`RUPlace`: Optimizing Routability via Unified Placement and Routing Formulation

Yifan Chen[1], Jing Mai[1], Zuodong Zhang[1,2], Yibo Lin[1,2,3]*
[1]School of Integrated Circuits, Peking University, Beijing, China
[2]Institute of Electronic Design Automation, Peking University, Wuxi, China
[3]Beijing Advanced Innovation Center for Integrated Circuits
Email: {chenyifan2019, jingmai, zuodongzhang, yibolin}@pku.edu.cn

***Abstract*—Placement plays a critical role in VLSI physical design, particularly in optimizing routability. With continuous advancements in semiconductor manufacturing technology, increased integration, and growing design complexity, managing routing congestion during placement has become increasingly challenging. Despite the widespread techniques to improve routability, these methods often lack theoretical guidance or sever the intrinsic connection between placement and routing optimization. In this paper, we propose `RUPlace`, an ADMM-based placer for unified optimization of placement and routing. Leveraging Wasserstein distance and bilevel optimization, our approach provides a unified framework for congestion optimization by alternately running global routing and incremental placement. Furthermore, we introduce a simple yet effective model for cell inflation-based global placement, where convex programming is employed to determine the optimal inflation ratio. Experimental results on a diverse set of open-source industrial benchmarks from `CircuitNet` and `Chipyard` demonstrate that our method achieves superior congestion reduction compared to widely used tools such as `OpenROAD`, `Xplace 2.0`, and `DREAMPlace 4.1`, while maintaining competitive wirelength and runtime.**

I. Introduction

Routability optimization in Placement plays a significant role in determining the overall performance and manufacturability of a chip. As semiconductor technology advances, increased complexity and high integration make congestion management more challenging. Routability placement usually consists of two components: routing congestion evaluation and congestion optimization.

Routing congestion evaluation is essential in routability-driven placement. Global routing [1], [2], [3], assigns temporary routing paths to estimate congestion, offering high accuracy. Statistical methods [4] predict congestion using probabilistic models, trading some accuracy for faster computation. Machine learning [5] approaches use data-driven models, though they may struggle with generalization.

For congestion optimization, there are three common approaches to addressing routing congestion: node inflation [1], [3], [6], [7], [8] based on tile congestion, where cell sizes are increased to spread them out; force-based placement [1], [8], ntuplace4h, which shifts cells to less congested areas; and machine learning-based methods [9], [10], which use data-driven models to predict and mitigate congestion.

However, node inflation methods often rely on heuristic approaches to adjust nodes based on congestion maps generated by global routing [1], [3] or statistical models [6], [7], [8]. These approaches depend heavily on human experience, lack theoretical modeling, and fail to capture the underlying connections between placement and routing outcomes. Force-based methods attempt to integrate routing considerations into placement modeling, such as introducing density constraints for route utilization [11] or incorporating congestion-weighted wirelength models [8]. However, to make such models tractable for optimization, they typically rely on simple statistical models to estimate routing outcomes, which limits their ability to embed precise routing predictions into placement. Machine learning-based methods [9], [10], on the other hand, aim to predict accurate routing results using neural networks and integrate them into placement optimization. However, these methods treat routing as a black box, disregarding the structural properties of the routing problem itself and lacking theoretical guidance.

To overcome these limitations, in this work, we propose `RUPlace`, a novel unified framework that seamlessly integrates mathematical models of both placement and routing. We formulate a unified placement and routing optimization problem that simultaneously minimizes congestion while maintaining placement quality. Rather than treating routing as a black box, this unified formulation explicitly models routing constraints and objectives alongside placement considerations. We enhance an alternating direction method of multipliers (ADMM) framework with Wasserstein distance metrics and bilevel optimization techniques to solve the problem. We further develop a principled approach to node inflation by reformulating traditional heuristic methods into a rigorous convex programming model, providing a mathematically sound and structured mechanism for congestion reduction. The key contributions of this work are summarized as follows:

- We propose a unified placement and routing formulation for routability optimization, which enables fine-grained congestion minimization.
- We propose an ADMM-based framework to solve the unified formulation leveraging Wasserstein distance and bilevel programming.
- We introduce a convex programming-based node inflation method, incorporating modularity-based clustering to automatically determine the relationship between the inflation ratio and tile congestion.

Experimental results on open-source industrial benchmarks demonstrate that our proposed algorithm can significantly reduce congestion in terms of both occurrence frequency and severity levels. Compared to the widely-used open-source EDA tool `OpenROAD` [12], our algorithm achieves $4.74\times$ ($3.47\times$) smaller horizontal (vertical) congestion with 7% better wirelength and a $3.67\times$ speedup in runtime, demonstrating that our proposed algorithm significantly improves routability while achieving better wirelength with reduced runtime.

The rest of the paper is organized as follows. Sec. II introduces the basic background and problem formulation; Sec. III explains the details of the proposed algorithm; Sec. IV validates the algorithm with experimental results; Sec. V concludes the paper.

II. Preliminaries

In this section, we provide an overview of key concepts which form the basis of our approach.

