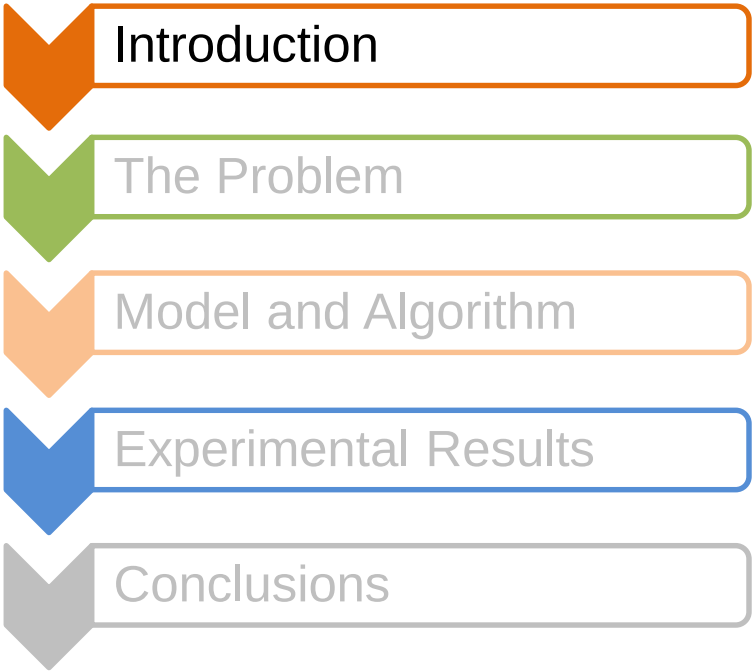
Analytical Mixed-Cell-Height Legalization Considering Average and Maximum Movement Minimization

Xingquan Li[1], Jianli Chen[2], Wenxing Zhu[2], and Yao-Wen Chang[3]

[1]Minnan Normal University
[2]Fuzhou University
[3]National Taiwan University

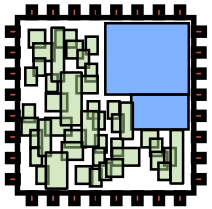
International Symposium on Physical Design 2019
San Francisco, California, USA

Outline


Introduction
The Problem
Model and Algorithm
Experimental Results
Conclusions

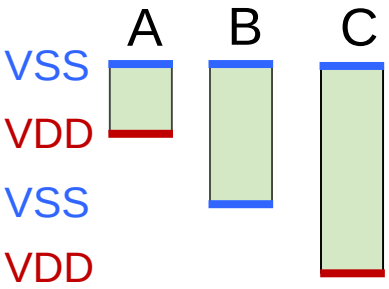
Mixed-Cell-Height Legalization

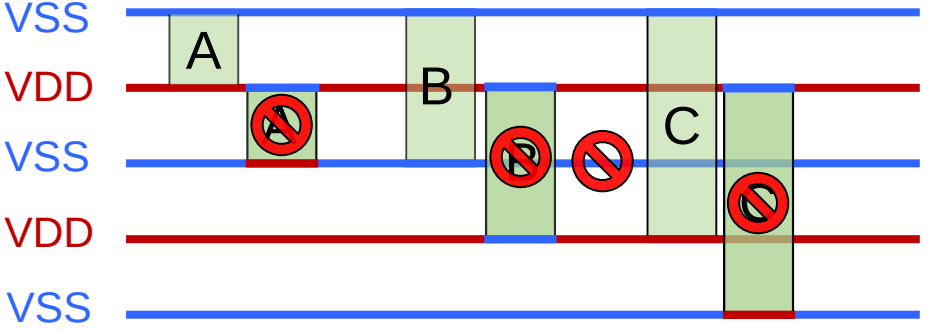
- Given: Global placement
- Objective: Minimize cell displacement
- Constraints:
  - 1) Chip region
  - 2) No cell overlaps
  - 3) Placement sites on rows
  - 4) Power-rail alignment
- Power-rail constraint



Global placement:
Place cells to desired positions, ignoring cells overlaps

Legalization:
Place cells into rows & removes overlaps

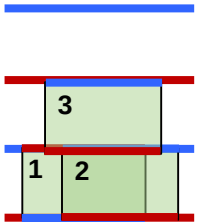




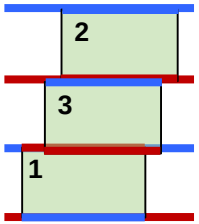
NP-hard problem!

Motivation

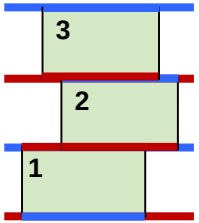
- For the tiny routability gap before and after legalization, a desirable legalization result should be with **minimized average cell movement and minimized maximum cell movement**



A circuit



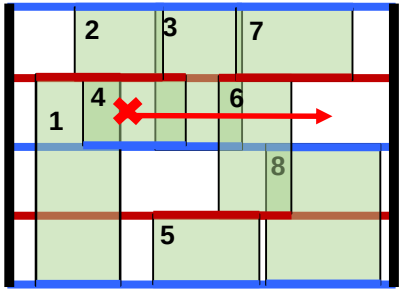
Minimizing average movement



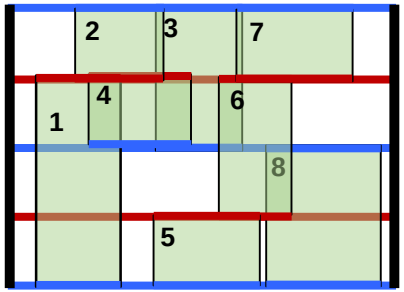
Minimizing average Movement and maximum movement

Motivation

- Previews works spread cell in the vertical or horizontal directions separately
  - Shifting a cell may cause cell overlaps in other rows
  - Large cell total movement or maximum movement
  - Cell may be placed outside the placement region
- In this work, we design an analytical algorithm to spread cells in both the vertical and horizontal directions simultaneously



After x- legalization

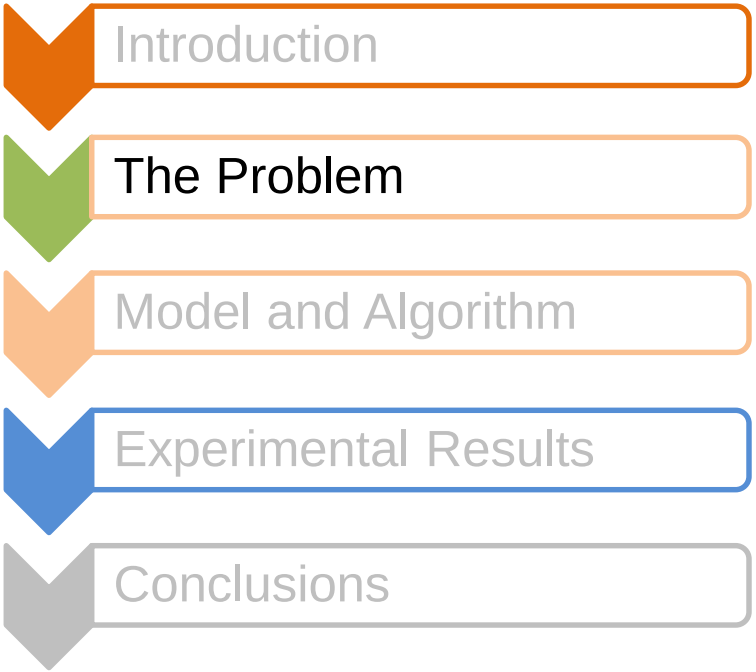


After x- and y- legalization

Our Contributions

- Consider not only the average cell movement, but also the maximum cell movement
- Spread cells in both the vertical and horizontal directions simultaneously
- Mixed integer quadratic program (MIQP) formulation for mixed-cell-height legalization
  - Analyze and remodel the objective function and constraints
- Relax the MIQP into a quadratic programming problem (QP)
  - Relax discrete constraints to linear ones

