Stronger Mixed-Size Placement Backbone Considering Second-Order Information

Yifan Chen[1], Zaiwen Wen[2], Yun Liang[1,3,4], Yibo Lin[1,3,4*]
[1]School of Integrated Circuits, Peking University, Beijing, China
[2]Beijing International Center for Mathematical Research, Peking University, Beijing, China
[3]Institute of Electronic Design Automation, Peking University, Wuxi, China
[4]Beijing Advanced Innovation Center for Integrated Circuits
Email: {chenyifan2019, wenzw, ericlyun, yibolin}@pku.edu.cn

***Abstract*—Macro placement is a critical step in modern very large-scale Integration (VLSI) physical design. Placing macros with varying sizes significantly impacts the eventual quality of results. Many studies attempt to improve macro placement solutions leveraging existing analytical placement algorithms as the backbone. However, existing analytical placement algorithms may fail to converge for mixed-size designs if the parameters are not well-tuned. In this work, we propose a stronger mixed-size placement backbone with robust global placement convergence and macro legalization. Experimental results show that our method outperforms state-of-the-art works with better solution quality and fewer optimization iterations on various benchmarks including MMS, ISPD2005, and TILOS.**

I. Introduction

Macros are pre-designed and pre-verified building blocks in VLSI circuits. As macros can range from small arithmetic units to large memory blocks and even complex subsystems, their locations in a chip layout are crucial to the eventual design quality of results (QoR) [1]. Macro placement determines the locations of these macros in the layout. With the increasing complexity of VLSI designs, placing macros with heterogeneous sizes becomes more and more challenging due to their diverse shapes and high impacts on the locations of standard cells.

In the early years, macros are manually placed according to designers' experience. However, as designs get larger and more intricate, creating high-quality macro placement with manual efforts becomes unaffordable. The industry calls for high-performance techniques to place macros.

As the locations of macros have high impacts on standard cells, literature has explored mixed-size placement techniques to simultaneously optimize the locations of macros and cells. Typical mixed-size placement algorithms leverage mathematical optimization, such as mPL6 [2], NTUplace [3], ePlace-MS [4]. Figure 1 (left) shows a two-stage mixed-size placement flow adopted by the open-source TILOS project [1], where mixed-size placement is first performed as prototyping to determine the macro locations before standard cell placement.

Given the mixed-size placement as the initial prototyping, another line of studies focuses on developing efficient data structures to represent a macro placement, such as CG [5], MP-tree [6], CP-tree [7], and [8], [9]. With efficient representations, searching-based approaches can be adopted to legalize and incrementally refine the locations of macros based on heuristic objectives.

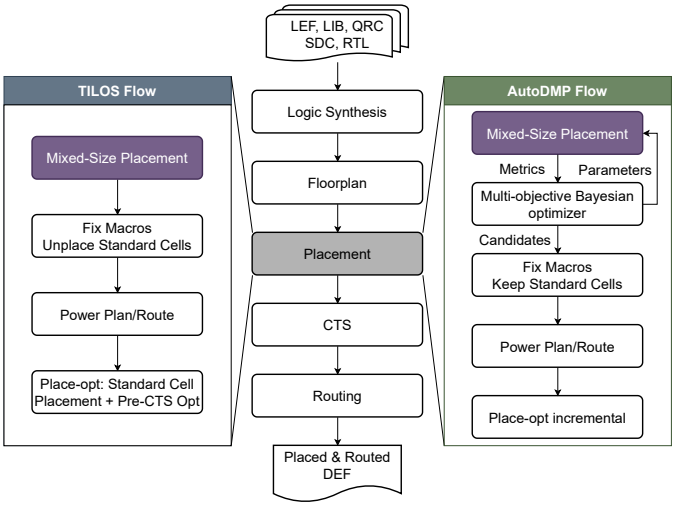


Fig. 1: A typical VLSI design flow. The left is TILOS flow, and the right one is AutoDMP flow. Both of them require a strong mixed-size placement backbone.

Recently, machine learning techniques and open-source datasets [10] have been incorporated into VLSI CAD tasks, including macro placement. Such as selection of macro placement solutions based on routability prediction [11], [12], graph learning based initialization for mixed-size placement [13], and reinforcement learning for macro refinement [14], [15], [16], [17]. A recent study from TILOS [1] has shown that the mixed-size placement prototyping significantly impacts the quality of solutions from reinforcement learning approaches. Besides previous techniques, AutoDMP based on multi-objective Bayesian optimization has achieved the state-of-the-art macro placement results that are comparable to commercial tools [18]. It essentially performs hyper-parameter searching upon a mixed-size placement engine DREAMPlace [19] to find the Pareto front of solutions considering congestion and wirelength, as shown in Figure 1 (right).

It can be seen that all the above-mentioned methods rely on mixed-size placement as their backbone, creating a growing demand for a stronger mixed-size placement backbone that

This work was supported in part by the National Science Foundation of China (Grant No. T2293700 and T2293701) and the 111 Project (B18001).

*Corresponding author

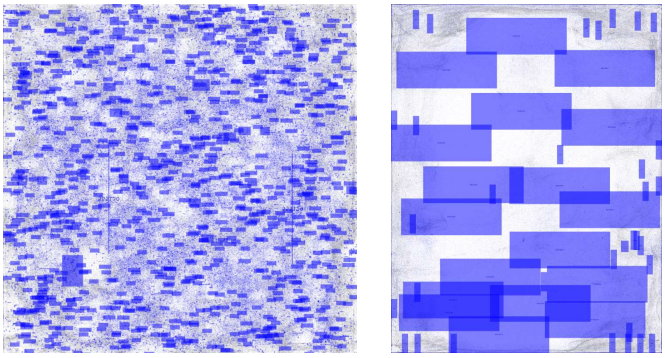


Fig. 2: Diverged examples from DREAMPlace. The two designs, I/O-freed `bigblue2` (left) and `newblue3` (right), come from the ISPD2005 and MMS benchmarks.

is robust and efficient to generate high-quality prototypes. However, macros with heterogeneous sizes pose challenges in optimization convergence. An ill-tuned placer may be prone to divergence and produce extremely low-quality solutions, as shown in Figure 2. The main reason is that the macros' motion will cause drastic changes in the gradient due to their heterogeneous sizes. As most mixed-size placement algorithms rely on first-order gradient information [2], [3], [4], [20], drastic gradient changes are more likely to cause instability and divergence. Therefore, we aim to address the convergence issue by considering second-order information.

In this work, we propose a stronger mixed-size placement backbone considering second-order information to enable robust and high-quality placement optimization. The major contributions of our work are summarized as follows:

- We propose a mixed-size placement backbone with Barzilai-Borwein method enabled optimization. This novel approach achieves robust convergence with fewer placement iterations and better quality, while incurring only minimal runtime overhead per iteration.
- We present a robust macro legalization algorithm combining multiple strategies that can achieve less displacement, contributing to higher-quality macro placement.
- The experimental results demonstrate that our proposed framework outperforms the widely-used mixed-size placement backbone, DREAMPlace [20], offering better solution quality and requiring fewer optimization iterations across a wide range of benchmarks. That is, 6.5%, 29.6%, and 33.3% lower wirelength than the default DREAMPlace on MMS [21], ISPD2005 [22], and TILOS [1] benchmarks. By integrating our backbone into AutoDMP, we can also improve the Pareto front curves on TILOS benchmarks.