A. Analytical Placement

Analytical placement typically involves three stages: global placement (GP), legalization (LG), and detailed placement (DP). Global placement spreads instances across the layout, legalization resolves overlaps, and detailed placement refines the solution. Given the importance of GP to the overall placement quality, we focus on the GP stage. Global placement minimizes wirelength under density constraints, as in `ePlace` [13] and `NTUPlace` [11]:

$$\mathcal{BP} : \min_x \quad WL(x), \tag{1a}$$

$$\text{s.t.} \quad d_i(x) \le d_t, \forall i \in B, \tag{1b}$$

where x is the instance coordinate, $WL(\cdot)$ is a wirelength cost function, and $d(\cdot)$ is the density function, B is the set of bins, and d_t is the whitespace in each bin.

B. Wasserstein Distance

The Wasserstein distance is used to measure the distance between two probability distributions. It is particularly useful when distributions have minimal or no overlap. Let a and b be two discrete distributions defined on a 2-dimensional grid $M \times N$. The Wasserstein distance of order p between a and b is defined as:

$$W_p(a,b) = \left\{ \inf_{\pi \in \Pi(a,b)} \sum_{(x_a,x_b) \in M \times N} \|x_a - x_b\|^p \, d\pi(x_a,x_b) \right\}^{1/p}, \tag{2}$$

where $\Pi(a,b)$ represents the set of all possible transport plans π that match the distributions a and b, $\|x_a - x_b\|^p$ is the cost of transporting mass from point x_a to point x_b raised to the power of p. In other words, the Wasserstein distance represents the minimum cost required to transform one probability distribution into another through mass transportation.

C. Hypergraph Modularity

Hypergraph modularity quantifies the quality of partitioning or clustering in hypergraphs by evaluating how well the clusters are separated while maintaining strong internal connections [14]. Consider a hypergraph $G = (V, E)$, where V represents the set of nodes and E denotes the set of hyperedges. For a given clustering result A, the hypergraph modularity Q is defined as:

$$Q = \frac{1}{|E|} \left(EC - \sum_{d=2}^{D} E_d \sum_{A_i \in A} \frac{|\mathrm{Vol}(A_i)|}{|\mathrm{Vol}(V)|}^d \right), \tag{3}$$

where EC represents the number of nets contained within a single cluster, E_d denotes the set of hyperedges with degree d, D is the maximum net degree in G, and $|\mathrm{Vol}(V)|$ represents the sum of degrees of all nodes in set V.

III. Algorithm

In this section, we will further detail our algorithm.

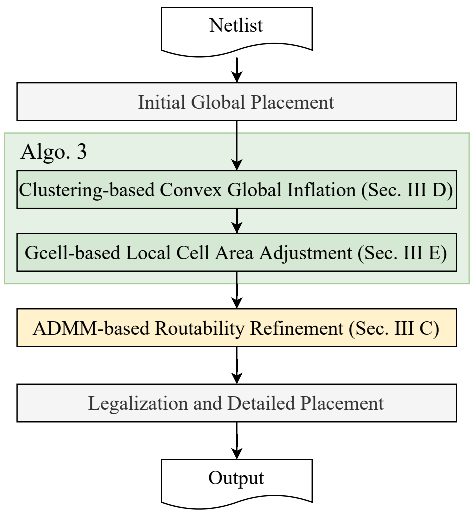
Netlist
Initial Global Placement
Algo. 3
Clustering-based Convex Global Inflation (Sec. III D)
Gcell-based Local Cell Area Adjustment (Sec. III E)
ADMM-based Routability Refinement (Sec. III C)
Legalization and Detailed Placement
Output

Fig. 1: The overall flow of our method.

A. Overview

The overall flow is shown in Fig. 1. The proposed algorithm initiates with a base global placement engine. When the density overflow is lower than a predefined threshold during GP, we perform cell inflation based on a convex optimization formulation for coarse-grained routability enhancement (see Sec. III-D). As the cells are inflated, the density overflow will increase again, we invoke the GP engine to perform incremental placement to spread cells. Once the density constraints are satisfied, we perform ADMM-based routability refinement for fine-grained improvement via a unified placement and routing formulation. The final stage involves legalization and detailed placement, which are performed using `DREAMPlace` [15].

B. Unified Placement and Routing Formulation

Given the placement solution x of Eq. (1), the routing problem can be formulated as an Integer Linear Programming (ILP) problem, denoted as problem $\mathcal{R}$ [16]:

$$\mathcal{R} : \min_f \quad \sum_{e \in E} \sum_{n \in N} c_e f_{n,e}, \tag{4a}$$

$$\text{s.t.} \quad \sum_{n} f_{n,e} \le cap_e, \forall e \in E, \tag{4b}$$

$$Af = h(x) \tag{4c}$$

$$f_{n,e} \in \{0,1\}, \forall n \in N, e \in E, \tag{4d}$$

In this formulation, E represents the routing grid edges, e is a specific routing edge, and N is the set of all nets. The variable cap_e represents the routing capacity of edge e, and the binary variable $f_{n,e}$ indicates whether net n routes through edge e ($f_{n,e} = 1$) or not ($f_{n,e} = 0$). The constraint $Af = h(x)$ represents the flow conservation constraints for the nets, where A is a coefficient matrix for the flow conservation constraints and $h(x)$ is defined as:

$$h(x)_{n,u} = \begin{cases} +1, & \text{if the source pin of net } n \text{ in gcell } u, \\ -1, & \text{if the sink pin of net } n \text{ in gcell } u, \\ 0, & \text{otherwise,} \end{cases} \tag{5}$$

As the solution f is likely to be infeasible due to congestion, we define a congestion function as the total violation of

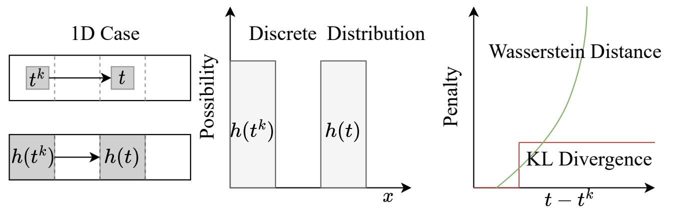

Fig. 2: Comparison of Wasserstein distance and KL divergencel. KL Divergence remains constant when two distributions have no overlap.

