DREAMPlace 3.0: Multi-Electrostatics Based Robust VLSI Placement with Region Constraints

Jiaqi Gu[1], Zixuan Jiang[1], Yibo Lin[2] and David Z. Pan[1]

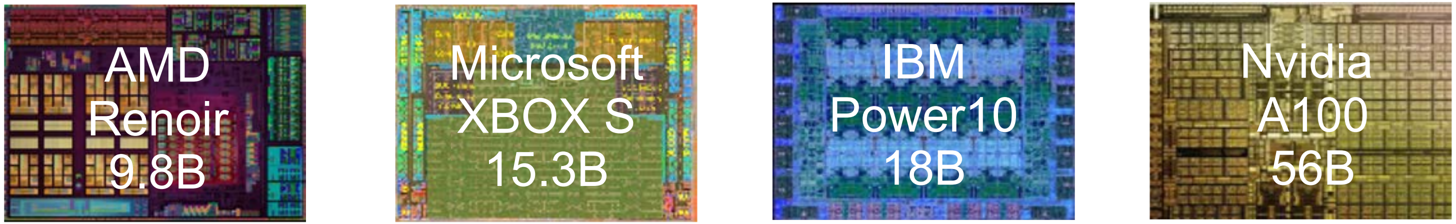
[1]ECE Department, The University of Texas at Austin

[2]Peking University

jqgu@utexas.edu

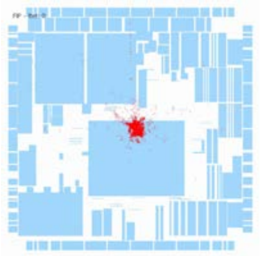
VLSI Placement and Challenges

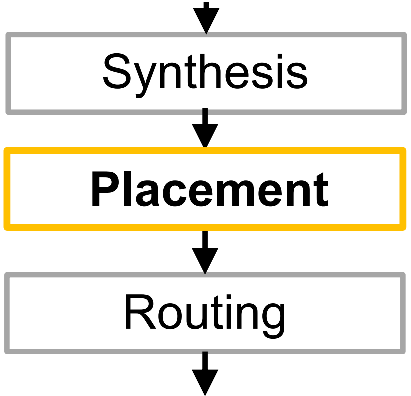
- Modern VLSI scale and design complexity grow rapidly
  - Billion-cell design
  - More design rules and constraints
  - Higher performance requirements



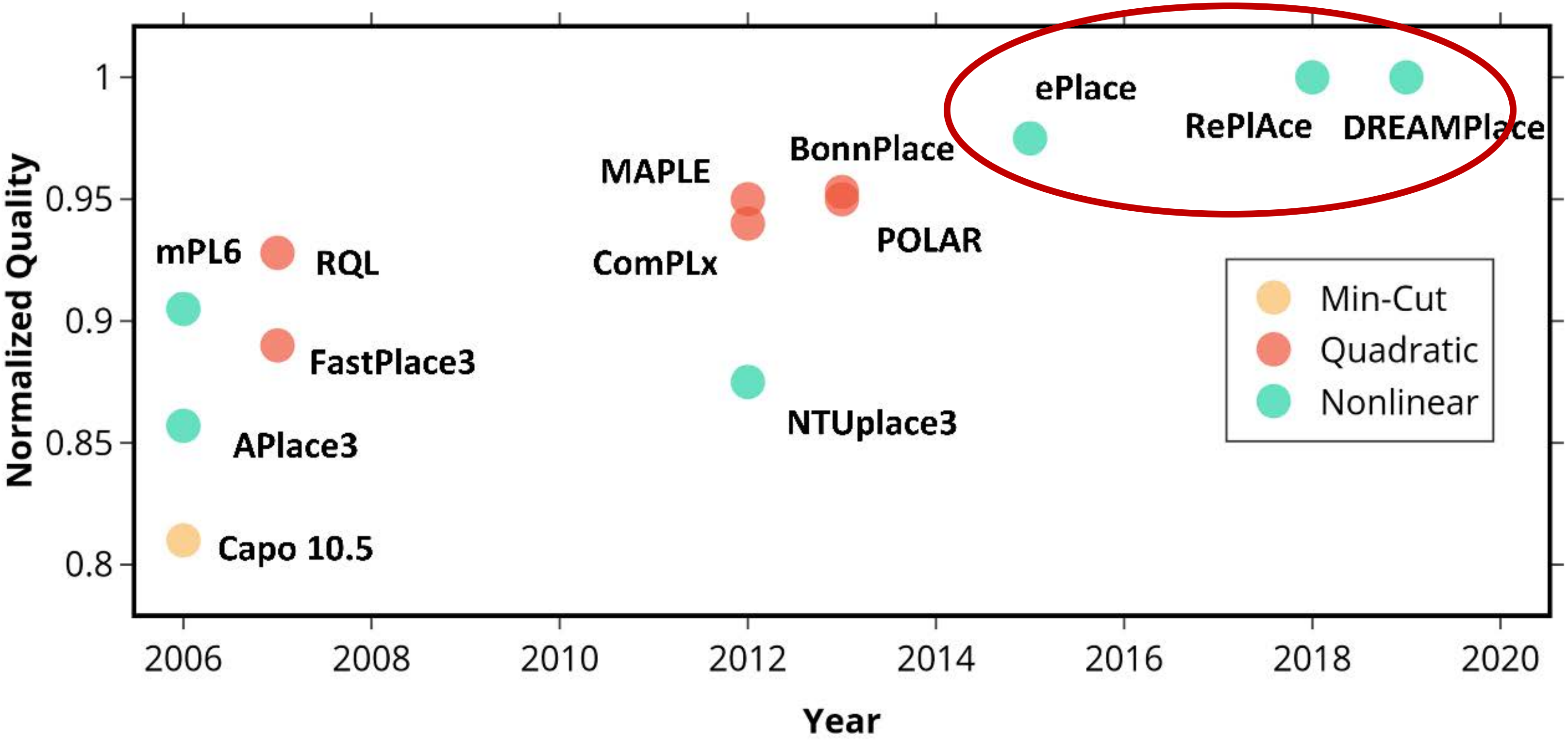
- Placement plays a ***critical*** role in design closures
  - Wirelength
  - Congestion / Routability
  - Timing
  - …

[*Courtesy RePlAce*]





