DiffSP: Differentiable Sequence Pair-based Analog Placement

Abstract

Analog placement with compact representations is traditionally solved using heuristic or simulated annealing methods that are difficult to integrate with a differentiable optimization engine. This paper introduces DiffSP, a differentiable sequence-pair-based analog placement method that bridges discrete combinatorial representation with continuous gradient optimization. We derive a smooth relaxation of the sequence pair constraint graph via Gumbel–Sinkhorn relaxation, which allows area, wirelength, and symmetry objectives to be jointly optimized via automatic gradient calculation. A MILP-based legalization stage then enforces exact geometric and symmetry constraints. Experiments on industrial-level OTA benchmarks show that DiffSP achieves better placement quality and post-layout performance metrics than state-of-the-art analog placers with significantly reduced runtime.

1 Introduction

Analog circuit placement critically affects performance, yet achieving an effective layout requires substantial time and expertise. Unlike digital designs, analog circuits are susceptible to layout parasitics and coupling, which complicates the relationship between layout and performance. Traditionally, the analog placement problem is formulated similarly to the digital floorplan problem with additional geometric constraints. The most widely used type of constraints is the variations of symmetry constraint [1–3], which enforce certain cells to be placed symmetrically. Similar constraints on enforcing the placed locations of matched cells include common centroid [4, 5], regularity [6, 7], proximity[8–10], etc.

Analog placement has been an active research area for decades. From an algorithmic standpoint, stochastic methods are a classical paradigm for analog placement, such as the work of [1, 5, 10–13]. These approaches typically encode layouts using predefined data structures and optimize them with randomized algorithms, most commonly simulated annealing [14]. Simulated annealing relies on compact spatial representations such as slice trees [15], B* trees [16], and sequence pairs (SP) [17, 18], with sequence pairs being the most widely adopted. A key limitation of this paradigm is its lack of a direct mechanism to optimize design objectives [19]. Alternative formulations include mixed-integer linear programming (MILP) [13, 20, 21] and nonlinear programming (NLP) [22–24], which directly target metrics such as area and wirelength. MILP methods often struggle to scale due to their computational complexity. In contrast, NLP benefits from differentiable objectives and matrix-based solvers, enabling faster convergence and better scalability. However, NLP typically relies on relaxed geometric constraints, which can cause undesirable area expansion during optimization [24]. This issue is especially pronounced in NLP-based frameworks such as MAGICAL, where practical analog layouts must satisfy tight area budgets [25].

Despite substantial progress, current analog placement frameworks still struggle to reconcile compact layout representations with differentiable optimization. Stochastic methods (e.g., ALIGN) capture rich geometric relations in a compact representation of sequence pairs and thus preserve area efficiency [19]. However, their discrete

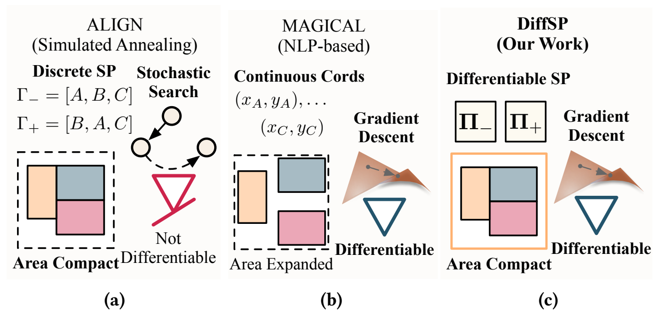


Figure 1: Comparison of analog global placers: (a) SA-based is compact but non-differentiable; (b) NLP-based is differentiable but area-expansive; (c) DiffSP makes SP differentiable via Gumbel–Sinkhorn while preserving compactness.

search hinders direct optimization of complex objectives, which is computationally inefficient with limited convergence guarantees. By contrast, NLP methods (e.g., MAGICAL) enable direct optimization with custom gradients and often exhibit favorable convergence in continuous spaces [3]. However, they typically achieve differentiability by relaxing geometric constraints and discarding compact representations, leading to excessive area overhead, weaker control of local geometric relations, and difficulty preserving both topology and layout quality simultaneously. Motivated by this gap, we pose the following question: ***Can we obtain the best of both worlds by extending the sequence-pair representation into a differentiable optimization framework?*** A resulting challenge is how to formulate discrete sequence pairs in a differentiable manner, enabling the optimization of objectives such as area, wirelength, and symmetry constraints.

To address this, we propose DiffSP, a differentiable relaxation of SP, as shown in Figure 1. DiffSP uses the Gumbel–Sinkhorn operator to map discrete permutations to doubly stochastic matrices, from which we build soft horizontal/vertical constraint graphs. A log-sum-exp–based longest-path computation then yields differentiable estimates of area and half-perimeter wire length (HPWL) objectives. DiffSP further incorporates smooth symmetry losses during training and enforces exact symmetry via a downstream MILP legalization step. In this way, DiffSP preserves SP's compactness while enabling scalable gradient-based optimization driven by design and performance constraints.

Our contributions are summarized as follows:

- We propose **DiffSP**, a differentiable analog placement framework that retains the classical sequence-pair representation while enabling end-to-end gradient-based continuous optimization.
- We relax SP by mapping permutations to doubly stochastic matrices via the Gumbel–Sinkhorn operator, from which we derive soft ranking masks and relaxed horizontal/vertical constraint graphs.
- We develop a differentiable objective that jointly optimizes area, wirelength, and symmetry via propagation on soft constraint graphs, and incorporate a MILP-based legalization step to satisfy symmetry constraints.

- On multiple experimental benchmarks, DiffSP achieves better placement quality and post-layout performance than state-of-the-art simulated-annealing and nonlinear placers.

2 Preliminaries

2.1 Analog Placement Representation

The *Sequence Pair* representation provides a compact representation for non-overlapping placement [26]. As shown in Figure 2, it captures spatial relations among modules using two ordered sequences of all module indices, denoted as Γ_+ and Γ_-. For any pair of modules (i, j), their relative placement is determined by their ordering in the two sequences:

- If i precedes j in both Γ_+ and Γ_-, module i is placed **to the left** of j;
- If i precedes j in Γ_+ but follows j in Γ_-, module i is placed **below** j.