Outline


Introduction
The Problem
Model and Algorithm
Experimental Results
Conclusions

The Problem

- Given: a global placement result of n standard cells $C = \{c_1, c_2, \ldots, c_n\}$, where each cell c_i has the respective height and width h_i and w_i, and its bottom-left coordinate (x_i^0, y_i^0), $\forall\, i$, $1 \le i \le n$, and each even-row-height cell has a boundary power-rail type VDD or VSS
- The objective of the multi-deck cell legalization is placing each cell c_i to a coordinate (x_i, y_i), such that the average cell movement and the maximum cell movement are minimized, and the following constraints are satisfied:
  - 1) cells must be non-overlapping
  - 2) cells must be placed inside the placement region
  - 3) cells must be located at placement sites on rows
  - 4) cells must be aligned to correct power rails
  - 5) others, e.g., pin access, pin short, edge spacing

Integer Program Formulation

$$\min_{x,y} \ \frac{1}{n'} \sum_{c_i \in C} (|x_i - x_i^0| + |y_i - y_i^0|) + \omega \cdot \max_{c_i \in C} (|x_i - x_i^0| + |y_i - y_i^0|) \qquad \textbf{(1)}$$

s.t.

$$y_i = k_i' R_h, \ \forall c_i \in C,$$

$$k_i' \in \begin{cases} \{0, 1, 2, \ldots\} \text{ if } c_i \text{ is of an odd-row height;} \\ \{0, 2, 4, \ldots\} \text{ (VDD boundaries) or} \\ \{1, 3, 5, \ldots\} \text{ (VSS), otherwise;} \end{cases}$$

$$x_i + w_i \leq x_j, \ \forall c_i, c_j \in C,$$

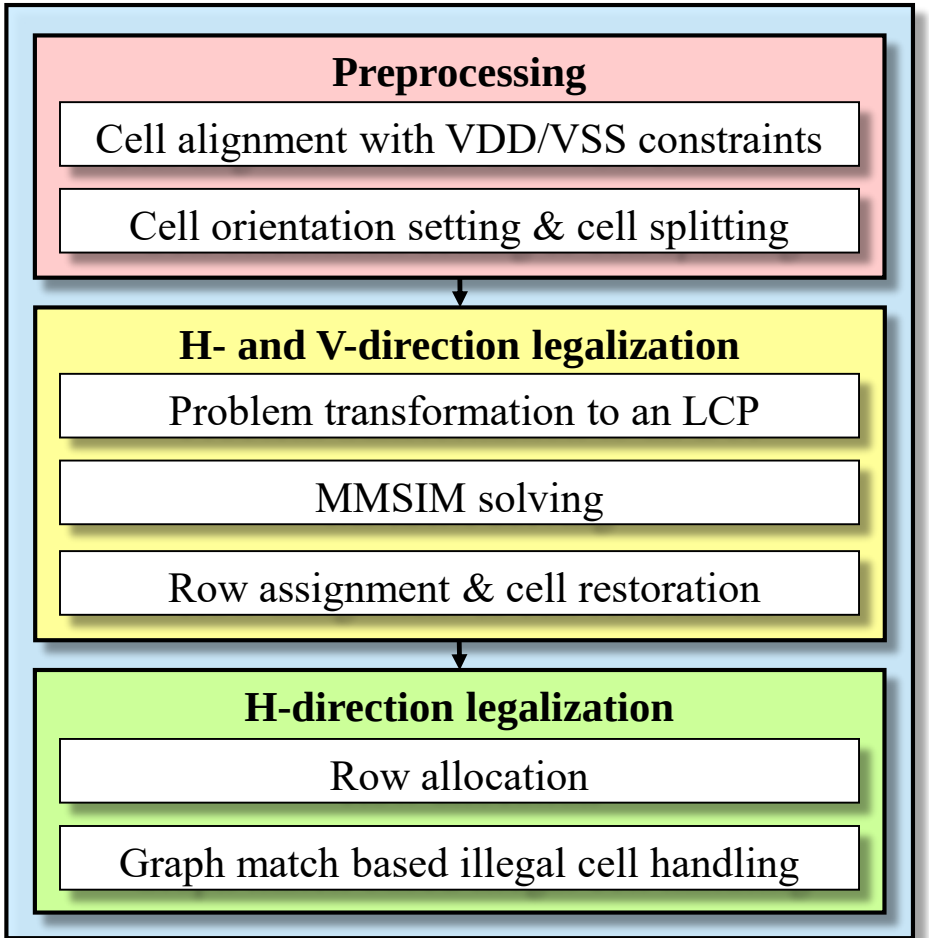
if c_i and c_j are in the same row, and $x_i \leq x_j$;

$$0 \leq x_i, x_i + w_i \leq W, \ 0 \leq y_i, y_i + h_i \leq H, \ \forall c_i \in C;$$

$$x_i = l_i S_w, l_i \in \{0, 1, 2, \ldots\}, \ \forall c_i \in C,$$

Site width S_w and site height R_h are two given constants. The place region is a regular sheet with (0,0)X(W,H)

Our Legalization Flow



Align each cell to the nearest correct row and set an orientation for each cell

Formulate the H- and V- legalization to a MIQP and relax it as a QP and further convert it to LCP,
Duplicate variables to ensure MMSIM convergence & solve the linear system efficiently

Handle illegal cells by graph matching and use Tetris to align cells to nearest placement sites

Outline

- Introduction
- The Problem
- Model and Algorithm
- Experimental Results
- Conclusions

1. Objective Reformulation

$$\min_{x,y} \frac{1}{n'} \sum_i (|x_i - x_i^0| + |y_i - y_i^0|) \quad + \omega \max_i (|x_i - x_i^0| + |y_i - y_i^0|)$$

- The objective is minimizing the weighted sum of the average cell movement and the maximum cell movement, which is transformed as:

$$\min_{x,y} \sum_{c_i \in C} \frac{\alpha_i}{2} ((x_i - x_i^0)^2 + (y_i - y_i^0)^2),$$

- α_i can be seen as a weight on the movement of cell c_i . In fact, if α_i is assigned a proper value, above function includes minimizing the maximum cell movement

1. Objective Reformulation

$$\alpha_i = \left(\frac{(x_i - x_i^0)^2 + (y_i - y_i^0)^2}{\frac{1}{n'} \sum_{c_j \in C} (|x_j - x_j^0| + |y_j - y_j^0|)} \right)^{\kappa}$$

- The parameter $\kappa \geq -1$ is used to make a trade-off between the average cell movement and the maximum cell movement.
  - If $\kappa \gg 1$, it focuses on minimizing the maximum cell movement;
  - if $\kappa = -1$, it focuses on minimizing the average cell movement.
- x_i and y_i are set as the latest iteration results (coordinates) of cell c_i in our algorithm.