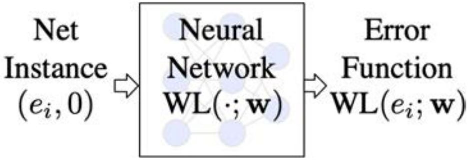
Recent Development of VLSI Placement



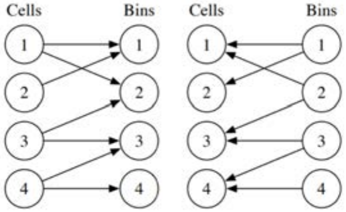
*Data collected from RePlAce [Cheng+, *TCAD'18*] and http://vlsi-cuda.ucsd.edu/~ljw/ePlace/ on ISPD 2005 benchmarks

DREAMPlace Evolution

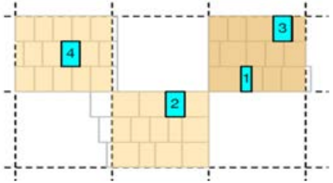
0.0
- DAC 2019
- DREAMPlace: VLSI placement using DL framework



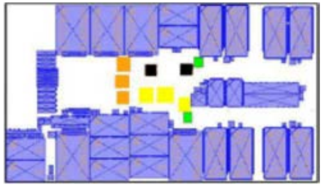
1.0
- TCAD 2020
- Improved kernels; Routabilty-driven placement



2.0
- TCAD 2020
- ABCDPlace: accelerated detailed placement



3.0
- **ICCAD 2020**
- **Multi-electrostatics-based placement**



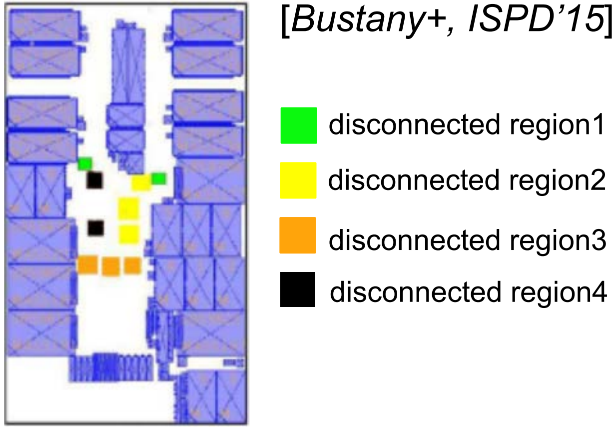
Placement with Region Constraints

- Place cells with the same function in a confined subregion
  - Support voltage islands
  - Improve manufacturability
  - Reduce datapath delay
  - Decrease clock power

- *Fence region*
  - Member-hard and non-member-hard
  - Cell assignment is exclusive
  - Hard constraints

- Severe quality loss if not considered

[*Bustany+, ISPD'15*]



mgc_superblue11_a

ISPD 2015
mgc_des_perf_a

Placement Formulation with Fence Region

$$\min_{\mathbf{x},\mathbf{y}} \sum_{e \in E} \mathrm{WL}(e; \mathbf{x}, \mathbf{y}) \quad \text{s.t.} \quad \mathcal{D}(\mathbf{x}, \mathbf{y}) \le \hat{\mathcal{D}},$$

$$v_k = (\mathbf{x}_k, \mathbf{y}_k) \in r_k, \quad k = 0, \cdots, K$$

Relax

$$\min_{v} \sum_{e \in E} \mathrm{WL}(e; v) + \langle \lambda, \mathcal{D}(v, r) \rangle$$

$$\lambda = (\lambda_0, \cdots, \lambda_K)$$

$$\mathcal{D}(v, r) = (\mathcal{D}(v_0, r_0), \cdots, \mathcal{D}(v_K, r_K))$$

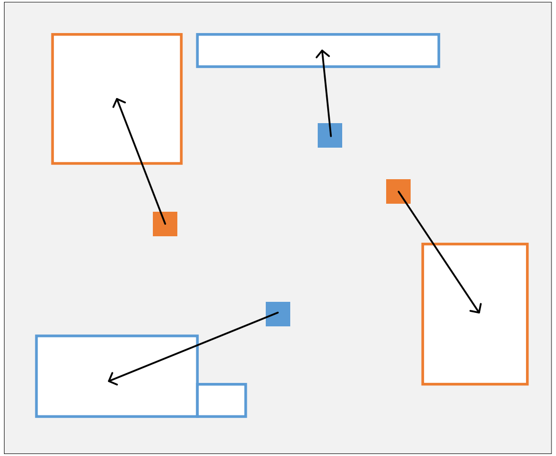
Previous solutions

- `NTUplace4dr`: region-aware clustering + new wirelength model [*Huang+, TCAD'18*]
- `Eh?Placer`: upper-bound-lower-bound + look-ahead legalization [*Darav+,TODAES'16*]
- `RippleDR`: upper-bound-lower-bound + look-ahead legalization [*Chow+, SLIP'17*]
- `ePlace`-family: not supported

Challenge: **Efficient and robust region-aware placement with a global view**

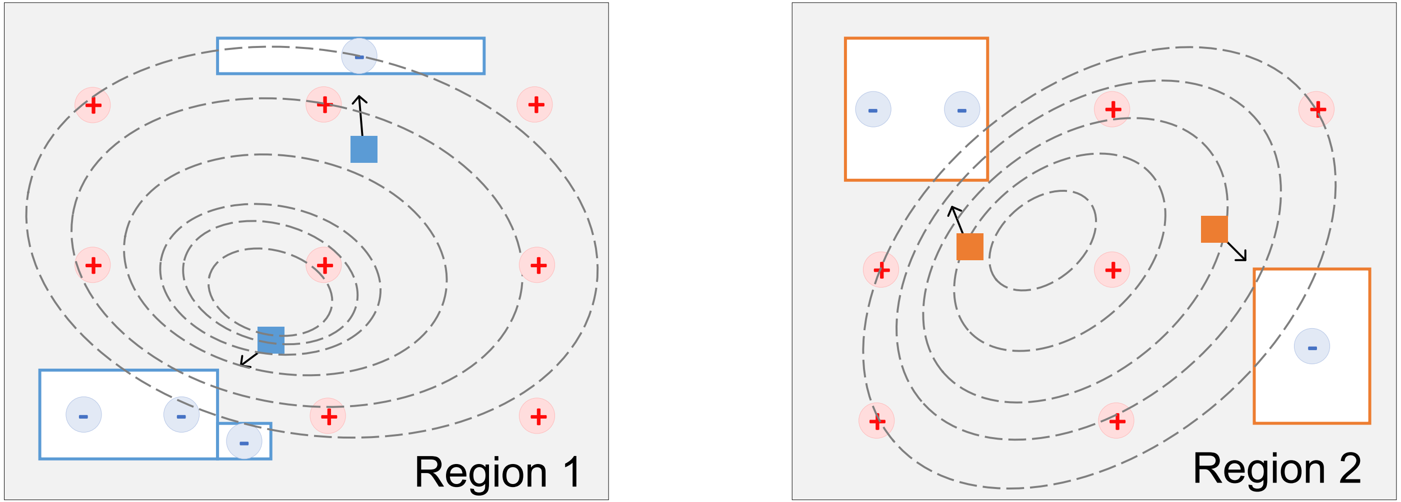
Intuition Behind Cell Assignment

- Clustering & Partitioning [NTUplace4dr]
  - Local view ×
  - Region capacity aware √
  - Suboptimal solution ×