The pairwise ordering relations in the SP define horizontal and vertical constraint graphs. Module coordinates are obtained by longest-path evaluation on these graphs or linear programming legalization, guaranteeing non-overlapping placement [11]. The SP representation thus offers a compact and expressive search space, where each permutation pair corresponds to a feasible geometric topology.

2.2 Analog Placement Constraints

Analog layouts impose additional geometric constraints (e.g., symmetry, common centroid, and proximity) to preserve device matching and reduce sensitivity to layout-dependent parasitics [4, 6, 7, 10]. Among these, the *symmetry constraint* is often the most critical, requiring device pairs or symmetric groups to be placed as mirror images with respect to a reference axis. Within the discrete SP representation, symmetry feasibility follows the condition described in [2, 10], which enforces symmetry ordering between symmetric modules in the two sequences.

For any pair of cells x, y with a horizontal symmetry relation, the sequence pair (Γ_+, Γ_-) is symmetric-feasible if:

$$\Gamma_+^{-1}(x) < \Gamma_+^{-1}(y) \iff \Gamma_-^{-1}(\mathrm{sym}(y)) < \Gamma_-^{-1}(\mathrm{sym}(x)), \quad \forall\, x \neq y. \tag{1}$$

Here, $\Gamma_+^{-1}(\cdot)$ and $\Gamma_-^{-1}(\cdot)$ denote the indices of a cell in Γ_+ and Γ_-, respectively. This horizontal stacking property ensures that any feasible perturbation, like local swaps satisfying the above conditions, will automatically preserve symmetry. During stochastic optimization, candidate SPs violating these symmetry relations are immediately rejected. A linear-programming-based legalization step is then applied to obtain the final, constraint-satisfying placement[2].

poe

2.3 Gumbel-Sinkhorn Relaxation

To enable gradient-based optimization over discrete permutations, we adopt the *Gumbel–Sinkhorn relaxation* [27]. This technique provides a differentiable approximation to permutation matrices, allowing end-to-end optimization through stochastic sampling. Formally, a random matrix $\mathbf{\Pi}$ on the space of permutation matrices is drawn from a Gumbel–Sinkhorn distribution parameterized by logits $\boldsymbol{\psi} \in \mathbb{R}^{n\times n}$ and temperature τ:

$$\mathbf{\Pi} \sim \text{Gumbel\_Sinkhorn}(\boldsymbol{\psi}, \tau). \tag{2}$$

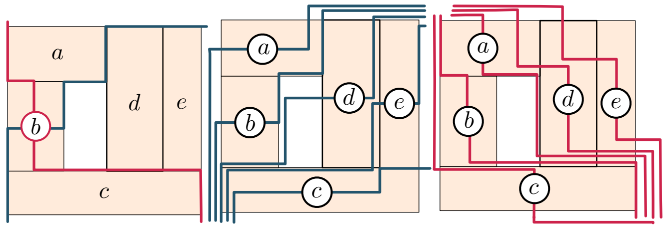


Figure 2: Sequence pair representation: (a) The positive step-line and negative step-line of module b; (b) Positive step-lines Γ_+: union of up-right and down-left step-lines; (c) Negative step-lines Γ_-: union of left-up and right-down step-lines.

A sample $\mathbf{\Pi}$ can be computed as:

$$\mathbf{\Pi} = S(\boldsymbol{\psi} + \mathbf{G}), \tag{3}$$

where $S(\cdot)$ is the *Sinkhorn operator*, which iteratively normalizes rows and columns to yield a doubly stochastic matrix, and $\mathbf{G}$ is a matrix of i.i.d. Gumbel noise. As the temperature $\tau \to 0$, the distribution converges to a discrete permutation distribution, commonly referred to as the *Gumbel–Matching* distribution. The discrete permutation obtained via the *Hungarian algorithm* [28] is used during the forward pass defined as

$$\hat{\mathbf{\Pi}} = \text{Hungarian}(\mathbf{\Pi}), \tag{4}$$

where the continuous $\mathbf{\Pi}$ is retained for gradient back-propagation. For differentiable sampling, a *straight-through estimator* [29] is often applied to estimate the gradient.

3 Problem Formulation

In the discrete SP representation, a placement is denoted by $\boldsymbol{s} = (\Gamma_+, \Gamma_-)$, and its cost is

$$\text{cost}(\boldsymbol{s}) = \text{area}(\boldsymbol{s}) + \alpha\, \text{HPWL}(\boldsymbol{s}) + \beta\, \text{SYM}(\boldsymbol{s}), \tag{5}$$

where area$(\cdot)$ is the total packing area, HPWL$(\cdot)$ estimates the total interconnect length, and α and β are weighting coefficients that balance these objectives. Feasibility requires $\boldsymbol{s}$ to satisfy all geometric constraints, including horizontal stacking property penalty for symmetry groups [10]. However, the combinatorial nature of $\boldsymbol{s}$ prevents gradient-based optimization.

To enable differentiability, **DiffSP** optimizes a paramerized distribution $p_\theta(\boldsymbol{s})$ over sequence pairs rather than a fixed discrete configuration:

$$\min_\theta\ \mathcal{L}(\theta) = \mathbb{E}_{\boldsymbol{s}\sim p_\theta}[\,\text{cost}(\boldsymbol{s})\,], \tag{6}$$

where sampling is achieved through the Gumbel–Sinkhorn relaxation that yields continuous doubly-stochastic matrices $(\mathbf{\Pi}_+, \mathbf{\Pi}_-)$ approximating permutations. Classical constraints are incorporated as differentiable penalties, allowing gradient-based learning of symmetry-preserving, area-efficient analog placement results.

