - Bad placement → Much more wire
  - More wire: bigger, slower chip
  - If placement is *very* bad, next tool in the flow—the **router**—unable to connect all wires, or meet timing



Recap: Placement Problem (cont'd)

- **Grid-based model of the chip**
  - A simple grid – like a chess board
  - Cells (gates) go in grid slots
  - Pins (connect off-chip, fixed at edges)
- **Grid-based model of gates**
  - All gates are exactly the same size.
  - (Unrealistic, but simplifies things)
  - Each grid slot can hold 1 gate

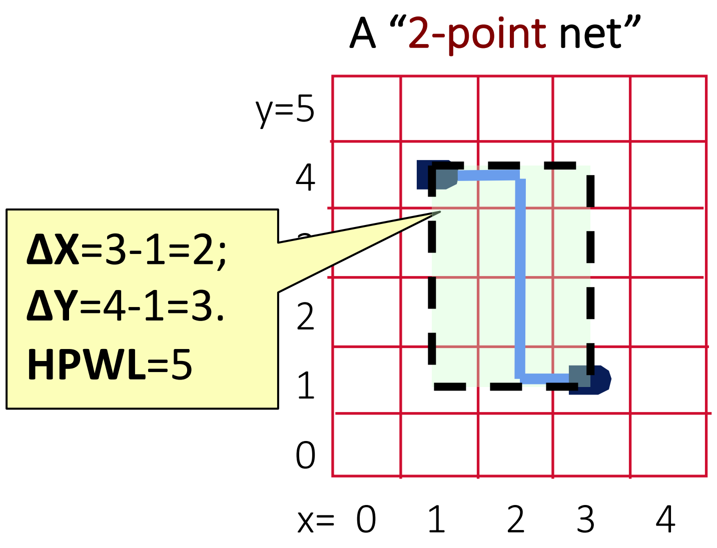


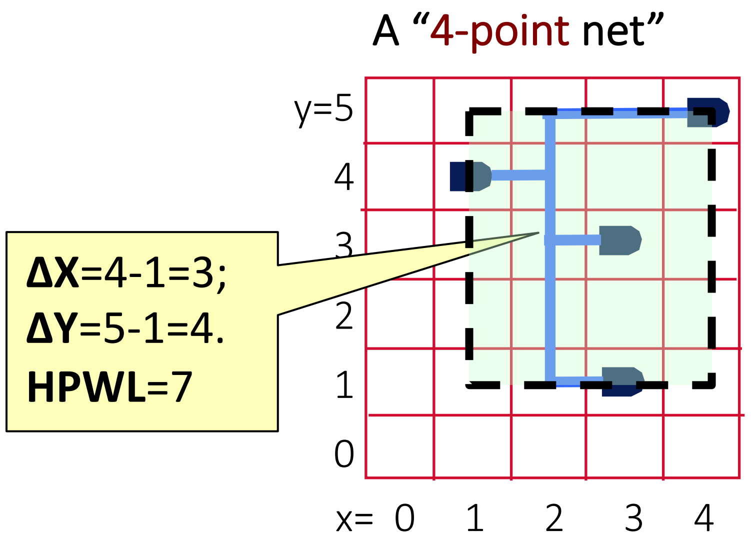
Recap: Placement Problem (cont’d)

- **Placer optimizes the ability of router to connect all the nets**
  - But routers are computationally expensive tools. We can’t run one “inside” placer
  - We need a simplified **approximation at this stage**
- **Every real placer minimizes expected wirelength**
  - For each wire in the design, **estimate the expected length** of the routed wire
  - Minimize this objective: $\sum_{\text{wires } W_i}$ **EstimatedLength(Wi)**
- **Placer finds gate locations to minimize this objective**

Recap: Wirelength Estimation

- **Most placers adopt Half-Perimeter Wirelength (HPWL)**
  - Also know as Bounding Box (BBOX) wirelength
  - Put smallest "bounding" box around all gates
  - Assume gate lives in "center" of the grid slot
  - Add width (ΔX) and height (ΔY) of the BBOX for the wirelength estimate





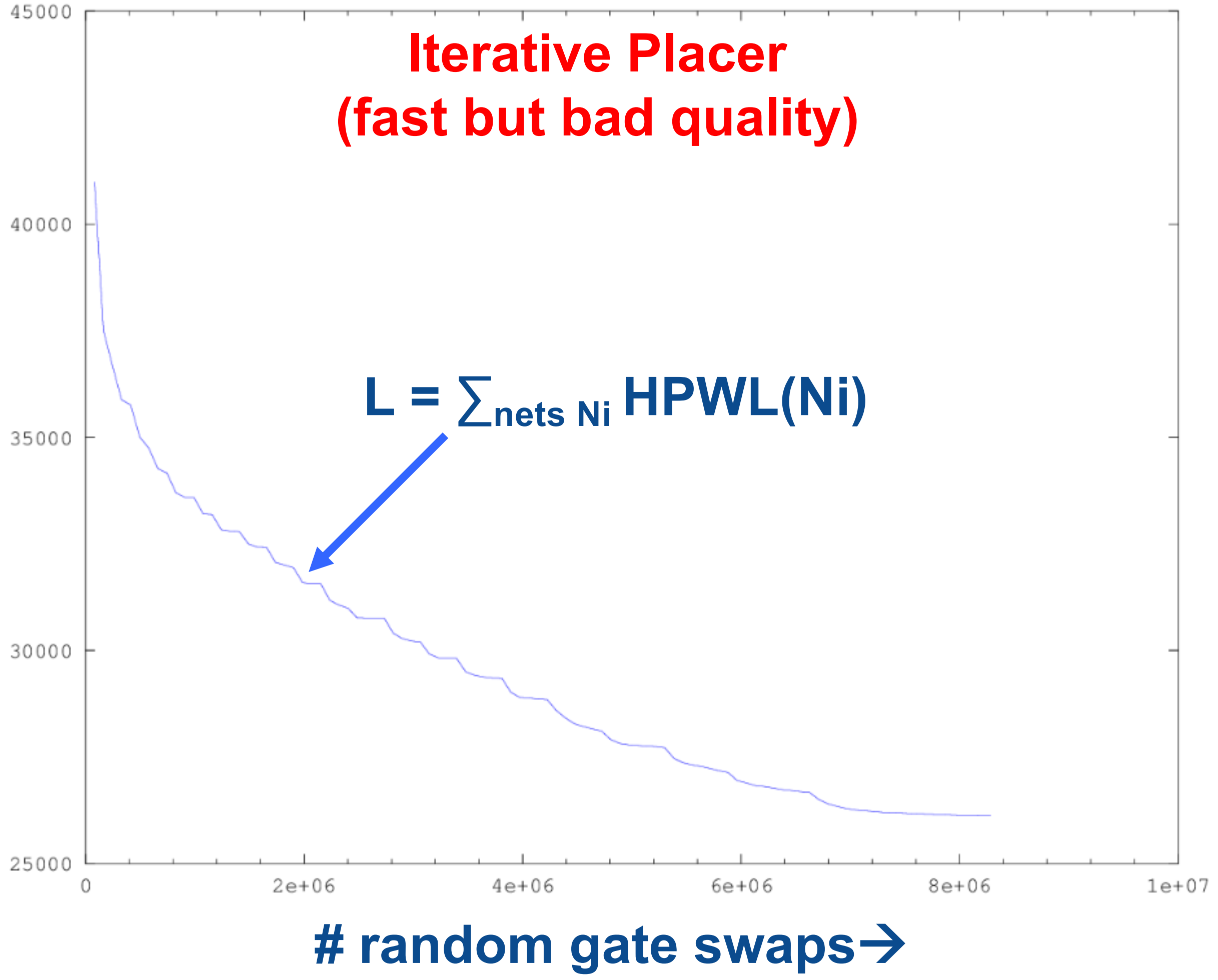
Recap: Wirelength Estimation (cont'd)