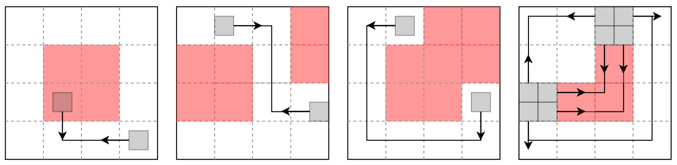

Fig. 3: Examples of how unified placement and routing resolve congestion under different scenarios. From left to right, the cases shown are the L shape net, Z shape net, 3-bend net, and long-wire congestion. Solid lines denote global routing paths. Arrows denote the intended directions of movement. The red area represents the congested region.

capacity constraints in Eq. (4b):

$$CONG(f) = \|\max(\sum_n f_{n,e} - cap_e, 0)\|. \quad (6)$$

Thus a final form of the Eq. (4) should be written as follows,

$$\mathcal{R}: \min_f \quad R(f) = \sum_{e\in E} c_e \sum_{n\in N} f_{n,e} + \mu CONG(f), \quad (7a)$$
$$\text{s.t.} \quad Af = h(x), \quad (7b)$$
$$f_{n,e} \in \{0,1\}, \forall n \in N, e \in E, \quad (7c)$$

where μ is a large weighting factor.

In routability-driven placement, we aim to optimize the post-routing result in the placement stage. Here we solve placement and routing concurrently so the objective function should follow Eq. (7a). Now we can define the concurrent placement and routing problem $\mathcal{CP}$:

$$\min_{x,f} R(f). \quad (8)$$

We expand problem $\mathcal{CP}$ considering the constraints in problem $\mathcal{BP}$ and problem $\mathcal{R}$ to obtain problem $\mathcal{UCP}$:

$$\mathcal{UCP}: \min_{x,f} \quad R(f), \quad (9a)$$
$$\text{s.t.} \quad d_i(x) \le d_t, \forall i \in B, \quad (9b)$$
$$Af = h(x), \quad (9c)$$
$$f_{n,e} \in \{0,1\}, \forall n \in N, e \in E, \quad (9d)$$

Eq. (9c) is the only constraint that couples x and f. This problem is the unified format of routability optimization for placement and routing.

C. ADMM-based Routability Refinement

Note that given x, the $\mathcal{UCP}$ is reduced to $\mathcal{R}$. Writing the optimal objective function value of $\mathcal{R}$ as $q(x)$, i.e., $q(x) = R(f^*(x); x)$, then adding $q(x)$ to the objective of $\mathcal{UCP}$ will not change the optimal solution of $\mathcal{UCP}$. Let $x = t$, $D = \{x | d_i(x) \le d_t\}$, $S = \{(f,t) | Af = h(t)\}$ and I_D, I_S be the indicator function of D, S, respectively, meaning $I_D(x) = 1$ if and only if $x \in D$, and $I_S(x) = 1$ if and only if $x \in S$. Then we can rewrite $\mathcal{UCP}$ to the following form,

$$\min_{x,f} \quad q(x) + I_D(x) + R(f) + I_S(f,t), \quad (10a)$$
$$\text{s.t.} \quad x = t, \quad (10b)$$
$$f_{n,e} \in \{0,1\}, \forall n \in N, e \in E, \quad (10c)$$

The augmented Lagrangian function is,

$$L(x,t,f,\lambda,\sigma) = q(x) + I_D(x) + R(f) + I_S(f,t) \quad (11a)$$
$$+ \lambda^\dagger(x-t) + \frac{\sigma}{2}(x-t)^2. \quad (11b)$$

We can leverage the alternating direction method of multipliers (ADMM) [17] to solve the problem,

$$t^{k+1}, f^{k+1} = \text{argmin}_{f,t} \quad L(x^k, t, f, \lambda^k, \sigma) \quad (12a)$$
$$x^{k+1} = \text{argmin}_x \quad L(x, t^{k+1}, f^{k+1}, \lambda^k, \sigma) \quad (12b)$$
$$\lambda^{k+1} = \lambda^k + \sigma(x^{k+1} - t^{k+1}) \quad (12c)$$

Approximating $q(x)$ by $WL(x)$ (note that $WL(x) \le q(x)$), we can expand Eq. (12b) to an analytical placement problem with new penalty terms,

$$\min_x \quad WL(x) + \lambda^{k\dagger}(x - t^{k+1}) + \frac{\sigma}{2}(x - t^{k+1})^2,$$
$$\text{s.t.} \quad d_i(x) \le d_t, \forall i \in B. \quad (13)$$