4 Differentiable SP-based Analog Placement

4.1 Overview

DiffSP converts the conventional discrete sequence-pair model for analog placement into a differentiable and stochastic optimization formulation. Instead of exploring SP permutations via heuristic moves or simulated annealing, which are non-differentiable and often slow,

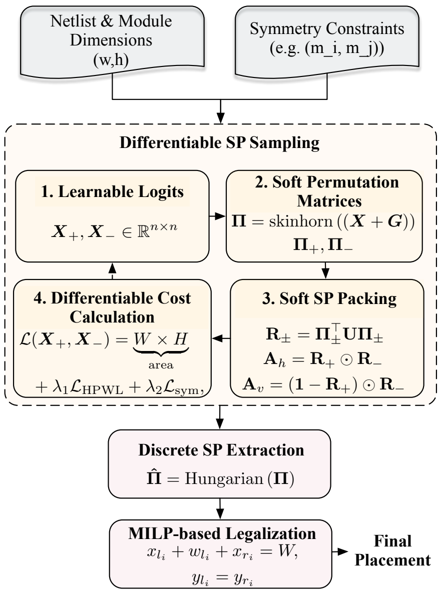

Netlist & Module Dimensions (w,h)
Symmetry Constraints (e.g. (m_i, m_j))
Differentiable SP Sampling
1. Learnable Logits
$X_+, X_- \in \mathbb{R}^{n\times n}$
2. Soft Permutation Matrices
$\Pi = \text{skinhorn}\left((X+G)\right)$
Π_+, Π_-
3. Soft SP Packing
$R_\pm = \Pi_\pm^\top U \Pi_\pm$
$A_h = R_+ \odot R_-$
$A_v = (1 - R_+) \odot R_-$
4. Differentiable Cost Calculation
$\mathcal{L}(X_+, X_-) = \underbrace{W \times H}_{\text{area}}$
$+ \lambda_1 \mathcal{L}_{\text{HPWL}} + \lambda_2 \mathcal{L}_{\text{sym}},$
Discrete SP Extraction
$\hat{\Pi} = \text{Hungarian}(\Pi)$
MILP-based Legalization
$x_{l_i} + w_{l_i} + x_{r_i} = W,$
$y_{l_i} = y_{r_i}$
Final Placement

Figure 3: The overall flow of our proposed method.

DiffSP learns a *continuous probabilistic distribution* over SPs. This enables direct gradient-based optimization of area, wirelength, and symmetry objectives.

As shown in Figure 3, the method operates in four conceptual stages in each loop: (1) differentiable SP sampling via Gumbel Sinkhorn relaxation; (2) construction of soft horizontal and vertical constraint graphs; (3) continuous estimation of layout area, HPWL, and symmetry objectives through relaxed shortest-path propagation; (4) gradient-based optimization of SP parameters aligning with analog placement-related constraints. A discrete SP is then extracted via the Hungarian algorithm, followed by MILP-based legalization to enforce exact geometric constraints and produce the final placement.

4.2 Sequence Pair Ranking Representation

A classical SP represents an analog placement by two permutations (Γ_+, Γ_-) over the module set. The relative position of any pair (a, b) is encoded by the order in these two sequences as horizontal constraints and vertical constraints.

Ranking-matrix Representation. Instead of working directly with the sequences, we encode each SP as a *ranking matrix*. For a discrete permutation Γ_+, its ranking matrix $\mathbf{R}_+ \in \{0,1\}^{n\times n}$ is defined by

$$(\mathbf{R}_+)_{ij} = \begin{cases} 1, & \text{if module } i \text{ precedes } j \text{ in } \Gamma_+, \\ 0, & \text{otherwise,} \end{cases} \tag{7}$$

and similarly for $\mathbf{R}_-$ from Γ_-. Thus, each row/column of $\mathbf{R}_\pm$ summarizes pairwise precedence relations induced by the SP. Because a sequence pair can be seen as a base permutation transformed by a single composite permutation, we can start from a base ranking matrix that is an upper-triangle matrix denoted as $\boldsymbol{U}$ and directly optimize the permutation matrices to obtain new SP representations.

Gumbel–Sinkhorn Relaxation of Permutation on SPs.. Direct optimization over discrete permutations of SPs is not differentiable. We therefore relax permutation matrices to *doubly stochastic* matrices. Following the Gumbel–Sinkhorn method [27], we sample soft permutation matrices $\mathbf{\Pi}_+, \mathbf{\Pi}_-$ as:

$$\mathbf{\Pi}_+ = \mathcal{S}\big((\mathbf{X}^+ + \boldsymbol{\varepsilon}^+)/\tau\big), \quad \mathbf{\Pi}_- = \mathcal{S}\big((\mathbf{X}^- + \boldsymbol{\varepsilon}^-)/\tau\big), \tag{8}$$

where $\mathbf{X}^+, \mathbf{X}^-$ are learnable logits, $\boldsymbol{\varepsilon}^\pm$ are i.i.d. Gumbel perturbations, and $\mathcal{S}(\cdot)$ is the Sinkhorn normalizer enforcing row/column stochasticity. As $\tau \to 0^+$, $\mathcal{S}(\mathbf{X}/\tau)$ converges to a discrete permutation matrix, yielding a reparameterizable, differentiable model of SP orderings.

Soft Ranking Matrices and Constraint Graphs. The soft permutation matrices induce *continuous* ranking matrices:

$$\mathbf{R}_+ = \mathbf{\Pi}_+^\top \mathbf{U}\, \mathbf{\Pi}_+, \tag{9}$$

$$\mathbf{R}_- = \mathbf{\Pi}_-^\top \mathbf{U}\, \mathbf{\Pi}_-, \tag{10}$$

where $\mathbf{U}$ is the fixed upper-triangular matrix encoding canonical precedence as a default initialization. In the discrete format, $(\mathbf{R}_\pm)_{ij} = 1$ if i precedes j and 0 otherwise; with soft permutations, these entries become values in $[0,1]$ that can be interpreted as probabilities or soft rankings.

We then construct horizontal and vertical adjacency matrices:

$$\mathbf{A}_h = \mathbf{R}_+ \odot \mathbf{R}_-, \quad \mathbf{A}_v = (\mathbf{1} - \mathbf{R}_+) \odot \mathbf{R}_-, \tag{11}$$

where $\odot$ is elementwise multiplication. Each entry of $\mathbf{A}_h$ (resp. $\mathbf{A}_v$) gives the soft likelihood that module i must precede j horizontally/vertically. These matrices generalize SP constraint graphs to a differentiable ranking matrix representation, enabling gradient-based optimization over analog placements.

4.3 Continuous Relaxation and Cost Calculation

The differentiable constraint graphs $\mathbf{A}_h$ and $\mathbf{A}_v$ play the same role as the *adjacency matrices* in the Cascade Algorithm [30]. Although the Cascade algorithm is primarily intended for shortest-path computation, on a DAG, its dynamic programming structure can be directly repurposed for the longest-path problem by substituting the min operator in the relaxation procedure with max. Thus, the longest-path distances in a DAG are propagated by repeated max–plus matrix products via the Cascade algorithm:

$$(\mathbf{C})_{ij} = \max_k (a_{ik} + b_{kj}). \tag{12}$$

To obtain gradients, DiffSP replaces the discrete max and + with their smooth counterparts based on the log-sum-exp operator LSE_τ:

$$(\mathbf{C})_{ij} = \tau \log \sum_k \exp\big((a_{ik} + b_{kj})/\tau\big), \tag{13}$$

which converges to the classic max–plus semiring as $\tau \to 0^+$. Thus, the forward and backward passes of Cascade are realized in a fully differentiable manner.