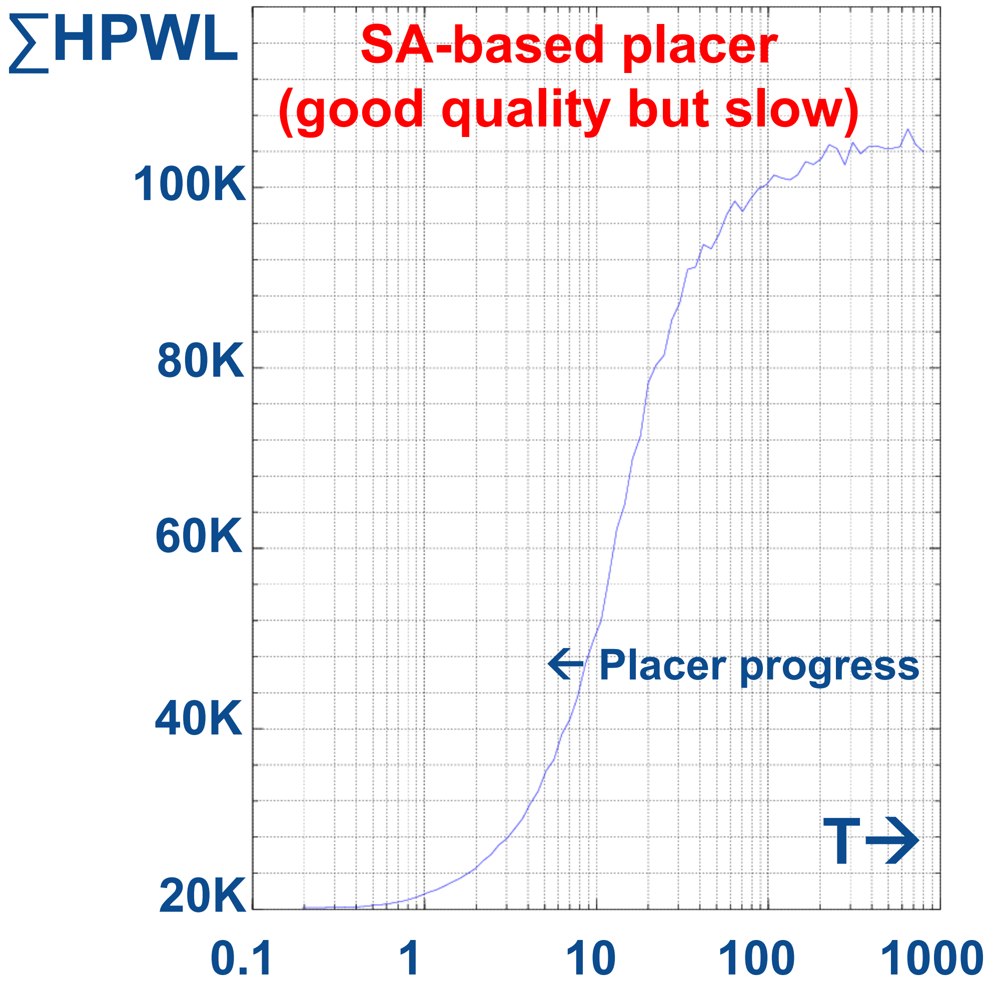
- **Easy to calculate, even for a multi-point net:**

  [max{X coordinates of all gates) – min{X coordinates of all gates}]
  + [max{Y coordinates of all gates) – min{Y coordinates of all gates}]

- **Always a lower bound on the real wire length**
  - No matter how complex the final routed wire path is…
  - …you need at least this much wire to connect everything
  - Aside: all wiring on big chips (and most boards) is strictly horizontal & vertical – no "arbitrary angles" for manufacturing reasons which is another reason HPWL is good