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

II. Preliminaries

In this section, we review the background and the motivation.

A. Analytical Mixed-Size Placement

Analytical placement usually consists of three stages: global placement (GP), legalization (LG), and detailed placement (DP). Global placement spreads out instances in the layout; legalization removes the remaining overlaps between instances and aligns instances to placement sites; detailed placement performs incremental refinement to further improve the quality. Since the quality of the final placement solution largely depends on the global placement and legalization stage, we mainly focus on them in this work.

Global placement aims at minimizing the wirelength cost subjecting to density constraints. The density constraints are relaxed to a density penalty term computed as the potential energy of an electrostatic system where cells are modeled as charges in ePlace [23],

$$\min \sum_{e \in E} w_e WL(e; x, y) + \lambda D(x, y) \tag{1}$$

where E is the set of nets, (x, y) are the coordinates of all the instances, w_e is the weight of net e. $WL(\cdot;\cdot)$ is a differentiable wirelength cost function that takes net e and returns the wirelength, and $D(\cdot)$ is the density penalty to spread cells out in the layout. The density constraints can be satisfied by gradually increasing the weight λ. From an optimization point of view, the formulation can be extended to mixed-size placement as well. Thus, analytical mixed-size global placement is solving an unconstrained optimization problem with differentiable objective function.

B. Macro Legalization

In the global placement stage, the density penalty cannot ensure a non-overlapping solution. Therefore, we need to perform legalization to remove the remaining overlaps between instances with minimum displacement. Since macros are typically much larger than standard cells, the legalization stage is usually divided into macro legalization and standard cell legalization. In this work, we mainly focus on macro legalization, as the displacement of macros would largely affect standard cell placement. Towards minimizing absolute displacement, macro legalization can be formulated as flowing,

$$\begin{aligned} \min \quad & \|x - \hat{x}\|_1 + \|y - \hat{y}\|_1 \\ \text{s.t.} \quad & x_i + w_i \le x_j \vee x_i - w_j \ge x_j \vee y_i + h_i \le y_j \vee y_j - h_j \ge y_j \\ & W_l \le x_i \le W_h - w_i \\ & H_l \le y_i \le H_h - h_i \end{aligned} \tag{2}$$

where $(\hat{x}, \hat{y})$ are the coordinates before legalization, (w_i, h_i) is the width and height of instance i, W_l, W_h are the left and right boundaries of the layout, H_l, H_h are the bottom and top boundaries of the layout. This constraint implies that the instance i is either to the left of or to the right of or above or below the instance j.

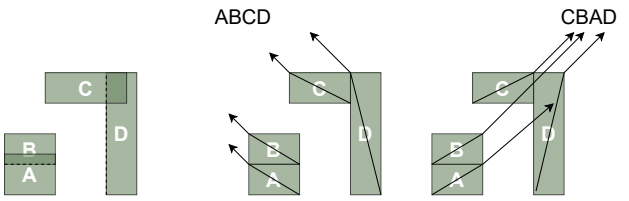


Fig. 3: Sequence pair example. The left is the layout with overlap, and we remove the overlap by cutting off the macros [24]. The right is the sweep line algorithm for sequence pair generation.

C. Sequence Pair

The sequence pair representation was introduced in [25], and many works have focused on efficiently generating a legal layout from a given sequence pair. A sequence pair $S(S^+, S^-)$ includes a pair of permutations of macros $\{1, 2, .., n\}$. Here is a sequence pair of 4 macros as shown in Figure 3,

$$\begin{aligned} S^+ &= \langle C, B, A, D \rangle \\ S^- &= \langle A, B, C, D \rangle \end{aligned} \tag{3}$$

Let $a \prec_x b$ denote a is to the left of b and $a \prec_y b$ denotes a is below b, the position of macro a in $S+$ is $S^+(a)$, the position of macro a in $S-$ is $S^-(a)$. Then, the relative relation of macros can be represented by sequence pair as following,

- $a \prec_x b \iff S^+(a) < S^+(b)$ and $S^-(a) < S^-(b)$.
- $b \prec_x a \iff S^+(a) > S^+(b)$ and $S^-(a) > S^-(b)$.
- $a \prec_y b \iff S^+(a) > S^+(b)$ and $S^-(a) < S^-(b)$.
- $b \prec_y a \iff S^+(a) < S^+(b)$ and $S^-(a) > S^-(b)$.

In macro legalization, converting a placement with overlaps to a sequence pair can be done in $O(n \log n)$ time using cutting-based overlap removal and sweep line algorithm [24]. Besides, the longest common sub-sequence algorithm [26] can check if a sequence pair is legal in $O(n \log \log n)$ time.

III. Algorithm

In this section, we will further detail our algorithm. There are two major parts: the Barzilai-Borwein method enabled Nesterov algorithm in Section III-B and proposed macro legalization algorithm in Section III-C.

A. Overview

Our mixed-size placement flow first uses Barzilai-Borwein method enabled Nesterov algorithm to solve the mixed-size analytical global placement problem. Then, we launch our macro legalization algorithm to remove the overlaps between macros towards minimum displacement. After the macros are legalized, we fix them and scale down the density penalty weight λ to restart global placement. In the restarted global placement, only the cells are movable and they start their motion from the current position instead of relaunching the whole standard cell global placement. The overall flow of our framework is shown in Figure 4.

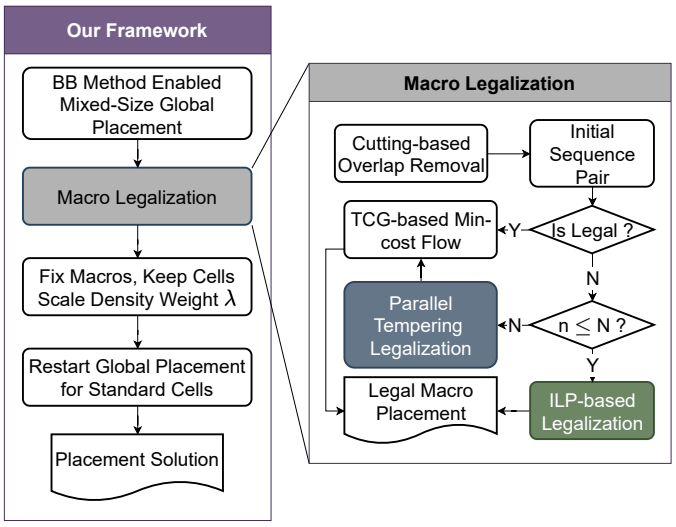


Fig. 4: The overall flow of our framework. It can be roughly viewed as a two-stage flow, including 1) mixed-size global placement and macro legalization, and 2) standard cell placement.