Differentiable Path Propagation. Given the soft precedence matrices $\mathbf{A}_h$ and $\mathbf{A}_v$, we first derive a topological order of the combined constraint graph. In the discrete setting, this would correspond to a standard topological sort of the horizontal/vertical constraint DAG; in our relaxed formulation, the same ordering is used as the scan order for dynamic programming.

Let $\mathbf{w}$ and $\mathbf{h}$ denote the vectors of module widths and heights. DiffSP propagates cumulative horizontal and vertical distances by a log-sum-exp dynamic program along the topologically sorted graph, which implements a softmax-type approximation of the max operator at each node:

$$\mathbf{d}_x^{(t+1)} = \tau \log\Big(\exp(\mathbf{d}_x^{(t)}/\tau) + \exp\big((\mathbf{A}_h \mathbf{d}_x^{(t)} + \mathbf{w})/\tau\big)\Big), \tag{14}$$

$$\mathbf{d}_y^{(t+1)} = \tau \log\Big(\exp(\mathbf{d}_y^{(t)}/\tau) + \exp\big((\mathbf{A}_v \mathbf{d}_y^{(t)} + \mathbf{h})/\tau\big)\Big), \tag{15}$$

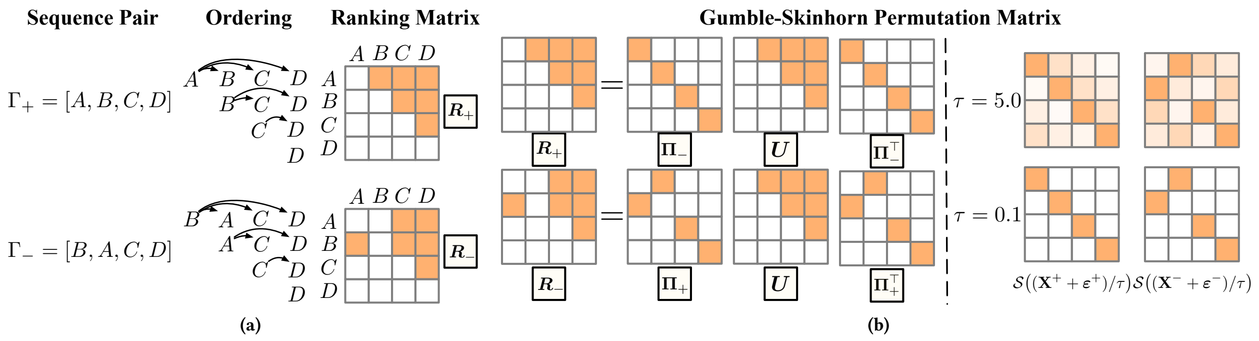

Sequence Pair
Ordering
Ranking Matrix
Gumble-Skinhorn Permutation Matrix
$\Gamma_+ = [A, B, C, D]$
$\Gamma_- = [B, A, C, D]$
R_+
R_-
Π_-
U
$\Pi_-^\top$
Π_+
$\Pi_+^\top$
$\tau = 5.0$
$\tau = 0.1$
$\mathcal{S}((\mathbf{X}^+ + \varepsilon^+)/\tau)\mathcal{S}((\mathbf{X}^- + \varepsilon^-)/\tau)$
(a)
(b)

Figure 4: From sequence pair to differentiable permutations: (a) The SP induces ordering and ranking matrices R^+ and R^-. (b) Gumbel–Sinkhorn converts logits to doubly stochastic permutation matrices, while lower temperature approaches discrete permutations.

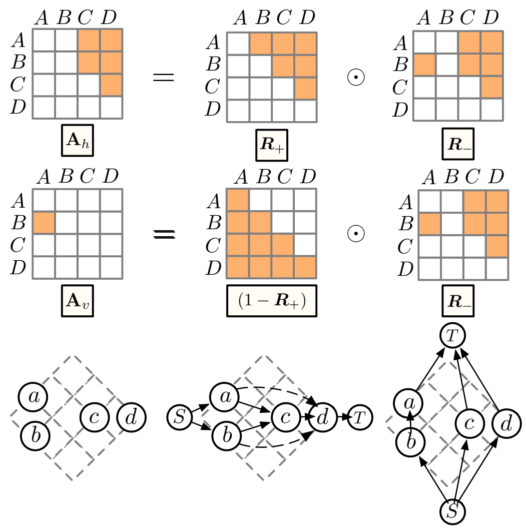


Figure 5: Soft constraint graphs from relaxed SP. Corresponding horizontal/vertical constraint graphs and their combined DAG for longest-path (packing) evaluation.

where $\mathbf{d}_x^{(t)}$ and $\mathbf{d}_y^{(t)}$ collect the current soft distances from a virtual source node to each module in the horizontal and vertical graphs, respectively. This iteration is performed for a fixed number of steps N_m, which equals the number of placed modules.

After the final iteration, instead of taking hard maxima over all modules, we use a global log-sum-exp to obtain softmax-type approximations of the overall layout width and height:

$$W = \tau \log \sum_i \exp\left((\mathbf{d}x)i/\tau\right),\ H \quad = \tau \log \sum_i \exp\left((\mathbf{d}y)i/\tau\right), \tag{16}$$

where $\mathbf{d}x$ and $\mathbf{d}y$ are the horizontal and vertical distance vectors at the last iteration. In the zero-temperature limit, the log-sum-exp operator converges to the hard max, and these quantities coincide with the source-to-target longest paths in the horizontal and vertical constraint DAGs.

