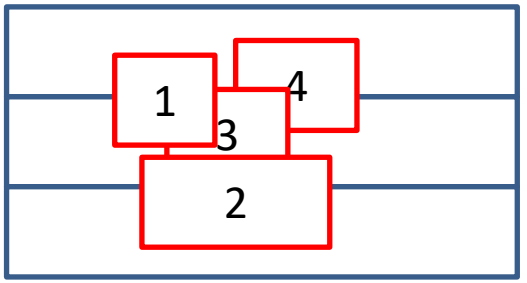
Greedy Methods

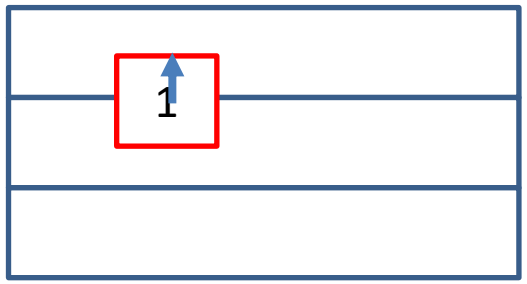
- Legalize one cell at a time
- **Tetris** [Hill, US Patent 6370673, 2002]
  - Sort cells in ascending x-coordinates
  - Pack cells to the left one at a time into a row so as to minimize the displacement
  - Already legalized cells are never moved again
  - Pros: Commonly used and extremely fast
  - Cons: May result in very uneven row length and may fail to pack all cells inside the placement region
- **Abacus** [Spindler, ISPD 2008]
  - Similar to Tetris
  - **Already legalized cells can be moved**
  - Get better results than Tetris

Example for Tetris

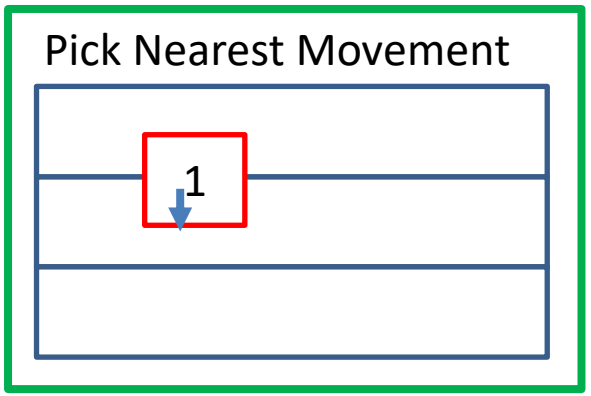
Global Placement

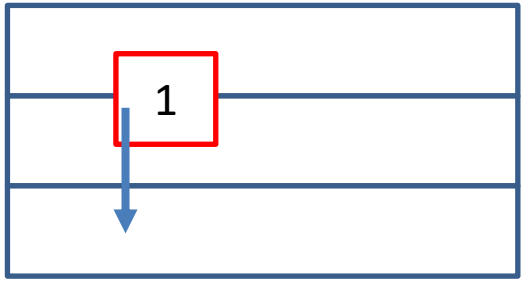


Legalize cell 1:

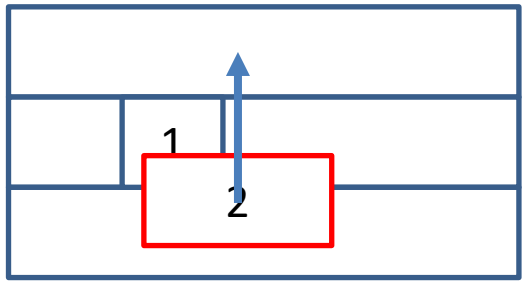


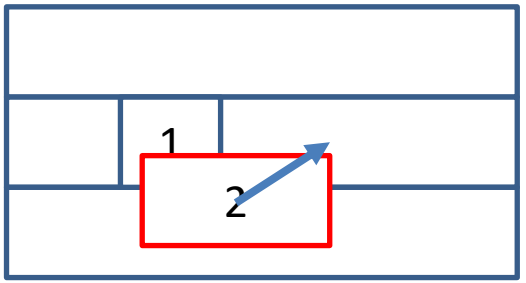
Pick Nearest Movement

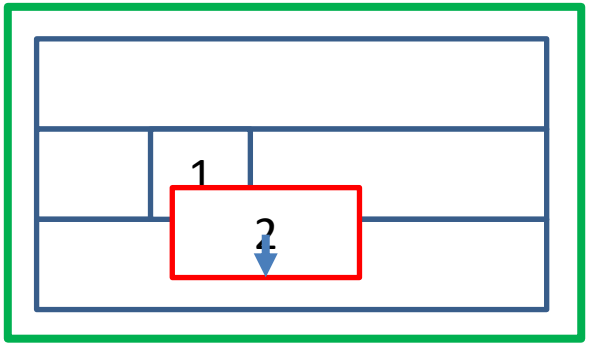




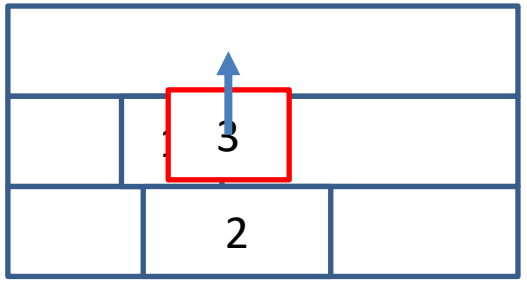
Legalize cell 2:

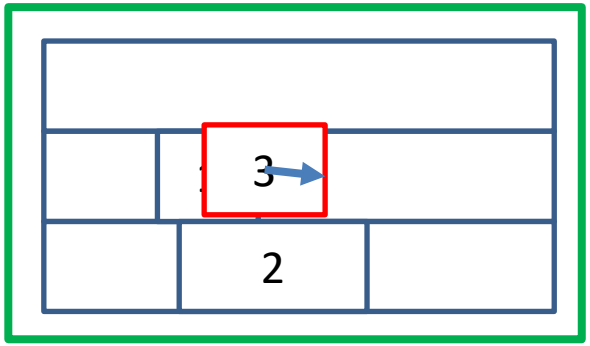


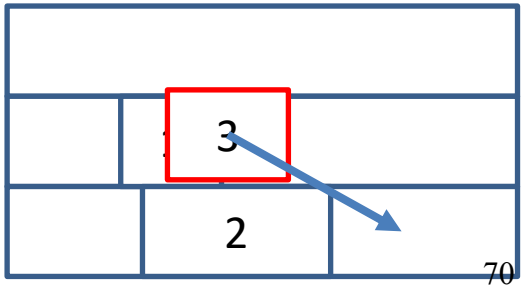




Legalize cell 3:



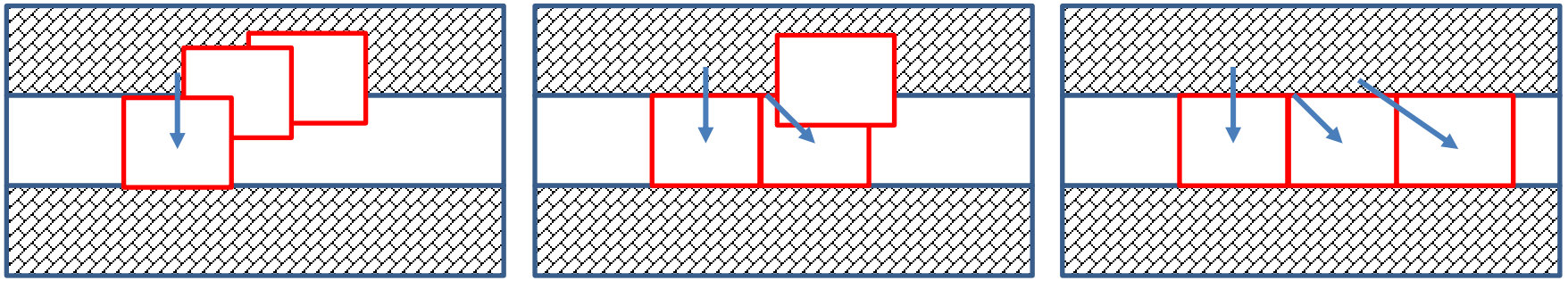




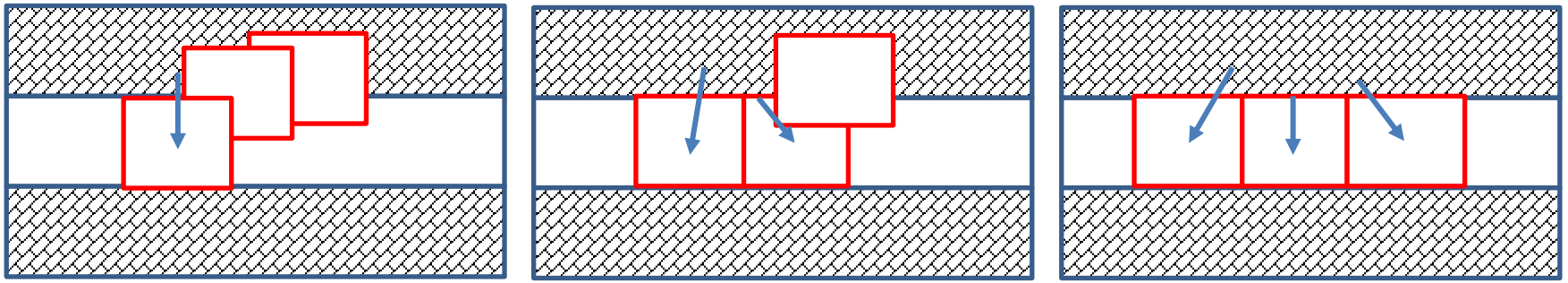
Abacus

- Similar to Tetris: sort cells and legalize one cell at a time
- Legalization of one cell: move the cell over the rows, and place the cell to the best/nearest row
- **PlaceRow** (difference from Tetris): move already legalized cells within one row to minimize total movement

Tetris:



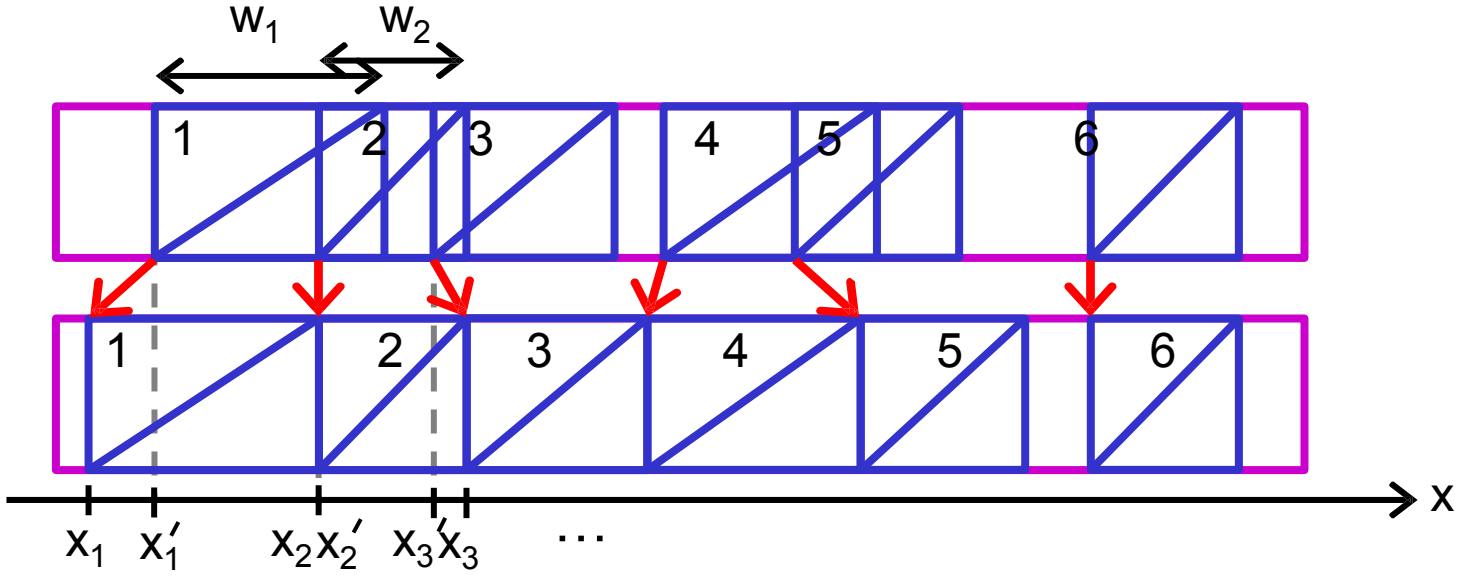
Abacus:



PlaceRow

Input: one row with N cells, x-pos of cells: global placement (x_i')

Output: new (legal) x-pos of cells (x_i) such that the overlap is removed and the total quadratic movement is minimized



QP:

$$\min \sum_{i=1}^{N} e_i \left(x_i - x_i'\right)^2 \quad \text{s.t.} \quad \underbrace{x_i \geq x_{i-1} + w_{i-1}}_{\text{no overlap}} \leftarrow \text{width}$$

