EAPLACE-VLM: EVOLUTIONARY MACRO PLACEMENT VIA VISION-LANGUAGE TOPOLOGICAL DIAGNOSIS AND GRID-FREE CONTINUOUS REFINEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Macro placement is a fundamental yet highly challenging problem in modern chip design, characterized by large-scale combinatorial complexity, high-dimensional constraints, and strong sensitivity to early decisions. Existing methods, including analytical optimization, reinforcement learning, and heuristic-based approaches, have achieved notable progress but still suffer from inherent limitations such as difficulty in handling discrete constraints, sample inefficiency, and susceptibility to local optima caused by sequential decision-making. In particular, heuristic-driven relocation strategies often exhibit limited global awareness, leading to sub-optimal solutions. To address these challenges, this paper proposes EAPlace-VLM, a novel dual-engine evolutionary framework that integrates vision-language models (VLMs) into macro placement. The proposed method extends traditional heuristic optimization into a unified paradigm combining multimodal semantic reasoning and continuous physical refinement. Specifically, a VLM is incorporated as a low-frequency topological diagnostician within the evolutionary loop to guide macro relocation decisions, thereby alleviating the spatial myopia of conventional heuristics. To enable effective cross-modal reasoning, we introduce a topology-to-color visual encoding scheme that transforms hypergraph connectivity into perceptually meaningful patterns, allowing the VLM to perform zero-shot spatial analysis and identify problematic macro clusters. Furthermore, a grid-free continuous physical refinement module is developed to eliminate discretization artifacts via legality-aware sub-pixel optimization in continuous space, improving layout compactness and physical fidelity. Experimental results on the standard ISPD 2005 benchmarks demonstrate that EAPlace-VLM achieves the state-of-the-art performance. Compared to the leading baseline EGPlace, our framework significantly reduces wirelength by up to 25% while maintaining a negligible refinement computational overhead of less than 2.5%, securing the best average rank in both pure macro and mixed-size global placement tasks.

1 INTRODUCTION

Macro placement is a critical stage in modern chip design, directly impacting design performance, quality, and manufacturability (MacMillen et al., 2000; Kim & Markov, 2012). Fundamentally, it can be formulated as a large-scale combinatorial optimization problem: assigning appropriate locations to macros within a fixed chip layout while satisfying constraints such as non-overlap (Wang et al., 2009). This problem is characterized by high-dimensional constraints and strong sensitivity to boundary conditions, which significantly increases its complexity. Although recent advances in machine learning have improved placement efficiency to some extent (Mirhoseini et al., 2021; Lin et al., 2019), the vast design space still makes macro placement one of the most challenging core problems in electronic design automation (EDA) (Majumdar et al., 2024).

With continued research progress, approaches ranging from constructive methods (Shi et al., 2023; Deng et al., 2025) to reinforcement learning agents (Lai et al., 2022; Geng et al., 2024), and further to Transformer-based models (Lai et al., 2023), driving continuous progress in this field. Notably,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

these methods can generally be formulated as sequential decision-making processes. For instance, WireMask-BBO (Shi et al., 2023) determines the initial placement order based on the total macro area involved in nets, while MaskPlace (Lai et al., 2022) follows heuristic strategies such as place larger macros first and prioritize denser regions. However, in sequential placement, early decisions directly constrain the feasible placement space for subsequent macros. A suboptimal choice in the early stage may trigger a "domino effect", irreversibly shrinking the solution space and leading to local optima (Ross & Bagnell, 2010; Kahng et al., 2011). In practical placement workflows, engineers often iteratively adjust and refine macro positions, a process that can be viewed as a form of visual–action coordinated decision-making.

To address the local optimality issue caused by sequential decision-making, Liu et al. (2008) applied evolutionary algorithms to optimize the placement order of macros. However, evaluating each individual in the population incurs a prohibitively high computational cost. In addition, Majumdar et al. (2024) explored a potential direction of introducing backtracking repair models that learn to identify and correct earlier mistakes. Similarly, Deng et al. (2025) employ heuristic methods to identify poorly placed macros and perform relocation to improve the final layout. However, these approaches suffer from notable limitations. On the one hand, training a model capable of general backtracking and repair requires extensive exploration and incurs high sample complexity (Majumdar et al., 2024). On the other hand, heuristic-based strategies rely on fixed rules for selecting macros, limiting their flexibility and generalization capability.

In recent years, large language models (LLMs) have shown strong generalization and reasoning ability, with growing applications in optimization, automated algorithm design, and learning-system design (Achiam et al., 2023; Wu et al., 2025; Liu et al., 2024a; Yu & Liu, 2026; Mo et al., 2025b; Yu & Liu, 2025). LLMs have also been increasingly explored in chip design, showing their potential to assist complex EDA workflows (Liu et al., 2023c;d; Thakur et al., 2023; Fu et al., 2023; Liu et al., 2024b; Saha et al., 2024). Recent studies further introduce LLMs into physical design, including floorplanning and macro placement (Yao et al., 2025; Mo et al., 2025a; 2026). Vision-language models (VLMs) can process rendered layout images alongside textual inputs, providing a multimodal interface for macro-selection decisions (Liu et al., 2023a; Bai et al., 2023). However, how to systematically incorporate VLMs into macro placement and fully leverage their decision-guiding capability remains an open research question.