For problem Eq. (12a) its exact formulation is,

$$t^{k+1}, f^{k+1} = \text{argmin}_{Af=h(t)} R(f) + \lambda^{k\dagger}(x^k - t) + \frac{\sigma}{2}(x^k - t)^2 \quad (14)$$

which yields a problem in finding a new placement solution near x^k to minimize the routing objective. Given t, define the routing problem as $\mathcal{R}(t)$ and its solution as $\Psi(t) = \{f | f = \text{argmin}_{Af=h(t)} R(f)\}$. We can reformulate the Eq. (14) as following,

$$t^{k+1} = \text{argmin}_t \quad \mathcal{R}(t) + \lambda^{k\dagger}(x^k - t) + \frac{\sigma}{2}(x^k - t)^2 \quad (15)$$

which is a bilevel optimization problem [18]. Thus it can be decomposed into two subproblems,

$$f^{k+1} = \Psi(t^k), \quad (16a)$$
$$t^{k+1} = \text{argmin}_t \quad \hat{\mathcal{R}}(f^{k+1}, t) + \lambda^{k\dagger}(x^k - t) + \frac{\sigma}{2}(x^k - t)^2 \quad (16b)$$

where $\hat{\mathcal{R}}(f^{k+1}, t)$ serves as an linear approximation of $\mathcal{R}(t)$ in the Neighborhood of f^{k+1}. In the approximation function, we assume $h(t)$ is close to $h(t^k)$ and we only move t along the routed wires f^{k+1} so that the routed wires $\Psi(t)$ keeps the same topological structure as f^{k+1}. With this assumption, $\hat{\mathcal{R}}(f^{k+1}, t)$ is a linear function of t because if we move pins along the routed wire, the wirelength and congestion would be linearly reduced.

To ensure that $h(t)$ remains close to $h(t^k)$, a regularization penalty is required. It is important to note that a simple quadratic term, such as $(t - t^k)^2$, cannot be directly applied. This is because when certain cells are near the gridlines, even a small variation in t can lead to significant changes in $h(t)$. Since each item in $h(t)$ is in $\{-1, 0, 1\}$, we can view $h(t)$ as a distribution if we depart $h(t)$ to the positive part and negative part $h^+(t), h^-(t)$. However, certain distance metrics

Algorithm 1 ADMM-based Routability Refinement

```
1: Input: Initial position $x^0$
2: Output: Optimized position $x$
3: $\lambda \leftarrow 0$
4: while Not Converge do
5:     Set $t^k \leftarrow x^k$
6:     Solve global routing for $f^{k+1}$                ▷ Eq. (16a)
7:     Compute $t^{k+1}$ via gradient descent             ▷ Eq. (17)
8:     Solve analytical placement for $x^{k+1}$           ▷ Eq. (13)
9:     Update $\lambda^{k+1}$                              ▷ Eq. (12c)
10:    $k \leftarrow k+1$
   return Final $x^k$
```

between distributions, such as KL divergence, are not suitable in this context. As illustrated in Fig. 2, when two distributions have no overlap, the penalty remains unchanged, resulting in a meaningless guiding measure. Hence, we introduce a Wasserstein distance regularization penalty to penalize the distance between $h(t)$ and $h(t^k)$, that is,

$$t^{k+1} = \text{argmin}_t \quad \hat{\mathcal{R}}(f^{k+1}, t) + \eta W_2(h^+(t), h^+(t^k))^2 + \eta W_2(h^-(t), h^-(t^k))^2 + \lambda^{k\dagger}(x^k - t) + \frac{\sigma}{2}(x^k - t)^2 \quad (17)$$

We know for each net n, the positive part $h_n(x)$ (the supply of n), written as $h_n^+(x)$, is a discrete Dirac distribution $\delta(x - x_n, y - y_n)$ where the source pin is located at grid (x_n, y_n). So Wasserstein distance can be easily calculated,

$$W_2(h_n^+(t), h_n^+(t^k))^2 = (x_n - x_n^k)^2 + (y_n - y_n^k)^2 \quad (18)$$

which is a quadratic penalty term aware of the discrete structure of $h(\cdot)$. Thus Eq. (17) is an unconstrained quadratic programming. Instead of solving it directly, we perform a single optimization step using gradient descent. The overall flow is written in Algo. 1. The entire optimization process is intuitively illustrated in Fig. 3. It can be understood as moving cells along the routed wires, enabling the cells and nets to shift away from congested regions. This also enables the unified framework to mitigate global congestion caused by long wires.

Although a unified placement and routing framework can optimize routability in various scenarios, it faces challenges when an entire net resides within a congested region, as the router cannot bypass the congested area. Consequently, this method struggles to move cells out of the congested region. To address such situations, particularly those involving large congested areas, we further propose a cell inflation technique to mitigate local congestion.

D. Clustering-based Convex Global Inflation

In this section, we globally detect the clusters with high connectivity which is likely to cause highly congested area, then uniformly inflate the nodes within same module to achieve global inflation. It is important to note that this clustering result is only used for cell inflation purposes and is not utilized during the global placement process. After detecting clusters, we assign an inflation ratio to each cluster. Following the inflation, we apply incremental placement to spread the cells. The overall flow of the Clustering-Based Convex cell inflation process is outlined in Algo. 3 line 3∼8.

