Lecture 14: Placement – IV

Tsung-Wei (TW) Huang

Department of Electrical and Computer Engineering

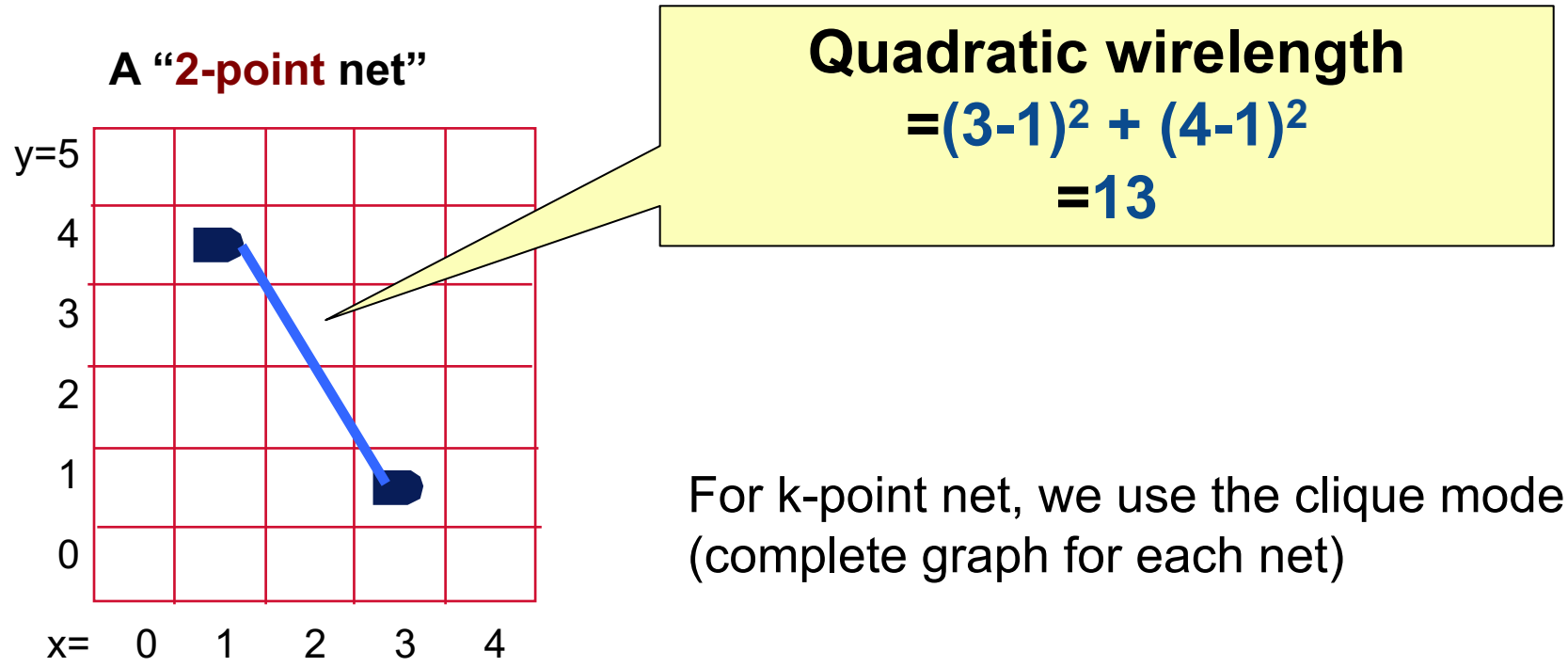
University of Utah, Salt Lake City, UT

Recap: 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)**:
    $$F(x_1, x_2, \ldots x_{1M}, y_1, y_2, \ldots y_{1M})$$
  - Solve analytically for $X^*=(x_1, x_2, \ldots x_{1M})$, $Y^*=(y_1, y_2, \ldots y_{1M})$ to minimize **F( )**
  - 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

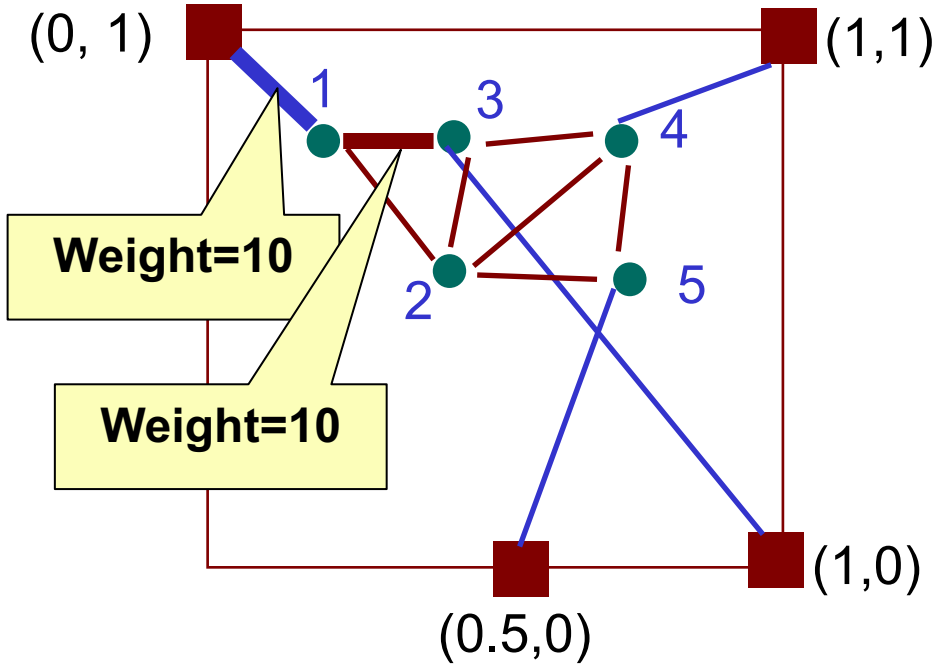
Recap: Quadratic Wirelength Model

- We optimize squared length of "distance" line between points: $(x1-x2)^2 + (y1-y2)^2$



For k-point net, we use the clique model (complete graph for each net)

Recap: Quadratic Placement Formulation



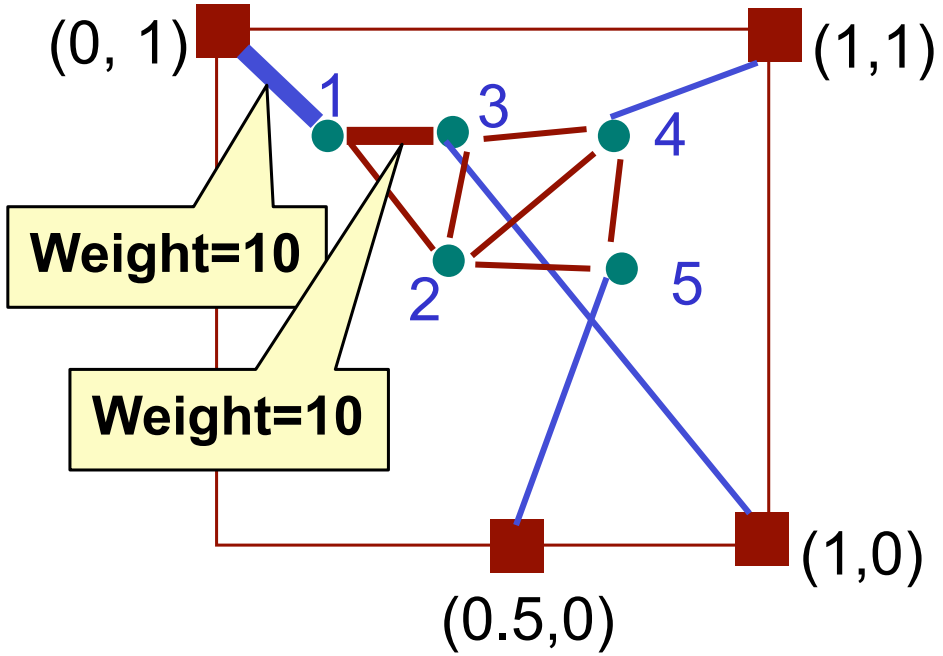
All wire weights = 1 *except* two highlighted:
gate1 to **pad** and **gate1** to **gate2**

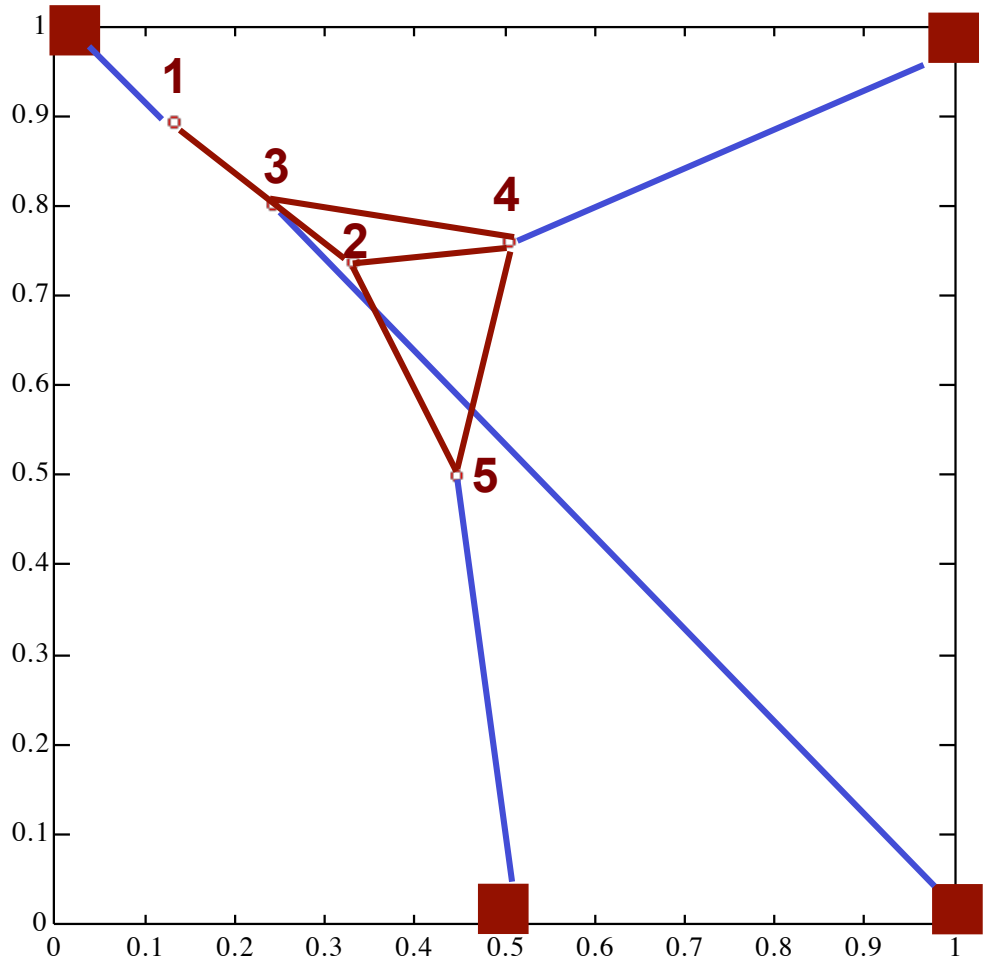
$$C = \begin{pmatrix} 0 & 1 & 10 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 10 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix} \quad A = \begin{pmatrix} 21 & -1 & -10 & 0 & 0 \\ -1 & 4 & -1 & -1 & -1 \\ -10 & -1 & 13 & -1 & 0 \\ 0 & -1 & -1 & 4 & -1 \\ 0 & -1 & 0 & -1 & 3 \end{pmatrix}$$