Cell Assignment via Multi-Electrostatics

- Multi-electrostatic system
  - Global view for cell assignment √
  - Low computation complexity √
  - Region capacity aware √



Proposed Method

- Multi-Electrostatics based placement

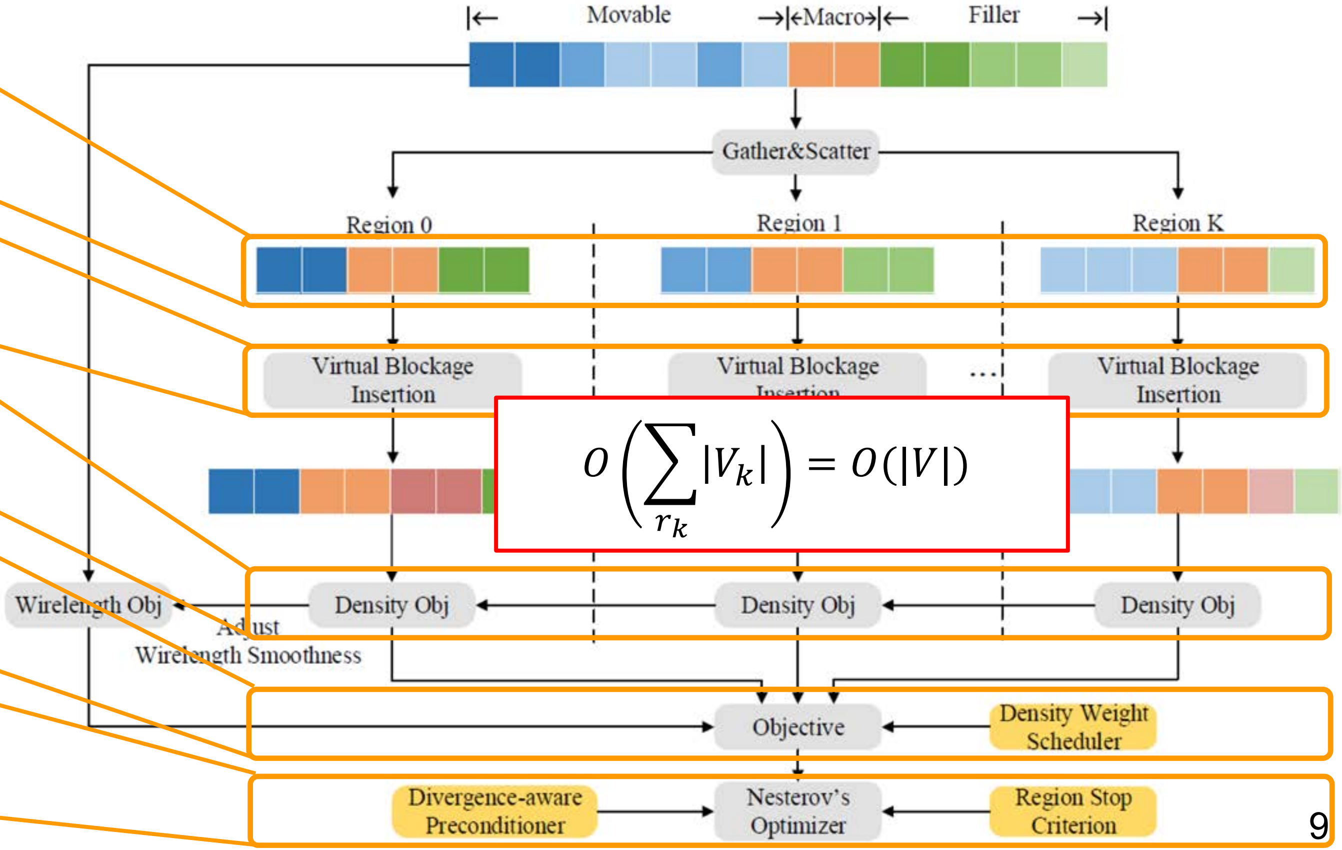
Subregion Partitioning

Insert Virtual Blockage

Isolated Density Optimization

Individual Density Weight Scheduling

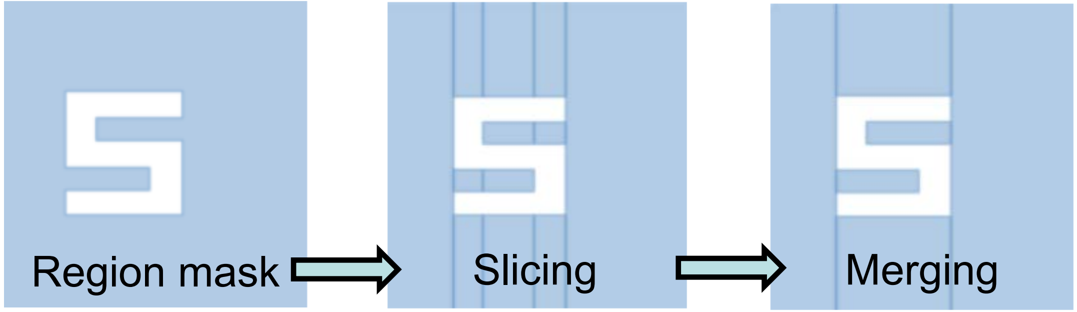
Robust Optimizer with Entropy Injection

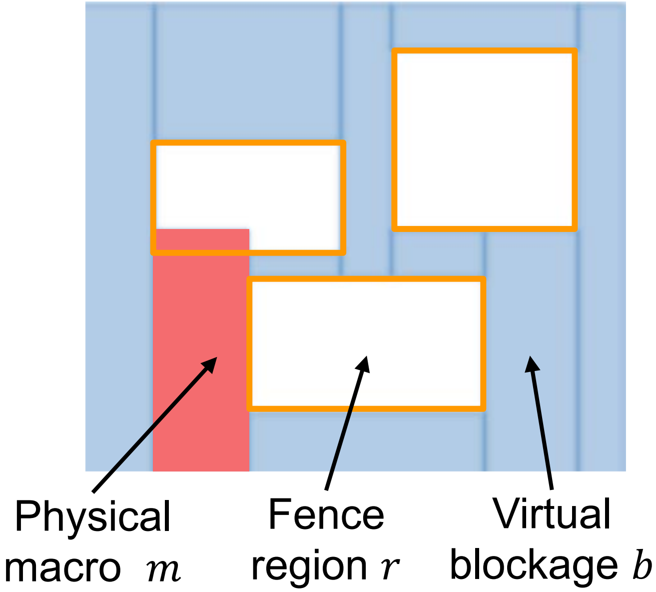


$$O\left(\sum_{r_k}|V_k|\right) = O(|V|)$$

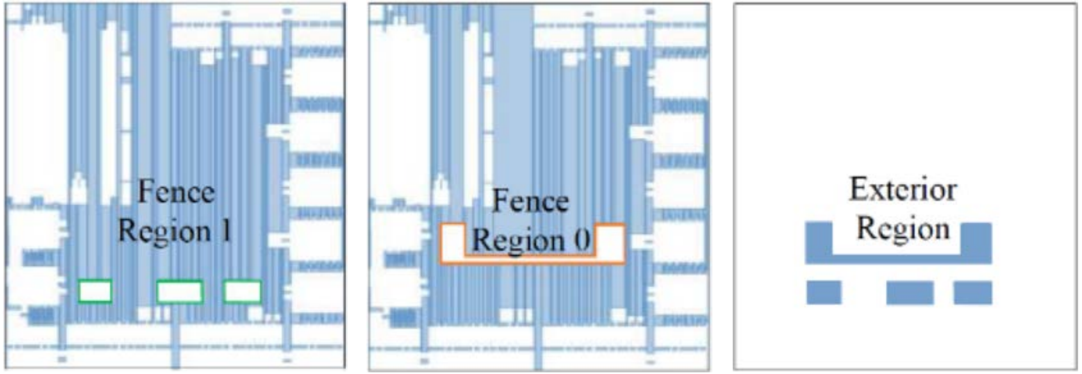
Virtual Blockage Insertion

- Virtual blockage insertion
  - Rectangle slicing





ISPD2015 superblue_16a



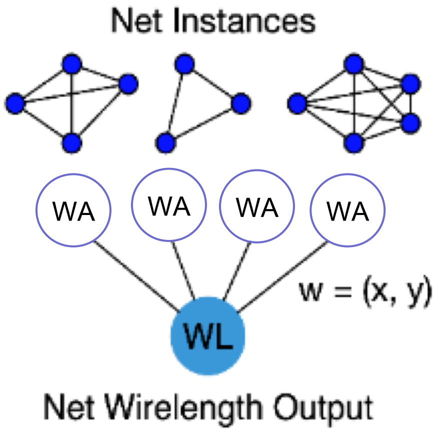
$$\hat{\mathcal{D}}_k = \max\left(LocalAreaUtil + \epsilon, \hat{\mathcal{D}}\right) = \max\left(\frac{Area(v_k)}{Area(r_k \backslash m)} + \epsilon, \hat{\mathcal{D}}\right)$$

Quadratic Density Penalty

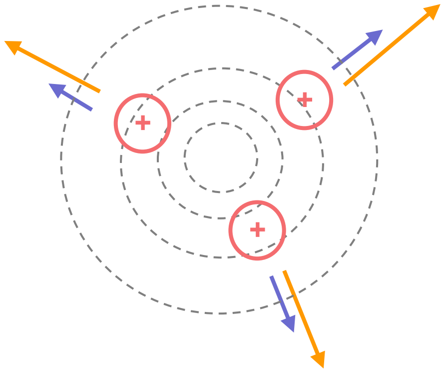
- Modified augmented Lagrangian formulation [*Zhu+, DAC 2018*]

$$f = \sum_{e \in E} \mathrm{WL}(e; v) + \left\langle \lambda, \mathcal{D}(v, r) + \frac{1}{2}\mu \mathcal{P}_\lambda \odot \mathcal{D}^2(v, r) \right\rangle$$

- Wirelength [*Hsu+, TCAD 2013*]
  - Weighted-average WL model with smoothness control