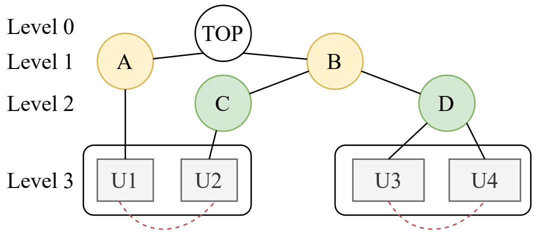

Fig. 4: Example of virtual clustering based on hypergraph modularity and logical hierarchy. We merge cells with high connectivity (red dashed lines) incrementally, level by level.

Algorithm 2 Modularity-Based Hierarchical Clustering

```
1: Input: Cell Position $x$, Hierarchy tree $T$, netlist $H$
2: Output: Clustering results
3: Construct initial clustering result $P$ using the leaf nodes of $T$.
4: Compute initial modularity $Q(P, H)$
5: for each level $l$ in $T$ do
6:     while nodes can be merged do
7:         for each leaf node $v$ in $T$ do
8:             $v$ belongs to module $m$ in level $l$
9:             for each leaf nodeu in the module $m$ do
10:                Compute $\Delta Q$ for merging $v$ and $u$ in $P$
11:                if $\Delta Q > 0$ and dist$(v, u) < 4$ row_height then
12:                    Merge node $v$ and $u$ in $P$ and $T$
13:                    Update modularity $Q(P, H)$
   return $P$
```

1) Vritual Clustering Based on Hypergraph Modularity and Logical Hierarchy

We first construct a logical hierarchy tree like that in [19], shown in Fig. 4. Note [19] does not consider routability optimization and we only follow the method of constructing the logical hierarchy tree. To detect modules with high connectivity, we then proceed to merge leaf nodes level by level, guided by the hypergraph modularity. At each level, leaf nodes within the same module are merged if the modularity gain is positive and the distance betwee them lower than 4 row_height. Every merge operation is executed to ensure an increase in the overall modularity. The complete algorithm is presented in Algo. 2.

2) Convex Global Inflation

Let x be the coordinates, let $dmd_e(x)$ represent the routing demand distribution contributed by a net e, and let $D_g(x)$ denote the routing demand distribution contributed by a cluster g. To ensure that the overall routing demand distribution remains unchanged, we assume the relationship between the two distributions is governed by the equation,

$$D_g(x) = \sum_{e \in E} \frac{p(e, g)}{|e|} dmd_e(x), \quad (19)$$

where $p(e, g)$ denotes the number of pins shared by net e and cluster g, and $|e|$ is the total number of pins in net e.

Introducing an inflation ratio l_g to cluster g leads to a modified routing demand distribution $D'_g(x)$. Assuming uniform inflation across the cluster, the relationship between the original distribution $D_g(x)$ and the new distribution $D'_g(x)$ is given by

$$D'_g(x) = \frac{1}{l_g} D_g(\frac{x}{l_g}). \quad (20)$$

Algorithm 3 Global Inflation and Local Area Adjustment

```
1: Input: Cell Positions, Congestion Map
2: Output: Cell Inflation Ratios
3: // Global inflation once
4: Modularity-based clustering                                   ▷ Algo. 2
5: Compute congestion distribution for each cluster              ▷ Eq. (19)
6: Compute intra-cluster wirelength
7: Solve Eq. (22) using the augmented Lagrangian method
8: Inflate cells and perform incremental placement
9: // Local area adjustment
10: while #inflate $< 6$ and congestion larger than 1% do
11:     Adjust cell area                                         ▷ Eq. (23b)
12:     Incremental placement
```

Here, the value of $D'_g(x)$ decreases, but the affected area expands. Assume that the routing demand contribution of gcell b after inflation originates from the same gcell b' before inflation. Then the new total routing demand $Demand'(b)$ for a gcell b after inflation as follows:

$$Demand'(b) = \sum_{g \in G} D'_g(b) \sim \sum_{g \in G} \frac{1}{l_g} D_g(b'), \tag{21}$$

To ensure that the routing demand does not exceed the available routing capacity, we can formulate the cell inflation problem as a convex optimization problem:

$$\mathcal{F}: \quad \min_{v} \quad \sum_{g} \frac{WL_g}{v_g}, \tag{22a}$$

$$\text{s.t.} \quad \sum_{g} D_g(b) \cdot v_g \le \text{cap}(b), \tag{22b}$$

where $v_g = \frac{1}{l_g}$, WL_g is the intra-cluster wirelength. The objective is to minimize the total wirelength and the constraints are designed to manage congestion by ensuring that the demand in each gcell does not exceed its capacity. Each subterm of the objective function WL_g/v_g is convex, and the constraints are linear, making this problem convex.

To solve this convex optimization problem, we apply the augmented Lagrangian algorithm, a well-established numerical method for handling constraints in optimization tasks.

E. Gcell-based Local Cell Area Adjustment

Convex global inflation assumes uniform scaling across clusters, which may mismatch routing demand and capacity. Some regions face residual congestion requiring inflation, while others have excess capacity needing shrinking. To address this, node areas are adjusted using the local demand $ldmd(b)$ and global demand $gdmd(b)$ for each gcell b. Here, $gdmd(b)$ represents the demand from nets crossing gcell b, while $ldmd(b)$ accounts for intra-gcell nets. If $ldmd(b) + gdmd(b) > cap(b)$, additional inflation is applied. Conversely, if $ldmd(b) + gdmd(b) < cap(b)$, the area is reduced.