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

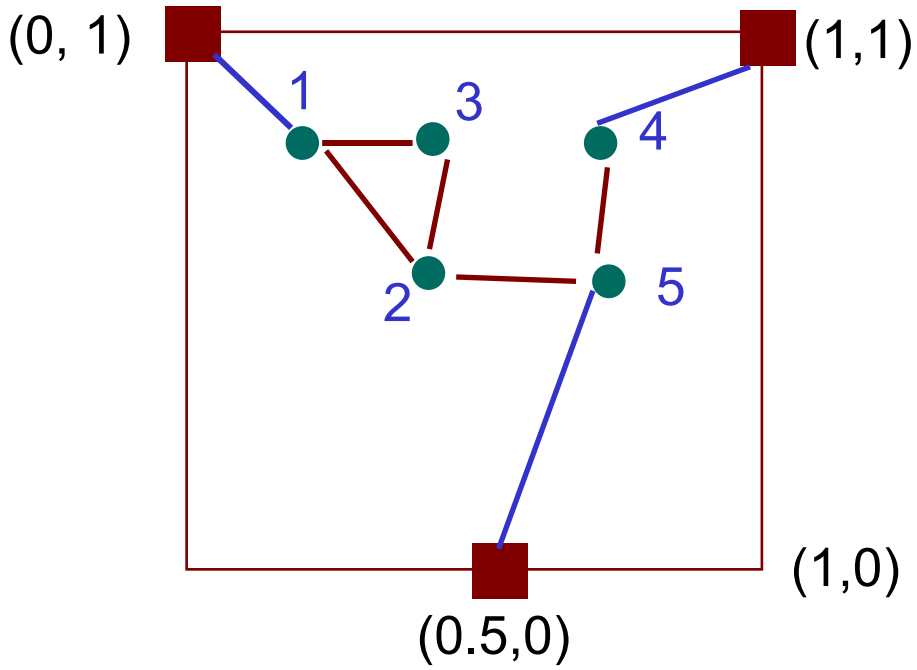
Recap: Quadratic Placement Result

Placement → Solving a linear system

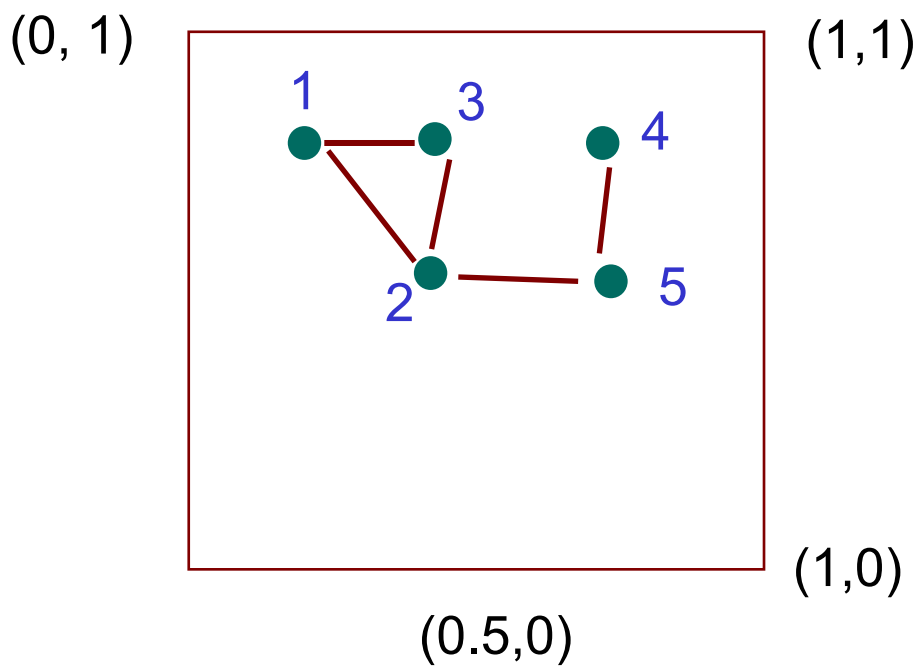




Recap: Two Practice Examples



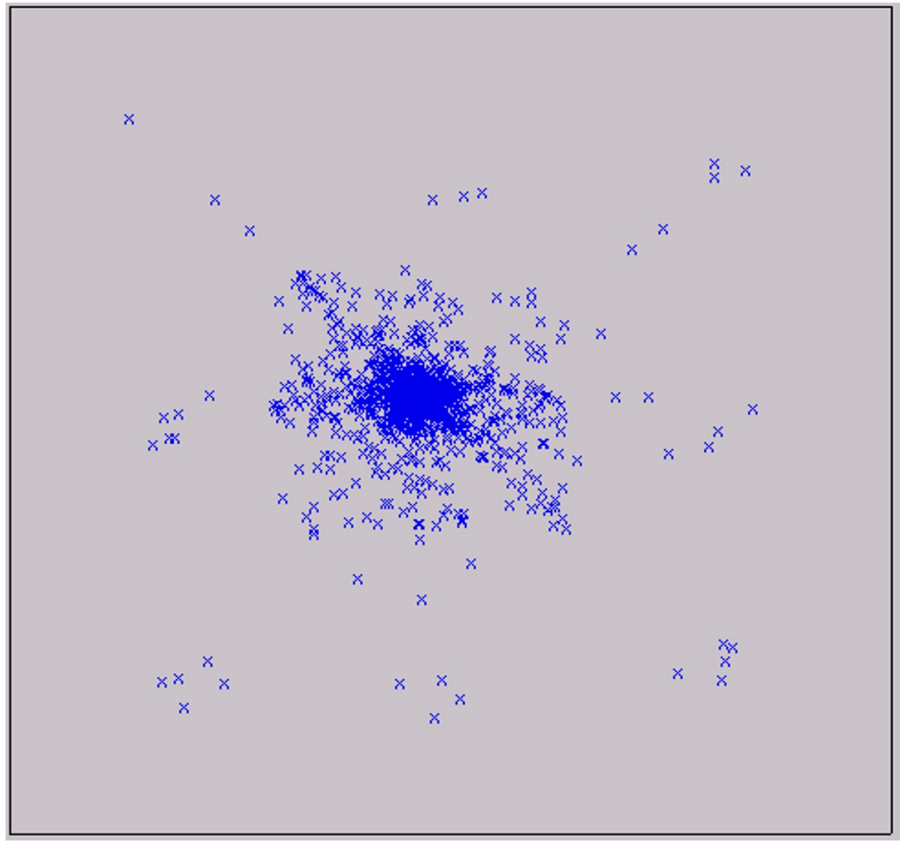
All wire weights = 1



All wire weights = 1

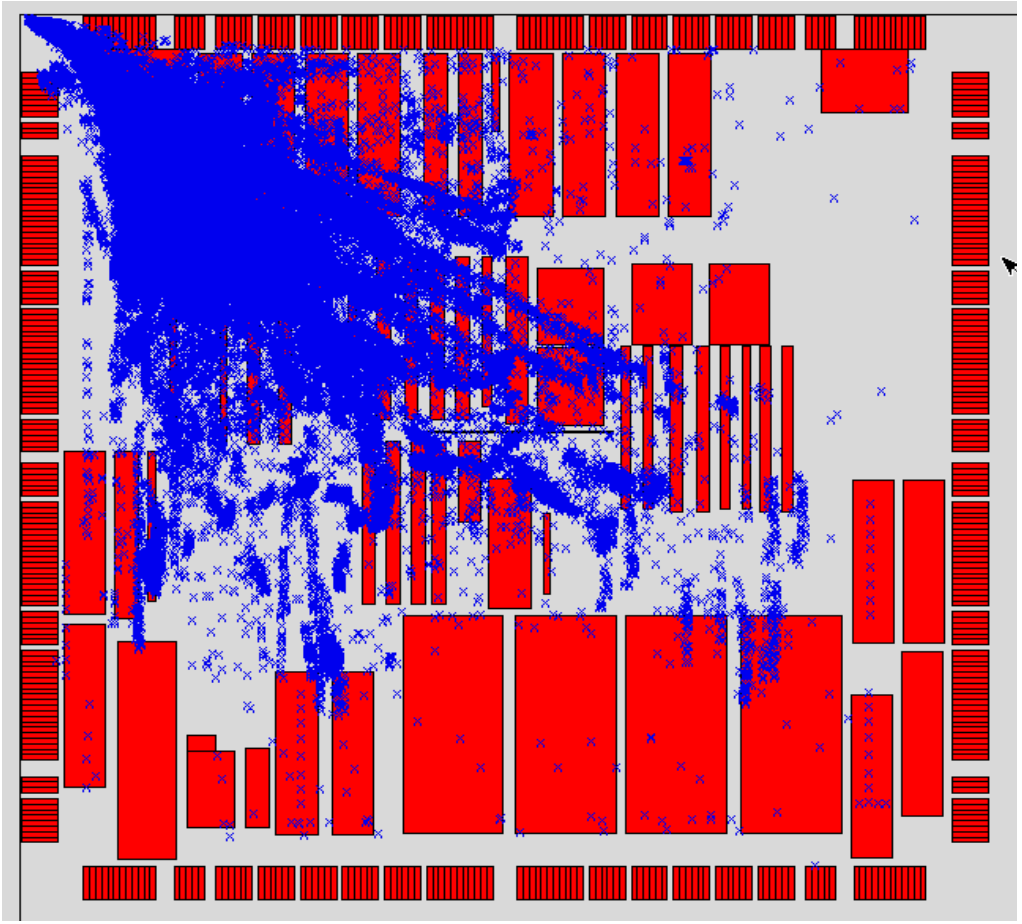
How does Quadratic Placement Look?