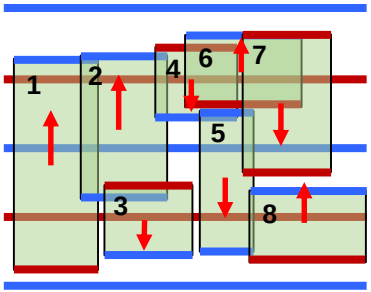
2. VSS/VDD Alignment

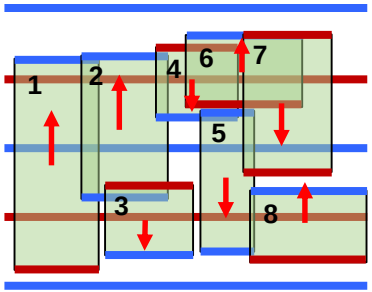
- Cells should be aligned to correct VDD/VSS rails:

$$y_i = k_i' R_h,\ \forall c_i \in C,$$

$$k_i' \in \begin{cases} \{0, 1, 2, \ldots\} \text{ if } c_i \text{ is of an odd-row height;} \\ \{0, 2, 4, \ldots\} \text{ (VDD boundaries) or} \\ \{1, 3, 5, \ldots\} \text{ (VSS), otherwise;} \end{cases}$$

- First align each cell to its nearest correct VSS/VDD lines





2. VSS/VDD Alignment

- After aligning cells to nearest VSS/VDD, if cells densely distribute in some rows, some cells should be realigned to adjacent rows
- Range of cell movement in the vertical direction:

$$y_i \in [y_i' - k_i^l R_h, y_i' + k_i^u R_h],\ \forall sc_i \in SC,$$

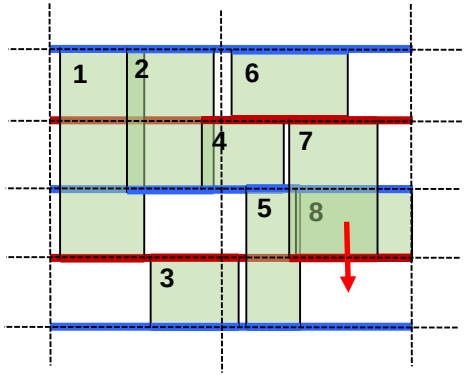
$$\text{where } k_i^l \in \{0, 1, 2, \ldots, \lfloor \frac{y_i'}{R_h} \rfloor\} \text{ and } k_i^u \in \{0, 1, 2, \ldots, \lfloor \frac{H - y_i'}{R_h} \rfloor\}.$$

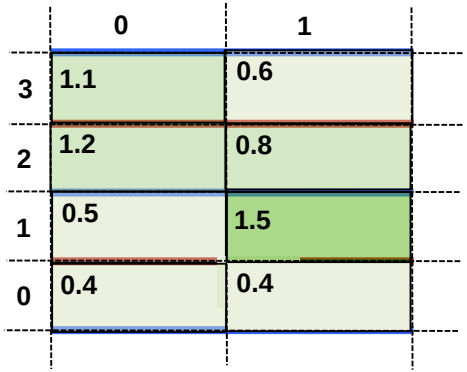
- Further, we preset the orientation of cell movement

$$y_i \in \begin{cases} [y_i' - k_i^l R_h, y_i'], & \text{if } o_i \text{ is downward;} \\ [y_i', y_i' + k_i^u R_h], & \text{if } o_i \text{ is upward,} \end{cases}$$

3. Cell Balance Aware Orientation Presetting

- Moving orientation of a cell is determined by row density
- Circuit layout is split into $I \times J$ bins, $(row_i)_I \times (col_j)_J$
  - Bin size: 1 $Site_h$ × 20 $Site_w$
  - Density $d_{i,j}$ of bin $b_{i,j}$: (the total width of cells in $b_{i,j}$) / (20 $Site_w$)
- Cells may be moved up or down to adjacent rows, such that max{$d_{i,j}$}-min{$d_{i,j}$} is minimized for col_j

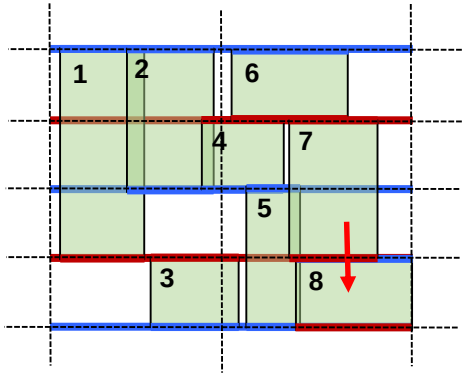


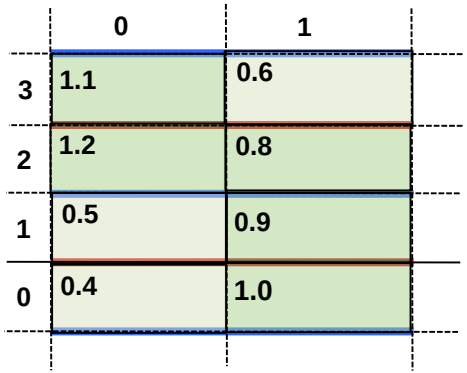


3. Cell Balance Aware Orientation Presetting

- If moving cell c_k up (or down) is good for minimizing the value of max$\{d_{i,j}\}$-min$\{d_{i,j}\}$, then the orientation of cell c_k is set as upward (or downward)

- For example, moving cell *8* down can reduce the value of max$\{d_{i,j}\}$-min$\{d_{i,j}\}$ from 1.1 to 0.4, then the orientation of cell *8* is set as downward



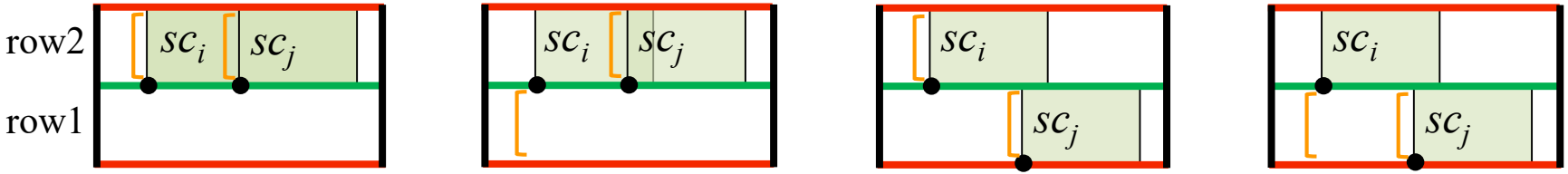


4. x- and y- Non-verlapping Constraints

- Cells spread out along x-coordinate until non-overlapping
  - If $x_i^0 \le x_j^0$, then $x_i + w_i \le x_j \iff x_i + w_i \le x_j + M(1 - z_{ij})$
    $z_{ij} = 1, if\ cell\ i\ and\ j\ are\ in\ the\ same\ row;\ otherwise = 0.$
- $z_{ij}R_h$ reflects the overlapping length of sub-cells sc_i and sc_j in the vertical direction
  - If z_{ij} = 1, i.e., $|y_i - y_j| = 0$, then the vertical overlapping length is R_h;
  - if z_{ij} = 0, i.e., $|y_i - y_j| \ge R_h$, then the vertical overlapping length is 0

$$z_{ij} = \max\left\{1 - \frac{|y_i - y_j|}{R_h}, 0\right\}, \text{ and}$$
$$x_i + w_i \le x_j + M \cdot \min\left\{\frac{|y_i - y_j|}{R_h}, 1\right\}, \text{ if } x_i^0 \le x_j^0$$