For cells in gcell b, let $l_c(b)$ denote the inflation ratio produced by convex global inflation or the previous iteration of cell area adjustment, and let $\hat{l}_c(b)$ denote the cell inflation ratio for the current iteration of node area adjustment. The new inflation ratio can be computed as follows:

TABLE I: Benchmark Statistics [20], [21].

Design	#Macros	#Cell / K	#Nets / K	#Pins / K	Utilization
OPENC910	33	735	751	3029	0.50
NVDLA_S	45	113	123	442	0.40
NVDLA_L	376	1021	1098	3900	0.43
VORTEX_S	108	269	289	1051	0.42
VORTEX_L	80	1539	1695	6036	0.43
GEMMINI	737	926	981	3540	0.63
LARGEBOOM	636	737	756	2918	0.56

$$l'_c(b) = \max(1, \frac{ldmd(b)}{cap(b) - gdmd(b)}), \tag{23a}$$

$$\hat{l}_c(b) = (1-\gamma) l_c(b) + \gamma \min(\text{max\_inflate}, l'_c(b)), \tag{23b}$$

Where γ is a fixed weighting factor controlling the balance between the old and newly computed inflation ratio $l'_c(b)$. In experiments we set $\gamma = 0.2$. We iteratively perform node area adjustments until the congestion lower than 1% or reach max iteration. The overall flow of the gcell-Based cell area adjustment process is outlined in Algo. 3 line 10∼12.

IV. Experimental Results

A. Experimental Settings

We implement our proposed algorithms based on `DREAMPlace` [15] and use `HeLEM-GR` [23] as the global router for a full GPU acceleration flow. Our experiments were conducted on a 64-bit Linux workstation equipped with an Intel(R) Xeon(R) Platinum 8358 CPU @ 2.60GHz (32 cores) and an NVIDIA A800 GPU with 80GB memory.

We compared our proposed algorithm with three state-of-the-art placers: `OpenROAD` (CPU) [1] [12], `Xplace 2.0` (GPU) [2] [3], and `DREAMPlace 4.1` (GPU) [3] [22]. Table I presents a comprehensive overview of our diverse benchmark suite, comprising open-source industrial designs from `CircuitNet` [20] and `Chipyard` [21]. All designs were synthesized and implemented using an advanced commercial 14nm Process Design Kit (PDK). The benchmark suite features two notable designs: `NVDLA`, a sophisticated deep learning accelerator, and `VORTEX`, an advanced RISC-V based GPGPU architecture. Each design comes in two variants - large versions (`NVDLA_L`, `VORTEX_L`) optimized for maximum performance, and small versions (`NVDLA_S`, `VORTEX_S`) tailored for energy efficiency in resource-limited environments. The inclusion of these diverse implementations underscores the comprehensive nature and versatility of our benchmark suite.

We demonstrate the effectiveness and efficiency of our proposed algorithm by evaluating three critical aspects: horizontal/vertical congestion, routed wirelength, and runtime. The metrics are reported using the industry-standard commercial tool `Innovus` [24] and its `earlyGlobalRoute` command to obtain accurate measurements.

B. Results and Analysis

Table II presents the comparison results of our proposed algorithm with three state-of-the-art placers. We can see that our proposed algorithm consistently and significantly achieves better congestion metrics than other placers, i.e., reducing

[1]With routability-driven placement enabled.
[2]With cell inflation enabled.
[3]With RUDY-based routability optimization enabled.

TABLE II: Routed Wirelenth (rWL/um), Horizontal Congestion (C_H/%), Vertical Congestion (C_V/%), and Runtime (RT/min) comparison with state-of-the-art placers.

Design	OpenROAD [12]				Xplace 2.0 [3]				DREAMPlace 4.1 [22]				RUPlace			
	rWL	C_H	C_V	RT	rWL	C_H	C_V	RT	rWL	C_H	C_V	RT	rWL	C_H	C_V	RT
OPENC910	1.34e7	7.17	4.18	20.4	1.47e7	7.27	2.68	2.8	1.22e7	10.56	5.47	1.6	1.56e7	2.02	0.72	4.3
NVDLA_S	4.98e6	0.90	0.26	4.1	4.67e6	1.01	0.37	0.7	4.43e6	1.54	0.49	0.8	4.95e6	0.09	0.09	1.8
NVDLA_L	3.92e7	3.67	0.55	28.0	3.80e7	3.78	0.69	4.3	3.58e7	4.78	1.36	3.3	4.43e7	1.36	0.23	7.1
VORTEX_S	2.63e6	2.42	0.94	5.8	1.64e6	0.85	0.34	0.5	1.59e6	1.22	0.59	0.3	1.71e6	0.28	0.16	0.8
VORTEX_L	1.17e7	0.17	0.08	12.6	1.12e7	0.24	0.14	1.6	1.10e7	0.60	0.29	2.2	1.09e7	0.13	0.10	4.9
GEMMINI	1.68e7	2.56	1.78	10.7	9.38e6	0.10	0.21	1.1	9.04e6	0.08	0.10	2.0	1.04e7	0.01	0.01	4.6
LARGEBOOM	1.20e7	0.06	0.02	10.5	1.00e7	0.97	0.51	1.4	9.78e6	1.55	0.93	1.7	1.17e7	0.31	0.11	4.0
Geo. Mean	1.07	4.74	3.47	3.67	0.93	4.11	3.88	0.45	0.88	5.91	5.80	0.43	1.00	1.00	1.00	1.00