B. Barzilai-Borwein Method Enabled Nesterov Algorithm

In global placement, we have an unconstrained differentiable function $f(\cdot)$, and our target is to solve flowing optimization problem,

$$\min f(x) \tag{4}$$

Let $g^{(k)} = \nabla f(x^{(k)})$ and $H^{(k)} = \nabla^2 f(x^{(k)})$. There are two widely-used methods to solve such kind of problem, the gradient method and the Newton's method.

Gradient Method: $x^{(k+1)} = x^{(k)} - \alpha_k g^{(k)}$. The Gradient Method is a first-order optimization method that uses the gradient of the objective function to update the solution at each iteration. It is relatively simple and computationally efficient, but its convergence rate can be slow, particularly for non-convex problems. It can be a good choice for large-scale problems when the objective function is expensive to compute, as it only requires the computation of the gradient. But it can be sensitive to the choice of the step size.

Original DREAMPlace [20] uses the predicted Lipschitz step size accelerated Nesterov method as ePlace [23],

$$\alpha_{lip}^{(k)} = \frac{\|x^{(k)} - x^{(k-1)}\|}{\|g^{(k)} - g^{(k-1)}\|} \sim \frac{1}{L}. \tag{5}$$

where L is the Lipschitz constant. The predicted Lipschiz step size works well for standard cell placement, but it is still prone to divergence for mixed-size placement, even if the line search scheme is enabled, as shown in Figure 2. The main reason is that when the macros are also free to move, the gradient of the objective function will also change drastically as the macros move, which indicates that first-order information of the objective function is no longer enough for mixed-size placement.

Newton's Method: $x^{(k+1)} = x^{(k)} - (H^{(k)})^{-1} g^{(k)}$. Newton's Method is a second-order optimization method that uses the gradient and the Hessian of the objective function to update the solution at each iteration. It has a faster convergence rate than

the Gradient Method, particularly for convex problems. However, it requires the computation and storage of the Hessian matrix, which can be computationally expensive and memory-intensive for large-scale problems. Quasi-Newton methods like Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) [27] improve Newton's method by leveraging approximated Hessian matrix to reduce computation and storage overhead,

but they are still not suitable for mixed-size placement,

- The wirelength function (e.g. log-sum-exp or weighted-average) is iteratively changed due to the dynamically adjusted smoothing coefficient γ.
- The penalty factor λ on the energy (density) function is iteratively changed for the runtime force balancing between wirelength and density

Those issues lead to the need to recompute the gradients of previous iterations which takes a lot of time.

Thus, we propose to approximate the inverse of the Hessian matrix of the objective function based on Barzilai-Borwein method [28], using the difference between two consecutive gradients.

The Barzilai-Borwein method uses the following update rule to find the next solution approximation:

$$x^{(k+1)} = x^{(k)} - \alpha_{bb}^{(k)} g^{(k)} = x^{(k)} - ((\alpha_{bb}^{(k)})^{-1} I)^{-1} g^{(k)}. \quad (6)$$

The step size $\alpha_{bb}^{(k)}$ is calculated using the Barzilai-Borwein step, which approximates the inverse of the Hessian matrix using the gradient information. That is $(\alpha_{bb}^{(k)})^{-1} I \sim H^{(k)}$.

Let $s^{(k-1)} = x^{(k)} - x^{(k-1)}$ and $y^{(k-1)} = g^{(k)} - g^{(k-1)}$. Assuming $x^{(k)}$ is very close to $x^{(k-1)}$, the Hessian matrix approximately satisfies

$$H^{(k)} s^{(k-1)} = y^{(k-1)} \quad (7)$$

Therefore, we choose step size $\alpha_{bb}^{(k)}$ such that

$$(\alpha_{bb}^{(k)})^{-1} I s^{(k-1)} \sim y^{(k-1)} \quad (8)$$

or alternatively

$$s^{(k-1)} \sim \alpha_{bb}^{(k)} I y^{(k-1)} \quad (9)$$

This leads to a least-squares problem

$$\begin{aligned} (\alpha_{bb}^{(k)})^{-1} &= \arg\min_{\beta} \|s^{(k-1)}\beta - y^{(k-1)}\|^2 \\ \implies \alpha_{bb1}^{(k)} &= \frac{(s^{(k-1)})^T s^{(k-1)}}{(s^{(k-1)})^T y^{(k-1)}} \end{aligned} \quad (10)$$

or alternatively

$$\begin{aligned} \alpha_{bb}^{(k)} &= \arg\min_{\alpha} \|s^{(k-1)} - y^{(k-1)}\alpha\|^2 \\ \implies \alpha_{bb2}^{(k)} &= \frac{(s^{(k-1)})^T y^{(k-1)}}{(y^{(k-1)})^T y^{(k-1)}} \end{aligned} \quad (11)$$

The three kinds of step sizes have the following relation,

$$\alpha_{bb2}^{(k)} \leq \alpha_{lip}^{(k)} \leq \alpha_{bb1}^{(k)} \quad (12)$$

The convergence to the global optimal of BB method for quadratic problems has been shown by [29]. But using BB method alone cannot ensure convergence for the general case, because BB method cannot ensure a monotone decrease in the objective function. [30] has shown that pairing up BB with a non-monotone line search [31] can achieve global convergence.

***Theorem** 1:* Assume that $\Omega_0 = \{x : f(x) \leq f(x_0)\}$ is a bounded set. Let $f : R_n \to R$ be continuously differentiable in some neighborhood N of Ω_0. Let $\{x_k\}$ be the sequence generated by the BB method with non-monotone line search algorithm. Then either $g(x_j) = 0$ for some finite j, or the following properties hold:

- $\lim_{k\to\inf} \|g_k\| = 0$.
- no limit point of $\{x_k\}$ is a local maximum of f.
- if the number of stationary points of f in Ω_0 is finite, then the sequence $\{x_k\}$ converges.

In our framework, we choose the short BB step since in non-convex optimization the $(s^{(k-1)})^T y^{(k-1)}$ term can be zero and pair it with Zhang-Hager's line search [32]. Note that the short Barzilai-Borwein step can be negative when f is non-convex, which means the Hessian matrix is negative definite. Thus, we start the non-monotone line search from the predicted Lipschitz step size when the short Barzilai-Borwein step is non-positive. The algorithm is shown in Algorithm 1.

Algorithm 1 Barzilai-Borwein Method Enabled Nesterov Algorithm

Input: major solution $u^{(k)}$, reference solution $v^{(k)}$, optimization parameter a_k.
$g^{(k-1)} \leftarrow \nabla f(v^{(k-1)})$
$g^{(k)} \leftarrow \nabla f(v^{(k)})$
$s^{(k-1)} \leftarrow v^{(k)} - v^{(k-1)}$
$y^{(k-1)} \leftarrow g^{(k)} - g^{(k-1)}$
$\alpha_{bb} = \frac{(s^{(k-1)})^T y^{(k-1)}}{(y^{(k-1)})^T y^{(k-1)}}$
$\alpha_0 = \alpha_{bb}$ if $\alpha_{bb} > 0$ else $\frac{\|s^{(k-1)}\|}{\|y^{(k-1)}\|}$
$\alpha = \text{LineSearch}(v^{(k)}, g^{(k)}, \text{starts from } \alpha_0)$
$u^{(k+1)} = v^{(k)} - \alpha g^{(k)}$
$a_{k+1} = (1 + \sqrt{4a_k^2 + 1})/2$
$v^{(k+1)} = u^{(k)} + \frac{a_{k+1}-1}{a_{k+1}}(u^{(k+1)} - u^{(k)})$
return $u^{(k+1)}$, $v^{(k+1)}$, $ak + 1$

C. Macro Legalization

The macro legalization flow is illustrated in Figure 4 (right), which combines multiple strategies. We first use cutting based overlap removal to build the initial sequence pair [24]. If the initial sequence pair is feasible, we use transitive closure graph based min-cost flow algorithm to obtain macro positions. If the initial sequence pair is infeasible, we choose either ILP-based macro legalization or parallel tempering based macro legalization to find a feasible solution according to problem sizes.