Recap: Placement Algorithm



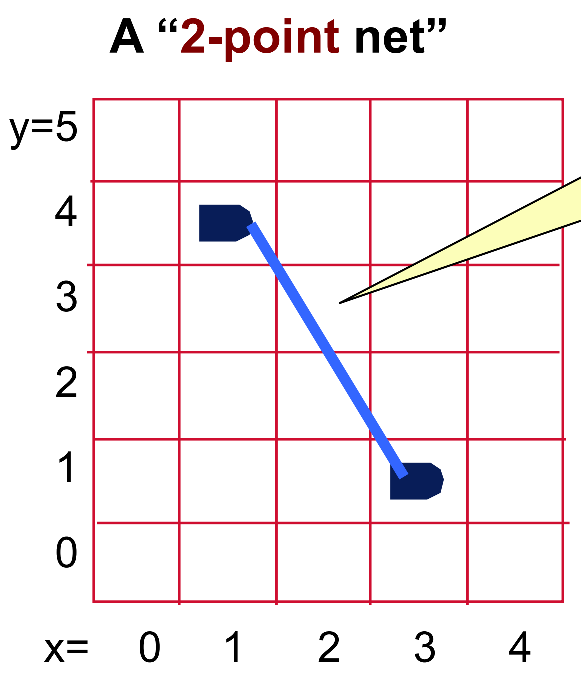


Analytical Placer

- **Write an *equation* whose *minimum* is the placement**
  - If you have a million gates, need a million **(xi, yi)** values as result
  - Formulate an appropriate **cost function** for all the gate-level **(xi, yi)**:
    $F(x_1, x_2, \dots x_{1M}, y_1, y_2, \dots y_{1M})$
  - Solve analytically for $X^*=(x_1, x_2, \dots x_{1M})$, $Y^*=(y_1, y_2, \dots y_{1M})$ to minimize $F(x_1, x_2, \dots x_{1M}, y_1, y_2, \dots y_{1M})$
  - The resulting values of **X*, Y*** give you the placement of all 1M gates
- **This sounds sort of crazy… but it works great**
  - All modern placers for big ASICs and SOCs are “analytical”
  - Big trick is write the wirelength in mathematically “friendly” form we can optimize

Key: Optimize *Quadratic* Wirelength Model

- For 2-point net, we optimize squared length of “distance” line between points: **$(x1-x2)^2 + (y1-y2)^2$**



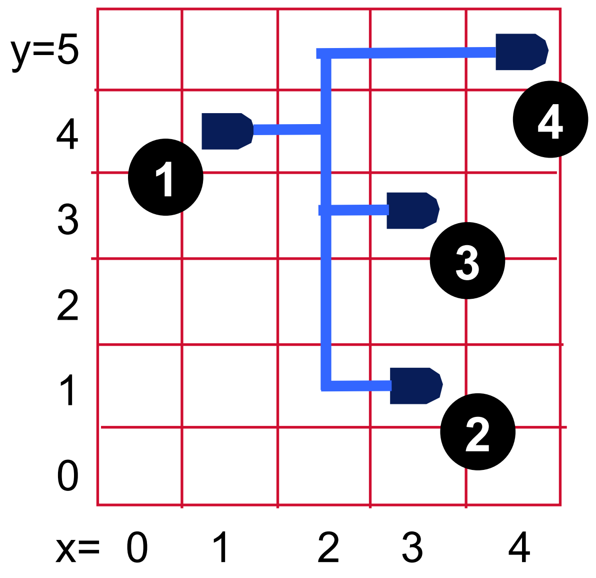
Quadratic wirelength
$=(3-1)^2 + (4-1)^2$
$=13$

BUT… what happens if your net has *more* than 2 points in it?

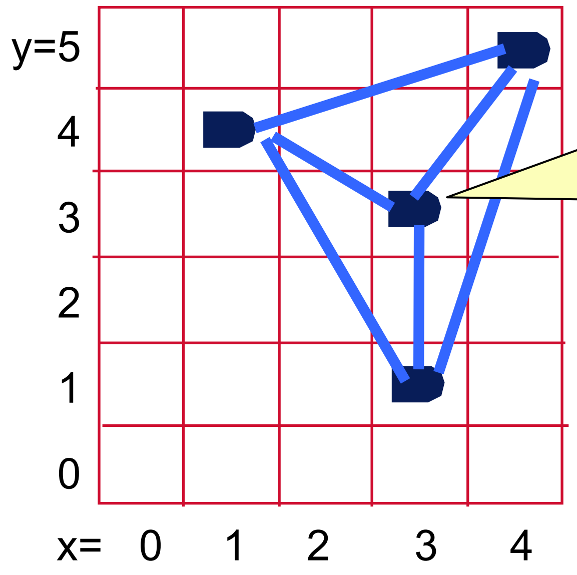
What About k-point Net, k>2?

- Replace one "real" net with **k(k-1)/2** 2-point nets and add a new net between every pair of points–*called a fully-connected clique model*

A "k=4-point net"



Clique model

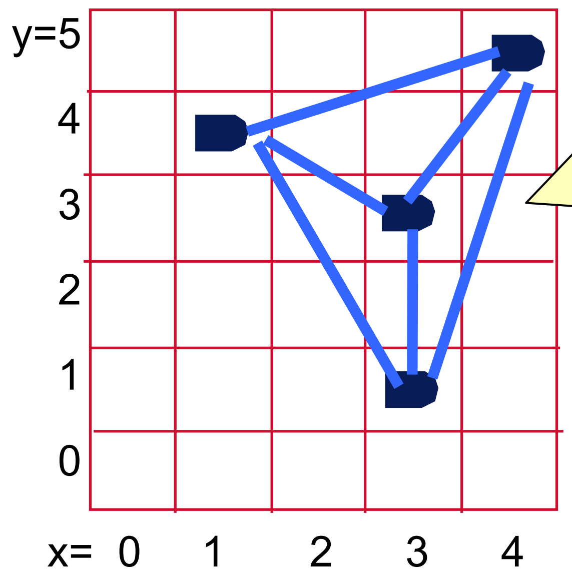


k(k-1)/2
=4(4-1)/2
=6 2-point nets

What About k-point Net, k>2? (cont'd)

- **Each new 2-point net is *weighted* by 1/(k-1)**
  - Why? 1 net became k(k-1)/2 nets. Need to ***compensate*** so we don't "overestimate"

Clique model



Quadratic estimate:

$$
\begin{aligned}
&(1/3)[(4-1)^2 + (5-4)^2] \\
&+(1/3)[(4-3)^2 + (5-1)^2] \\
&+(1/3)[(3-1)^2 + (4-1)^2] \\
&+(1/3)[(3-1)^2 + (4-3)^2] \\
&+(1/3)[(4-3)^2 + (5-3)^2] \\
&\underline{+(1/3)[(3-3)^2 + (3-1)^2]}
\end{aligned}
$$