- **Like the right figure:**
  - Small IBM ASIC, few thousand gates
  - All lumped together to give the minimum wirelength
- **New problem:**
  - Quadratic model minimizes wirelength for big netlists, in a numerical way
  - But ignores that gates have physical size, cannot be on top of each other
  - Now, we have to fix this…
  - Our solution: recursive partitioning

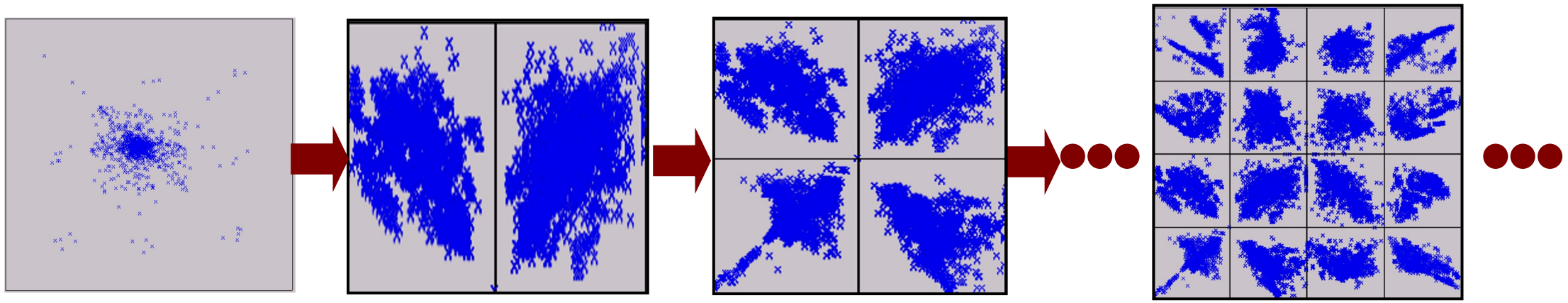


A Bigger Industrial Example

- **This benchmark is from IBM**
  - 210,904 gates (**blue**)
  - 543 fixed blocks (**red**) –like SRAMS
  - Image shows where gates “want” to go if we model blocks like “big pads”
  - This is a quadratic placement of gates
  - Additional problems: gates need to go between these blocks!
- **Why we are showing this?**
  - Example of how badly imbalanced the quadratic placement can be



Recursive Partitioning



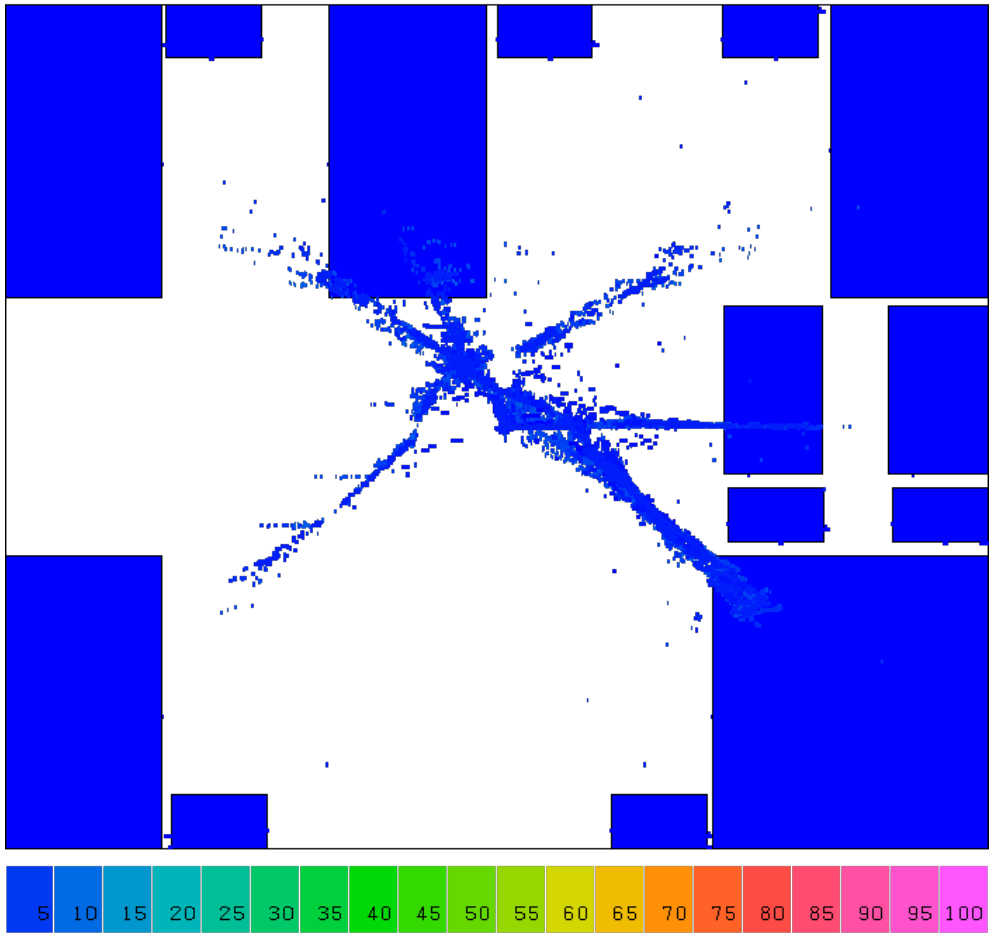
1st quadratic place (**QP**) solve

Partition chip into left / right.
Select which gates on each side.
Solve 2 new, smaller **QP** tasks

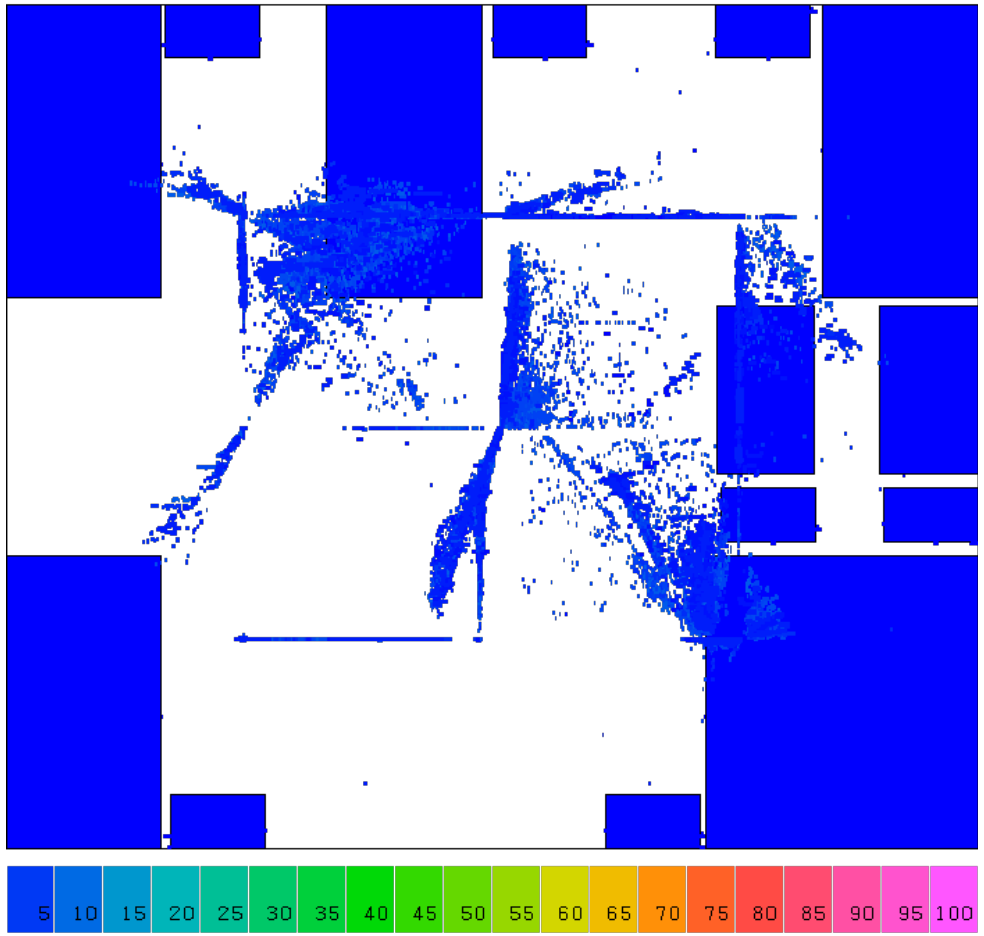
Repeat.
Partition each side top/bottom.
Solve 4 new smaller **QP** solves