1) Dual Min-Cost Flow based Macro Legalization for Feasible Initial Solutions: In the macro legalization stage, our target is to remove overlaps towards minimum displacement. A heuristic is to keep the order of macros, or the relative relation of macros in other words, like what most standard

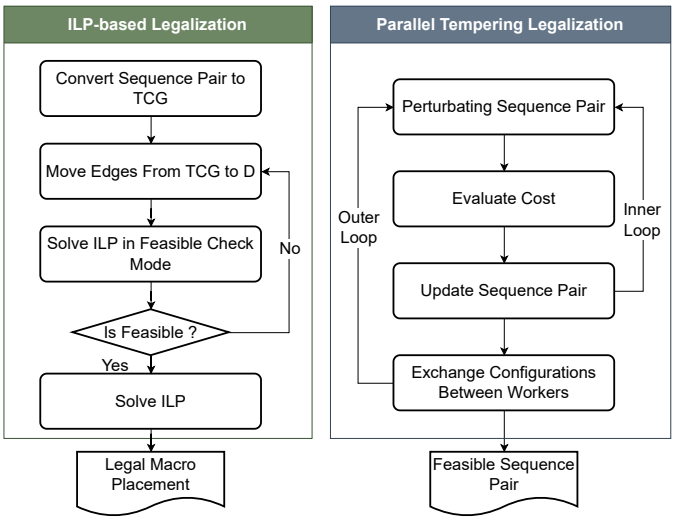


Fig. 5: Our ILP-based and parallel tempering macro legalization algorithm.

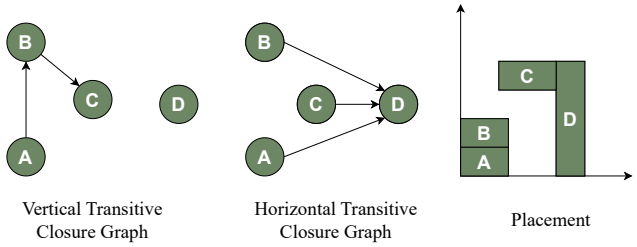


Fig. 6: An example of transitive closure graph of 4 macros.

cell legalization works do. Thus, we convert the placement with overlaps to an initial sequence pair which represents the relative relation between macros. Once the relative relation of macros is determined, the original formulation Eq. (2) would be reduced to a linear programming problem,

$$\begin{aligned} \min \ & \|x-\hat{x}\|_1+\|y-\hat{y}\|_1 \\ \text{s.t.} \ & x_i+w_i \le x_j, \forall i \prec_x j \\ & x_i-w_j \ge x_j, \forall j \prec_x i \\ & y_i+h_i \le y_j, \forall i \prec_y j \\ & y_j-h_j \ge y_j, \forall j \prec_y i \\ & W_l \le x_i \le W_h - w_i \\ & H_l \le y_i \le H_h - h_i \end{aligned} \tag{13}$$

It can be noticed that every pair of (i,j) needs to satisfy one of the four relative relations, which results in $\Theta(n^2)$ constraints. Solving such a problem by a general linear programming solver would take too much time. There are many redundant constraints. For example, if $i \prec_x j$ and $j \prec_x k$, there is no need to add a constraint for $i \prec_x k$. Hence, we perform transitive closure graph based constraints reduction. Some previous works [33], [34] use transitive closure graph (TCG) to represent the relation between macros without redundancy. They keep two directed transitive closure graphs G_x and G_y, and let $i \prec_x j \iff \exists$ path from i to $j \in G_x$, so as G_y. By the property of transitive closure graph, for every pair (i,j), there exist paths either in G_x or G_y that connect them. An example of 4 macros is shown in Figure 6.

Thus the number of final constraints can be reduced a lot. The formulation is,

$$\begin{aligned} \min \ & \|x-\hat{x}\|_1+\|y-\hat{y}\|_1 \\ \text{s.t.} \ & x_i+w_i \le x_j, \forall e_{ij} \in G_x \\ & y_i+h_i \le y_j, \forall e_{ij} \in G_y \\ & W_l \le x_i \le W_h - w_i \\ & H_l \le y_i \le H_h - h_i \end{aligned} \tag{14}$$

Then the problem is how to convert an initial sequence pair to transitive closure graph G_x and G_y. Take G_x as an example, the edge direction is always set from the macro on the left to the macro on the right. Therefore, we can visit the macros by the order in S^+ and add edge e_{ik} to G_x for all $i<k$ if $S^-(i)<S^-(k)$. To keep the G_x a transitive closure graph, before adding e_{ik} to G_x we need to ensure that there is no macro j such that $e_{ij} \in G_x$ and $j \prec_x k$. The process would take only $O(V(V+E))$ time, where V, E are the number of nodes and edges in G_x. By the way, obtaining transitive closure graph directly from the layout needs to use a depth-first search to check connectivity before adding an edge, which results in a much larger time cost. The generated transitive closure graph is equivalent to the original sequence pair, and if the sequence pair is legal then the linear programming Eq. (14) is feasible.

Since x, y are independent, they can be solved separately. Here we solve x first then y. Notice that, we can easily use dynamic programming to compute the possible range of a macro. Let $b_i^{xl}, b_i^{xh}, b_i^{yl}, b_i^{yh}$ be the lower bound, upper bound of macro i in the x-axis and the lower bound, upper bound of macro i in the y-axis. We have the following equation,

$$b_i^{xh} = \max\{\hat{x}_i, \max_{j \prec_x i}\{b_j^{xh}\}\} + w_i. \tag{15}$$

Other bounds can be obtained similarly. If there is no overlap between the possible range of two macros, we do not need to consider the constraint between them. So, the formulation when solving x is

$$\begin{aligned} \min \ & \|x-\hat{x}\|_1 \\ \text{s.t.} \ & x_i+w_i \le x_j, \forall e_{ij} \in G_x \text{ and } [b_i^{yl}, b_i^{yh}] \cap [b_j^{yl}, b_j^{yh}] \ne \emptyset \\ & W_l \le x_i \le W_h - w_i \end{aligned} \tag{16}$$

Besides, when x is fixed, the movement in y axes would not cause new overlaps in x axes. Thus, when solving y, we can only consider the constraints between macros overlapping in x axes.

$$\begin{aligned} \min \ & \|y-\hat{y}\|_1 \\ \text{s.t.} \ & y_i+h_i \le y_j, \forall e_{ij} \in G_y \wedge \text{i and j overlap in } x \text{ axes} \\ & H_l \le y_i \le H_h - h_i \end{aligned} \tag{17}$$

The dual problem of Eq. (16) and Eq. (17) are associated with the min-cost flow problem [35]. We solve the dual min-cost flow problem by network simplex algorithm, which can be much faster than general linear programming solvers like Gurobi [36]. The transitive closure graph based ordered macro legalization is very efficient and can legalize a design with more than 20k macros within 1 second.