=sum of 6 weighted 2-point lengths

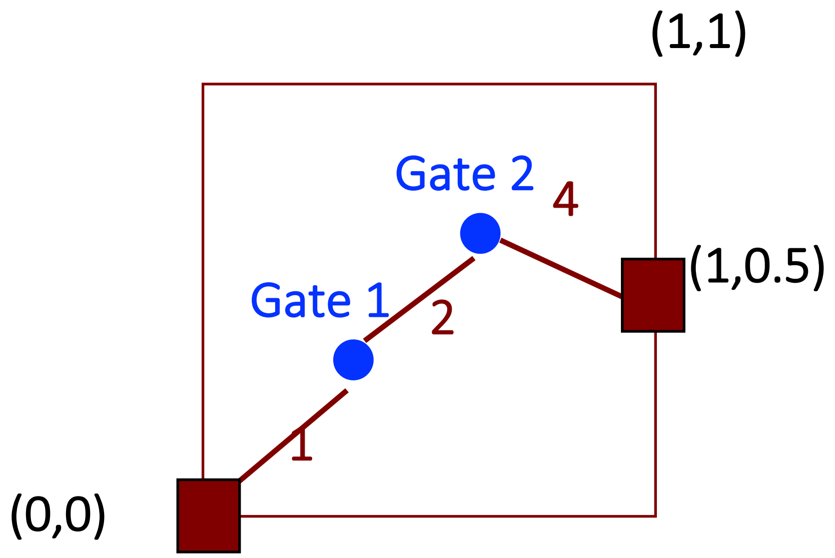
Note also: when k=2, this weight is just (2-1)=1, so no special treatment for common 2-point nets

Gates as Points

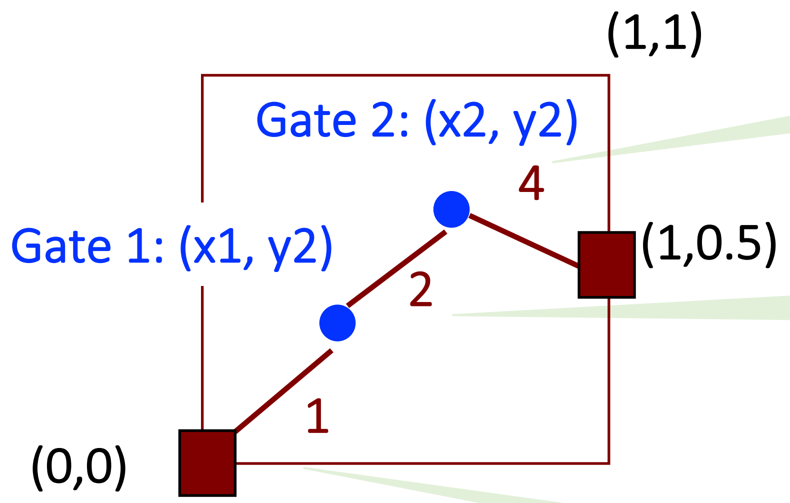
- **To make the math work out easily, one more simplification:**
  - Ignore the physical size of all the gate – pretend *gates are dimensionless points*
  - And, we will *ignore* (for now…) constraint that gates can't go on top of each other
- **Sounds strange… Why?**
  - Allow us to write a very simple, very elegant "equation" for the placement
  - Allow us to solve the problem very quickly and effectively

Easiest to See with Small Example

- **Chip surface is a rectangle**
  - **X** from 0 to 1; **Y** from 0 to 1
  - This it totally arbitrary, btw
- **2 gate "points", index 1 and 2**
- **3 nets, each with a weight**
  - Each net is 2 points to keep manual example small and easy
  - Weights are **1, 2, 4** in diagram
- **2 pads**
  - **Pad** = **fixed** pin (red square) on the edge of the chip. These do not move.



Easy to Write the Quadratic Wirelength



$4(x2-1)^2 + 4(y2-0.5)^2$

$2(x2-x1)^2 + 2(y2-y1)^2$

$1(x1-0)^2 + 1(y1-0)^2$

Next step: How do we optimize this equation?

How Do We Minimize This?

- **Basic calculus! Differentiate, set derivative to 0, then solve!**
  - But this is multiple variables? So, we do *partial derivatives*, set each to 0, solve x and y independently!

Q(X): $4(\mathbf{x2}-1)^2 + 2(\mathbf{x2}-\mathbf{x1})^2 + 1(\mathbf{x1}-0)^2$

$$\Downarrow$$

$$\begin{aligned} \partial Q/\partial \mathbf{x1} &= 0 + 4(\mathbf{x2}-\mathbf{x1})(-1) + 2(\mathbf{x1}) \\ &= 6\,\mathbf{x1} - 4\,\mathbf{x2} = 0 \end{aligned}$$

$$\begin{aligned} \partial Q/\partial \mathbf{x2} &= 8(\mathbf{x2}-1) + 4(\mathbf{x2}-\mathbf{x1}) + 0 \\ &= -4\,\mathbf{x1} + 12\,\mathbf{x2} - 8 = 0 \end{aligned}$$

Q(Y): $4(\mathbf{y2}-0.5)^2 + 2(\mathbf{y2}-\mathbf{y1})^2 + 1(\mathbf{y1}-0)^2$

$$\Downarrow$$

$$\begin{aligned} \partial Q/\partial \mathbf{y1} &= 0 + 4(\mathbf{y2}-\mathbf{y1})(-1) + 2(\mathbf{y1}) \\ &= 4\,\mathbf{y1} - 4\,\mathbf{y2} - 8 = 0 \end{aligned}$$

$$\begin{aligned} \partial Q/\partial \mathbf{y2} &= 8(\mathbf{y2}-0.5) + 4(\mathbf{y2}-\mathbf{y1}) + 0 \\ &= -4\,\mathbf{y1} + 12\,\mathbf{y2} - 4 = 0 \end{aligned}$$

How Do We Minimize This? (cont'd)

- These are linear equations! We know how to solve these!
  - Two matrix equations: $Ax=b_x$ and $Ay=b_y$ → N gates gives NxN matrix
  - Same matrix for X,Y, different b vectors → X, Y, b are Nx1 vectors