To address these challenges, we propose EAPlace-VLM, a dual-engine evolutionary framework that combines heuristic macro relocation with multimodal candidate selection and continuous physical refinement. To complement the scalar-driven heuristic, EAPlace-VLM incorporates a VLM into the evolutionary process as a low-frequency topological diagnostician that proposes macro subsets for relocation. Specifically, a topology-to-color visual encoding maps dominant-net connectivity to color patterns that provide partial connectivity cues for zero-shot candidate selection. The VLM outputs macro identifiers rather than coordinates, leaving coordinate assignment and geometric legality to a deterministic mask-guided insertion operator. This separation avoids relying on model-generated coordinates, but it does not guarantee correct visual interpretations or optimal subset choices. Furthermore, to reduce quantization-induced whitespace, a grid-free continuous physical refinement module is applied to the best grid placement after the evolutionary search. It performs legality-aware sub-pixel local search in physical $\mathbb{R}^2$ space. In summary, EAPlace-VLM distinguishes itself from previous works through the following key features:

- **VLM-guided relocation paradigm.** We use VLMs as high-level topological diagnosticians to propose macro subsets from net-aware layout images and structured macro information. The model selects identifiers, while deterministic greedy insertion assigns coordinates and enforces geometric legality.
- **Asymmetric dual-engine evolutionary framework.** We combine high-frequency heuristic evolution with low-frequency VLM intervention. Both branches share the same insertion operator, allowing multimodal proposals to complement scalar-driven selection within one evolutionary search.
- **Grid-free continuous physical refinement.** The final stage of EAPlace-VLM performs legality-aware sub-pixel local search in physical $\mathbb{R}^2$ space to reduce quantization-induced whitespace. Its effect is evaluated on grid-level placements produced by EGPlace and by the evolutionary search stage of EAPlace-VLM.

- **Empirical evaluation of search and refinement.** EAPlace-VLM and its downstream DREAMPlace flow achieve the best average ranks among the compared methods on the six-benchmark macro-only and mixed-size tasks, respectively. Separate search-stage and matched-refinement comparisons distinguish these results from the method's overall gains over EGPlace.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 introduces the problem formulation. Section 4 presents EAPlace-VLM, Section 5 reports the experimental results, and Section 6 concludes. Appendix A provides the VLM prompt and response example, Appendix B discusses the benchmark, and Appendices C–D provide implementation details and additional experiments.

2 Related Work

2.1 Macro Placement Methods

Existing macro placement methodologies can generally be categorized into three groups: analytical, reinforcement learning-based, and constructive approaches. Analytical methods, such as ePlace (Lu et al., 2015) and DREAMPlace (Lin et al., 2019), formulate placement as a continuous optimization problem and employ gradient-based techniques to minimize objectives such as wirelength and density. While computationally efficient, these methods often struggle to accurately handle the discrete nature of macro non-overlap constraints and other non-differentiable components of the problem. RL-based methods, pioneered by AlphaChip (Mirhoseini et al., 2021) and further developed in works such as MaskPlace (Lai et al., 2022) and ChiPFormer (Lai et al., 2023), model macro placement as a sequential decision-making process, typically framed as a Markov Decision Process (Alagoz et al., 2010). Although these approaches have demonstrated strong performance, they generally require substantial computational resources for training and may suffer from sample inefficiency. Constructive and hybrid methods, exemplified by WireMask-EA (Shi et al., 2023), combine global search strategies (e.g., evolutionary algorithms (Cohoon & Paris, 1987)) with deterministic greedy solvers. In particular, WireMask-EA employs a mask-guided greedy insertion strategy to efficiently generate feasible placements.

2.2 Vision-Language Models in Physical Design

Most recently, the scope of foundation models in EDA has expanded from text-based large language models (Wu et al., 2024) to multimodal VLMs (Liu et al., 2023a). Early integrations of LLMs in IC design primarily focused on generating register-transfer level code or logic synthesis scripts rather than direct spatial layout generation (Liu et al., 2023b). Model-generated outputs can contain hallucinations (Huang et al., 2025), so a placement workflow should not rely on unverified coordinate predictions to satisfy geometric constraints. VLMs offer an interface for combining layout images with structured text. Building upon this direction, **EAPlace-VLM** uses a VLM to propose macro subsets for relocation rather than to generate coordinates. Its topology-to-color encoding supplies partial connectivity cues, while a deterministic mask-guided insertion operator assigns coordinates and enforces geometric legality. This division of responsibilities does not eliminate perceptual errors or validate the model's verbal rationale, as illustrated by the recorded interaction in Appendix A.