However, not every initial sequence pair is feasible. We use

ILP-baed legalization and parallel tempering legalization to find a feasible sequence pair.

2) Integer linear programming (ILP) based Macro Legalization for Infeasible Initial Solutions. : If the initial sequence is not feasible, we need to change the relative relation between some macros. Let D be the set of macro pairs of which the relative relation needs to change. We remove the relations in D from transitive closure graph and let the remaining relations unchanged. Then, we get 'or' constraints as the original 'or' formulation Eq. (2) and 'or' constraints can be converted to integer programming by introducing binary variables. Let G'_x and G'_y be the remaining graph, Eq. (2) can be transformed to an integer linear programming formulation,

$$\begin{aligned} \min \ & \|x-\hat{x}\|_1 + \|y-\hat{y}\|_1 \\ \text{s.t.} \ & x_i + w_i \le x_j + W(x_{ij}+y_{ij}), \forall (i,j) \in D \\ & x_i - w_j \ge x_j - W(1+x_{ij}-y_{ij}), \forall (i,j) \in D \\ & y_i + h_i \le y_j + H(1-x_{ij}+y_{ij}), \forall (i,j) \in D \\ & y_j - h_j \ge y_j - H(2-x_{ij}-y_{ij}), \forall (i,j) \in D \\ & x_i + w_i \le x_j, \forall e_{ij} \in G'_x \\ & y_i + h_i \le y_j, \forall e_{ij} \in G'_y \\ & W_l \le x_i \le W_h - w_i, H_l \le y_i \le H_h - h_i \\ & x_{ij} \in \{0,1\}, y_{ij} \in \{0,1\} \end{aligned} \tag{18}$$

where W, H are two very large numbers such that only one of the 4 constraints for pairs in D would be active. Let gap_i^x and gap_i^y be the maximum movable range in x-axis and y-axis of macro i defined in previous work [34]. To achieve a feasible transitive closure graph, we need to ensure the gap is non-negative for any macro. If a macro has a low gap, it would be more likely to affect feasibility. Another heuristic is that there is no need to change the relative relationship between macros that are far away. Thus, we add the macro pairs that both have $gap \le \kappa$ and dist$(i,j) \le l$ to D and increase κ, l until the ILP is feasible. The flow of our ILP-based macro legalization is shown in Figure 5 (left). The experiments show that ILP-based legalization can handle mixed-size designs with hundreds of macros using modern ILP solvers in a few seconds. In this work, we choose the commercial solver Gurobi [36]. Any other solver can be used as an alternative. If the number of macros is too large to be solved by the ILP solver, we use parallel tempering on sequence pair to search for a feasible solution.

3) Parallel Tempering Macro Legalization for Infeasible Initial Solutions. : Parallel tempering is a population-based algorithm that employs multiple replicas or 'temperatures' in parallel. Each replica is associated with a different temperature, where higher temperatures correspond to a flatter distribution of states. In parallel tempering, replicas perform independent searches but periodically exchange configurations between adjacent temperatures. This exchange mechanism facilitates the escape from local minima and improving convergence towards the global minimum. The flow of our parallel tempering macro legalization is shown in Figure 5 (right).

We adopt the following operations to perturb a sequence pair

- Swap$^+$: swap two macros in S^+.
- Swap$^-$: swap two macros in S^-.
- Swap: swap two macros in both S^+ and S^-.
- Move$^+$: move one macro to the previous position of another macro in S^+.
- Move$^-$: move one macro to the previous position of another macro in S^-.
- Move: move one macro to the previous position of another macro in both S^+ and S^-.

For infeasible sequence pairs, we cannot calculate exactly displacement. Thus, we remove the right/top boundaries of the layout and greedily place each macro in topological order in the transitive closure graph (generated by sequence pair) at the position with minimal displacement. Let 'GreedyDisp' denote the obtained displacement via the greedy method, 'TNG' denotes the sum of the absolute value of the negative gap of all macros mentioned in Section III-C2, and 'Disp' denote the exact displacement by solving min-cost flow, the cost function is defined as

$$\text{Cost} = \begin{cases} \alpha\text{TNG} + \beta\text{GreedyDisp}, \text{if TNG} > 0 \\ \alpha\text{TNG} + \gamma\text{Disp}, \text{if TNG} \le 0 \end{cases} \tag{19}$$

where α, β, γ are three constants such that α is large enough to dominate the cost function and β, γ are chosen to narrow the gap between GreedyDisplacement and Displacement. In this work, we set $\alpha = n(W+H), \beta = \gamma = 1$ and where n is the number of macros and (W, H) is the size of the placeable area.

IV. Experimental Results

In this section, we present the details of the experiments and the analysis of the results.

A. Implementation and Benchmarks

We implement the framework on top of DREAMPlace, using PyTorch for the BB method enabled Nesterov optimizer and C++ for the macro legalizer.

To demonstrate the robustness and effectiveness of our approach to modern mixed-size designs, we use three benchmarks, MMS [21], ISPD2005 [37], and TILOS [1]. The MMS benchmark frees all the macros in ISPD2005/ISPD2006 and set the sizes of all I/O objects to zero. To construct more complex mixed-size designs, we modify the ISPD2005 benchmark such that all macros and I/O objects are freed to movable without changing their shape [13]. Besides, the I/O objects in ISPD2005 are treated as macros and are not allowed to overlap with other instances. Note that these designs with freed I/O are quite challenging due to their large number and variable shape. The TILOS benchmark [1] includes `Ariane` (a single-core RISC-V CPU), the `MemPool` group and `BlackParrot` designs (many-core RISC-V CPUs with large amounts of on-chip SRAMs), and an `NVDLA` partition. TILOS repository provides the netlist of `Ariane133` (ASAP7 [38]), `Ariane136` (ASAP7), `MemPool` (ASAP7), and `NVDLA` (NanGate45 [39]), so we just use them. But for `BlackParrot` we synthesize it by yosys [40] using open-source process design kit NanGate 45nm [39] since the netlist is not provided in TILOS repository recently.

TABLE I: Results on MMS Benchmarks.