Q(X): $4(x2-1)^2 + 2(x2-x1)^2 + 1(x1-0)^2$

Minimize

$$\begin{bmatrix} 6 & -4 \\ -4 & 12 \end{bmatrix} \begin{bmatrix} x1 \\ x2 \end{bmatrix} = \begin{bmatrix} 0 \\ 8 \end{bmatrix}$$

x1 = 0.571 **x2** =0.857

Q(Y): $4(y2-0.5)^2 + 2(y2-y1)^2 + 1(y1-0)^2$

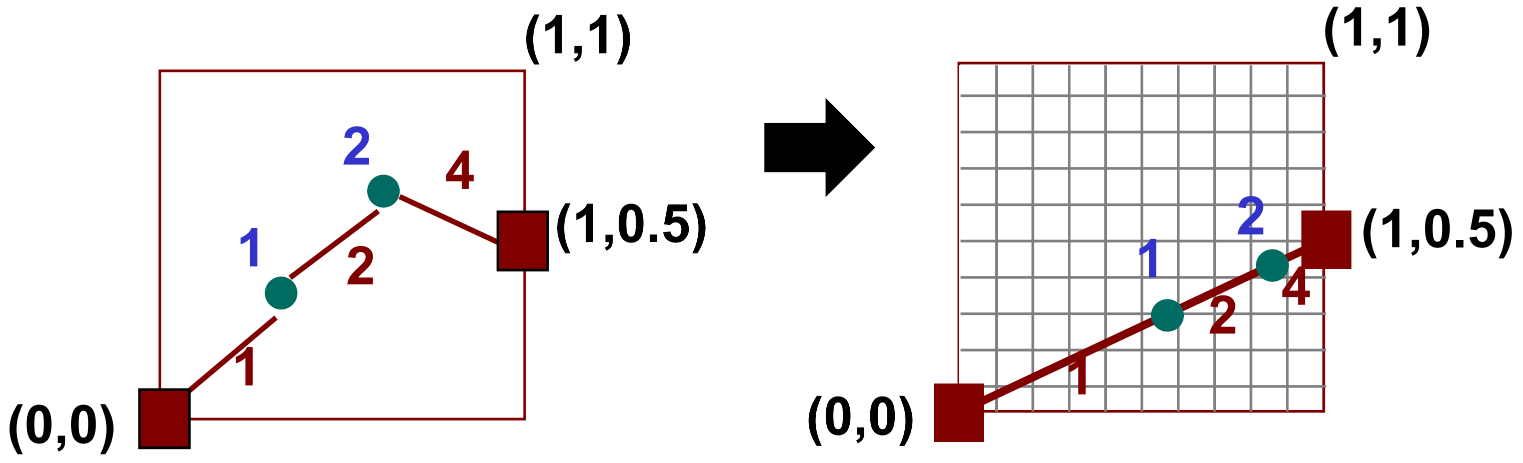
Minimize

$$\begin{bmatrix} 6 & -4 \\ -4 & 12 \end{bmatrix} \begin{bmatrix} y1 \\ y2 \end{bmatrix} = \begin{bmatrix} 0 \\ 4 \end{bmatrix}$$

y1 = 0.286 **y2** =0.429

Placement Result

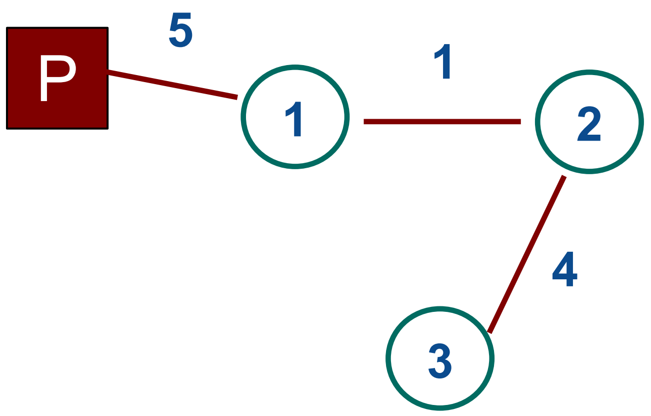
- **Observation: placement result makes visual sense!**
  - All points on a straight line between the pads
  - Each 2-point wire is like a spring–*placement minimizes all spring lengths*
- **Bigger** weight on the wire → **shorter** wire. Gives us lots of control over placement
- ***Same*** matrix, ***different*** right-hand-side **b** vectors. Why? Different **x, y** pad coordinates



x1 = 0.571 **x2** =0.857
y1 = 0.286 **y2** =0.429

What is Matrix A? (cont'd)

- **Surprisingly simple recipe to build the required A matrix**
  - First, build the **NxN** connectivity matrix, called **C**
  - If gate **i** has a 2-point wire to gate **j** with weight **w**, **c[i,j] = w**, else = **0**
- **Another example of 3 gates, 3 wires, and 1 I/O pad (P)**

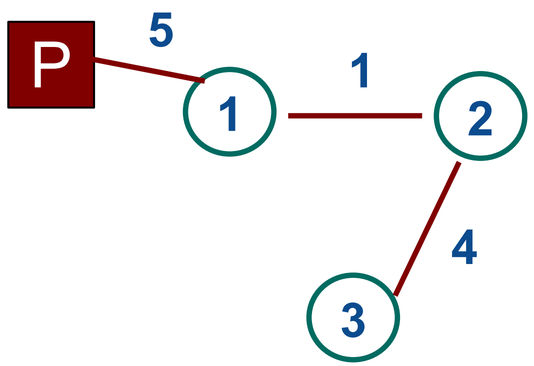


$$C = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 4 \\ 0 & 4 & 0 \end{bmatrix}$$

Note: **C** matrix ignores the pads

What is Matrix A?

- **Use the connectivity C matrix to build A matrix**
  - Elements **a[i,j]** not on the matrix diagonal are just **a[i,j] = -c[i,j]**
  - Elements on the diagonal are **a[i,j]=** $\sum_{j=1,n}$ **c[i,j] + (weight of any pad wire)**
    - …ie, add up the **i**th row of **C** and then add in weight on a (possible) wire to pad

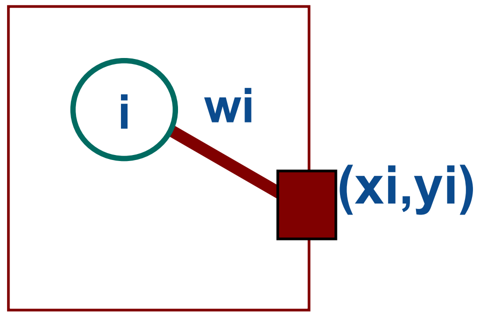


$$C = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 4 \\ 0 & 4 & 0 \end{pmatrix} \qquad A = \begin{pmatrix} 6 & -1 & 0 \\ -1 & 5 & -4 \\ 0 & -4 & 4 \end{pmatrix}$$

diagonal

Note: **A** matrix accounts for pads

What is Vector b?