- Quadratic term
  - Accelerate initial spreading



- Density weight $\lambda = (\lambda_0, \cdots, \lambda_K)$
  - Independent for each region
  - Also controls quadratic term

Density Weight Scheduling

- Update Lagrangian multiplier λ
  - Normalized preconditioned sub-gradient descent

$$\hat{\nabla}_\lambda f = \nabla_\lambda f \odot \mathcal{P}_\lambda$$

$$\lambda \leftarrow \min\left(\lambda_{max}, \lambda + \alpha \frac{\hat{\nabla}_\lambda f}{||\hat{\nabla}_\lambda f||_2}\right)$$

- Adaptive step size α
  - Exponentially increased step size based on density

$$\alpha \leftarrow \gamma(\mathcal{D}, \mathcal{P}_\lambda)\alpha$$

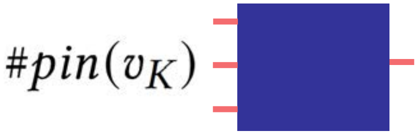
Preconditioned Nesterov's Optimizer

- Multi-field divergence-aware preconditioning
  - Stabilize optimization for the exterior region

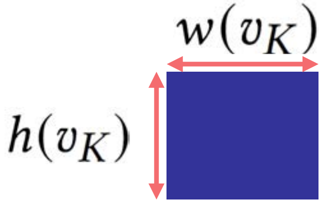
$$\hat{\nabla} f = \nabla f \odot \mathcal{P}$$

$$\mathcal{P}_K = \min\left(1, \left(\nabla^2_{v_K} \sum WL(e, v) + \beta \lambda_K \nabla^2_{v_K} \mathcal{D}(v_K, r_K)\right)^{-1}\right)$$

- Wirelength Hessian [*Courtesy ePlace*]
  - Estimate the diagonal by pin count of an instance



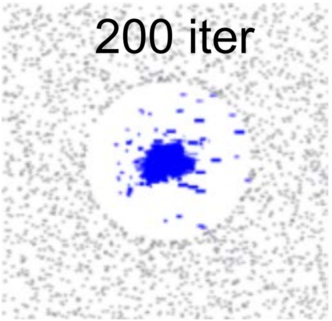
- Density Hessian [*Courtesy ePlace*]
  - Estimate the diagonal by instance area