Design	#Macros	Default DREAMPlace (Single-stage Flow)				Ours w/o BB Method (Two-stage Flow)				Ours w/ BB Method (Two-stage Flow)			
		Status	Iterations	HPWL	Runtime	Status	Iterations	HPWL	Runtime	Status	Iterations	HPWL	Runtime
adaptec1	63	Success	**607**	65.30	**17.79**	Success	832	65.24	29.58	Success	745	**64.63**	26.09
adaptec2	127	Success	**569**	79.29	**28.45**	Success	751	76.07	174.88	Success	762	**74.71**	43.88
adaptec3	58	Success	**659**	158.06	**44.58**	Success	946	**155.44**	83.01	Success	834	155.58	79.11
adaptec4	69	Success	**735**	**141.71**	**46.77**	Success	895	142.72	102.79	Success	783	142.48	85.94
adaptec5	76	Success	**1053**	326.30	**63.84**	Success	1424	307.35	103.63	Success	1300	**306.84**	92.71
bigblue1	32	Success	**646**	85.38	**21.26**	Success	877	**85.16**	39.29	Success	808	85.32	32.57
bigblue2	959	Success	**638**	125.35	**41.97**	Success	826	125.33	187.13	Success	775	**125.33**	164.33
bigblue3	2549	Success	**911**	279.33	**112.49**	Success	1205	**270.87**	181.51	Success	1095	273.84	194.47
bigblue4	199	Success	**1189**	648.84	**172.40**	Success	1606	643.55	233.31	Success	1513	**642.88**	282.51
newblue1	64	Success	**574**	62.82	**22.46**	Success	825	**58.61**	32.62	Success	743	59.33	34.45
newblue2	3748	Success	**730**	155.53	**34.82**	Success	948	**151.51**	83.24	Success	848	152.87	80.42
newblue3	51	Diverge, LG Failed*	1318	597.32	**55.71**	Diverge	1160	448.59	118.01	Success	**834**	**270.08**	92.84
newblue4	81	Success	**1009**	246.24	**52.61**	Success	1244	223.40	54.21	Success	1148	**223.24**	59.69
newblue5	91	Success	**1254**	444.20	**99.37**	Success	1444	**387.90**	132.79	Success	1344	388.98	160.38
newblue6	74	Success	**929**	410.61	**96.13**	Success	1234	**406.33**	148.31	Success	1145	406.82	168.85
newblue7	161	Success	**1077**	903.59	**184.07**	Success	1540	**879.37**	281.74	Success	1489	881.64	304.60
Ratio			**1.000**	1.000	**1.000**		1.298	0.953	2.135		1.186	**0.935**	1.784

* Default DREAMPlace fails to legalize the macros, so we make the macros fixed with overlapping and only perform cell legalization to evaluate the HPWL.

TABLE II: Results on I/O-freed ISPD2005 Benchmarks.

Design	#Macros	Default DREAMPlace (Single-stage Flow)				Ours w/o BB Method (Two-stage Flow)				Ours w/ BB Method (Two-stage Flow)			
		Status	Iterations	HPWL	Runtime	Status	Iterations	HPWL	Runtime	Status	Iterations	HPWL	Runtime
adaptec1	543	Success	**600**	101.27	**26.33**	Success	859	67.91	122.99	Success	749	**65.92**	127.88
adaptec2	566	LG Failed*	**588**	137.52	**40.63**	Success	824	79.74	145.01	Success	753	**77.65**	143.08
adaptec3	723	Success	**765**	179.51	**54.08**	Success	990	153.51	162.94	Success	898	**151.31**	173.44
adaptec4	1329	Success	876	153.27	**48.91**	Success	1200	213.86	86.86	Success	**873**	**141.45**	94.16
bigblue1	560	Success	**699**	86.18	**23.45**	Success	946	83.60	137.89	Success	810	**82.55**	132.94
bigblue2	23084	Diverge	1267	2426.69	679.43	Diverge	1314	273.28	151.88	Success	**918**	**99.92**	**149.06**
bigblue3	3778	Success	**1207**	330.15	**115.36**	Success	1432	301.70	161.26	Success	1302	**296.79**	197.04
bigblue4	8170	Success	**1581**	820.07	**239.64**	Success	2043	658.75	285.61	Success	1961	**620.00**	314.12
Ratio			**1.000**	1.000	**1.000**		1.295	0.787	2.715		1.112	**0.704**	2.802

* Default DREAMPlace fails to legalize the macros, so we make the macros fixed with overlapping and only perform cell legalization to evaluate the HPWL.

TABLE III: Results on TILOS Benchmarks.

Design	#Macros	Default DREAMPlace (Single-stage Flow)				Ours w/o BB Method (Two-stage Flow)				Ours w/ BB Method (Two-stage Flow)			
		Status	Iterations	HPWL	Runtime	Status	Iterations	HPWL	Runtime	Status	Iterations	HPWL	Runtime
Ariane133	133	Success	**573**	10.03	**11.26**	Success	790	9.76	28.90	Success	716	**9.36**	91.12
Ariane136	136	Success	**569**	13.80	**12.18**	Success	771	**12.81**	25.51	Success	700	12.94	28.29
MemPool	20	Success	**611**	10.92	**11.36**	Success	829	**10.67**	23.44	Success	779	10.68	26.36
BlackParrot	220	Success	**829**	342.59	**91.34**	Success	1092	**129.19**	123.82	Success	953	130.60	133.47
NVDLA	128	Diverge	**490**	326.83	**22.52**	Diverge	822	54.64	137.05	Success	742	**34.61**	60.25
Ratio			**1.000**	1.000	**1.000**		1.417	0.684	2.8331		1.283	**0.667**	3.3746

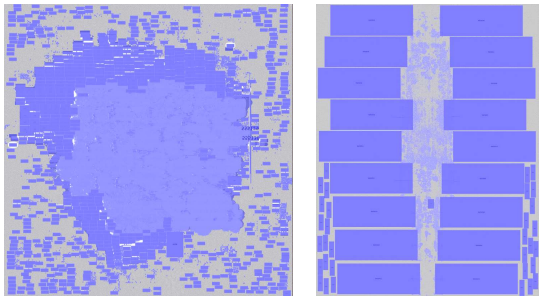
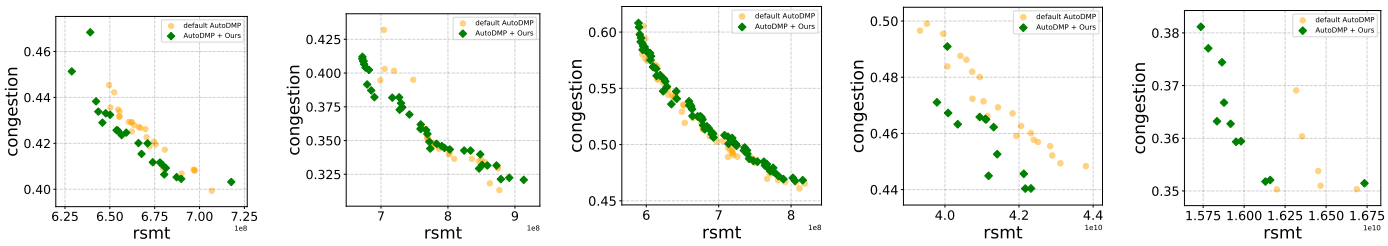


Fig. 7: The converged result of I/O freed bigblue2 in ISPD2005 and newblue3 in MMS.