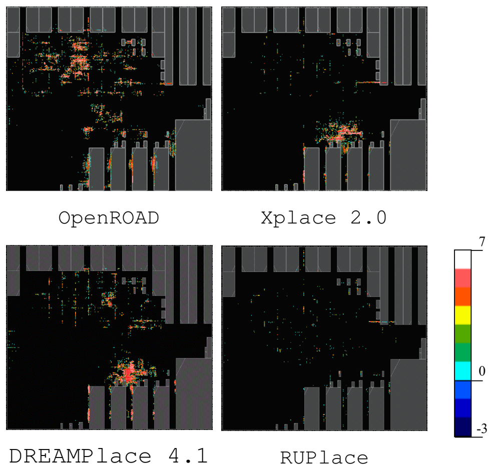

Fig. 5: Comparison of congestion distribution and severity across different placers on VORTEX_S. The colors indicate routing overflow, defined as the difference between routing demand and routing capacity. White regions represent severe congestion (demand exceeds capacity by more than 7), while blue regions indicate low congestion levels.

horizontal (vertical) congestion by 4.74× (3.47×) compared to OpenROAD, by 4.11× (3.88×) compared to Xplace 2.0, and by 5.91× (5.80×) compared to DREAMPlace 4.1. This demonstrates that our proposed algorithm can significantly reduce congestion across all test cases. Furthermore, compared to OpenROAD, our algorithm achieves 7% better routed wirelength, 4.74× smaller horizontal congestion, 3.47× smaller vertical congestion, and a 3.67× speedup in runtime, demonstrating that our proposed algorithm significantly improves routability while achieving better wirelength with reduced runtime.

Although our proposed algorithm achieves higher routed wirelength compared to Xplace 2.0 and DREAMPlace 4.1, the evaluation of placement quality requires considering both congestion and wirelength metrics. For cases with low congestion ($< 1\%$), further reduction does not provide significant improvement. However, for high congestion cases ($> 1\%$), lower congestion values are highly desirable. Our algorithm successfully maintains congestion values below 1% in almost all cases, except for hard cases like OPENC910 and NVDLA_L. Take the highly congeted case OPENC910 as an example. Our algorithm performs significantly better than the results of OpenROAD (7.17% and 4.18%), Xplace 2.0 (7.27% and 2.68%), and DREAMPlace 4.1 (10.56% and 5.47%). This demonstrates that our proposed algorithm can effectively reduce congestion even in highly congested designs. Additionally, our peak congestion values across all cases are significantly lower than those of other placers, demonstrating the practical viability of our approach in production workflows with acceptable runtime overhead.

C. Congestion Map Visualization

Fig. 5 visualizes congestion maps, highlighting differences across placers. We observe that OpenROAD exhibits more scattered overflow regions across the layout. In contrast, Xplace 2.0 and DREAMPlace 4.1 show more concentrated congestion areas with higher peak overflow values. Notably, Xplace 2.0 and DREAMPlace 4.1 share similar locations for their peak congestion regions, indicating these areas are inherently prone to congestion. The congestion map of our proposed algorithm demonstrates that it significantly alleviates congestion overall, achieving good results even in regions where previous placers struggled with severe congestion. This indicates that our algorithm effectively reduces both the occurrence and severity of congestion across the layout.

V. Conclusion

In this paper, we propose RUPlace, a unified framework for routability-driven placement in VLSI design, leveraging ADMM, Wasserstein distance, and bilevel optimization to jointly optimize placement and routing. Our approach provides a theoretically guided solution for congestion reduction, overcoming the limitations of traditional heuristic and probabilistic methods. Additionally, we introduced a convex programming-based cell inflation technique, incorporating modularity-based clustering to determine inflation ratios effectively, enhancing congestion mitigation beyond conventional heuristics. In experiments on CircuitNet and Chipyard benchmarks, our algorithm reduces horizontal (vertical) congestion by 4.74× (3.47×) compared to OpenROAD, 4.11× (3.88×) compared to Xplace 2.0, and 5.91× (5.80×) compared to DREAMPlace 4.1. It also achieves 7% lower wirelength and a 3.67× runtime speedup over OpenROAD, demonstrating its superiority in wirelength, efficiency, and routability optimization.

VI. Acknowledgment

This paper is supported in part by the National Natural Science Foundation of China (Grant No. T2293701), the Natural Science Foundation of Beijing, China (Grant No. Z230002), Grant QYJS-2023-2303-B, and the 111 project (B18001).

References

[1] C.-C. Huang, H.-Y. Lee, B.-Q. Lin, S.-W. Yang, C.-H. Chang, S.-T. Chen, Y.-W. Chang, T.-C. Chen, and I. Bustany, "Ntuplace4dr: A detailed-routing-driven placer for mixed-size circuit designs with technology and region constraints," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 3, pp. 669–681, 2018.

[2] C.-K. Cheng, A. B. Kahng, I. Kang, and L. Wang, "Replace: Advancing solution quality and routability validation in global placement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 9, pp. 1717–1730, 2019.