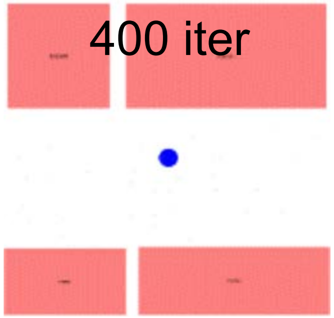
- Exponentially increased β factor to slow down large-cell movement

Intuition Behind Optimizer Robustness

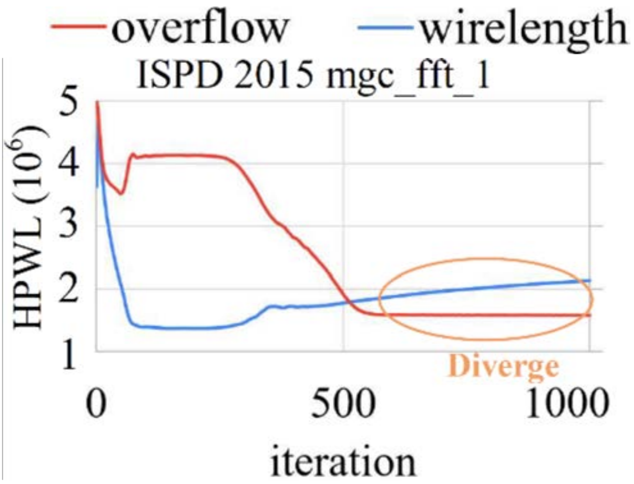
- Slow convergence
  - Slow spreading
  - 30%-50% runtime for spreading
- Optimizer divergence
  - Stagnant density overflow
  - Increasing wirelength
- Stuck in saddle-point
  - Saddle-point circle that harms the HPWL

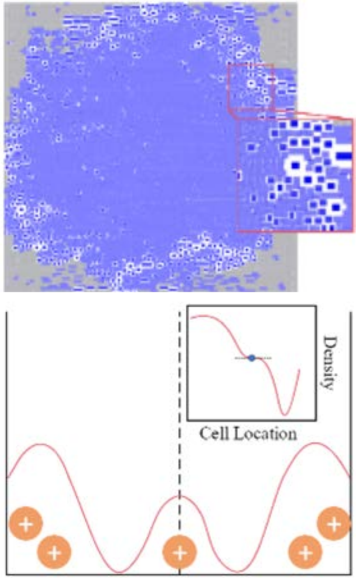


[ISPD'19 test1]



[ISPD'15 mgc_des_perf_a]





[ISPD'19 test1]

Robust Placement

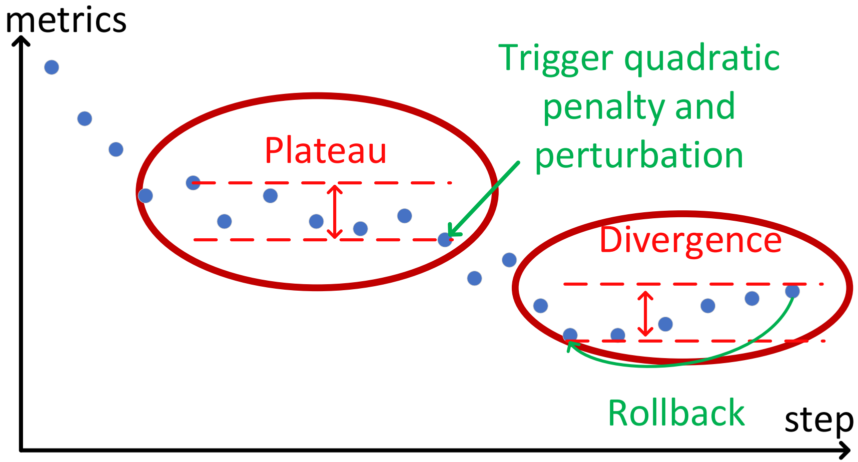
- Adaptive quadratic penalty and entropy injection
  - Window-based plateau detector

$$\mathrm{PLT} = \begin{cases} \dfrac{\max_L(OVFL) - \min_L(OVFL)}{\mathrm{avg}_L(OVFL)} < \delta_{PLT}, & OVFL > 0.9 \\ \mathrm{False}, & OVFL \leq 0.9, \end{cases}$$

  - Quadratic penalty with doubled density weight if triggered
  - Entropy injection as location perturbation and shrinking
    - Escape saddle-point
    - Faster convergence

$$\hat{x} = s\left(x - \frac{\sum_{i \in v} x_i}{|v|}\right) + \frac{\sum_{i \in v} x_i}{|v|} + \Delta x$$

- Divergence-aware rollback



Post-GP Placement

- Fence region aware legalization
  - Per region greedy legalization (`gl`) with virtual blockage