4. x- and y- Overlap Constraints



$$y_i \ominus y_j = \begin{cases} y_i - y_j, & \text{if } y_i \geq y_j \text{ for Cases (2) and (3),} \\ & \text{and if } y_i^0 \geq y_j^0 \text{ for Cases (1) and (4);} \\ y_j - y_i, & \text{if } y_i < y_j \text{ for Cases (2) and (3),} \\ & \text{and if } y_i^0 < y_j^0 \text{ for Cases (1) and (4).} \end{cases}$$

- The maximum value of $y_i \ominus y_j$ is $(k_i + k_j)R_h$

$$x_i + w_i \leq x_j + M \cdot \frac{y_i \ominus y_j}{(k_i + k_j)R_h}, \text{ if } x_i^0 \leq x_j^0$$

5. Multiple-Row-Height Cell and Range Constraints

- Range constraints
  - All x_i should be not less than 0, and all $x_i + w_i$ should be not greater than W
- Multiple-row-height cell constraints
  - A multiple-row-height cell c_l is split into single-row-height sub-cells $c_{l_1}, c_{l_2}, \ldots, c_{l_{ri}}$
  - The x-coordinates of these sub-cells should be equal, i.e., $x_{l_1} = x_{l_2} = \cdots = x_{l_{ri}}$
  - The y-coordinates of these sub-cells should satisfy: $y_{l_1} + (r-1)R_h = y_{l_2} + (r-2)R_h = \cdots = y_{l_{ri}}$

6. Quadratic Programming (QP)

$$\min_{x,y} \sum_{sc_i \in SC} \frac{\alpha_i}{2r_i}((x_i - x_i^0)^2 + (y_i - y_i^0)^2)$$

→ Objective function **(2)**

$$\text{s.t.} \quad x_i + w_i \le x_j + \frac{\beta_{ij} \cdot (w_i + w_j)}{(k_i + k_j)R_h} \cdot (y_i \ominus y_j),$$

$$\forall sc_i, sc_j \in SC, \text{if } VMI_i \cap VMI_j \neq \emptyset,$$

$$sc_i, sc_j \text{ are adjacent, and } x_i^0 \le x_j^0;$$

→ x- and y- overlap constraint

$$y_i \in \begin{cases} \left[y_i' - k_i^l R_h, y_i'\right], & \text{if } o_i \text{ is downward,} \\ \left[y_i', y_i' + k_i^u R_h\right], & \text{if } o_i \text{ is upward,} \end{cases}$$

→ VSS/VDD alignment constraint

$$x_i \ge 0, \ \forall sc_i \in SC;$$

→ Range constraint

$$y_{i1} + (r_i - 1)R_h = y_{i2} + (r_i - 2)R_h = \cdots = y_{ir_i},$$

$$x_{i1} = x_{i2} = \cdots = x_{ir_i}, \ sc_{i1}, \cdots sc_{ir_i} \text{ are from the same cell.}$$

→ Multiple-height-cell constraint

6. Quadratic Programming (QP)

- Let $x = (x_1, x_2, \ldots, x_n)^T$, $y = (y_1, y_2, \ldots, y_n)^T$, and $\mu = [x, y]^T$
- *Q* is a diagonal matrix with its elements $q_{i,i}=qn_{+i,n+i} = \frac{\alpha_i}{r_i}$
- *p* is a vector with $p_i = -\frac{\alpha_i x_i^0}{r_i}$ (i=1,2..n), and $p_i = -\frac{\alpha_i y_i^0}{r_{i-n}}$ (i=n+1,n+2..2n)
- *A* is the overlap constraint matrix with only four nonzero elements 1, -1, $\frac{\beta_{ij}(w_i+w_j)}{(k_i+k_j)R_h}$, and $-\frac{\beta_{ij}(w_i+w_j)}{(k_i+k_j)R_h}$ in each row

$$\min_{\mu} \quad \frac{1}{2}\mu^T Q\mu + p^T\mu \qquad (3)$$

$$\text{s.t.} \quad A\mu \geq b;$$

$$d \leq y \leq d^u;$$

$$x \geq 0;$$

$$E\mu = f,$$

$$\bar{y} = y - d,$$

$$\bar{\mu} = [x, \bar{y}]^T$$

$$\min_{\bar{\mu}} \quad \frac{1}{2}\bar{\mu}^T Q\bar{\mu} + \bar{p}^T\bar{\mu} \qquad (4)$$

$$\text{s.t.} \quad A\bar{\mu} \geq \bar{b};$$

$$-I\bar{y} \geq d - d^u;$$

$$\bar{\mu} \geq 0;$$

$$E\bar{\mu} = \bar{f},$$

6. Quadratic Programming (QP)

- To guarantee that the constraint matrix is of full row rank, we increase the number of variables by duplicating $\bar{y}$ to $\underline{y}$ with $\bar{y} = \underline{y}$.
- Let

$$\widetilde{\mu} = \begin{bmatrix} x \\ \bar{y} \\ \underline{y} \end{bmatrix}, \widetilde{Q} = \begin{bmatrix} Q & 0 \\ 0 & 0 \end{bmatrix}, \widetilde{p} = \begin{bmatrix} \bar{p} \\ 0 \end{bmatrix}, \widetilde{A} = \begin{bmatrix} A & 0 \\ 0 & -I \end{bmatrix}, \widetilde{b} = \begin{bmatrix} \bar{b} \\ d - d^u \end{bmatrix}, \widetilde{E} = \begin{bmatrix} E \\ E' \end{bmatrix}, \widetilde{f} = \begin{bmatrix} \bar{f} \\ 0 \end{bmatrix}$$

$$\min_{\widetilde{\mu}} \quad \frac{1}{2}\widetilde{\mu}^T \widetilde{Q}\widetilde{\mu} + \widetilde{p}^T\widetilde{\mu} \qquad \textbf{(5)}$$

$$\text{s.t.} \quad \widetilde{A}\widetilde{\mu} \geq \widetilde{b};$$

$$\widetilde{E}\widetilde{\mu} = \widetilde{f};$$

$$\widetilde{\mu} \geq 0.$$

$$\Longrightarrow$$

$$\min_{\widetilde{\mu}} \frac{1}{2}\widetilde{\mu}^T (\widetilde{Q} + \lambda\widetilde{E}^T\widetilde{E})\widetilde{\mu} + (\widetilde{p}^T - \lambda\widetilde{f}^T\widetilde{E})\widetilde{\mu} \qquad \textbf{(6)}$$

$$\text{s.t.} \quad \widetilde{A}\widetilde{\mu} \geq \widetilde{b};$$

$$\widetilde{\mu} \geq 0.$$

6. Quadratic Programming (QP)

$$\min_{\widetilde{\mu}} \quad \frac{1}{2}\widetilde{\mu}^T(\widetilde{Q} + \lambda\widetilde{E}^T\widetilde{E})\widetilde{\mu} + (\widetilde{p}^T - \lambda\widetilde{f}^T\widetilde{E})\widetilde{\mu} \qquad (6)$$

$$\text{s.t.} \quad \widetilde{A}\widetilde{\mu} \geq \widetilde{b};$$

$$\widetilde{\mu} \geq 0.$$

Proposition 1: In above problem, $\tilde{Q} + \lambda\tilde{E}^T\tilde{E}$ is a symmetric positive definite matrix.