Differentiable HPWL via Soft Penalty. The same relaxed distances also yield approximate module coordinates. Let x_i and y_i denote the soft center coordinates derived from $\mathbf{d}_x$ and $\mathbf{d}_y$ by subtracting half-widths and half-heights. For each net n with pin set $\mathcal{P}_n$, the half-perimeter wirelength is

$$\mathrm{HPWL}_n = (\max_{i\in\mathcal{P}_n} x_i - \min_{i\in\mathcal{P}_n} x_i) + (\max_{i\in\mathcal{P}_n} y_i - \min_{i\in\mathcal{P}_n} y_i).$$

To make this differentiable, DiffSP replaces the hard min/max with temperature-controlled log-sum-exp approximations:

$$\max_{i\in\mathcal{P}_n} x_i \approx \max^{\mathrm{soft}}_{i\in\mathcal{P}_n} x_i = \tau_x \log \sum_{i\in\mathcal{P}_n} \exp(x_i/\tau_x), \tag{17}$$

$$\min_{i\in\mathcal{P}_n} x_i = \approx \min^{\mathrm{soft}}_{i\in\mathcal{P}_n} x_i = -\tau_x \log \sum_{i\in\mathcal{P}_n} \exp(-x_i/\tau_x), \tag{18}$$

and similarly for y_i. The resulting soft HPWL for net n is:

$$\mathrm{HPWL}_n^{(\mathrm{soft})} = \left(\max^{\mathrm{soft}}_{i\in\mathcal{P}_n} x_i - \min^{\mathrm{soft}}_{i\in\mathcal{P}_n} x_i\right) + \left(\max^{\mathrm{soft}}_{i\in\mathcal{P}_n} y_i - \min^{\mathrm{soft}}_{i\in\mathcal{P}_n} y_i\right). \tag{19}$$

The total HPWL loss is then defined as:

$$\mathcal{L}_{\mathrm{HPWL}} = \sum_n \mathrm{HPWL}_n^{(\mathrm{soft})}. \tag{20}$$

Geometrically, this corresponds to computing the distance between the soft leftmost and rightmost pins (and bottommost/topmost pins) of each net.

Differentiable Cost Formulation. The overall optimization objective is

$$\mathcal{L} = \underbrace{W \times H}_{\text{area}} + \lambda_1 \mathcal{L}_{\mathrm{HPWL}} + \lambda_2 \mathcal{L}_{\mathrm{sym}}, \tag{21}$$

where $\mathcal{L}_{\mathrm{sym}}$ penalizes deviations from symmetry constraints as in Section 4.4. Every term is differentiable with respect to the permutation logits $\mathbf{X}^+$ and $\mathbf{X}^-$ through the soft permutation matrices, ranking matrices, and constraint graphs.

4.4 Symmetry Constraints Optimization and MILP Legalization

Analog layouts often demand strict geometric symmetries to guarantee device matching and minimize systematic mismatch. In DiffSP, symmetry relationships are handled in two complementary stages: (i) during differentiable sequence-pair optimization to guide learning, and (ii) during MILP-based legalization to ensure exact symmetry in the final packed layout.

Differentiable Symmetry Optimization. At the differentiable stage, symmetry is encouraged by penalizing mismatched ordering

Table 1: Post-layout Performance Comparison (The best results are in bold, and the runner-ups are colored brown).

Circuit	Schematic				ALIGN (SP + SA) [13]					MAGICAL (NLP optimizer) [31]					**DiffSP (Ours)**				
	CMRR (dB)↑	BW (MHz)↑	Gain (dB)↑	Noise (μV_{rms})↓	OV (μV)↓	CMRR (dB)↑	BW (MHz)↑	Gain (dB)↑	Noise (μV_{rms})↓	OV (μV)↓	CMRR (dB)↑	BW (MHz)↑	Gain (dB)↑	Noise (μV_{rms})↓	OV (μV)↓	CMRR (dB)↑	BW (MHz)↑	Gain (dB)↑	Noise (μV_{rms})↓
OTA1-A	153.3	112.1	38.26	214.5	4289	34.89	50.97	33.18	234.7	1186	**104.5**	**51.66**	36.96	224.6	**637.8**	97.99	37.60	**54.70**	**220.6**
OTA1-B	150.2	115.7	38.50	215.8	49700	23.20	25.53	14.62	277.0	**1277**	67.73	53.77	36.86	225.5	769.2	**75.43**	**57.50**	**37.87**	**218.8**
OTA1-C	143.3	118.8	38.61	217.8	30270	25.32	13.91	18.32	243.0	**1389**	24.89	53.73	**39.81**	247.5	8283	**48.14**	**59.38**	37.87	**219.8**
OTA1-D	115.3	121.0	38.43	221.8	17900	37.98	43.33	27.65	230.5	**964.4**	79.20	47.37	36.22	226.9	5190.5	**113.70**	**49.98**	**36.70**	**223.2**
OTA2-A	155.3	108.1	37.92	213.4	8165	32.04	37.01	30.31	299.9	6086	59.31	38.75	35.70	**258.0**	**1221**	**61.07**	**39.15**	**39.04**	280.6
OTA2-B	155.0	108.5	38.54	214.5	14130	38.38	29.62	23.49	**247.6**	55580	27.22	3.39	13.12	256.1	**10600**	**41.98**	**31.16**	**25.63**	249.6
OTA2-C	154.7	108.7	39.05	215.5	203400	22.46	6.928	8.243	219.4	35660	11.04	33.48	19.31	575.5	**91.55**	**61.22**	**39.42**	**40.68**	**281.0**
OTA2-D	154.4	108.7	39.48	216.4	13810	12.48	11.04	29.55	557.7	5731	36.48	37.69	36.08	**259.2**	**125.9**	**61.12**	**39.52**	**41.64**	282.1
OTA3-A	126.6	589.7	47.48	3024	12.38	99.52	164.9	6.223	2400	**8.80**	**113.3**	263.2	**6.670**	2359	31.31	92.57	**269.6**	6.067	**2325**
OTA3-B	136.1	595.1	47.60	3040	12.56	90.72	**271.7**	4.894	3897	**3.66**	90.03	270.3	6.670	2543	11170	**96.42**	214.9	**44.62**	**2248**
OTA4-A	88.31	607.3	22.20	2428	13.33	93.89	164.1	6.209	2398	**9.34**	**111.1**	262.8	**6.670**	2374	9.61	108.7	**270.2**	6.068	**2323**
OTA4-B	91.04	595.4	22.49	2484	14.34	86.53	40.73	6.103	2397	**3.86**	90.24	**270.0**	**6.670**	2560	31.65	**92.80**	269.40	6.067	**2321**
Avg					28476.38	49.78	71.65	17.40	1116.82	8991.59	67.92	**115.51**	23.40	1009.11	**3180.13**	**79.26**	114.82	**31.41**	**932.73**
Ratio					1.00	0.61	0.62	0.55	1.00	0.32	0.84	**1.00**	0.74	0.90	**0.11**	**1.00**	0.99	**1.00**	**0.84**

probabilities between mirrored module pairs in the soft precedence matrices. For left–right symmetric pairs (i, i') and (j, j'), we add hard symmetry constraints to each paired group:

$$\mathcal{L}_{\text{sym}} = \sum_{(i,i'),(j,j')\in\mathcal{M}_{\text{lr}}} \left\| \mathbf{A}_h[i,j] - \mathbf{A}_h[i',j'] \right\|_2^2 \tag{22}$$

and its vertical analogue on $\mathbf{A}_v$ softly guides the optimization toward symmetric placements.