Just keep going

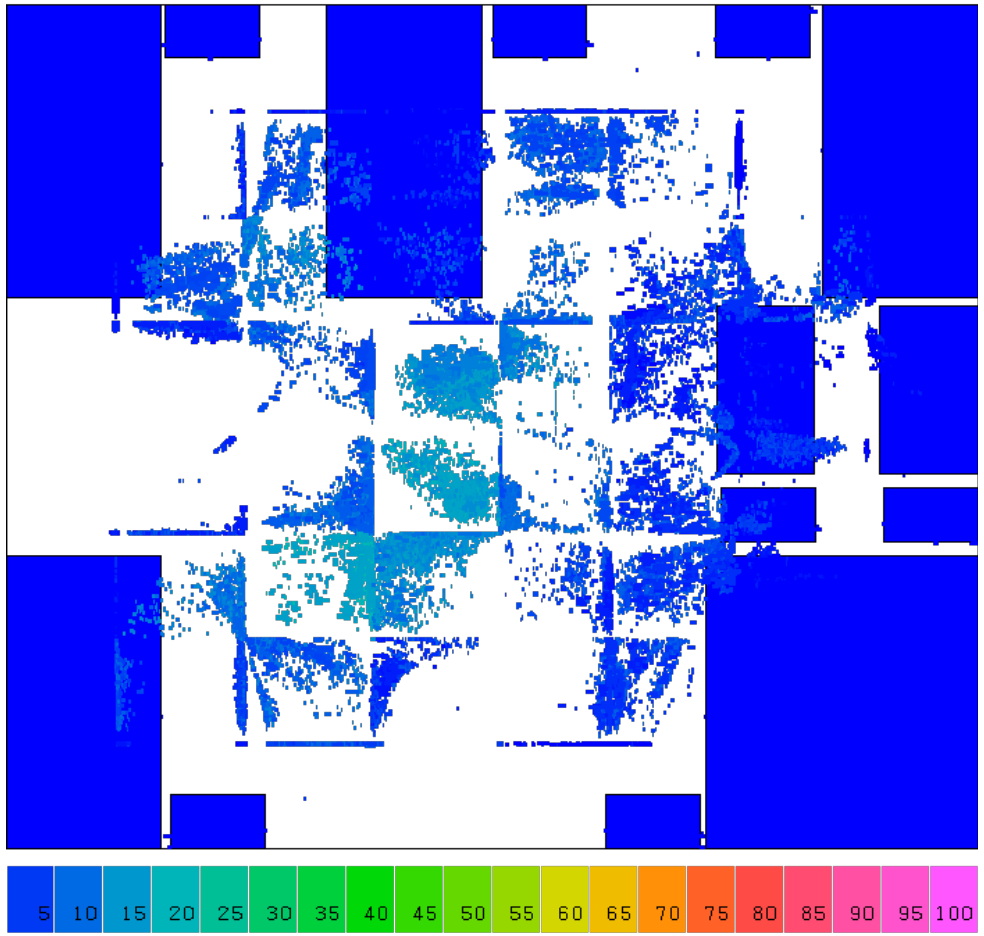
Example: Partition-based Placement



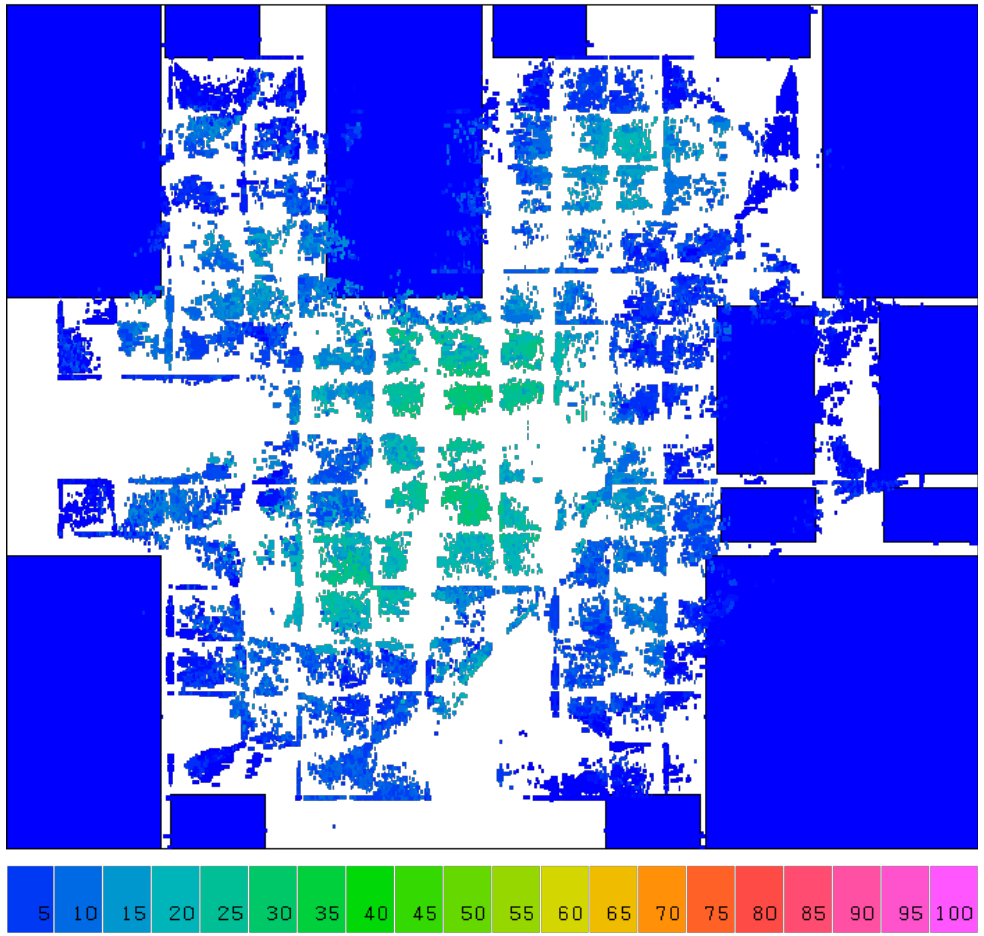
Example: Partition-based Placement



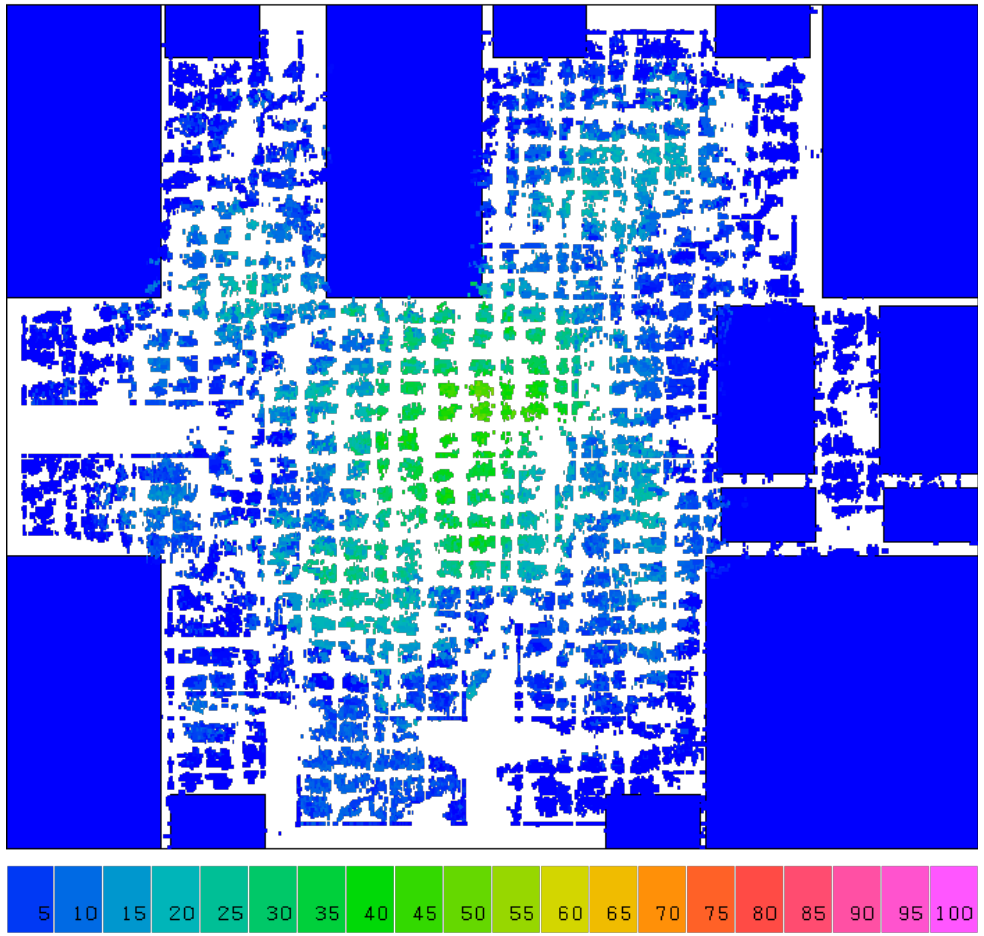
Example: Partition-based Placement



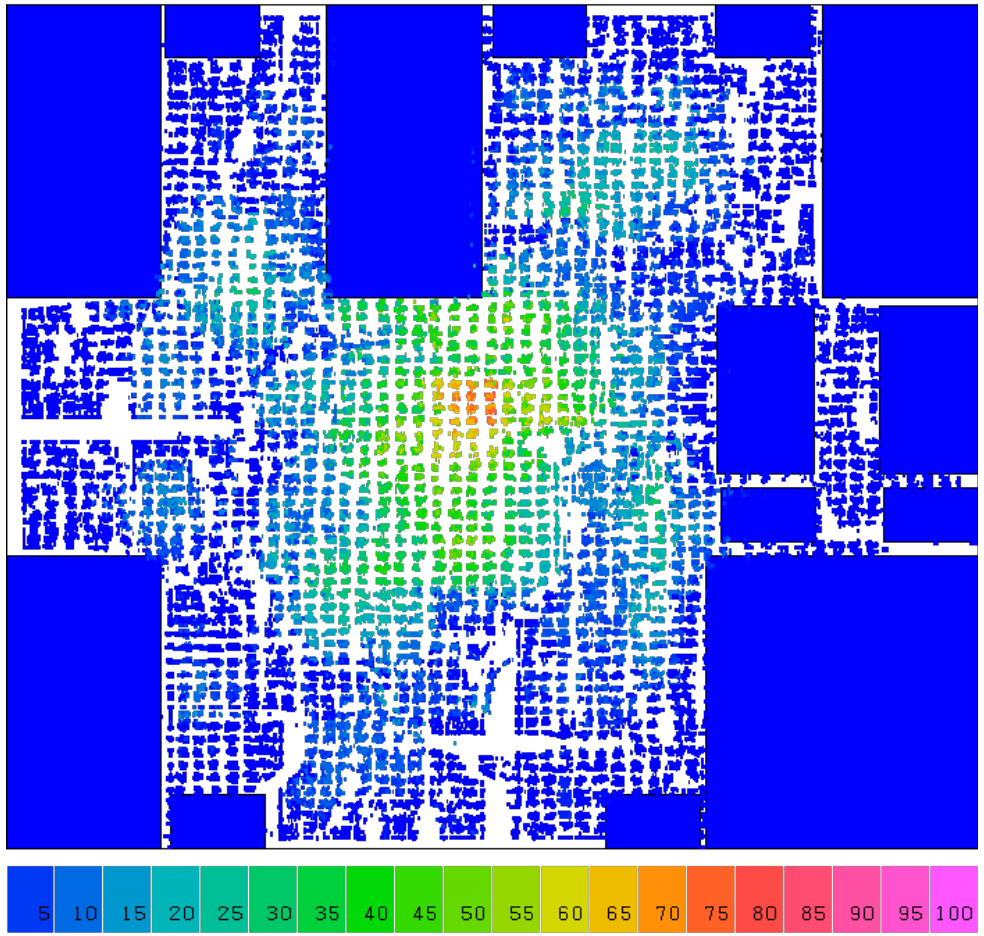
Example: Partition-based Placement



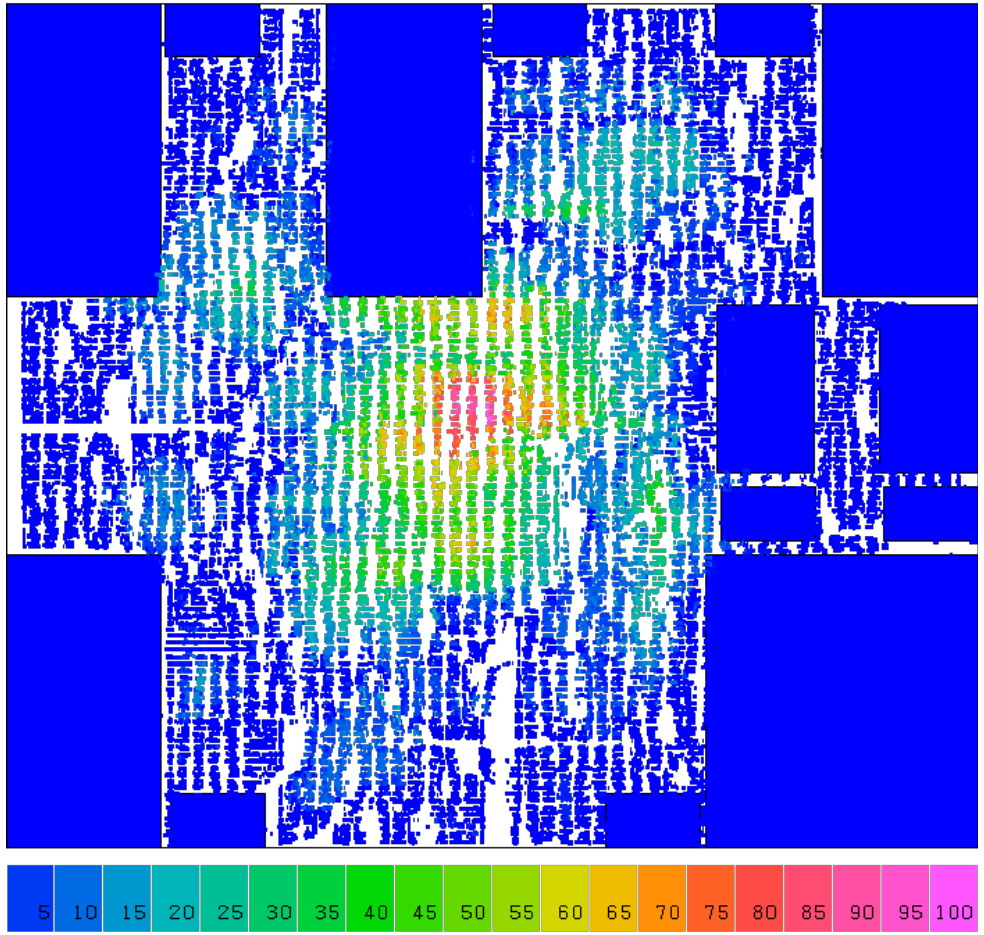
Example: Partition-based Placement



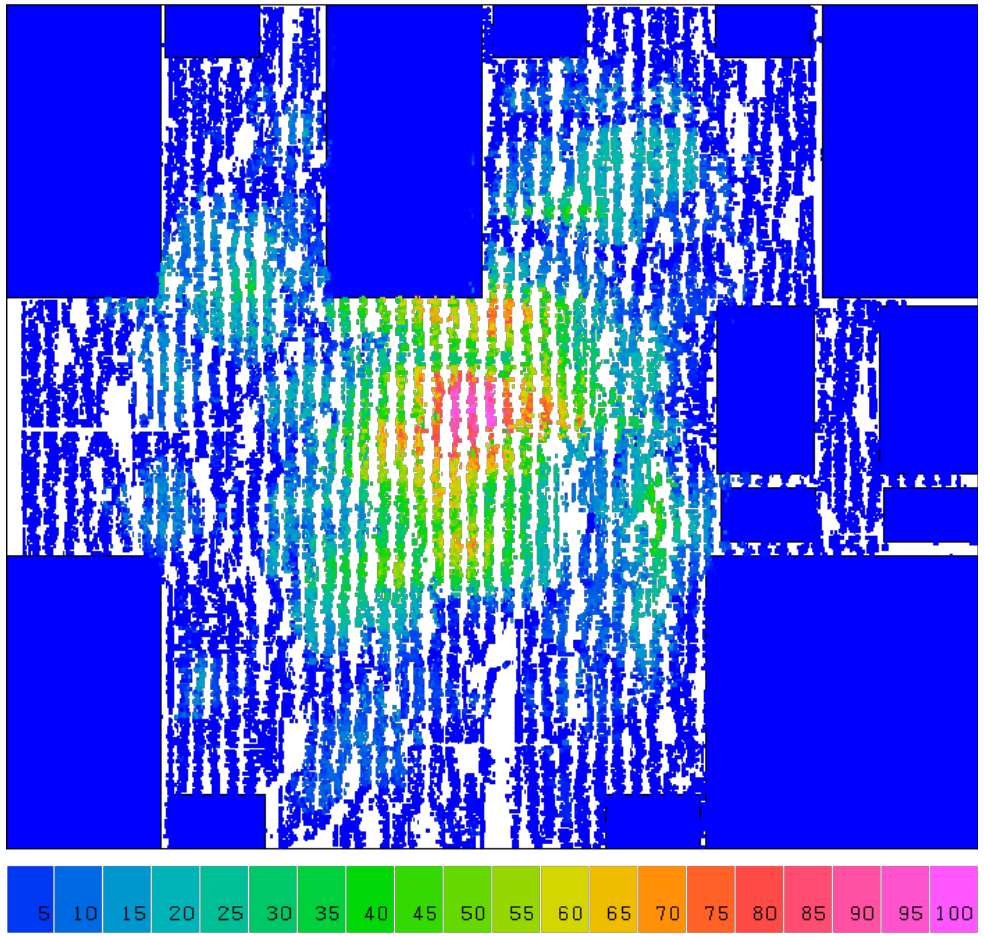
Example: Partition-based Placement



Example: Partition-based Placement