Proposition 2: In above problem, if $k_i^l = k_i^u = 1$ for all sub-cells, then matrix $\tilde{A}$ if of full row rank.

7. Linear Complementarity Problem (LCP)

- QP is converted into an LCP
  - LCP gives the optimal solution of the QP
- LCP(B, t):
  - Given: a large, sparse & real matrix $B = (b_{ij})^{n \times n}$, and a real vector $t = (t_1, t_2, \dots, t_n)^T \in R^n$
  - Goal: find a pair of real vectors w and $z \in R^n$ s. t.

$$w = Bz + t \geq 0, z \geq 0 \text{ and } z^T w = 0.$$

- The **M**odulus-based **M**atrix **S**plitting **I**teration **M**ethod (MMSIM) is the most effective, efficient method for solving LCP(B, t) [1]

[1] J. Chen, Z. Zhu, W. Zhu, and Y.-W. Chang. Toward optimal legalization for mixed-cell-height circuit designs. In Proceedings of ACM/IEEE Design Automation Conference, 2017.

8. Horizontal- and Vertical-Direction Legalization

Input: matrices: $M(\varepsilon)$, N, $B(\varepsilon)$; vectors: t, $s^{(0)}$, $z^{(0)}$; parameters: γ, δ, σ, κ;

Output: horizontal- and vertical-direction legalization result.

1: $l = 0$;
2: **do**
3: $x_i^{(l)} = z_i^{(l)}$, $y_i^{(l)} = z_{n+i}^{(l)} + d_i$, $i = 1, 2, \cdots, n$;
4: $\alpha_i = \left(\frac{(x_i^{(l)} - x_i^0)^2 + (y_i^{(l)} - y_i^0)^2}{\frac{1}{n}\sum_{j=1}^{n}(|x_j^{(l)} - x_j^0| + |y_j^{(l)} - y_j^0|)} \right)^{\kappa}$;
5: solve $(M(\varepsilon) + I)s^{(l+1)} = Ns^{(l)} + (I - B(\varepsilon))|s^{(l)}| - \gamma t$;
6: $z^{(l+1)} = \frac{1}{\gamma}(|s^{(l+1)}| + s^{(l+1)})$;
7: l++;
8: **until** $|z^{(l)} - z^{(l-1)}| < \delta$
9: obtain the coordinate (x, y) for each sub-cell by $z^{(l)}$;

Use previous iteration results to update α

MMSIM

8. Horizontal- and Vertical-Direction Legalization

9: obtain the coordinate (x, y) for each sub-cell by $z^{(l)}$;
10: $i = 0$;
11: **do**
12: **if** $\frac{|y_i - y'_i|}{R_h} < \sigma$, **then** $y_i = y'_i$;
13: **if** $\frac{y_i - (y'_i - kR_h)}{R_h} < 1 - \sigma$, **then** $y_i = y'_i - kR_h$;
14: **if** $\frac{(y'_i + kR_h) - y_i}{R_h} < 1 - \sigma$, **then** $y_i = y'_i + kR_h$;
15: i++;
16: **until** $i \geq n$
17: **Return** x, y.

Align cell to the nearest correct row

Outline

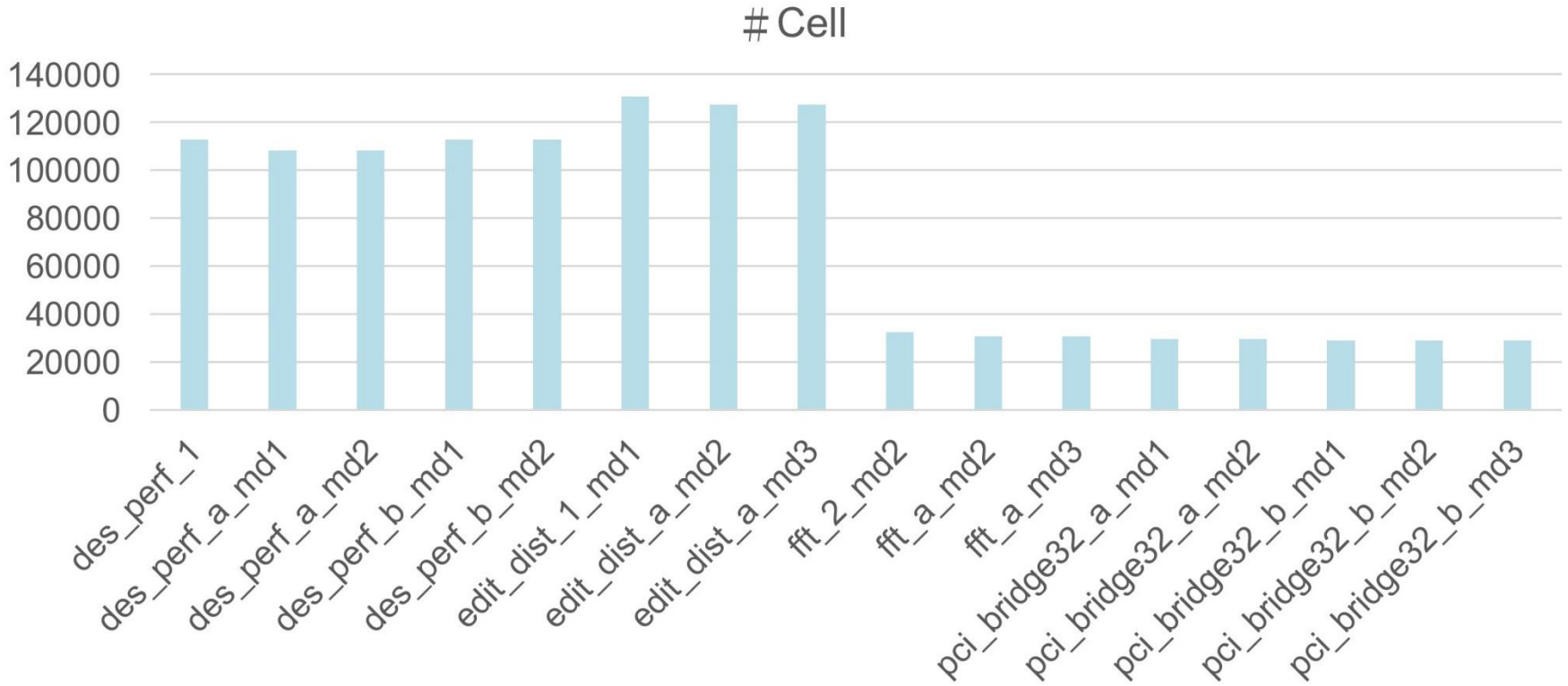
- Introductions
- Problem Formulation
- Model and Algorithm
- **Experimental Results**
- Conclusions

Experimental Settings

- Platform and parameters
  - C++ programming language
  - Unix machine with Intel Core 2.70 GHz CPU and 8 GB memory
  - IP solver: CPLEX
  - Parameter λ was set as 500. β^* and θ^* in the splitting matrices M and N were both set as 0.5, and γ, σ, and κ were set as 1, 0.4, and 1, respectively
- Sixteen Benchmarks
  - Modified from ICCAD-2017 CAD Contest on Multi-Deck Standard-Cell Legalization by omitting the fence-region constraints and the soft constraints
- Algorithms
  - DAC'17: Total cell movement, spread cell along row, MMSIM
  - MIQP: Total and maximum cell movement, global cell spreading, IP solver
  - LCP: Total and maximum cell movement, global cell spreading, MMSIM
- Indicators
  - HPWL(%), Average movement(sites), maximum movement(sites), CPU(s)

The Number of Cells

- The numbers of cells of benchmarks range from thirty thousand to one hundred and thirty thousand.