MILP-based Legalization with Hard Symmetry. After differentiable optimization, the continuous layout predicted by DiffSP is discretized and a MILP-based legalization step imposes hard geometric symmetry and legality. For each left–right symmetric pair (m_{l_i}, m_{r_i}) and top–bottom pair (m_{t_i}, m_{b_i}), the constraints:

$$x_{l_i} + w_{l_i} + x_{r_i} = W, \qquad y_{l_i} = y_{r_i}, \tag{23}$$

$$y_{t_i} + h_{t_i} + y_{b_i} = H, \qquad x_{t_i} = x_{b_i}, \tag{24}$$

mirror modules about the layout center lines while maintaining alignment, respectively. Together, the soft loss $\mathcal{L}_{\text{sym}}$ and the hard MILP constraints yield symmetry-aware learning and provably symmetric, non-overlapping final layouts.

5 Experimentals

We implement the proposed performance-driven analog layout framework in Python 3.7.5 and C++. We test our method and baseline method on the same environment, with an Intel Xeon Platinum 8368@2.40GHz CPU with Micron 64G-3200MHz RAM. The differentiable global placer is built with PyTorch 2.0.0, the legalization core and router are in C++, and MILP problems are solved using Gurobi [32]. For the routing engine, we build on a modified version of the MAGICAL framework [24], also implemented in C++.

Benchmarks. In the experiments, we use ten different OTA circuits as benchmarks, denoted as OTA1-A–D, OTA2-A–D, OTA3-A–B, and OTA4-A–B. These ten benchmark circuits differ in topology and function, and are designed by experienced circuit designers in TSMC 40nm technology, with the pre-layout performance provided in Table 1. OTA1-A–D and OTA2-A–D are different 2-stage Miller-compensated amplifier OTAs. OTA3-A–B and OTA4-A–B are 2-stage telescopic OTAs with a more complex topology. All designs are implemented in TSMC 40nm technology, extracted for parasitics with Calibre PEX, and simulated with Cadence Spectre.

Compared Methods. We compare DiffSP against two strong baseline methods: (1) **ALIGN** [13]: an analog layout framework proposed by the University of Minnesota [13]. Its global placement engine uses a sequence-pair representation combined with simulated annealing, followed by an MILP legalization stage to satisfy design constraints. Since the default ALIGN framework does not support the TSMC 40nm process, we re-implemented the SP+SA-based framework and integrated a unified MILP legalization step. We integrated a unified MILP legalization step so that all methods can be compared under a unified PDK. (2) **MAGICAL** [31]: an open-source analog layout framework proposed by UT Austin [24, 31]. Its global placement engine is optimized using an NLP-based solver, and it also employs MILP afterward to produce constraint-satisfying layouts.

Evaluation Metrics. We evaluate our analog global placer and baselines using layout-quality and post-layout circuit-performance metrics. For placement quality, we report area, HPWL, and wall-clock runtime, averaged over multiple runs. Area and HPWL indicate layout compactness and wirelength. All placement metrics are measured after global placement and before legalization. We then apply the unified legalization flow of [31] to enforce symmetry constraints and use a modified MAGICAL router [33] to obtain routed layouts. On the final layouts, we measure standard post-layout metrics for OTA designs: Offset Voltage (**OV**), Common-Mode Rejection Ratio (**CMRR**), DC Gain (**Gain**), Bandwidth (**BW**), and Noise. We report normalized area, HPWL, runtime, and a post-layout figure of merit (**FoM**),

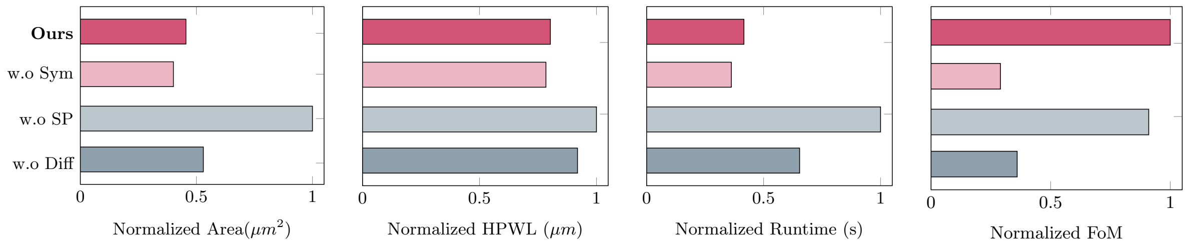
$$\text{FoM} = -0.001 \cdot \text{OV} + \text{CMRR} + \text{Gain} + \text{BW} - 0.001 \cdot \text{Noise},$$

where higher FoM indicates better circuit performance. These metrics collectively reflect both the effectiveness and efficiency of the different analog placement methods.

Placement Quality and Runtime. Table 2 reports area, HPWL, and runtime for ALIGN (SP+SA), MAGICAL (NLP), and DiffSP. ALIGN is area-efficient, whereas MAGICAL incurs, on average 2.70× larger area due to relaxed coordinates. DiffSP reduces the average area from 1212.10 μm^2 to 1140.99 μm^2 with 5.9% over ALIGN, and achieves the smallest area in OTA1-A/B/C/D, OTA3-A and OTA4-A, showing that the differentiable SP relaxation preserves compact packing. Although MAGICAL sometimes shortens wires by spreading cells, its HPWL remains worse than ALIGN, while DiffSP lowers ALIGN's average HPWL from 583.19 μm to 562.59 μm with 3.5% reduction, indicating effective wirelength optimization on soft constraint graphs without losing compactness. Runtime differences are larger: ALIGN takes 45.0 s and MAGICAL 31.8 s on average, whereas DiffSP converges

Table 2: Placement Performance Comparison (The best results are in bold, and the runner-ups are colored brown).