$$v_k^g \leftarrow \mathtt{gl}(v_k^m, m, b_k)$$

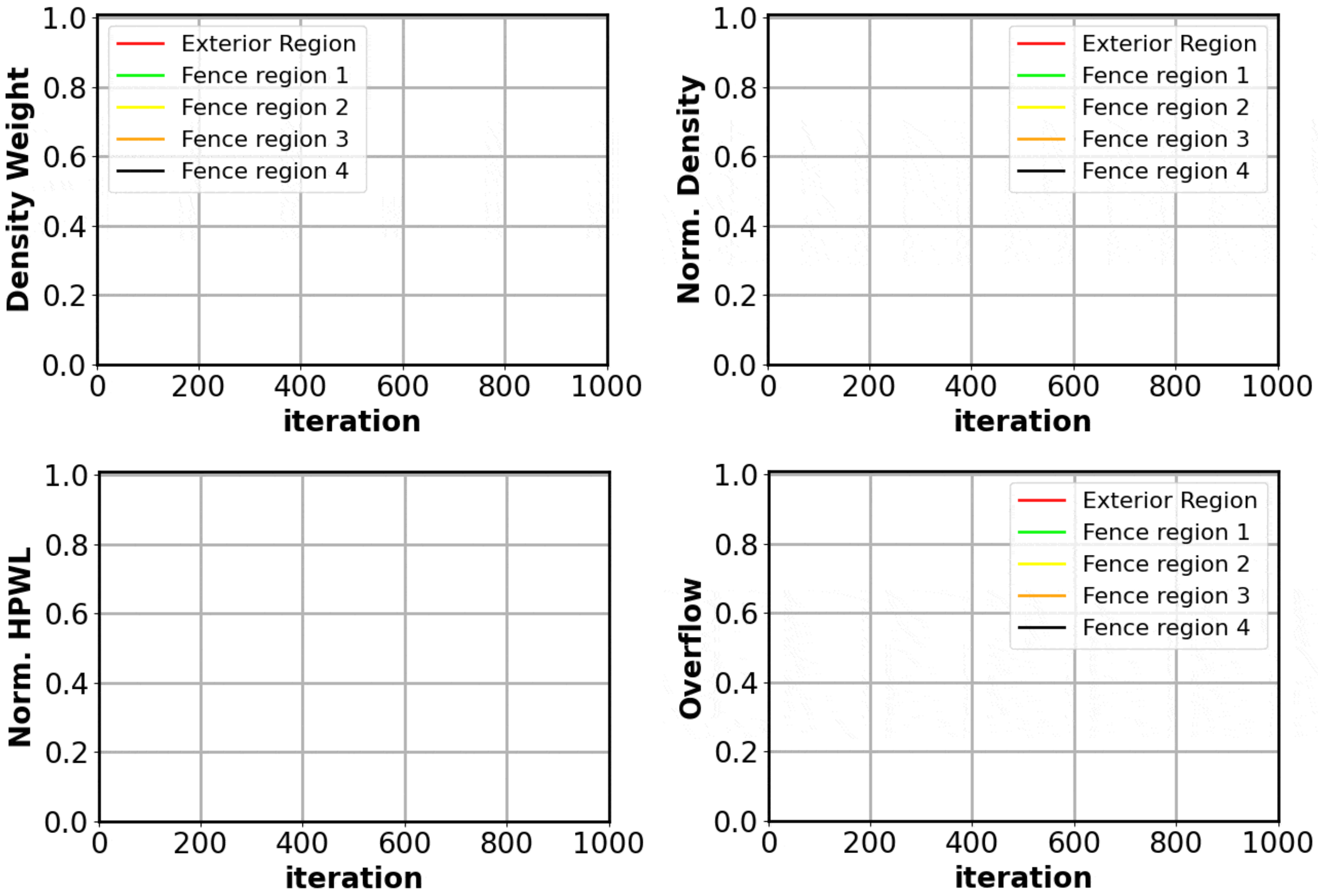
  - `Abacus` (`al`) [*Spindler+, ISPD'08*] algorithm to minimize displacement with virtual blockage

$$\tilde{v_k} \leftarrow \mathtt{al}(v_k^m, v_k^g, m, b_k)$$

- Finish the flow with detailed placement using `ABCDPlace` [*Lin+, TCAD 2019*]
  - Support fence region constraints

DREAMPlace 3.0 Animation

ISPD'15 mgc_superblue11_a

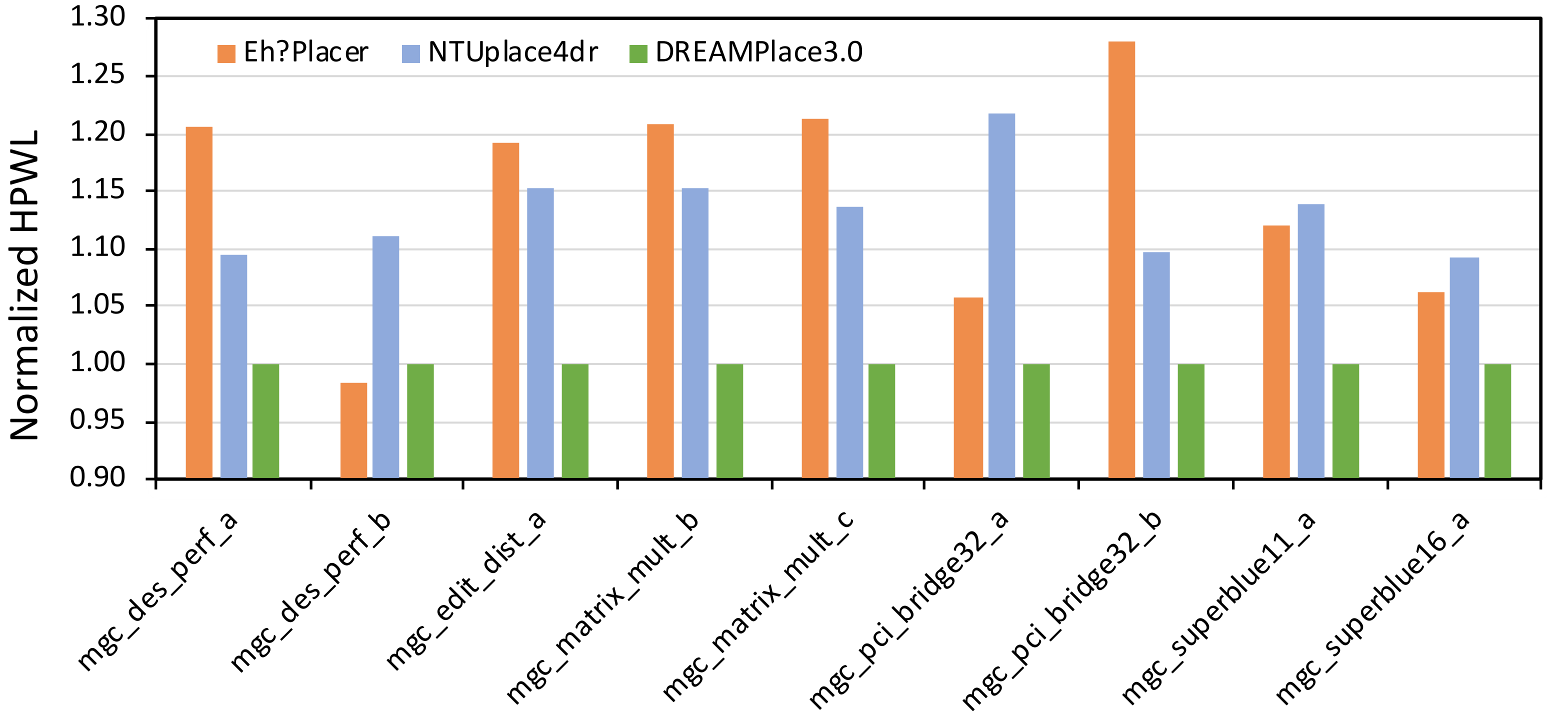


Experimental Setup

- Machine
  - Intel Core i9-7900X CPUs (3.3 GHz and 10 cores)
  - 128 GB RAM
  - NVIDIA TitanXp GPU
- Benchmark suits
  - ISPD 2015
  - ISPD 2019 (used as placement benchmarks)
  - ICCAD 2014
- Baseline
  - `DREAMPlace` [*Lin+, DAC 2019*] and `ABCDPlace` [*Lin+,TCAD 2020*]
- Placers for comparison
  - `NTUplace4dr` [*Huang+, TCAD 2018*]
  - `Eh?Placer` [*Darav+, TODAES 2016*]
  - `DREAMPlace` [*Lin+, DAC 2019*]