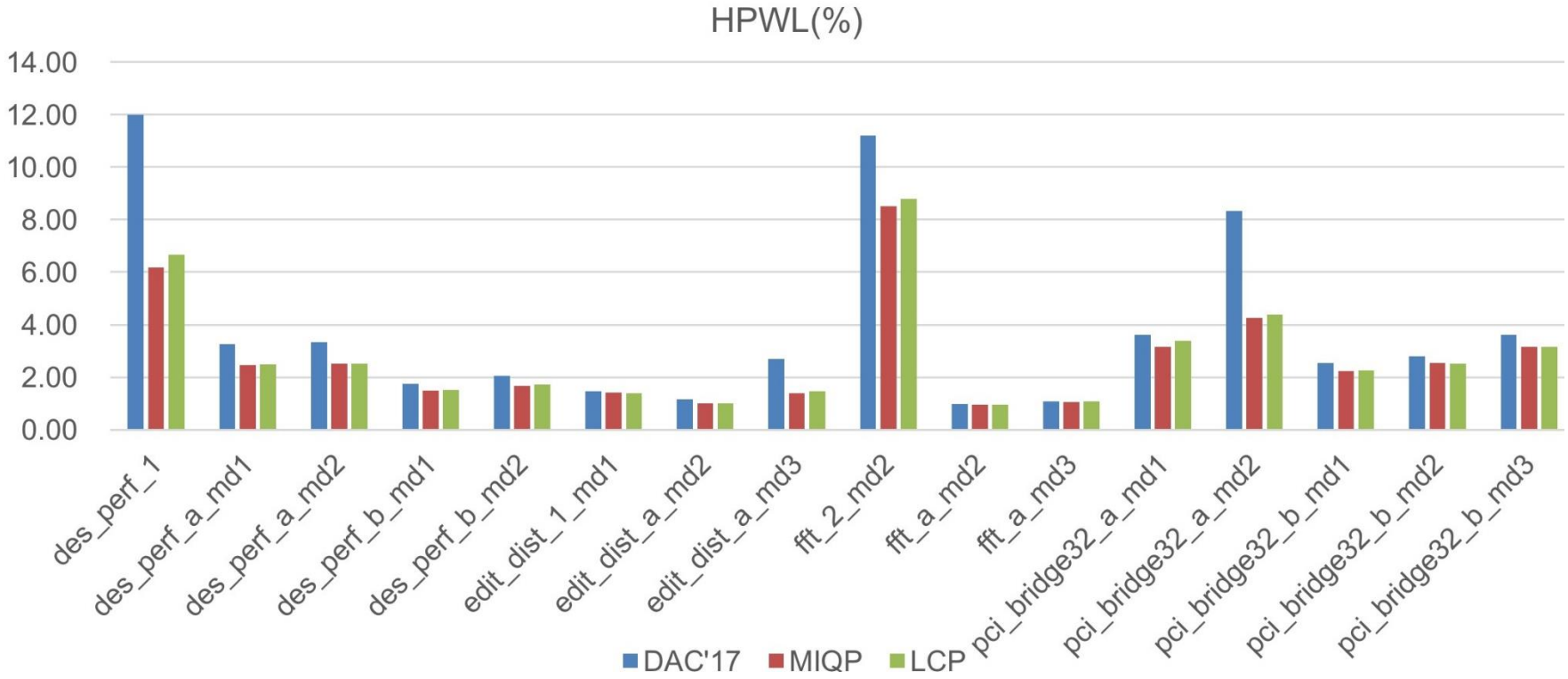


Total Experimental Results

Benchmarks	ΔHPWL(%)			Avg. Move. (sites)			Max. Move. (sites)			CPU(s)		
	DAC'17	MIQP	LCP	DAC'17	MIQP	LCP	DAC'17	MIQP	LCP	DAC'17	MIQP	LCP
des_perf_1	16.21	6.19	6.66	10.86	6.65	6.97	200.82	48.67	48.95	11.23	1672.24	11.75
des_perf_a_md1	3.27	2.47	2.48	6.71	5.93	5.94	607.30	607.30	607.30	2.30	917.02	2.79
des_perf_a_md2	3.35	2.52	2.51	6.77	5.90	5.93	403.86	403.86	403.86	2.19	839.70	6.82
des_perf_b_md1	1.75	1.50	1.52	5.17	4.75	4.77	79.34	51.14	38.45	2.01	905.74	3.64
des_perf_b_md2	2.05	1.68	1.72	5.74	5.22	5.25	198.74	54.75	39.76	2.31	991.30	3.12
edit_dist_1_md1	1.47	1.41	1.39	6.22	5.73	5.79	109.34	107.64	95.45	3.49	1300.40	5.19
edit_dist_a_md2	1.17	1.01	1.01	6.02	5.51	5.51	164.00	164.00	164.00	2.59	1313.57	2.24
edit_dist_a_md3	2.69	1.39	1.48	9.11	6.77	7.08	233.00	233.00	233.00	5.91	1357.08	15.68
fft_2_md2	11.21	8.52	8.78	8.84	7.44	7.54	102.94	62.69	73.60	0.70	69.52	2.89
fft_a_md2	0.98	0.95	0.95	5.03	4.86	4.86	345.50	345.50	345.50	0.69	57.03	0.60
fft_a_md3	1.08	1.07	1.08	4.73	4.54	4.55	109.62	109.62	109.62	0.63	55.56	0.40
pci_bridge32_a_md1	3.61	3.16	3.38	6.01	5.53	5.64	72.48	63.76	63.76	0.61	44.38	2.29
pci_bridge32_a_md2	8.33	4.26	4.38	9.43	7.01	7.14	186.08	121.35	121.35	0.53	54.57	3.34
pci_bridge32_b_md1	2.55	2.23	2.26	6.35	5.94	6.01	322.71	332.71	332.71	0.52	44.03	0.70
pci_bridge32_b_md2	2.80	2.54	2.53	5.92	5.52	5.53	640.12	430.04	430.04	0.50	42.11	0.66
pci_bridge32_b_md3	3.63	3.17	3.17	6.74	6.10	6.10	398.57	398.57	398.57	0.51	43.26	1.58
N.Average	1.32	0.98	1.00	1.14	0.98	1.00	1.61	1.02	1.00	0.67	158.25	1.00