[3] L. Liu, B. Fu, S. Lin, J. Liu, E. F. Young, and M. D. Wong, "Xplace: An extremely fast and extensible placement framework," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2023. [Online]. Available: https://github.com/cuhk-eda/Xplace/releases/tag/v2.2.1

[4] P. Spindler and F. M. Johannes, "Fast and accurate routing demand estimation for efficient routability-driven placement," in *2007 Design, Automation Test in Europe Conference Exhibition*, 2007, pp. 1–6.

[5] Z. Xie, Y.-H. Huang, G.-Q. Fang, H. Ren, S.-Y. Fang, Y. Chen, and J. Hu, "Routenet: Routability prediction for mixed-size designs using convolutional neural network," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2018, pp. 1–8.

[6] T. Lin and C. Chu, "Polar 2.0: An effective routability-driven placer," in *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2014, pp. 1–6.

[7] X. He, Y. Wang, Y. Guo, and E. F. Y. Young, "Ripple 2.0: Improved movement of cells in routability-driven placement," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 22, no. 1, sep 2016. [Online]. Available: https://doi.org/10.1145/2925989

[8] J.-M. Lin, C.-W. Huang, L.-C. Zane, M.-C. Tsai, C.-L. Lin, and C.-F. Tsai, "Routability-driven global placer target on removing global and local congestion for vlsi designs," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2021, pp. 1–8.

[9] S. Liu, Q. Sun, P. Liao, Y. Lin, and B. Yu, "Global placement with deep learning-enabled explicit routability optimization," in *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2021, pp. 1821–1824.

[10] S. Park, D. Kim, S. Kwon, and S. Kang, "Routability prediction and optimization using explainable ai," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 2023, pp. 1–8.

[11] M.-K. Hsu, Y.-F. Chen, C.-C. Huang, S. Chou, T.-H. Lin, T.-C. Chen, and Y.-W. Chang, "Ntuplace4h: A novel routability-driven placement algorithm for hierarchical mixed-size circuit designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 12, pp. 1914–1927, 2014.

[12] T. Ajayi, D. Blaauw, T.-B. Chan, C.-K. Cheng, V. A. Chhabria, D. K. Choo, M. Coltella, R. G. Dreslinski, M. Fogac¸a, S. M. Hashemi, A. A. Ibrahim, A. B. Kahng, M. Kim, J. Li, Z. Liang, U. Mallappa, P. I. Pénzes, G. Pradipta, S. Reda, A. Rovinski, K. Samadi, S. S. Sapatnekar, L. K. Saul, C. Sechen, V. Srinivas, W. Swartz, D. Sylvester, D. Urquhart, L. Wang, M. Woo, and B. Xu, "Openroad: Toward a self-driving, open-source digital layout implementation tool chain," 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:210937106

[13] J. Lu, P. Chen, C.-C. Chang, L. Sha, D. J.-H. Huang, C.-C. Teng, and C.-K. Cheng, "eplace: Electrostatics-based placement using fast fourier transform and nesterov's method," *ACM TODAES*, vol. 20, no. 2, p. 17, 2015.

[14] M. E. J. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences*, vol. 103, no. 23, p. 8577–8582, Jun. 2006. [Online]. Available: http://dx.doi.org/10.1073/pnas.0601602103

[15] Y. Lin, S. Dhar, W. Li, H. Ren, B. Khailany, and D. Pan, "DREAMPlace: Deep learning toolkit-enabled gpu acceleration for modern vlsi placement," in *Proc. DAC*, 2019.

[16] P. Yao, P. Zhang, and W. Zhu, "Pathfinding model and lagrangian-based global routing," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*, 2023, pp. 1–6.

[17] D. P. Bertsekas, *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[18] Y. Beck and M. Schmidt, "A gentle and incomplete introduction to bilevel optimization," 2021.

[19] A. Kahng, S. Kang, S. Kundu, K. Min, S. Park, and B. Pramanik, "Ppa-relevant clustering-driven placement for large-scale vlsi designs," in *Proceedings of the 61st ACM/IEEE Design Automation Conference*, ser. DAC '24. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Available: https://doi.org/10.1145/3649329.3655991

[20] Z. Chai, Y. Zhao, W. Liu, Y. Lin, R. Wang, and R. Huang, "Circuitnet: An open-source dataset for machine learning in vlsi cad applications with improved domain-specific evaluation metric and learning strategies," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2023.

[21] A. Amid, D. Biancolin, A. Gonzalez, D. Grubb, S. Karandikar, H. Liew, A. Magyar, H. Mao, A. Ou, N. Pemberton, P. Rigge, C. Schmidt, J. Wright, J. Zhao, Y. S. Shao, K. Asanović, and B. Nikolić, "Chipyard: Integrated design, simulation, and implementation framework for custom socs," *IEEE Micro*, vol. 40, no. 4, pp. 10–21, 2020.

[22] Y. Chen, Z. Wen, Y. Liang, and Y. Lin, "Stronger mixed-size placement backbone considering second-order information," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 2023, pp. 1–9. [Online]. Available: https://github.com/limbo018/DREAMPlace/releases/tag/4.1.0

[23] C. Zhao, Z. Guo, R. Wang, Z. Wen, Y. Liang, and Y. Lin, "Helem-gr: Heterogeneous global routing with linearized exponential multiplier method," in *2024 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. ACM, 2024.

[24] *Cadence Innovus Implementation System*, Cadence Design Systems, Inc., 2023, available: https://www.cadence.com.