- For **$Ax = b_x$** vector:
  - If gate **i** connects to a pad at **(xi, yi)** with a wire with weight **wi**
  - Then set **$b_x[i] = wi \cdot xi$**

$$\begin{bmatrix} \quad A \quad \end{bmatrix} \begin{bmatrix} x \end{bmatrix} = \begin{bmatrix} b_x \end{bmatrix}$$

ith element of b_x vector

- For **$Ay = b_y$** vector:
  - If gate **i** connects to a pad at **(xi, yi)** with a wire with weight **wi**
  - Then set **$b_y[i] = wi \cdot yi$**

$$\begin{bmatrix} \quad A \quad \end{bmatrix} \begin{bmatrix} y \end{bmatrix} = \begin{bmatrix} b_y \end{bmatrix}$$

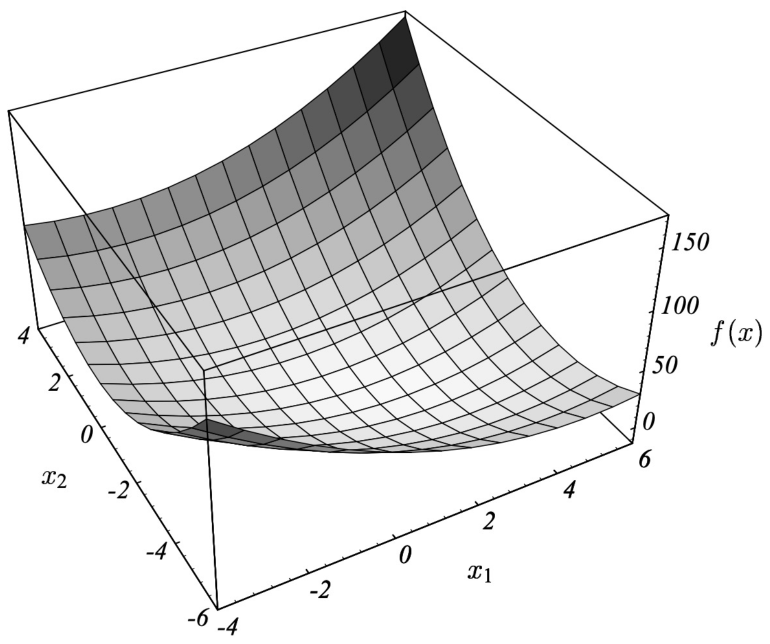
ith element of b_y vector

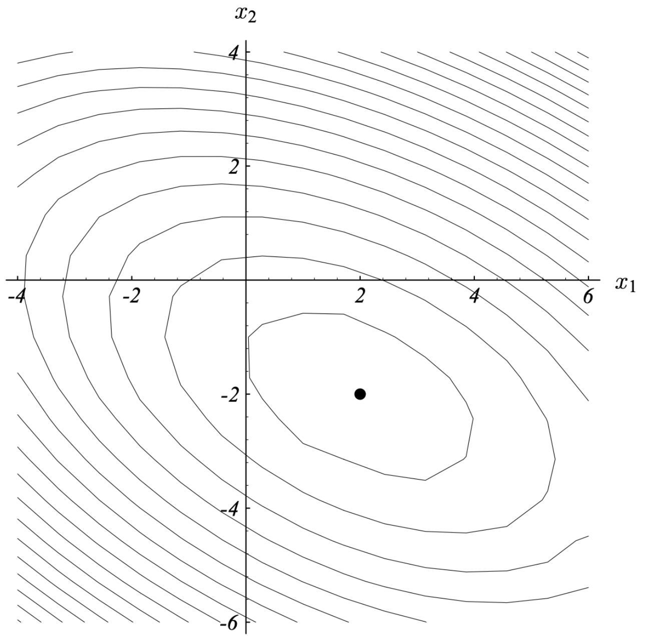
How to Solve Ax=b?

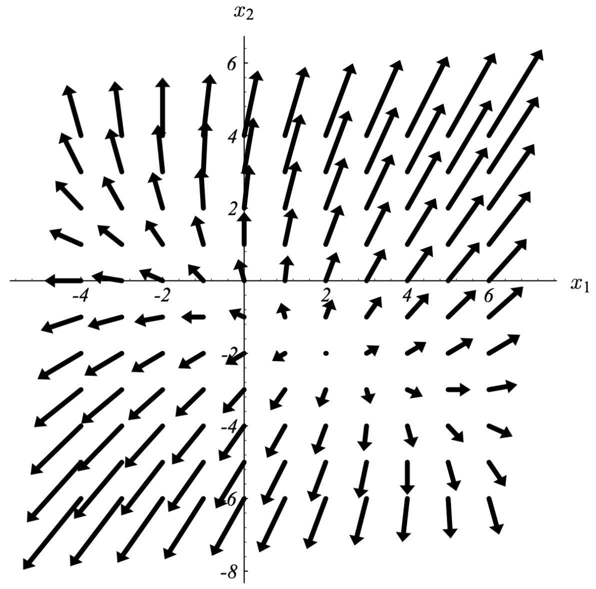
- **Are these *difficult* to do, in practice?**
  - If we have 1M gates, this is a 1M x 1M **A** matrix, with 1M element **x** and **b** vectors!
- **No – these are VERY EASY to solve, even when very large**
  - The **A** matrix has a special form—*it is sparse, symmetric, diagonally dominant*
  - Mathematically: **A** is positive semi-definite—*very simple to solve!*
  - We use iterative, approximate solvers, in practice (i.e., not Gaussian elimination but techniques like conjugate gradient)
    - This means the solver converges gradually to the right answer
    - But, also means that the answers can be a little bit “off”, not quite perfect