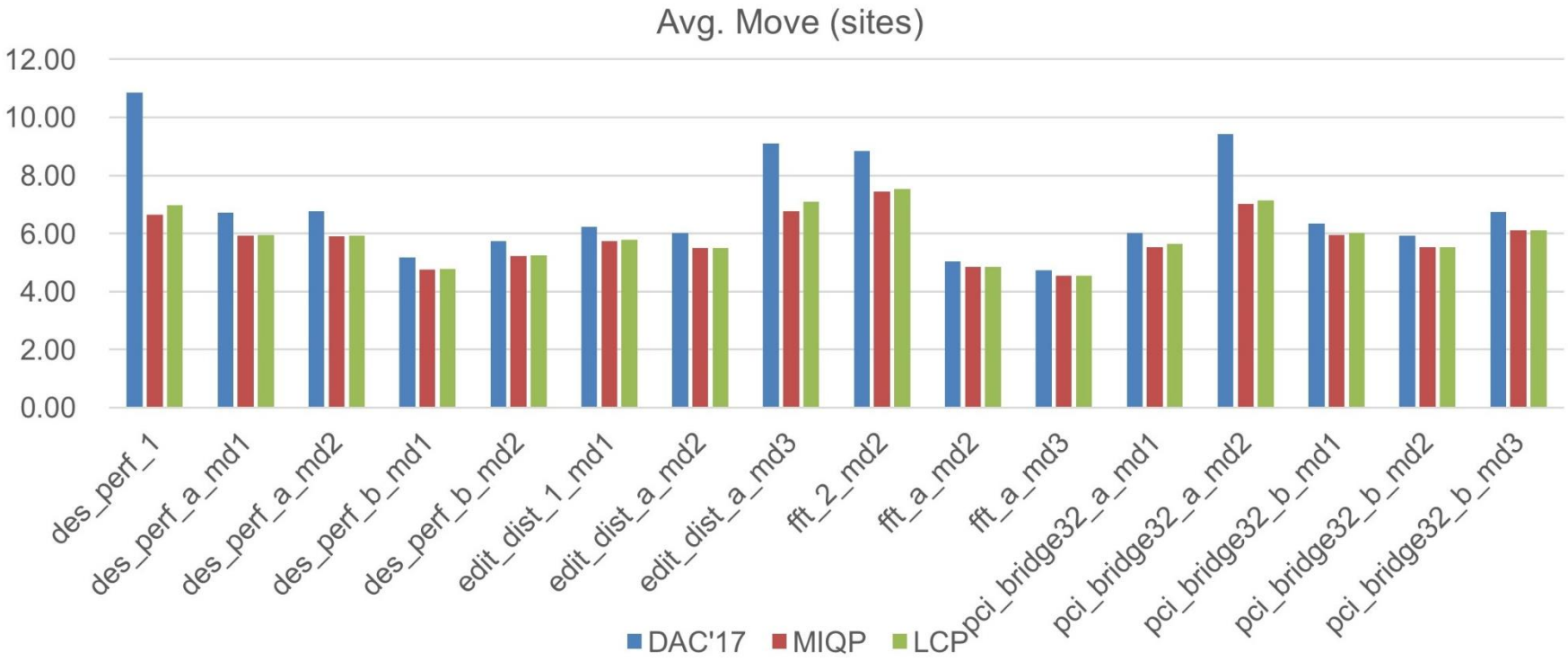
Comparison: HPWL

- Compared with “DAC’17,” “LCP” achieves **32%** improvement on HPWL
- Compared with optimal “MIQP”, “LCP” loses only **2%** on HPWL



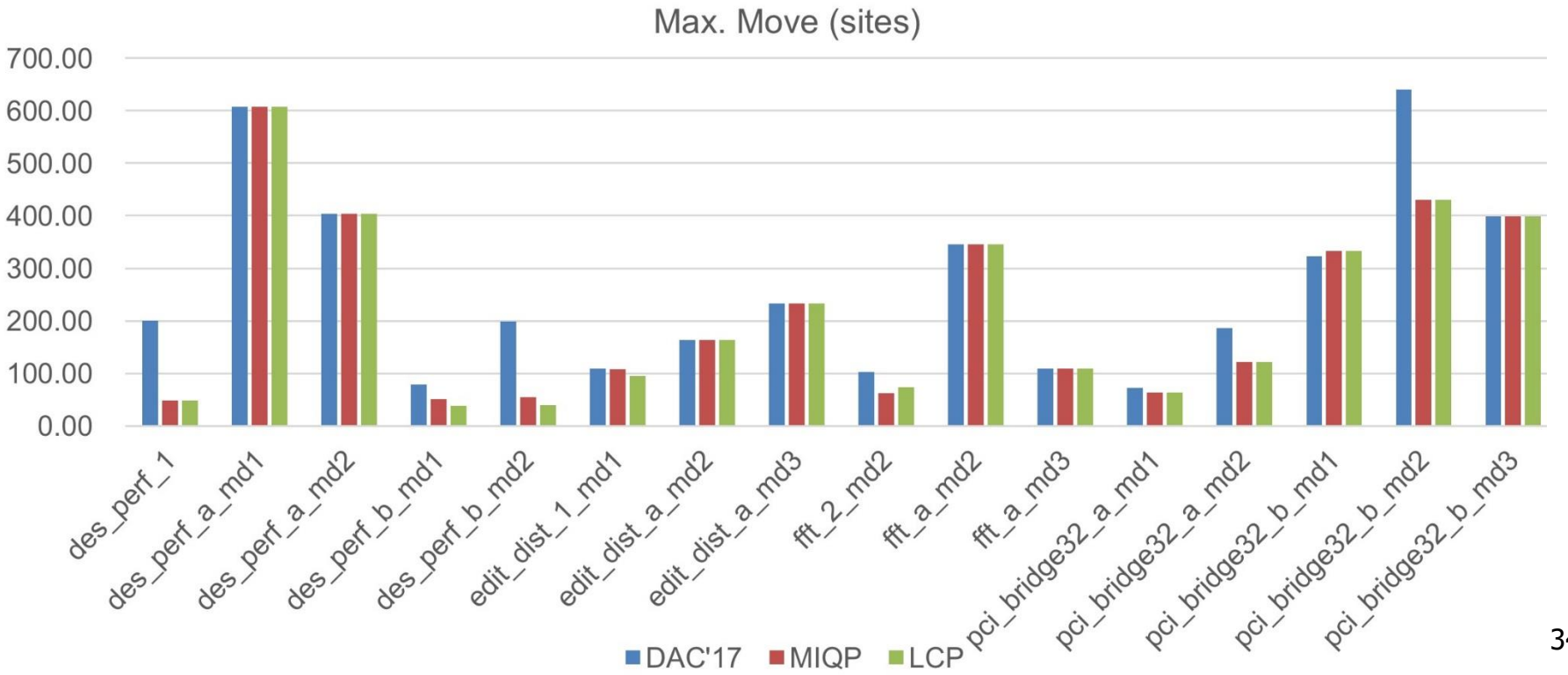
Comparison: Average Movement

- Compared with “DAC’17,” “LCP” achieves **14%** improvement on Avg. Move.
- Compared with optimal “MIQP”, “LCP” loses only **2%** on Avg. Move.