Circuit	ALIGN (SP + SA) [13]			MAGICAL (NLP optimizer) [31]			**DiffSP (Ours)**		
	Area (μm^2)	HPWL (μm)	Runtime (s)	Area (μm^2)	HPWL (μm)	Runtime (s)	Area (μm^2)	HPWL (μm)	Runtime (s)
OTA1-A	1453.84	**406.57**	21.00	2720.64	609.32	42.00	**1173.88**	416.32	**18.00**
OTA1-B	1406.50	602.28	**8.00**	2934.98	612.02	42.00	**1321.80**	**479.64**	15.00
OTA1-C	1637.37	541.94	59.00	3161.78	552.86	41.00	**1513.85**	**420.50**	**14.00**
OTA1-D	1824.52	644.13	**17.00**	3113.88	613.36	36.00	**1419.62**	**600.76**	20.00
OTA2-A	**1307.10**	460.28	35.00	3084.60	558.52	**33.00**	1427.67	**387.72**	18.00
OTA2-B	**1339.39**	**647.75**	**9.00**	3559.04	686.48	38.00	1483.76	650.37	15.00
OTA2-C	**1412.15**	580.94	27.00	3225.38	582.37	46.00	1532.26	**556.86**	**21.00**
OTA2-D	1393.77	**499.32**	**9.00**	3225.38	623.31	50.00	**1310.23**	504.69	14.00
OTA3-A	660.65	**643.07**	60.00	3377.58	728.78	18.00	**526.30**	652.39	**17.00**
OTA3-B	**640.91**	702.98	95.00	3653.24	763.78	**12.00**	653.79	**693.21**	17.00
OTA4-A	718.97	631.98	100.00	3595.21	740.58	**11.00**	**586.27**	**614.64**	17.00
OTA4-B	750.04	**637.01**	119.00	3595.21	756.98	**12.00**	**742.41**	773.97	18.00
Avg	1212.10	583.19	45.00	3270.58	652.36	31.75	**1140.99**	**562.59**	**17.00**
Ratio	1.000	1.000	1.000	2.698	1.119	0.705	**0.941**	**0.965**	**0.378**



(a) **(b)** **(c)** **(d)**

Figure 6: Ablation study of DiffSP on OTA1 (A-D). Averaged (a) area, (b) HPWL, (c) runtime, and (d) post-layout FoM for the full DiffSP and three variants without symmetry loss (w.o Sym), without sequence-pair representation via NLP optimizer (w.o SP), and without differentiable relaxation via SA (w.o Dif).

in 17.0s, which is 2.6× faster than MAGICAL and 5.9× faster than ALIGN. Thus, DiffSP jointly improves area, HPWL, and runtime over a strong SP+SA baseline while avoiding the severe area expansion of NLP-based placement.

Post-layout Performance Impact. Table 1 reports post-layout CMRR, bandwidth (BW), DC gain (Gain), noise, and offset voltage (OV) after routing and parasitic extraction. Overall, DiffSP consistently improves or matches circuit performance compared with ALIGN and MAGICAL. OV, which is very sensitive to mismatch and symmetry violations, is reduced from the ALIGN baseline to 0.55× by MAGICAL and further to 0.32× by DiffSP with about 68% reduction, while retaining better area efficiency. DiffSP also achieves higher or comparable CMRR and Gain, often close to schematic CMRR, indicating that symmetry-aware, compact placement effectively controls parasitic imbalance. DiffSP maintains more consistent BW and noise and offers a better trade-off among BW, noise, and OV. Overall, these geometric and symmetry-aware improvements translate into multiple post-layout performance gains, which are critical for analog layouts.

Ablation Studies. As shown in Figure 6, to understand the contribution of each component in DiffSP, we perform an ablation study on four variants: (1) **Ours**: the full DiffSP with differentiable SP, symmetry loss, and MILP legalization; (2) **w.o Sym**: removing the symmetry loss while keeping the differentiable SP formulation; (3) **w.o SP**: replacing the SP-based formulation with an NLP-style continuous coordinate optimizer; (4) **w.o Dif**: replacing the differentiable relaxation with a conventional SP+SA optimizer. Figure 6 shows that the full DiffSP achieves the best trade-off, with the smallest area/HPWL, highest FoM, and low runtime. Removing symmetry loss (w/o Sym) mainly reduces FoM, indicating its role in post-layout performance guarantee. Removing SP (w/o SP) expands the area, worsening HPWL, highlighting the value of SP's compact representation. Replacing the differentiable relaxation with SA (w.o Diff) keeps layouts compact but increases runtime and slightly degrades FoM, underscoring the benefit of gradient-based optimization. Overall, DiffSP's advantages stem from combining differentiable SP, symmetry-aware loss, and MILP enforcement.

6 Conclusion

In this work, we proposed DiffSP, a differentiable sequence-pair–based analog placement framework that combines compact geometric representations with gradient-based optimization. By relaxing SP permutations via the Gumbel–Sinkhorn operator and optimizing costs on soft constraint graphs, DiffSP provides differentiable estimates of area, HPWL, and symmetry, while MILP-based legalization enforces symmetric, legal layouts. On industrial OTA benchmarks in TSMC 40nm, DiffSP achieves 5.9% lower area, 3.5% lower HPWL, and 5.9× faster runtime than an SP+SA baseline, avoids the 2.7× area expansion of NLP-based placement, and improves multiple post-layout performance. Future work includes exploring additional constraints in DiffSP, integrating performance-driven objectives into the DiffSP framework, and extending it to hierarchical analog placement.

References

[1] F. Balasa, S. C. Maruvada, and K. Krishnamoorthy, “Efficient solution space exploration based on segment trees in analog placement with symmetry constraints,” in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2002.

[2] E. F. Y. Tam Yiu-Cheong and C. Chu., “Analog placement with symmetry and other placement constraints,” in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2006.

[3] K. Zhu, H. Chen, M. Liu, X. Tang, N. Sun, and D. Z. Pan, “Effective analog/mixed-signal circuit placement considering system signal flow,” in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2020.