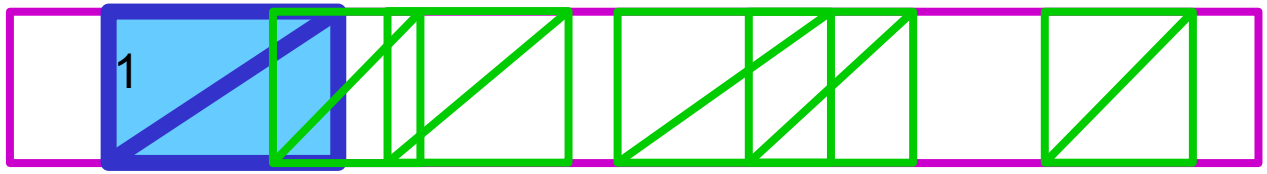
weight: e_i ↑ legal x-pos: x_i ↑ global x-pos: x_i'

PlaceRow: Dynamic Programming

PlaceRow:

- Solve QP by dynamic programming approach:
  solve sub problems optimally to obtain final solution
- Process cells from left to right

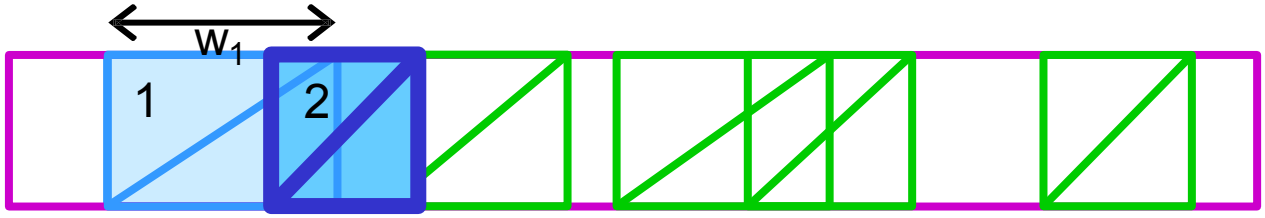
Cell 1:



first cell → do not move

PlaceRow: Cell 2

Cell 2:



overlap with previous cell? **yes** → **cluster** with previous cell

Clustering process:

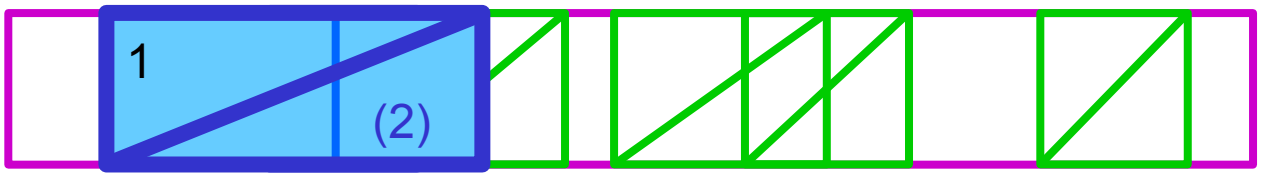
$$x_2 = x_1 + w_1$$

$$\min e_1(x_1 - x_1')^2 + e_2(x_2 - x_2')^2 \Rightarrow \min \underbrace{(e_1 + e_2)}_{\text{new } e_1}\left(x - \underbrace{\frac{e_1 x_1' + e_2(x_2' - w_1)}{e_1 + e_2}}_{\text{new } x_1'}\right)^2$$

Update x_1', e_1, and w_1:

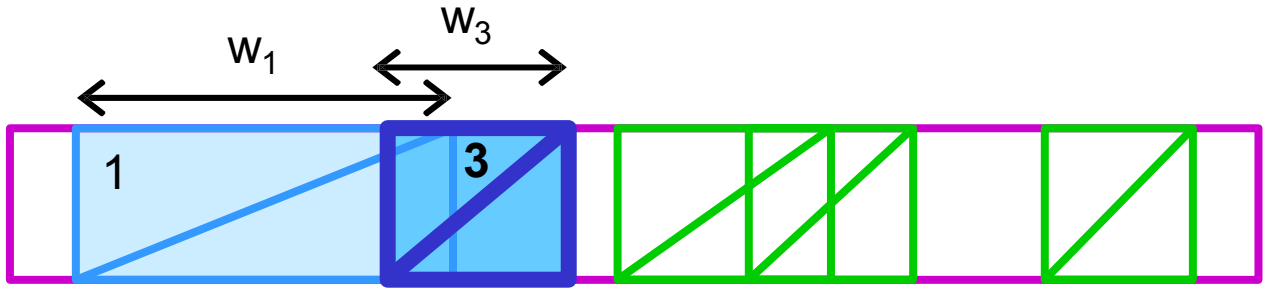
$$x_1' \leftarrow \frac{e_1 x_1' + e_2(x_2' - w_1)}{e_1 + e_2} \qquad e_1 \leftarrow e_1 + e_2 \qquad w_1 \leftarrow w_1 + w_2$$