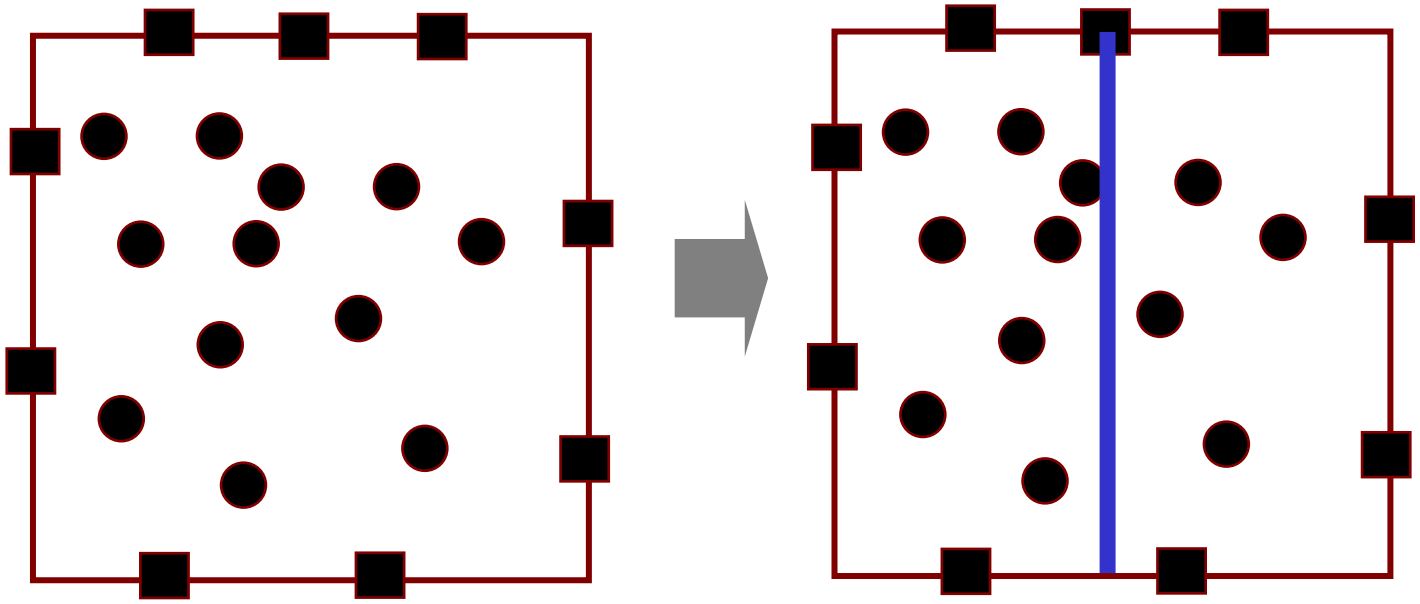


Example: Partition-based Placement



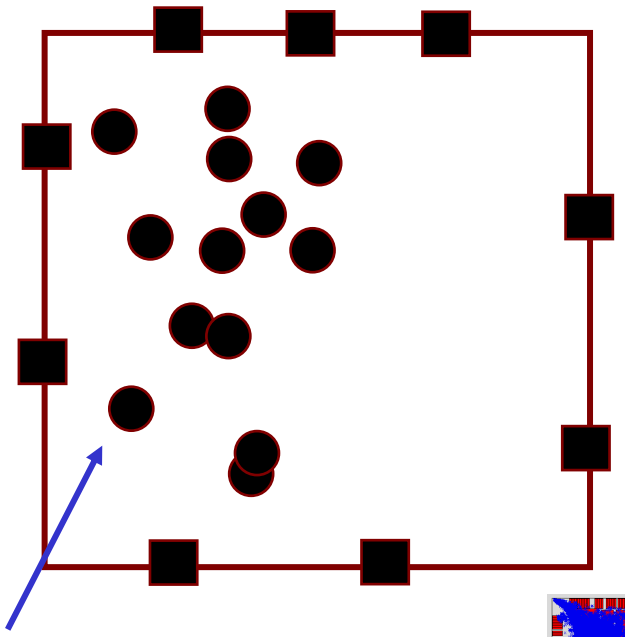
Recursive Partition

- After 1st quadratic placement (**QP**), divide chip area exactly in half, vertically (horizontal is ok too)
- We want *half* the gates on *each* side

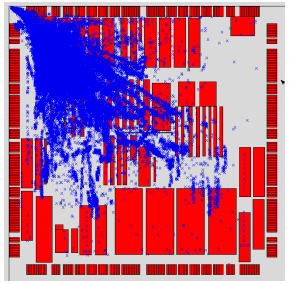


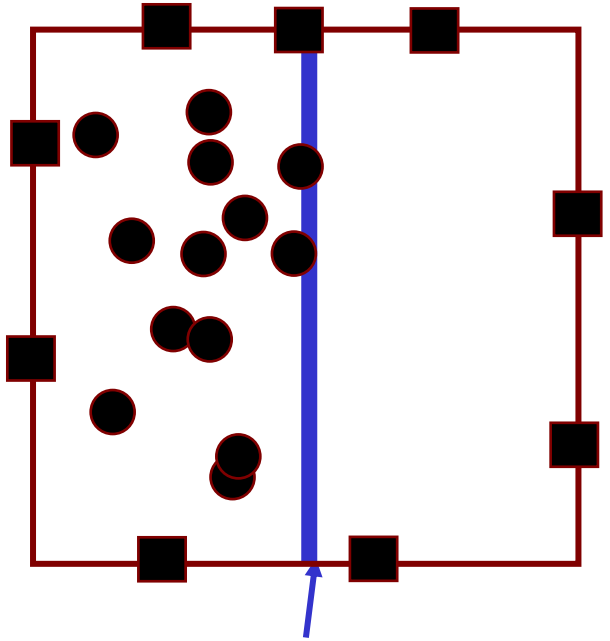
Recursive Partitioning: How to Assign?

- What if QP does not spread gates *evenly* between halves?



How do we know which gates to put left vs right if this is initial **QP**?