B. Performance and Comparison

1) Comparison on MMS, ISPD2005 and TILOS: The experiment results of default DREAMPlace, ours w/o BB method, and ours w/ BB method tested on MMS, ISPD2005, and TILOS benchmarks are shown in Table I, Table II and Table III. We run experiments on a Linux machine with 1 NVIDIA GeForce RTX 2080, one 10 cores Intel(R) Xeon(R) Silver 4210R CPU @ 2.40GHz, and 62 GB of memory. In the tables, 'Diverge' refers to the situations where density overflow cannot be reduced, even if the placement iterations have reached stopping criteria. In such cases, the resulting HPWL is exceptionally poor.

In a comprehensive comparison between default DREAMPlace and our framework using the BB method, our method achieves a reduction in HPWL of 6.5%, 29.6%, and 33.3%. Besides, our framework effectively addresses legalization and



(a) Ariane133 (b) Ariane136 (c) MemPool (d) BlackParrot (e) NVDLA

Fig. 8: Congestion-RSMT Pareto fronts from default AutoDMP and AutoDMP + Ours.

divergence issues, while default DREAMPlace diverges on `newblue3` in Table I, `bigblue2` in Table II, and `NVDLA` in Table III. A comparison between diverged and converged results of `bigblue2` in I/O freed ISPD2005 and `newblue3` in MMS can be seen in Figure 2 and Figure 7. This demonstrates the effectiveness of our framework in improving the solution quality of mixed-size placement. Default DREAMPlace has fewer iterations because it adopts the single-stage flow, while our method adopts a two-stage flow. The second stage of our framework brings only a small iteration overhead. Additionally, it is worth noting that our framework successfully legalizes all cases regardless of whether BB method is enabled or not, which indicates that our legalization algorithm is reliable.

Comparing our framework w/o and w/ BB method, we find that the BB method effectively improves the algorithm's convergence. There is no divergence observed in any of the three sets of benchmarks, while our framework w/o BB method diverges on the same designs as default DREAMPlace. Furthermore, compared with the framework w/o the BB method, the framework w/ BB method reduces HPWL by 2.4%, 14.1%, and 7.7%, respectively. It also decreases the iteration count by 9.0%, 14.4%, and 9.4%, respectively, indicating that the BB method, which considers second-order information, not only enhances convergence stability but also accelerates the convergence rate. Additionally, the time overhead incurred by using the BB method is only 5.5% and 38.0% in ISPD2005 and TILOS, for the MMS benchmark the BB method decreases the runtime by 5.3%. It is worth mentioning that the majority of this time difference arises from macro legalization, while the time cost of the BB method itself used in global placement is similar to the original Nesterov optimizer in default DREAMPlace.

2) Integration into AutoDMP: To further demonstrate the stability of our framework, we integrate it into AutoDMP. Default AutoDMP has the option to choose between Adam optimizer and Nesterov optimizer at runtime. We replace the Nesterov optimizer with our BB method enabled Nesterov. We run AutoDMP on a Linux machine with 4 NVIDIA GeForce RTX 2080Ti, two 20 cores Intel(R) Xeon(R) Gold 6230 CPU @ 2.10GHz, and 500 GB of memory. Running AutoDMP for 400 iterations, we focus on three optimization objectives: RSMT, Congestion, and Density. The results are presented in the form of the final generated Congestion-RSMT Pareto front, illustrated in Figure 8.

Analyzing the Pareto curves, we observe significant improvements in three designs, namely `Ariane133`, `BlackParrot`, and `NVDLA`. Meanwhile, two designs `Ariane136` and `MemPool` have similar Pareto curves to default AutoDMP. Table IV, where a 'Bad Run' means that the optimizer diverges and the objective of the last iteration is INF, highlights the robustness of our BB method enabled Nesterov optimizer, as it consistently avoids producing unfavorable runs with the infinity objective. It is noteworthy that for design `BlackParrot`, the number of Nesterov increased a lot after BB method is enabled, which means that BB method makes Nesterov perform better than Adam on this design.

TABLE IV: The number of 'Bad Runs' of AutoDMP on TILOS benchmark. Here a 'Bad Run' means that the optimizer diverges and the objective of the last iteration is INF.

Design	Default AutoDMP			AutoDMP + Ours		
	#Nesterov Runs	#Nesterov Bad Runs	Runtime	#Nesterov Runs	#Nesterov Bad Runs	Runtime
`Ariane133`	354	22	1714	354	**0**	2102
`Ariane136`	360	40	1587	365	**0**	1788
`MemPool`	356	27	1130	353	**0**	1721
`BlackParrot`	51	29	4108	357	**0**	5035
`NVDLA`	56	29	2356	44	**0**	2892

V. Conclusion

In this work, we propose a stronger mixed-size placement backbone that considers second-order information, enabling robust macro placement. In the global placement stage, we propose Barzilai-Borwein method considering second-order information to achieve robust convergence with fewer iterations, resulting in improved efficiency while incurring only minimal runtime overhead per iteration. Besides, a robust macro legalization algorithm is introduced, addressing macro legalization issues with fewer displacements and resulting in higher-quality macro placement solutions. Extensive experimental results demonstrate the superiority of the proposed framework over the default DREAMPlace tool. Specifically, the proposed framework achieves HPWL reductions of 6.5%, 29.6%, and 33.3% compared to the default DREAMPlace on MMS, ISPD2005, and TILOS benchmarks, respectively.

REFERENCES

[1] C.-K. Cheng, A. B. Kahng, S. Kundu, Y. Wang, and Z. Wang, "Assessment of reinforcement learning for macro placement," in *Proceedings of the 2023 International Symposium on Physical Design*, ser. ISPD '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 158–166. [Online]. Available: https://doi.org/10.1145/3569052.3578926

[2] T. F. Chan, K. Sze, J. R. Shinnerl, and M. Xie, "Mpl6: Enhanced multilevel mixed-size placement with congestion control," in *Modern Circuit Placement*. Springer, 2007.

[3] 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 TCAD*, vol. 33, no. 12, pp. 1914–1927, 2014.

[4] J. Lu, H. Zhuang, P. Chen, H. Chang, C.-C. Chang, Y.-C. Wong, L. Sha, D. Huang, Y. Luo, C.-C. Teng *et al.*, "ePlace-MS: Electrostatics-based placement for mixed-size circuits," *IEEE TCAD*, vol. 34, no. 5, pp. 685–698, 2015.

[5] J. Cong and M. Xie, "A robust detailed placement for mixed-size ic designs," in *Asia and South Pacific Conference on Design Automation, 2006.*, 2006, pp. 7 pp.–.

[6] T.-C. Chen, P.-H. Yuh, Y.-W. Chang, F.-J. Huang, and T.-Y. Liu, "Mp-trees: A packing-based macro placement algorithm for modern mixed-size designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 9, pp. 1621–1634, 2008.

[7] Y.-F. Chen, C.-C. Huang, C.-H. Chiou, Y.-W. Chang, and C.-J. Wang, "Routability-driven blockage-aware macro placement," in *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2014, pp. 1–6.