Result: $\min e_1(x_1 - x_1')^2 \rightarrow x_1 = x_1'$



PlaceRow: Cell 3

Cell 3:



Overlap with previous cell? **Yes** → **cluster** with previous cell

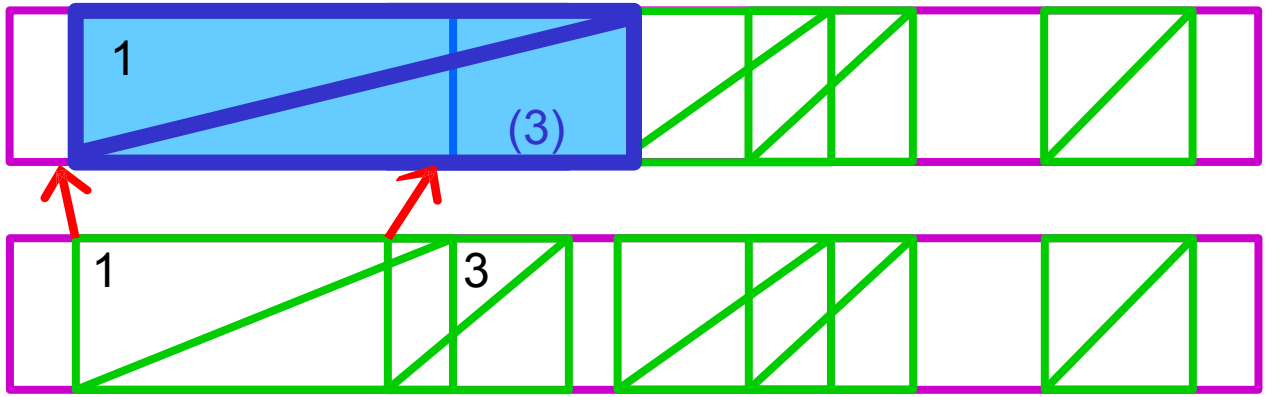
Update x_1', e_1, and w_1:

$$x_1' \leftarrow \frac{e_1 x_1' + e_3 (x_3' - w_1)}{e_1 + e_3} \qquad w_1 \leftarrow w_1 + w_3 \qquad e_1 \leftarrow e_1 + e_3$$

Move cell 1: $\min e_1 (x_1 - x_1)^2 \longrightarrow x_1 = x_1'$

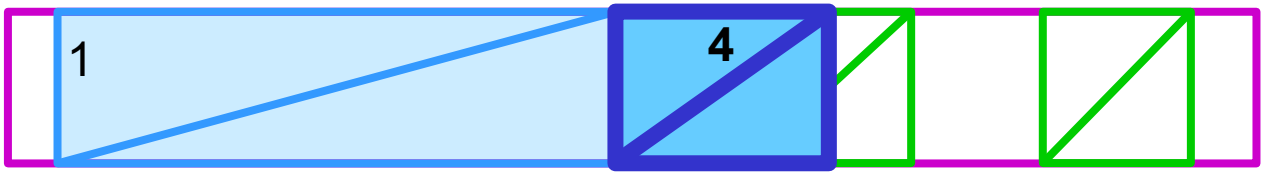
Result:

Prev:



PlaceRow: Cell 4

Cell 4:

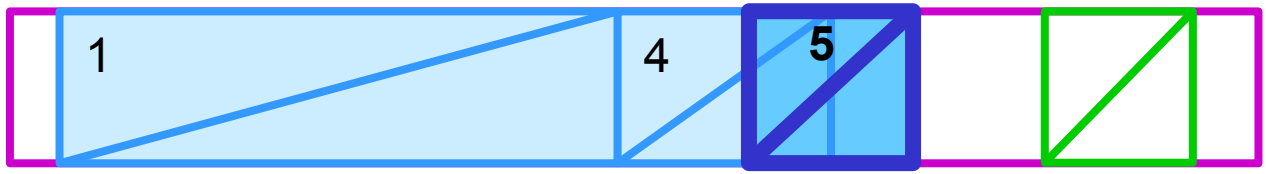


Overlap with previous cell? **No**

→ **no clustering, no movement**

PlaceRow: Cell 5

Cell 5:

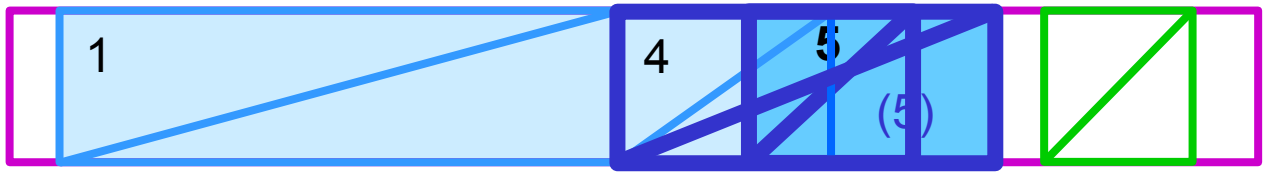


Overlap with previous cell? **yes** → **cluster** with previous cell 4

Update x_4', e_4, and w_4:

$$x_4' \leftarrow \frac{e_4 x_4' + e_5 (x_5' - w_4)}{e_4 + e_5} \qquad e_4 \leftarrow e_4 + e_5 \qquad w_4 \leftarrow w_4 + w_5$$

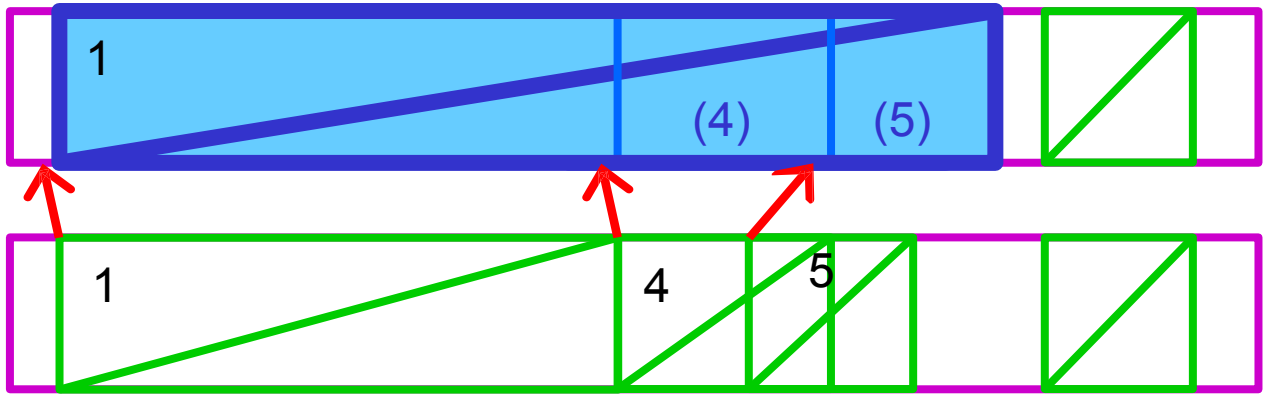
Move cell 4: $\min e_4 (x_4 - x_4')^2 \longrightarrow x_4 = x_4'$



Overlap with previous cell? **yes** → **cluster** with previous cell 1

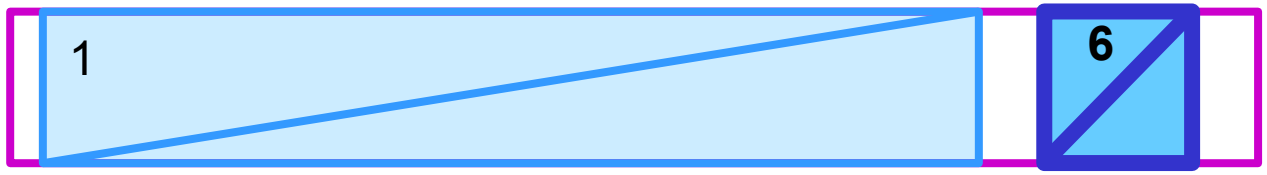
Result:

Prev:



PlaceRow: Cell 6

Cell 6:



Overlap with previous cell? **No**
→ **no clustering, no movement**

Last cell → done, PlaceRow finished

PlaceRow: Summary

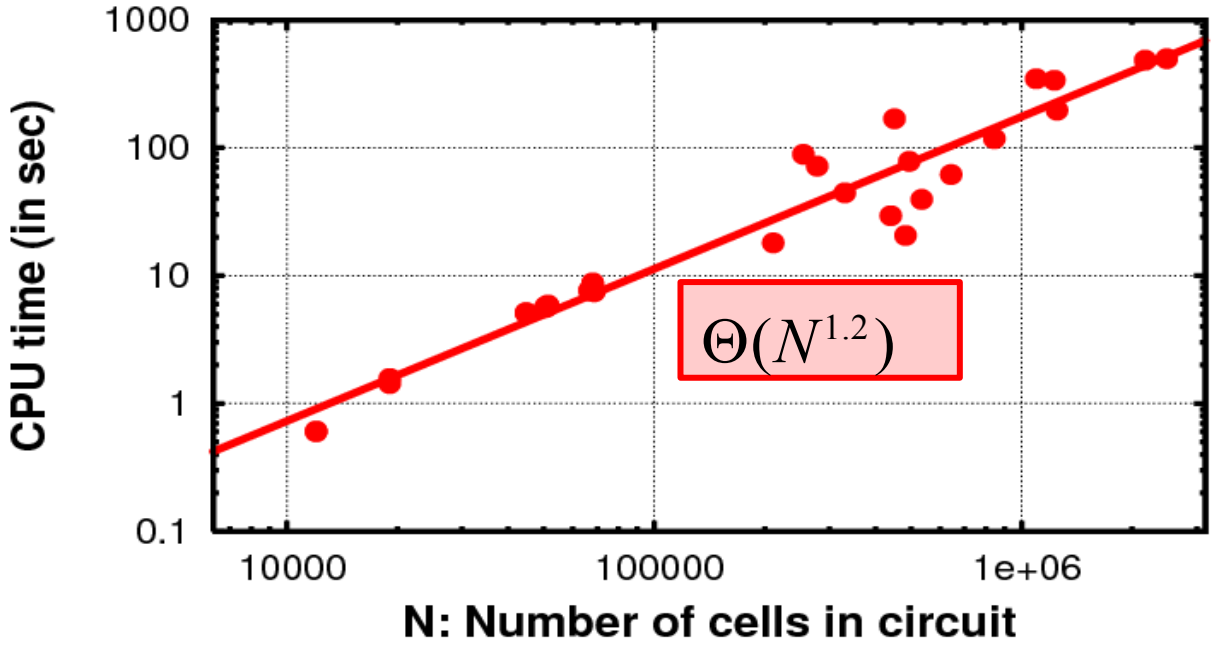
PlaceRow:

- Called several times for legalizing one cell
- Places cells aligned to one row:
minimize quadratic movement → quadratic program (QP)
- Solves QP by dynamic programming:
  - Process cells from left to right
  - If cell overlaps with previous cell:
    clustering → movement → further checks with left cells
  - Clustering: update width, weight, and global x-pos of cell
    constant execution time
- Linear worst-case complexity: O(N) N: number of cells in the row
At most N-1 clustering operations for N cells

Complexity

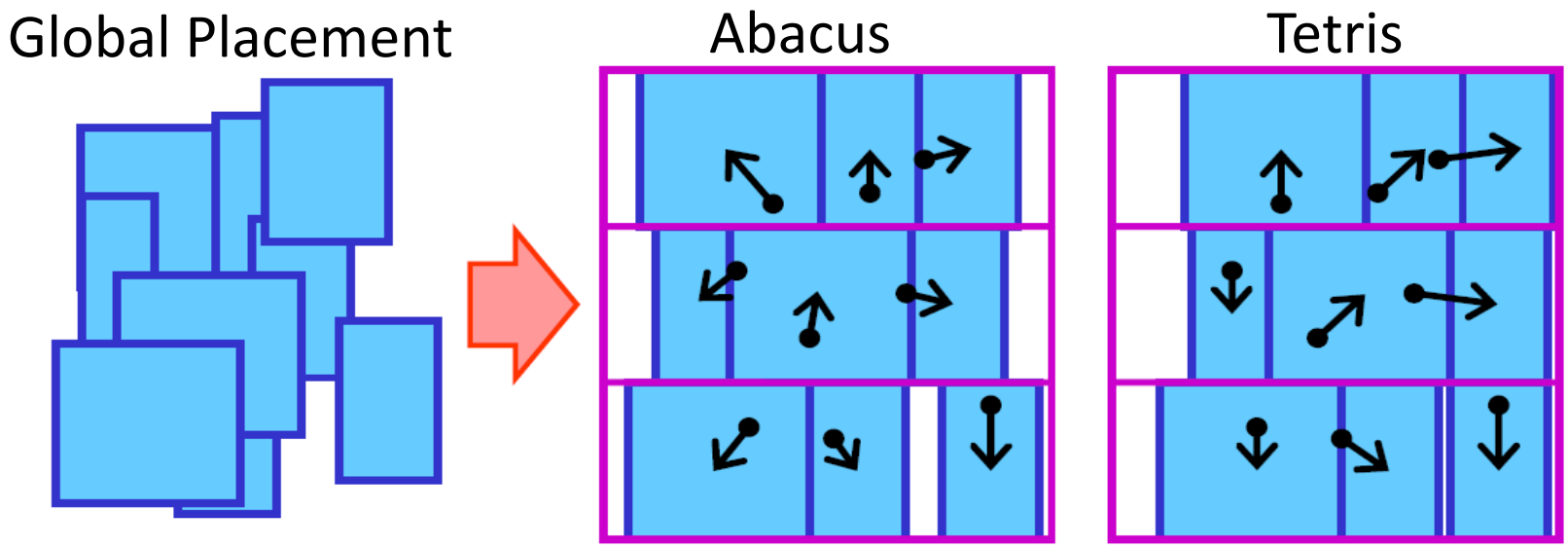
Worst-case for a complete circuit with N cells: $O(N^2)$