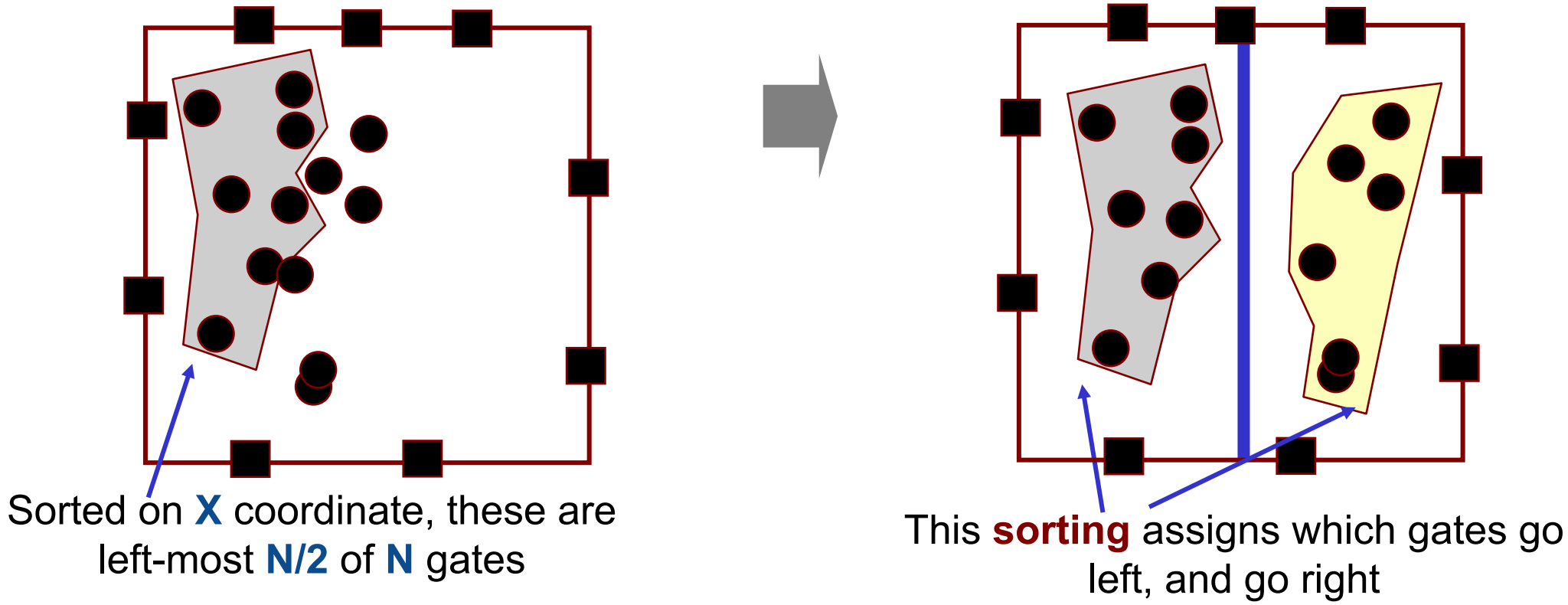




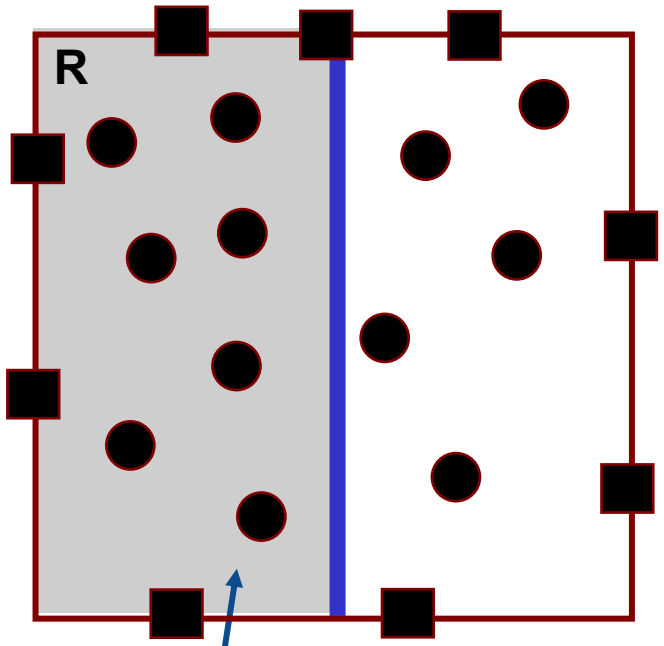
We partition region at exact center. How do we reformulate a new **QP** to re-do the gates on each side?

Recursive Partitioning: How to Assign?

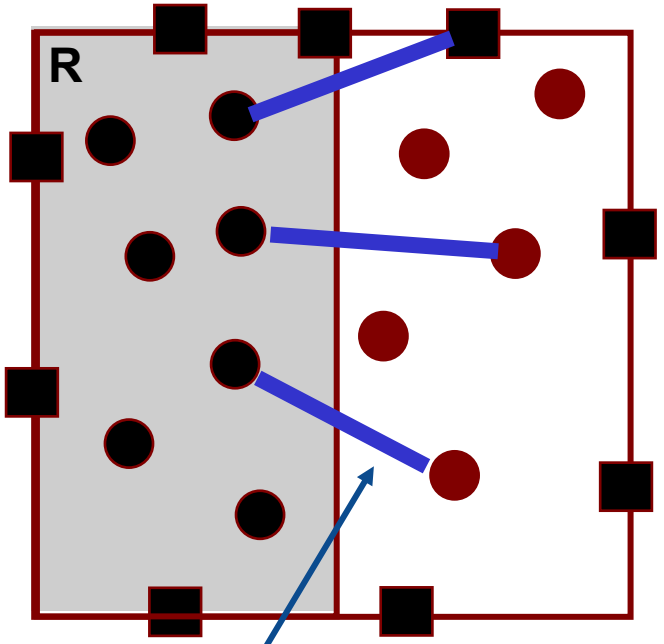
- Sort placed gates on **X** coordinate, then **Y** (for horizontal cut – sort on **Y** first, then **X**); If **N** total gates, then first **N/2** in sorted list go on left; others on right



Recursive Partitioning: How to Assign?



Focus on the gates inside this shaded region **R** on the **left** side of the cut.



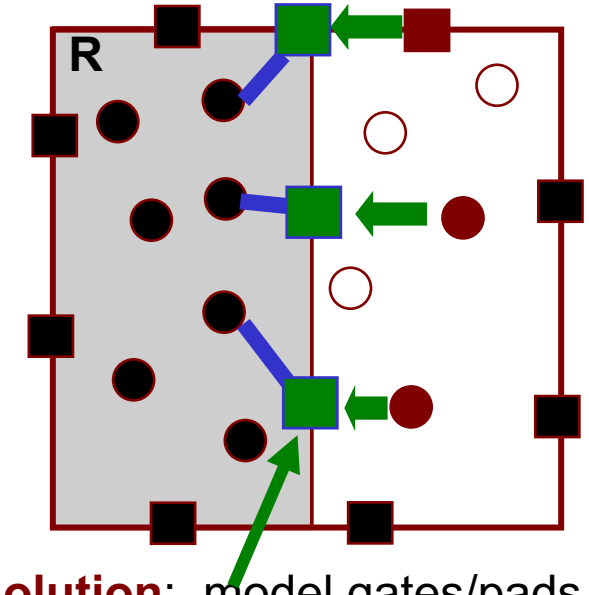
New big questions:

(1) How do we **keep** the gates assigned to left side actually **in** the left side?

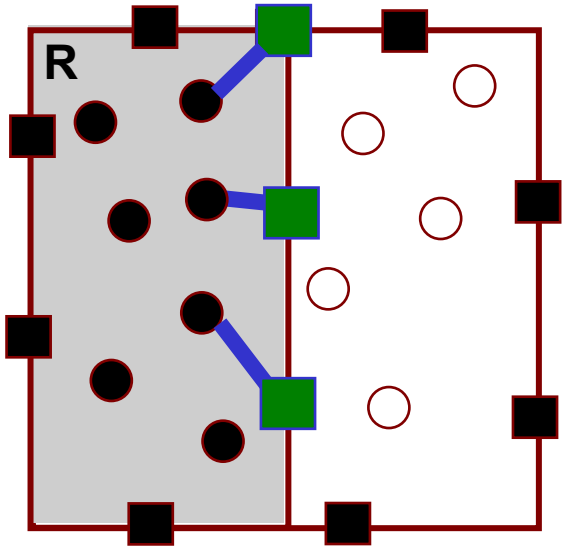
(2) How do we model wires that **connect** to gates/pads on **right**? Can't ignore these!

Recursive Partitioning: How to Contain?

- **Idea: Pseudo-pads**
  - Every gate & pad not inside region **R** is modeled as a pad on boundary of **R**
  - Propagate these outside gates using their current (x,y) location to nearest point on **R**;
  - For this example, we take the y coordinate, and put pad on center of cut line x



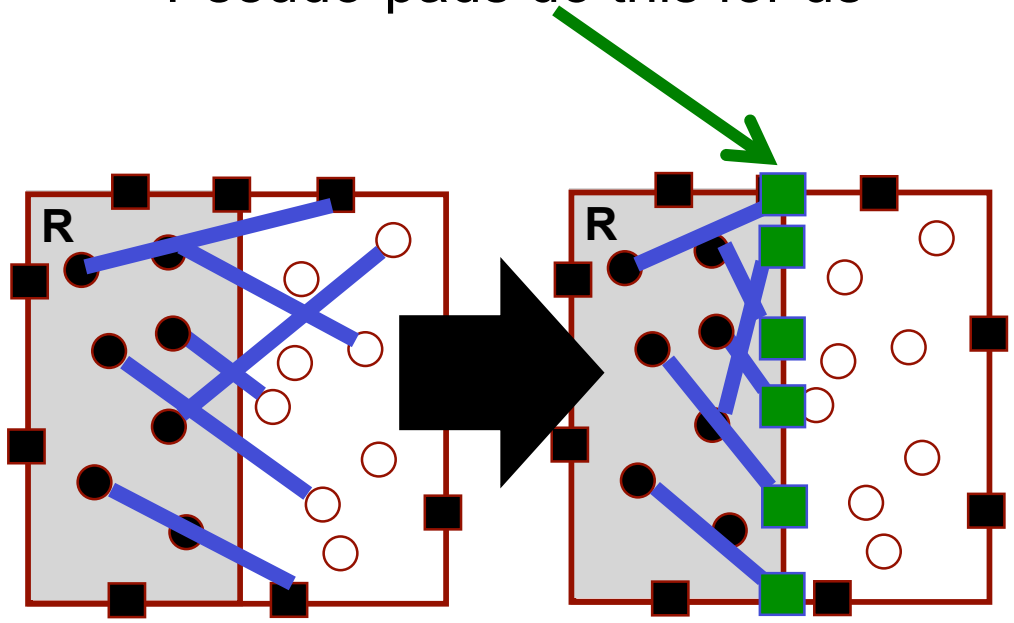
Solution: model gates/pads on right as "fake" pads on left.



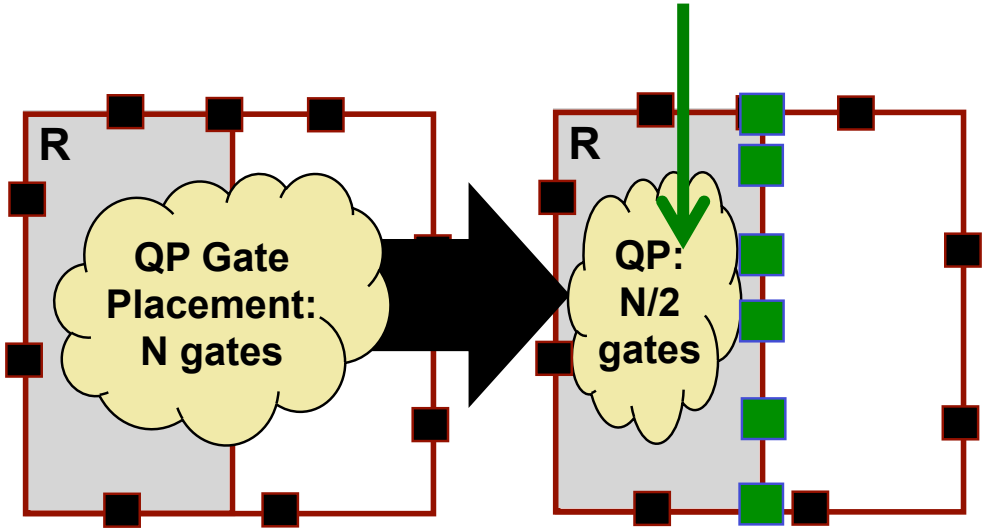
Resulting **new QP** problem for gates in left region

Why this Works?