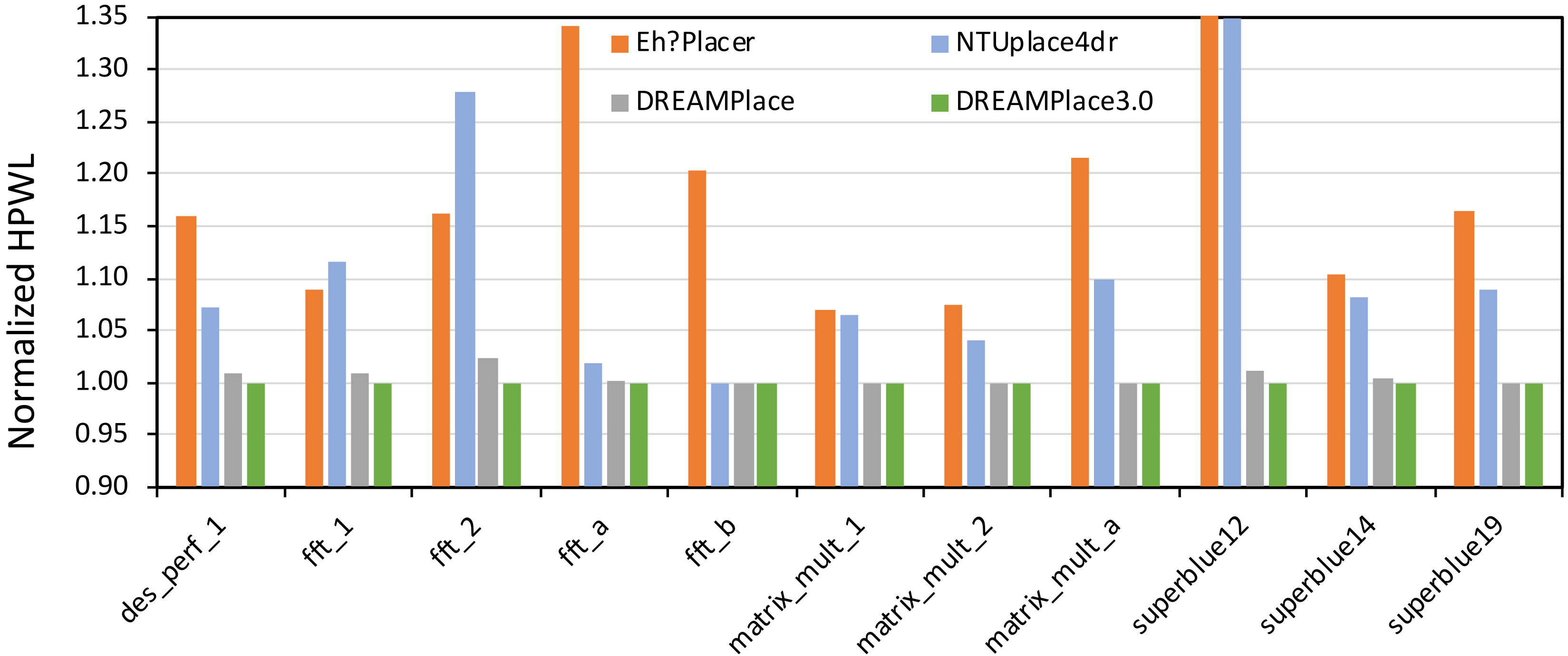
HPWL Comparison (w/ Region)

- DREAMPlace3.0 significantly outperforms other region-aware placers on ISPD15
  - 20.6% better than Eh?Placer
  - 13.3% better than NTUplace4dr



HPWL Comparison (w/o Region)

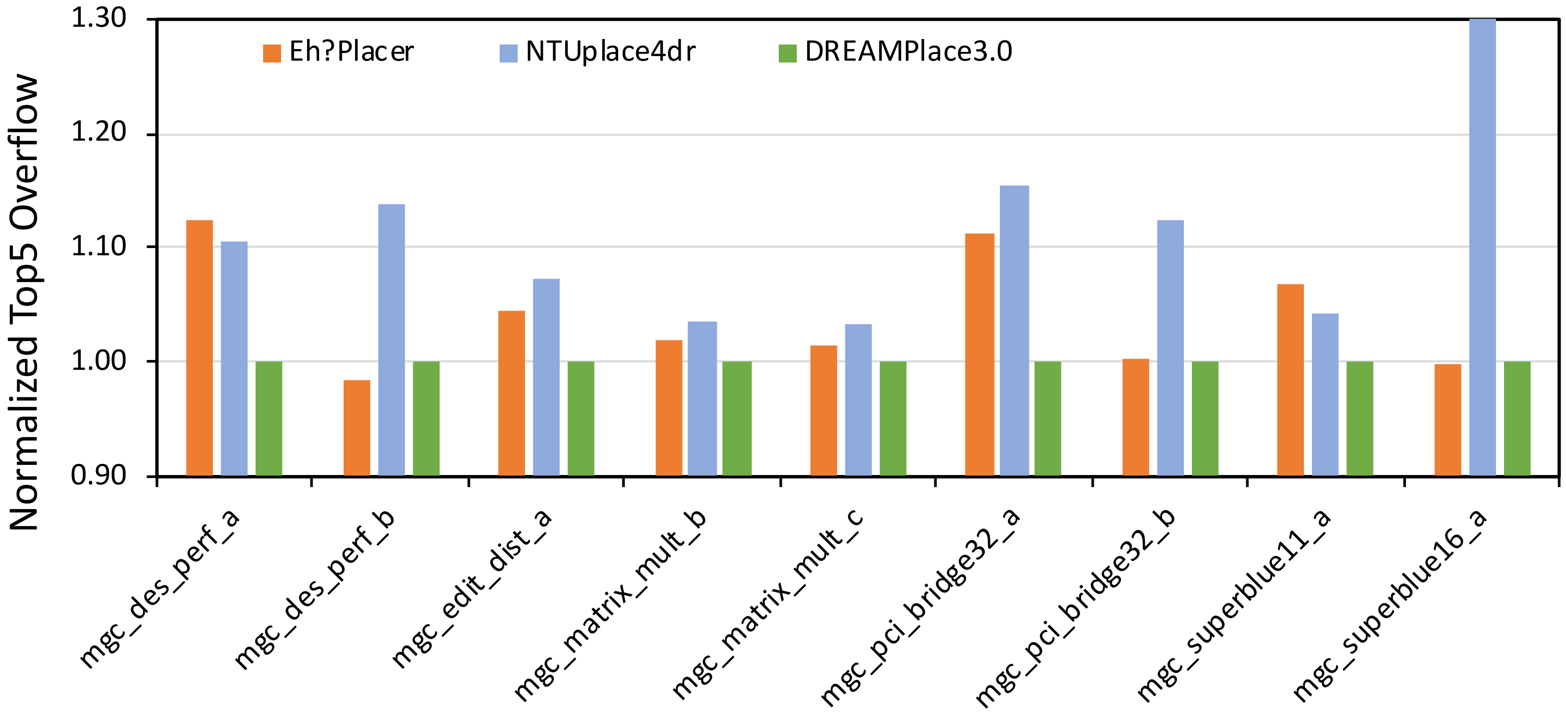
- DREAMPlace3.0 outperforms other placers on ISPD15
  - 17.0% better than Eh?Placer
  - 7.4% better than NTUplace4dr
  - 1.2% better than DREAMPlace