Average-case (experimental results):



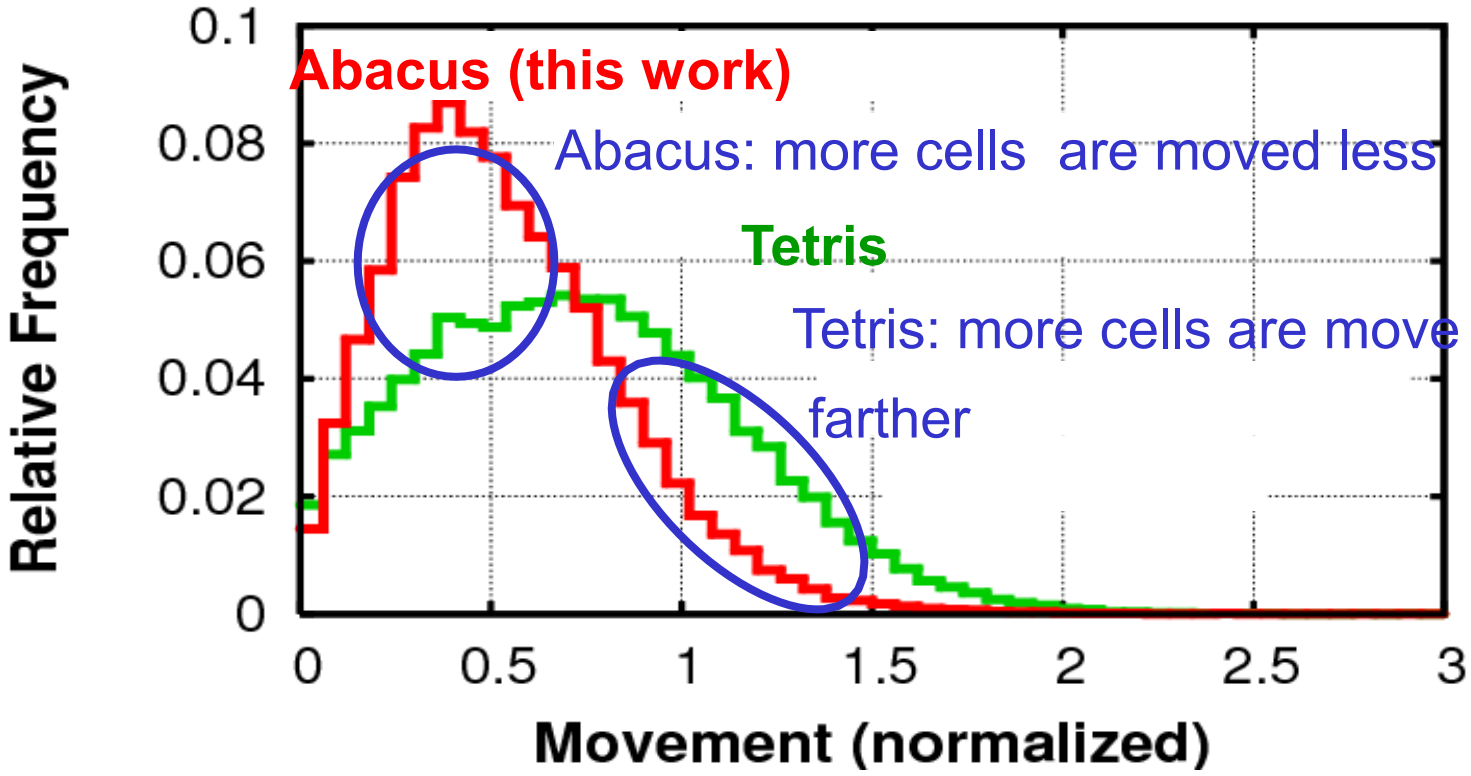
Tetris vs. Abacus

- Abacus can get shorter total movement and shorter maximum movement than Tetris
- Abacus can avoid overflow in each row
- The runtime of Abacus is longer than Tetris, but both can complete in few seconds for large designs



Comparison on Movement

Experimental results of one circuit:



→ **Lower movement with Abacus**