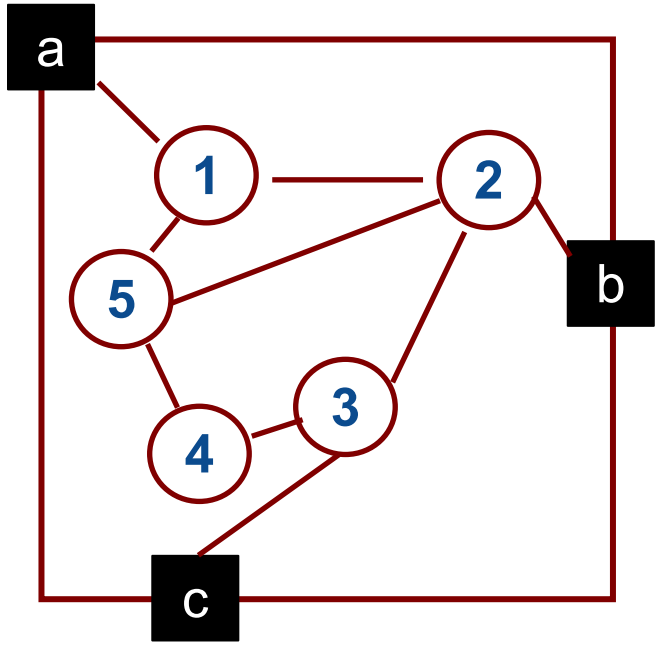
- **Cannot ignore gates outside region we are re-placing**
  - Want gates inside to *feel pull* from wires to *gates outside* region
  - Pseudo-pads do this for us



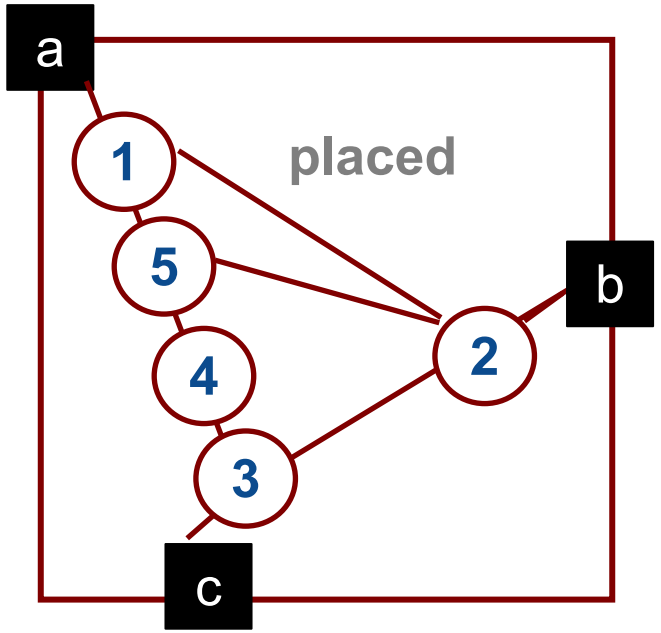
- **Pseudo-pads guarantee all gates re-locate inside region**
  - Think of wires as ‘springs’ that each pull gates *toward* other gates or pads
  - If pads (real & pseudo) are on edges of region – **QP keeps gates inside**



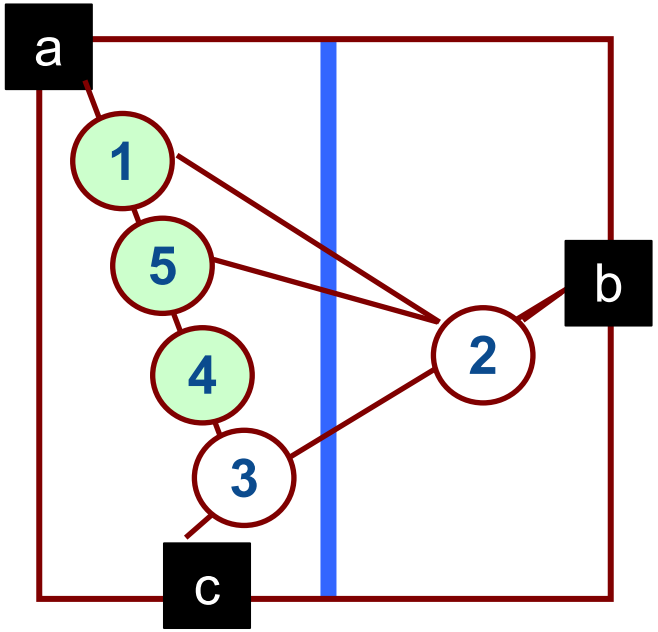
Example



1. Initial netlist
5 gates (1,2,3,4,5)
8 wires
3 pads (a,b,c)

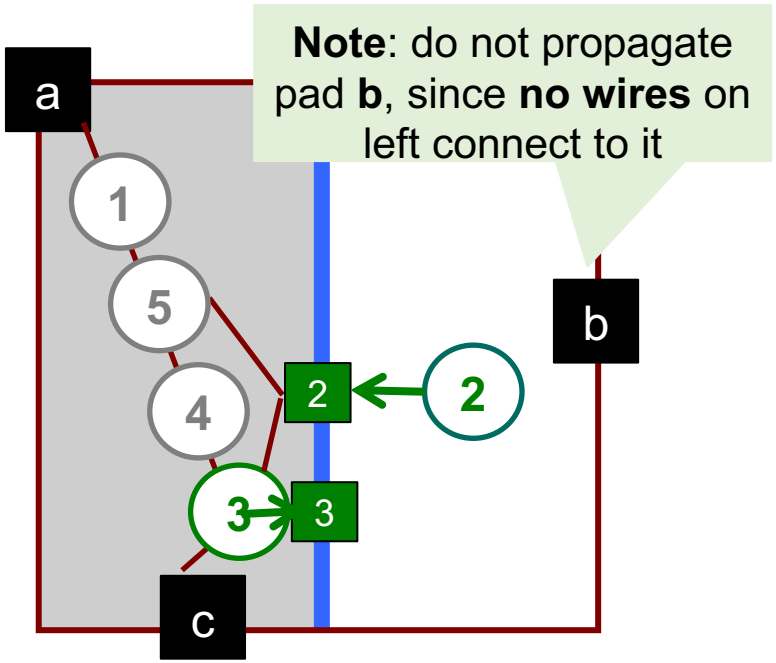


2. Initial QP

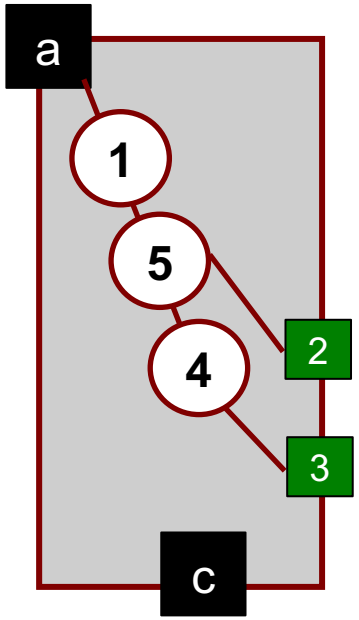


3. First partition
Sort on **X**:
Gate order 1 5 4 3 2
Pick: **1 5 4** on left

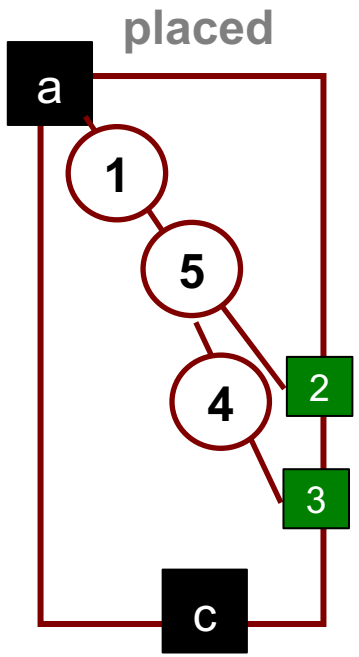
Example (cont'd)



4. Propagate gates/pads
Right-side gates: 2,3
Right-side pads: b
Push to cut, using **y** coordinates

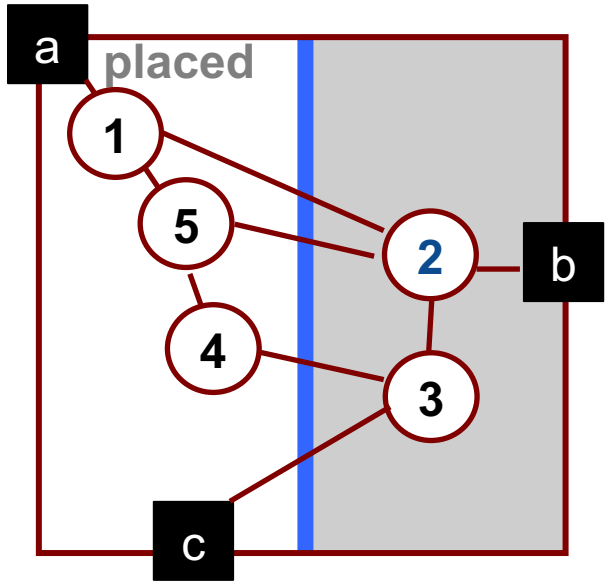


5. 2nd QP input
This is set up for this new smaller placement

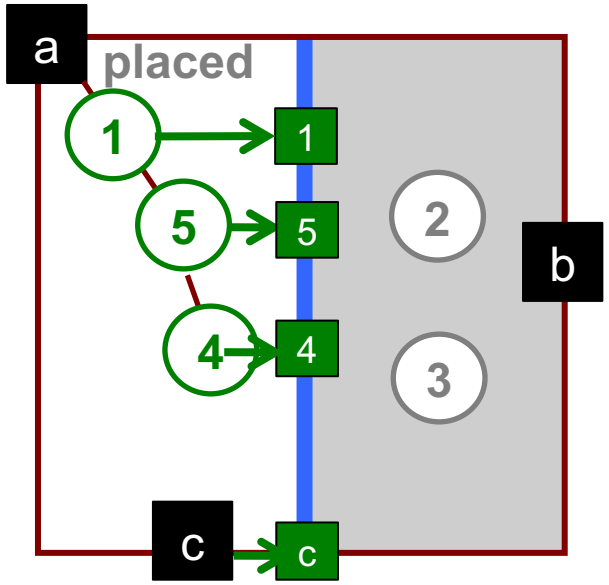


6. 2nd QP solved
New placement

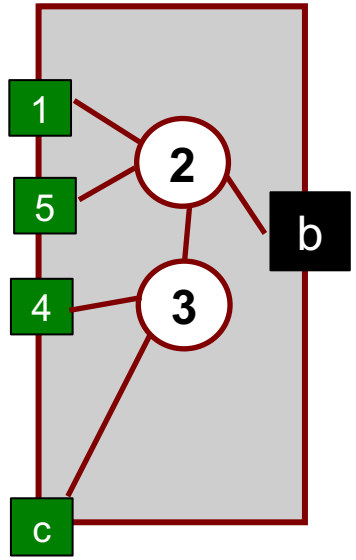
Example (cont'd)



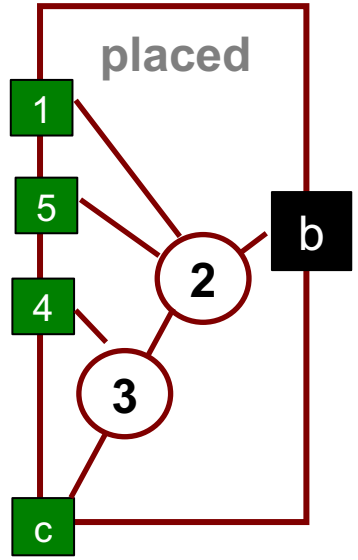
7. Left side placed.
Now, re-place right-side gates.