[4] P.-H. Lin, H. Zhang, M. D. Wong, and Y.-W. Chang, “Thermal-driven analog placement considering device matching,” in *ACM/IEEE Design Automation Conference (DAC)*, 2009.

[5] Q. Ma, E. F. Y. Young, and K. P. Pun, “Analog placement with common centroid constraints,” in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2007.

[6] P.-Y. Chou, H.-C. Ou, and Y.-W. Chang, “Heterogeneous B*-trees for analog placement with symmetry and regularity considerations,” in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2011.

[7] S. Nakatake, M. Kawakita, T. Ito, M. Kojima, M. Kojima, K. Izumi, and T. Habasaki, “Regularity-oriented analog placement with diffusion sharing and well island generation,” in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 2010.

[8] H.-C. C. Chien, H.-C. Ou, T.-C. Chen, T.-Y. Kuan, and Y.-W. Chang, “Double patterning lithography-aware analog placement,” in *ACM/IEEE Design Automation Conference (DAC)*, 2013.

[9] P.-H. Lin and S.-C. Lin, “Analog placement based on hierarchical module clustering,” in *ACM/IEEE Design Automation Conference (DAC)*, 2008.

[10] Q. Ma, L. Xiao, Y.-C. Tam, and E. F. Young, “Simultaneous handling of symmetry, common centroid, and general placement constraints,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 30, no. 1, pp. 85–95, 2010.

[11] F. Balasa and K. Lampaert, “Module placement for analog layout using the sequence-pair representation,” in *ACM/IEEE Design Automation Conference (DAC)*, 1999.

[12] F. Balasa, S. C. Maruvada, and K. Krishnamoorthy, “On the exploration of the solution space in analog placement with symmetry constraints,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 23, no. 2, pp. 177–191, 2004.

[13] K. Kunal, M. Madhusudan, A. K. Sharma, W. Xu, S. M. Burns, R. Harjani, J. Hu, D. A. Kirkpatrick, and S. S. Sapatnekar, “ALIGN: Open-source analog layout automation from the ground up,” in *ACM/IEEE Design Automation Conference (DAC)*, 2019.

[14] D. Henderson, S. H. Jacobson, and A. W. Johnson, “The theory and practice of simulated annealing,” in *Handbook of metaheuristics.* Springer, 2003, pp. 287–319.

[15] P.-C. Pan, C.-Y. Chin, H.-M. Chen, T.-C. Chen, C.-C. Lee, and J.-C. Lin, “A fast prototyping framework for analog layout migration with planar preservation,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 34, no. 9, p. 1373–1386, Sep. 2015.

[16] T.-C. Chen and Y.-W. Chang, “Modern floorplanning based on b/sup */-tree and fast simulated annealing,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 25, no. 4, pp. 637–650, 2006.

[17] X. Tang and D. F. Wong, “Floorplanning with alignment and performance constraints,” in *ACM/IEEE Design Automation Conference (DAC)*, 2002.

[18] L. Xiao, E. F. Young, X. He, and K.-P. Pun, “Practical placement and routing techniques for analog circuit designs,” in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2010.

[19] Y. Li, Y. Lin, M. Madhusudan, A. Sharma, W. Xu, S. S. Sapatnekar, R. Harjani, and J. Hu, “A customized graph neural network model for guiding analog IC placement,” in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2020.

[20] B. Xu, B. Basaran, M. Su, and D. Z. Pan, “Analog placement constraint extraction and exploration with the application to layout retargeting,” in *ACM International Symposium on Physical Design (ISPD)*, 2018.

[21] B. Xu, S. Li, X. Xu, N. Sun, and D. Z. Pan, “Hierarchical and analytical placement techniques for high-performance analog circuits,” in *ACM International Symposium on Physical Design (ISPD)*, 2017.

[22] H.-C. Ou, K.-H. Tseng, J.-Y. Liu, I.-P. Wu, and Y.-W. Chang, “Layout-dependent-effects-aware analytical analog placement,” in *ACM/IEEE Design Automation Conference (DAC)*, 2015.

[23] B. Xu, S. Li, C.-W. Pui, D. Liu, L. Shen, Y. Lin, N. Sun, and D. Z. Pan, “Device layer-aware analytical placement for analog circuits,” in *ACM International Symposium on Physical Design (ISPD)*, 2019.

[24] H. Chen, M. Liu, X. Tang, K. Zhu, A. Mukherjee, N. Sun, and D. Z. Pan, “MAGICAL 1.0: An open-source fully-automated ams layout synthesis framework verified with a 40-nm 1GS/s $\Delta\sum$ ADC,” in *IEEE Custom Integrated Circuits Conference (CICC)*, 2021.

[25] B. Xu, Y. Lin, X. Tang, S. Li, L. Shen, N. Sun, and D. Z. Pan, “WellGAN: Generative-adversarial-network-guided well generation for analog/mixed-signal circuit layout,” in *ACM/IEEE Design Automation Conference (DAC)*, 2019.

[26] X. Tang and D. Wong, “FAST-SP: A fast algorithm for block placement based on sequence pair,” in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 2001.

[27] G. Mena, D. Belanger, S. Linderman, and J. Snoek, “Learning Latent Permutations with Gumbel-Sinkhorn Networks,” in *International Conference on Learning Representations (ICLR)*, 2018.

[28] J. Munkres, “Algorithms for the assignment and transportation problems,” *Journal of the society for industrial and applied mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[29] M. Huh, B. Cheung, P. Agrawal, and P. Isola, “Straightening out the straight-through estimator: Overcoming optimization challenges in vector quantized networks,” in *International Conference on Machine Learning (ICML)*, 2023.

[30] B. Farbey, A. Land, and J. Murchland, “The cascade algorithm for finding all shortest distances in a directed graph,” *Management Science*, vol. 14, no. 1, pp. 19–28, 1967.

[31] K. Zhu, H. Chen, M. Liu, and D. Z. Pan, “Hierarchical analog and mixed-signal circuit placement considering system signal flow,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 8, pp. 2689–2702, 2022.

[32] Gurobi Optimization, LLC, “Gurobi Optimizer Reference Manual,” 2025. [Online]. Available: https://www.gurobi.com

[33] H. Chen, K. Zhu, M. Liu, X. Tang, N. Sun, and D. Z. Pan, “Toward silicon-proven detailed routing for analog and mixed-signal circuits,” in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2020.