[8] M. Moffitt, A. Ng, I. Markov, and M. Pollack, "Constraint-driven floorplan repair," in *2006 43rd ACM/IEEE Design Automation Conference*, 2006, pp. 1103–1108.

[9] C.-H. Chiou, C.-H. Chang, S.-T. Chen, and Y.-W. Chang, "Circular-contour-based obstacle-aware macro placement," in *ASP-DAC*. IEEE, 2016, pp. 172–177.

[10] Z. Chai, Y. Zhao, Y. Lin, W. Liu, R. Wang, and R. Huang, "Circuitnet: An open-source dataset for machine learning applications in electronic design automation (eda)," *SCIENCE CHINA Information Sciences*, vol. 65, no. 12, pp. 227 401–, 2022.

[11] Y.-H. Huang, Z. Xie, G.-Q. Fang, T.-C. Yu, H. Ren, S.-Y. Fang, Y. Chen, and J. Hu, "Routability-driven macro placement with embedded cnn-based prediction model," in *DATE*. IEEE, 2019, pp. 180–185.

[12] Y. Chen, J. Mai, X. Gao, M. Zhang, and Y. Lin, "Macrorank: Ranking macro placement solutions leveraging translation equivariancy," in *Proc. ASPDAC*, 2023, pp. 258–263.

[13] Y. Liu, Z. Ju, Z. Li, M. Dong, H. Zhou, J. Wang, F. Yang, X. Zeng, and L. Shang, "Graphplanner: Floorplanning with graph neural network," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 28, no. 2, dec 2022. [Online]. Available: https://doi.org/10.1145/3555804

[14] A. Mirhoseini, A. Goldie, M. Yazgan, J. W. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, A. Nazi *et al.*, "A graph placement methodology for fast chip design," *Nature*, vol. 594, no. 7862, pp. 207–212, 2021.

[15] R. Cheng and J. Yan, "On joint learning for solving placement and routing in chip design," *CoRR*, vol. abs/2111.00234, 2021. [Online]. Available: https://arxiv.org/abs/2111.00234

[16] Y. Lai, Y. Mu, and P. Luo, "Maskplace: Fast chip placement via reinforced visual representation learning," 2022.

[17] Q. Xu, H. Geng, S. Chen, B. Yuan, C. Zhuo, Y. Kang, and X. Wen, "Goodfloorplan: Graph convolutional network and reinforcement learning-based floorplanning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 10, pp. 3492–3502, 2022.

[18] A. Agnesina, P. Rajvanshi, T. Yang, G. Pradipta, A. Jiao, B. Keller, B. Khailany, and H. Ren, "Autodmp: Automated dreamplace-based macro placement," in *Proceedings of the 2023 International Symposium on Physical Design*, ser. ISPD '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 149–157. [Online]. Available: https://doi.org/10.1145/3569052.3578923

[19] 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.

[20] Y. Lin, S. Dhar, W. Li, H. Ren, B. Khailany, and D. Z. Pan, "Dreamplace: Deep learning toolkit-enabled gpu acceleration for modern vlsi placement," in *DAC*, 2019, pp. 1–6.

[21] J. Z. Yan, N. Viswanathan, and C. Chu, "Handling complexities in modern large-scale mixed-size placement," in *2009 46th ACM/IEEE Design Automation Conference*, 2009, pp. 436–441.

[22] G.-J. Nam, C. J. Alpert, P. Villarrubia, B. Winter, and M. Yildiz, "The ispd2005 placement contest and benchmark suite," in *Proceedings of the 2005 International Symposium on Physical Design*, ser. ISPD '05. New York, NY, USA: Association for Computing Machinery, 2005, p. 216–220. [Online]. Available: https://doi.org/10.1145/1055137.1055182

[23] 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.

[24] J. Egeblad, "Placement techniques for vlsi layout using sequence-pair legalization," *Master of Science Thesis, Department of Computer Science University of Copenhagen*, 2003.

[25] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "Vlsi module placement based on rectangle-packing by the sequence-pair," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 12, pp. 1518–1524, 1996.

[26] X. Tang, R. Tian, and D. Wong, "Fast evaluation of sequence pair in block placement by longest common subsequence computation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 12, pp. 1406–1413, 2001.

[27] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical programming*, vol. 45, no. 1-3, pp. 503–528, 1989.

[28] J. BARZILAI and J. M. BORWEIN, "Two-Point Step Size Gradient Methods," *IMA Journal of Numerical Analysis*, vol. 8, no. 1, pp. 141–148, 01 1988. [Online]. Available: https://doi.org/10.1093/imanum/8.1.141

[29] M. Raydan, "On the Barzilai and Borwein choice of steplength for the gradient method," *IMA Journal of Numerical Analysis*, vol. 13, no. 3, pp. 321–326, 07 1993. [Online]. Available: https://doi.org/10.1093/imanum/13.3.321

[30] ——, "The barzilai and borwein gradient method for the large scale unconstrained minimization problem," *SIAM Journal on Optimization*, vol. 7, 02 1997.

[31] L. Grippo, F. Lampariello, and S. Lucidi, "A nonmonotone line search technique for newton's method," *SIAM Journal on Numerical Analysis*, vol. 23, no. 4, pp. 707–716, 1986. [Online]. Available: http://www.jstor.org/stable/2157617

[32] H. Zhang and W. W. Hager, "A nonmonotone line search technique and its application to unconstrained optimization," *SIAM Journal on Optimization*, vol. 14, no. 4, pp. 1043–1056, 2004. [Online]. Available: https://doi.org/10.1137/S1052623403428208

[33] J.-M. Lin and Y.-W. Chang, "Tcg: a transitive closure graph-based representation for non-slicing floorplans," in *Proceedings of the 38th Design Automation Conference (IEEE Cat. No.01CH37232)*, 2001, pp. 764–769.

[34] H.-C. Chen, Y.-L. Chuang, Y.-W. Chang, and Y.-C. Chang, "Constraint graph-based macro placement for modern mixed-size circuit designs," in *2008 IEEE/ACM International Conference on Computer-Aided Design*, 2008, pp. 218–223.

[35] Y. Lin, B. Yu, X. Xu, J.-R. Gao, N. Viswanathan, W.-H. Liu, Z. Li, C. J. Alpert, and D. Z. Pan, "Mrdp: Multiple-row detailed placement of heterogeneous-sized cells for advanced nodes," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 6, pp. 1237–1250, 2018.

[36] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2023. [Online]. Available: https://www.gurobi.com

[37] G.-J. Nam, C. J. Alpert, P. Villarrubia, B. Winter, and M. Yildiz, "The ispd2005 placement contest and benchmark suite," in *Proc. ISPD*. ACM, 2005, pp. 216–220.

[38] L. T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha, B. Cline, C. Ramamurthy, and G. Yeric, "Asap7: A 7-nm finfet predictive process design kit," *Microelectronics Journal*, vol. 53, pp. 105–115, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S002626921630026X

[39] J. Knudsen, "Nangate 45nm open cell library," *CDNLive, EMEA*, 2008.

[40] C. Wolf, "Yosys open synthesis suite," https://yosyshq.net/yosys/.