Top 5 OVFL Comparison (w/ Region)

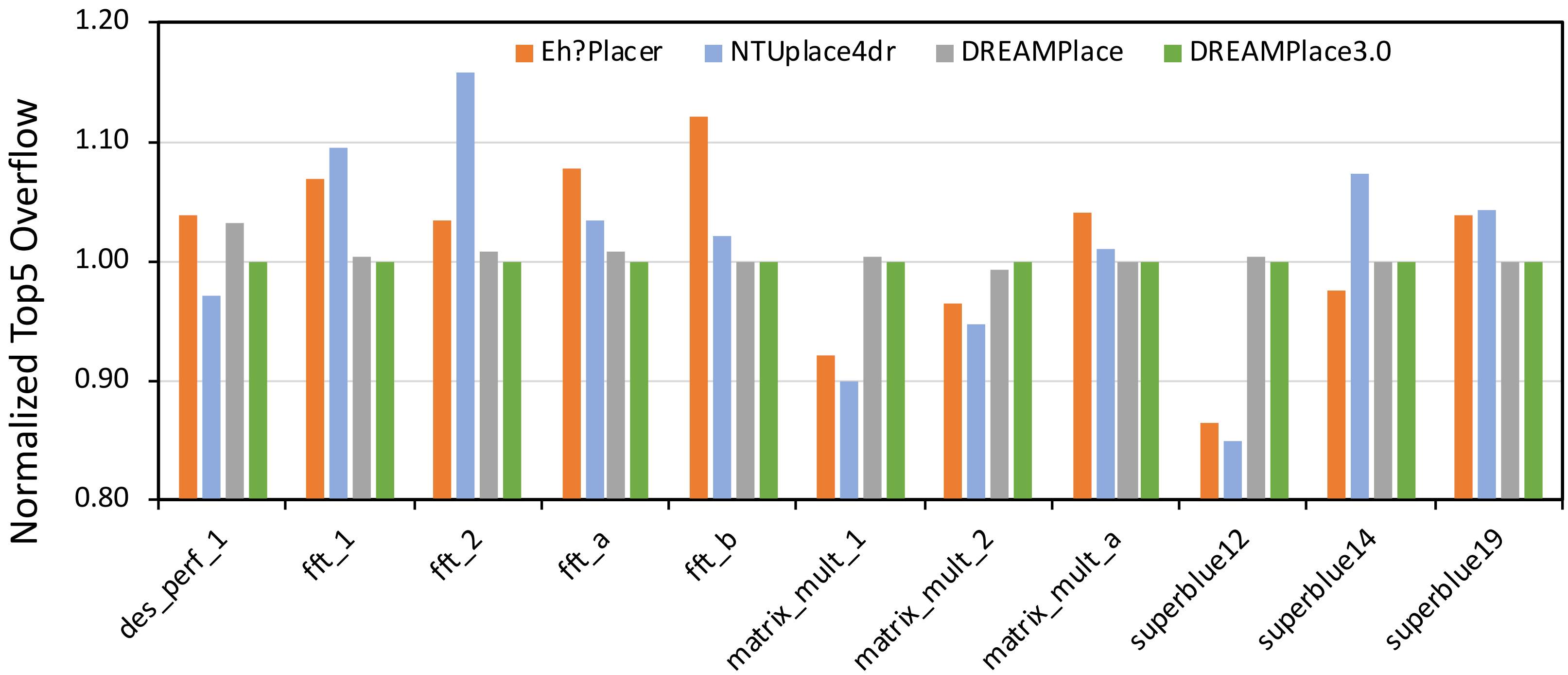
- DREAMPlace3.0 outperforms other region-aware placers on ISPD15
  - 12.4% better than Eh?Placer
  - 11.2% better than NTUplace4dr

*reported by NCTU-GR [*Dai+, TVLSI 2012*]



Top 5 OVFL Comparison (w/o Region)

- DREAMPlace3.0 outperforms other region-aware placers on ISPD15
  - 3.8% better than Eh?Placer
  - 3.3% better than DREAMPlace
  - 2.9% worse than NTUplace4dr



Runtime/Robustness Comparison

- On ISPD 2015 (w/ region), GPU-based `DREAMPlace 3.0` is
  - 3.7× faster than 8-threaded `Eh?Placer`
  - 34.8× faster than 8-threaded `NTUplace4dr`
- On ISPD 2015 (w/o region), GPU-based `DREAMPlace 3.0` is
  - 13.9× faster than 8-threaded `Eh?Placer`
  - 37.8× faster than 8-threaded `NTUplace4dr`
  - 1.9% faster than `DREAMPlace`
- On ISPD 2019 and ICCAD 2014, GPU-based `DREAMPlace 3.0` is
  - 10.8% faster than `DREAMPlace`
  - More stable in convergence with similar solution quality

Conclusion and Future Direction

- Conclusion
  - **Multi-electrostatics system**: handle fence region constraints with a *global view*
  - **Virtual blockage and field isolation**: parallel multi-region placement
  - **Adaptive quadratic penalty and entropy injection:** more stable convergence
  - **>13% better HPWL and 11% better overflow** than region-aware placers
  - **10% faster and more stable** than `DREAMPlace`

- Future direction
  - Honor more placement constraints
  - Other optimization algorithms
  - New acceleration methods in multi-field placement

Thank you!