Conjugate Gradient

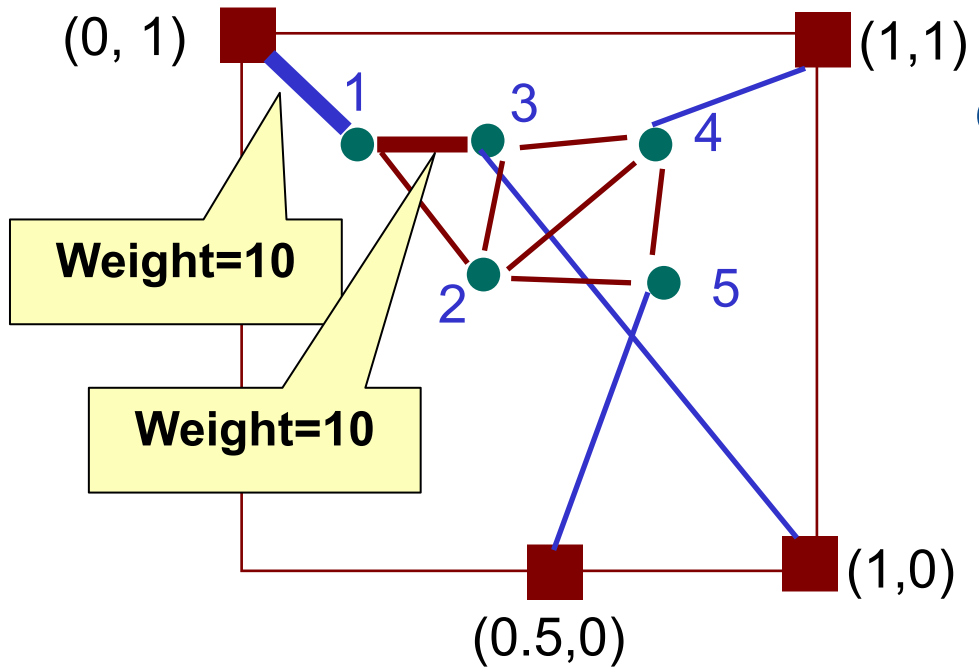
- Practical analytical placers apply conjugate gradient
  - https://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf







Another Example – Building A



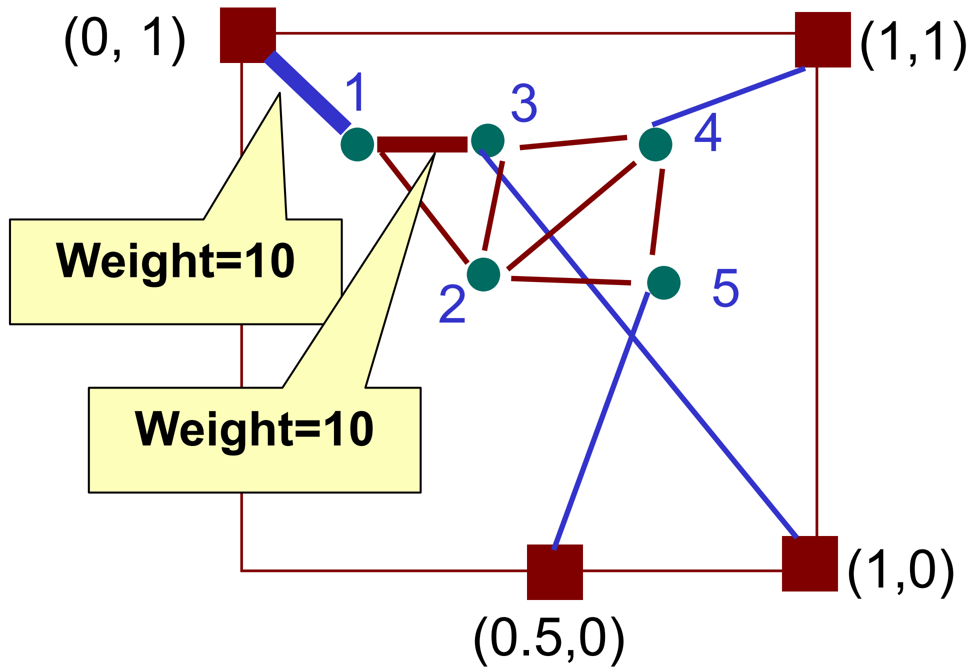
$$C = \begin{pmatrix} 0 & 1 & 10 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 10 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix} \qquad A = \begin{pmatrix} 21 & -1 & -10 & 0 & 0 \\ -1 & 4 & -1 & -1 & -1 \\ -10 & -1 & 13 & -1 & 0 \\ 0 & -1 & -1 & 4 & -1 \\ 0 & -1 & 0 & -1 & 3 \end{pmatrix}$$

Elements **a[i,j]** not on the matrix diagonal are just **a[i,j] = -c[i,j]**

Elements on the diagonal are **a[i,j]=** $\sum_{j=1,n}$ **c[i,j] + (weight of any pad wire)**

All wire weights = 1 *except* two highlighted: **gate1** to **pad** and **gate1** to **gate2**

Another Example – Building b

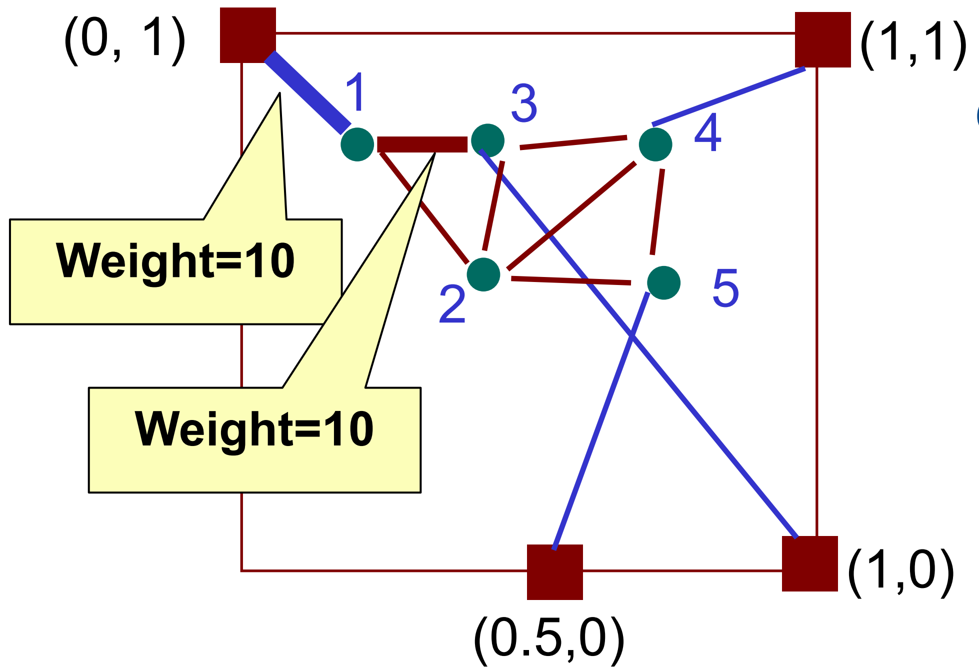


All wire weights = 1 *except* two highlighted:
gate1 to **pad** and **gate1** to **gate2**