8. Propagate gates/pads
This is set up for next, new smaller placement

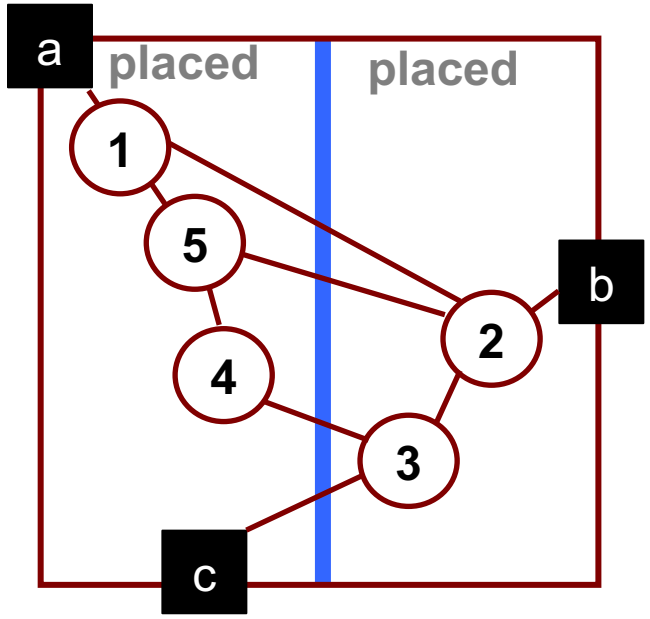


9. 3nd QP input
This is set up for this new smaller placement

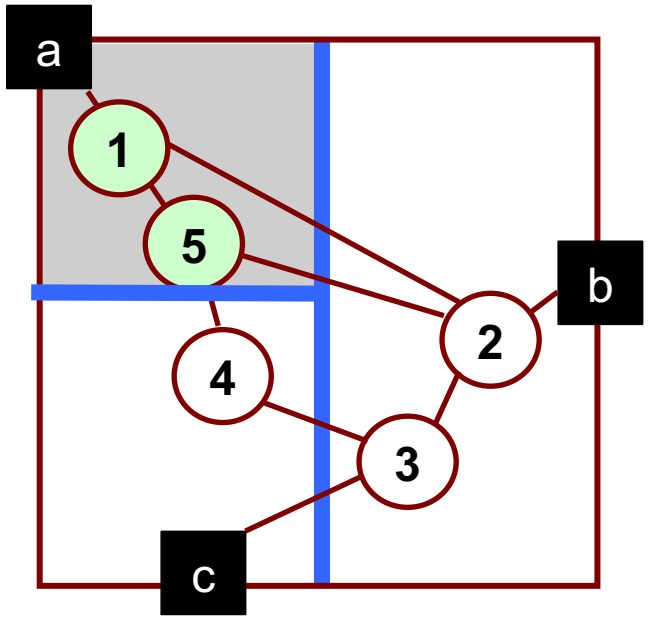


10. 3nd QP solve

Example (cont'd)



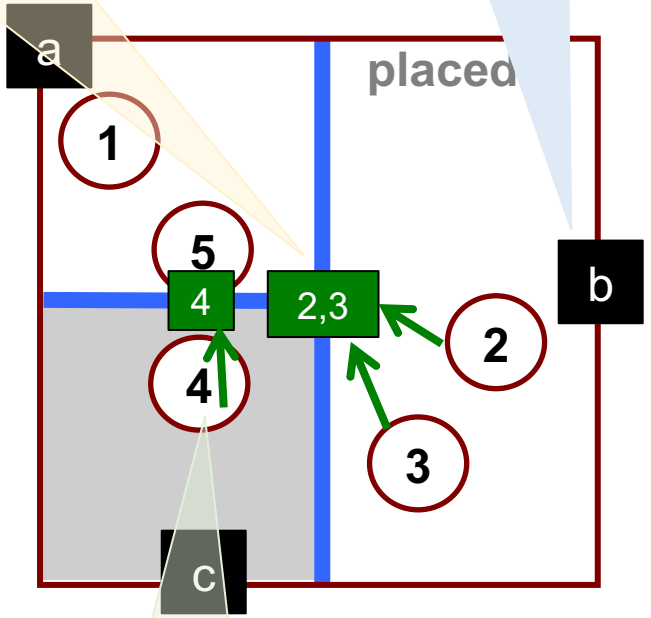
Repeat: Horizontal partition on left



Focus on top.
Sort gates on **Y**
Assign gates 1,5 to region.

Note: gates **2,3** propagate to **corner** of new region: closest point

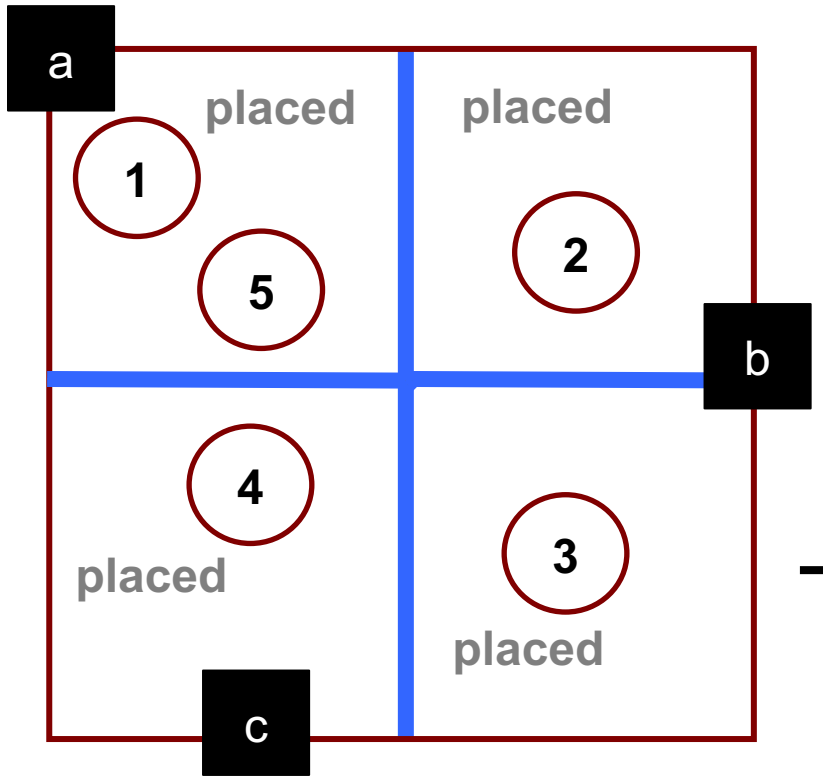
Note: do not propagate pad **b, no wires** to **1,5**



Propagate gates & pads

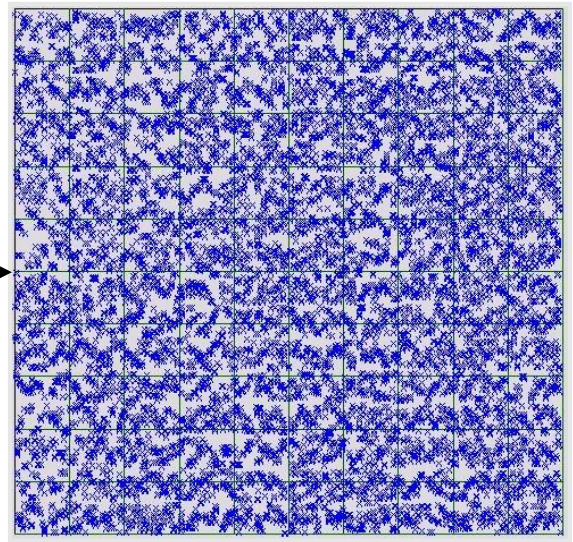
Note: gate **4** propagates up to **bottom** of new region

Keep Repeating this Recursion



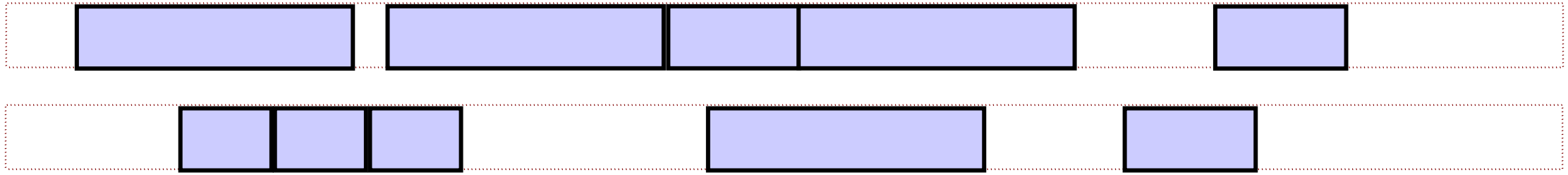
Continue…

- **Keep recursively partitioning…**
  - Usually, you continue until you have a "small" number of gates in each region
  - Small >> 1 typically. **10-100** for example
  - Get a good, "global" placement, but not a "final" placement



Final Placement Step: Legalization

- **Still need to force gates in precise rows for final result**
  - QP methods *cannot* force individual gates into standard cell rows, without overlaps



- **Solution step is called: Legalization**
  - Many different algorithms. One easy way to do this is by annealing!
  - Do local improvement based on swaps of nearby gates
  - To anneal, set **T=HOT** to be very small (cold), so don't disrupt QP result

Summary

- **We have discussed problems of plain quadratic placement**
  - Gates are easily lumped together to minimize wirelength
- **We have discussed a solution: partition-based placement**
  - Partition the placement region iteratively
  - Introduce auxiliary pads between partition boundaries
- **We have discussed legalization**
  - Remove all overlaps among gates
  - Improve wirelength in a local region (e.g., rows)