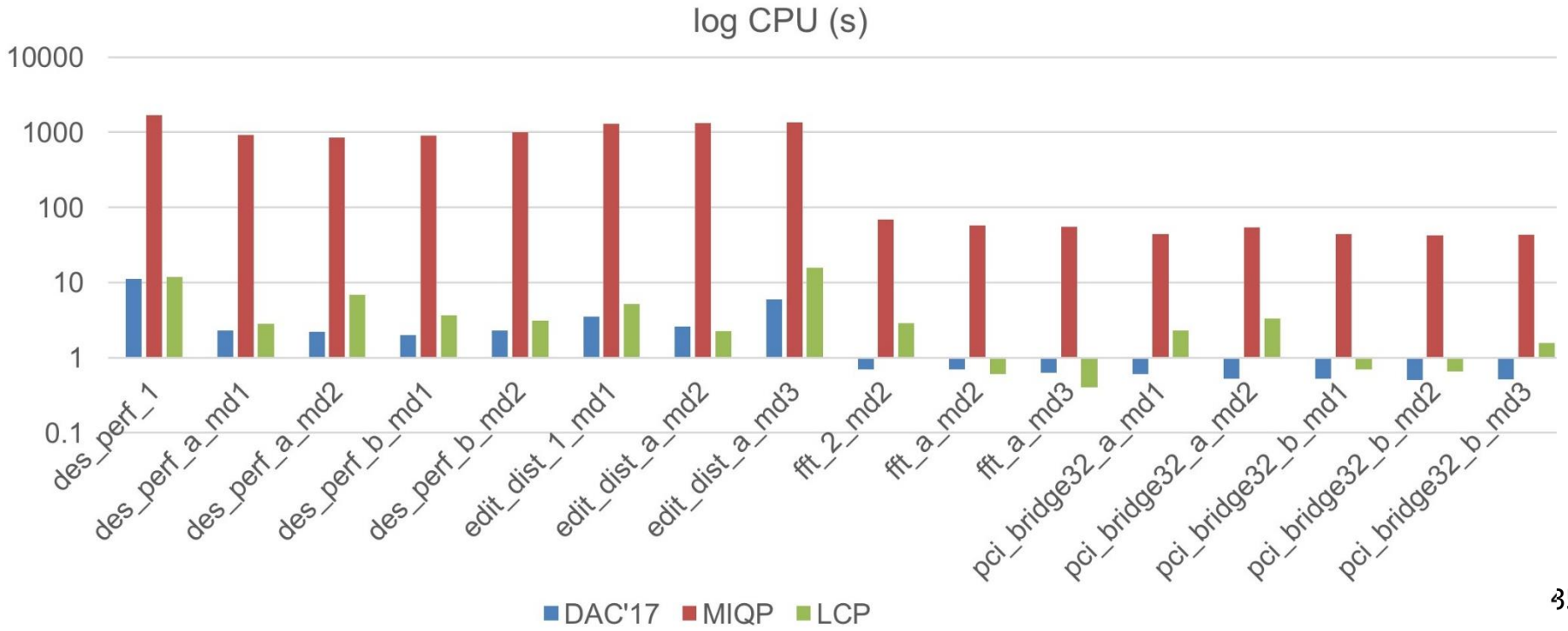
Comparison: Maximum Movement

- Compared with “DAC’17” and “MIQP”, “LCP” achieves **61%** and **2%** improvement on Max. Move.



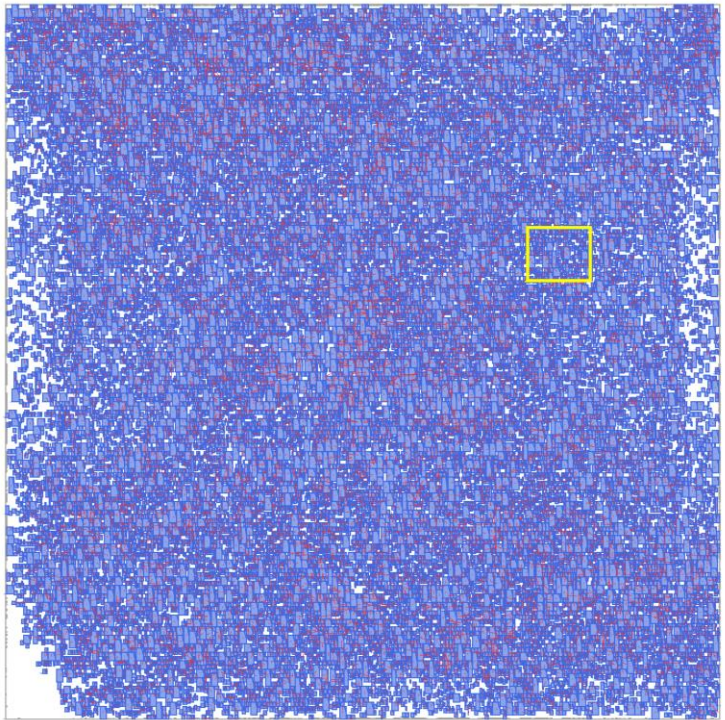
Comparison: Runtime

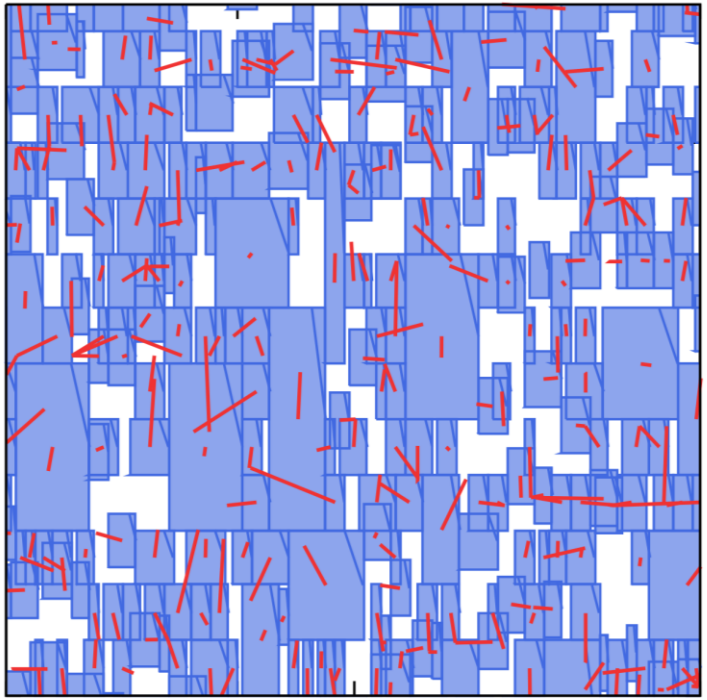
- "LCP" is **158X** faster than "MIQP", and "DAC'17" is **1.5X** faster than "LCP"



An Example

- Legalization result of the benchmark “ t_2_md2” from our algorithm
- Cells are in blue, and movement in red
- All cells are legalized and the total movement is small





Outline

- Introductions
- Problem Formulation
- Model and Algorithm
- Experimental Results
- Conclusions

Conclusions

- We formulate the mixed-cell-height standard-cell legalization problem as a mixed integer quadratic program (MIQP),
  - Average cell movement, Maximum cell movement
- By relaxing discrete constraints to linear ones, we convert the MIQP to a quadratic programming problem (QP),
  - Spreading cells continuously in both the horizontal and vertical directions
- We apply a series of operations to guarantee convergence of the MMSIM.
- Experimental results show that our analytical legalization method is effective in reducing the average and maximum cell movements.

Thank You!