- For **Ax = b_x** vector (similarly for y):
  - If gate **i** connects to a pad at **(xi, yi)** with a wire with weight **wi**
  - Then set **b_x[i] = wi • xi**

$$b_x = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0.5 \end{bmatrix} \qquad b_y = \begin{bmatrix} 10 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Another Example

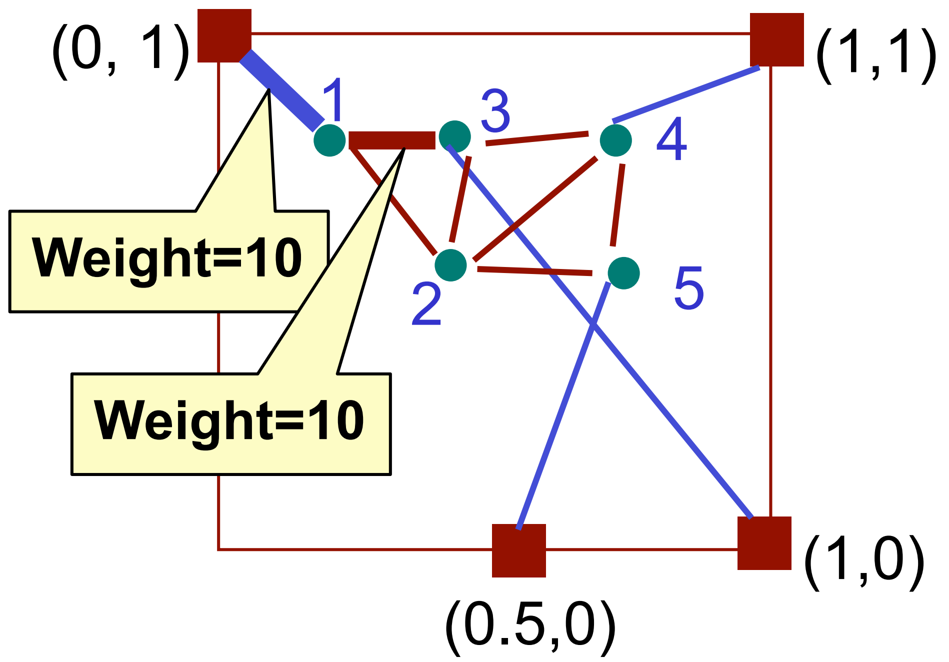


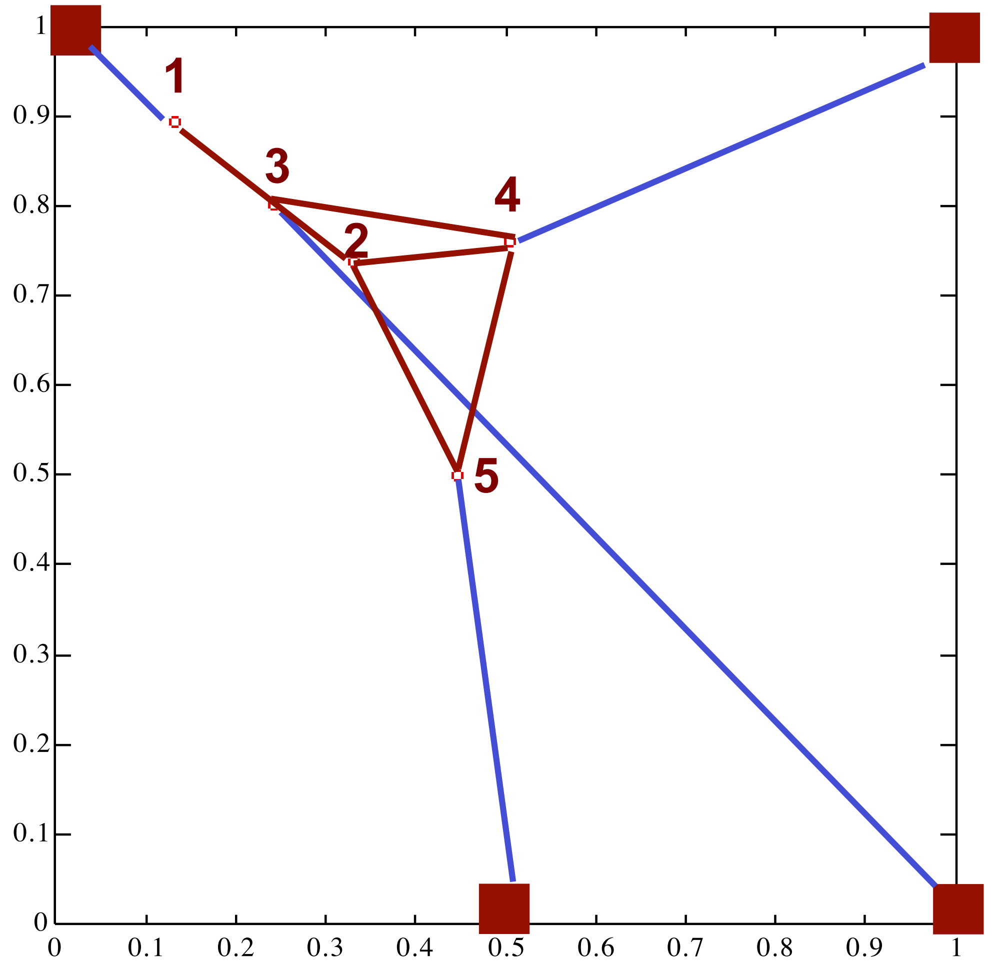
$$C = \begin{pmatrix} 0 & 1 & 10 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 10 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix} \qquad A = \begin{pmatrix} 21 & -1 & -10 & 0 & 0 \\ -1 & 4 & -1 & -1 & -1 \\ -10 & -1 & 13 & -1 & 0 \\ 0 & -1 & -1 & 4 & -1 \\ 0 & -1 & 0 & -1 & 3 \end{pmatrix}$$

$$b_x = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0.5 \end{pmatrix} \qquad b_y = \begin{pmatrix} 10 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

All wire weights = 1 *except* two highlighted:
gate1 to **pad** and **gate1** to **gate2**

Another Example: Result





Summary

- **We have discussed a new analytical placement algorithm**
- **We have discussed the quadratic wirelength model**
  - Construct the matrix A (NxN) for a placement problem of N gates
  - Construct the vector b_x and the vector b_y
  - Formulate the placement problem into a linear system
  - Solve the linear system and get the gate locations
- **We will dive into the quadratic placement problem next**