3 Preliminaries

A macro placement instance consists of macros $\mathcal{M}$, a connectivity hypergraph $\mathcal{E}$, fixed I/O ports $\mathcal{O}$, and a physical canvas $\mathcal{D}^{\mathrm{c}} = [0, W^{\mathrm{c}}] \times [0, H^{\mathrm{c}}]$. Macro m_i has fixed physical dimensions $(w_i^{\mathrm{c}}, h_i^{\mathrm{c}})$ and lower-left coordinate $\mathbf{p}_i^{\mathrm{c}}$. A legal placement keeps all macro rectangles inside the canvas without pairwise overlap. We minimize half-perimeter wirelength (HPWL), the sum of the horizontal and vertical spans of net bounding boxes. The full notation and formal definitions are in Appendix C.1.

The evolutionary search uses a $G_x \times G_y$ grid with physical cell sizes $r_x = W^{\mathrm{c}}/G_x$ and $r_y = H^{\mathrm{c}}/G_y$. Macro dimensions are conservatively quantized as

$$w_i = \lceil w_i^{\mathrm{c}}/r_x \rceil, \qquad h_i = \lceil h_i^{\mathrm{c}}/r_y \rceil. \tag{1}$$

We use $\mathbf{P}$ for a grid placement and $\mathbf{P}^{\mathrm{c}}$ for a physical placement. Projection Π scales grid coordinates by (r_x, r_y) and restores true macro dimensions. Conservative footprints preserve legality on projection but can introduce artificial gaps, such as $r_x w_i - w_i^{\mathrm{c}} \geq 0$ between horizontally adjacent macros. The continuous refiner targets this geometric waste.

Two masks support greedy insertion. PositionMask V_i filters coordinates that violate the boundary or overlap fixed macros. WireMask W_i scores the HPWL increment on nets incident to m_i. The insertion coordinate is

$$\mathbf{p}_i^{\star} = \arg \min_{\mathbf{p}_i:\, V_i(\mathbf{p}_i)=1} W_i(\mathbf{p}_i). \tag{2}$$

Grid-level HPWL guides search, and physical HPWL evaluates the final placement.

4 Methodology

4.1 Framework Overview

EAPlace-VLM combines frequent heuristic evolution, periodic VLM diagnosis, and final physical refinement (Figure 1). Initialization constructs legal placements by inserting randomly ordered macros and retaining a population $\mathcal{Q}$ of at most N candidates. Each iteration selects a parent, removes a replace-set $\mathcal{R}_t$, and reinserts these macros while keeping the others fixed. The resulting child enters the population. After the search budget expires, the best grid placement is projected and refined. Algorithm 1 summarizes this procedure, with the complete pseudocode provided in Appendix C.2.

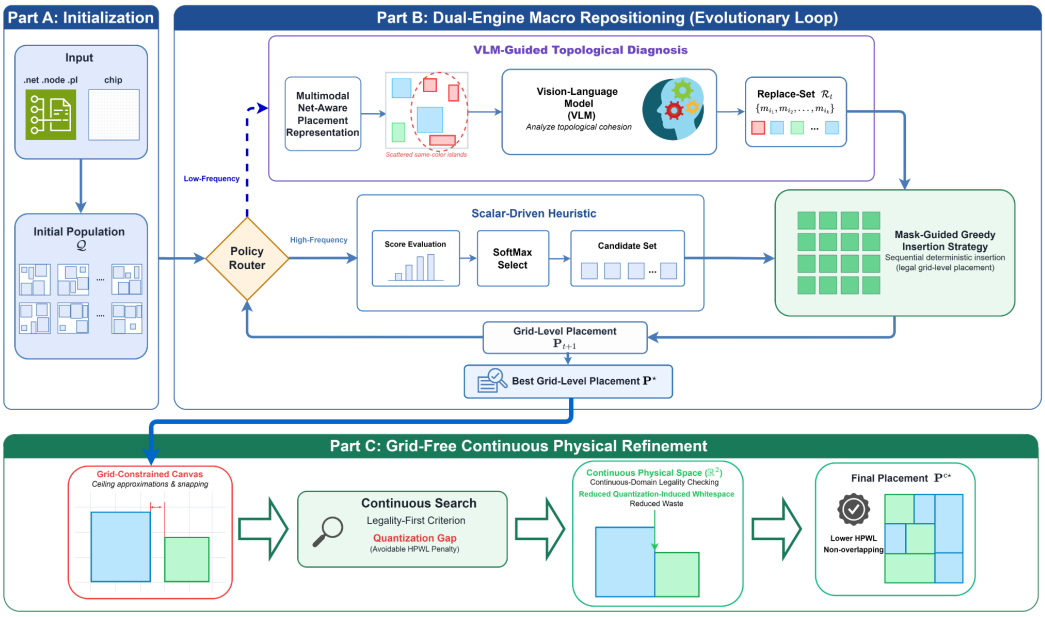

Figure 1: Overview of EAPlace-VLM: heuristic evolution and periodic VLM diagnosis share a legal greedy insertion operator, followed by physical refinement.

4.2 Multimodal Net-Aware Placement Representation

The VLM receives a rendered placement and structured macro information. Rectangles encode grid coordinates and quantized sizes, while white space marks unoccupied regions and a light grid provides spatial reference. To expose connectivity without cluttered flylines, each macro is colored by its dominant incident net:

$$e^{\star}(m_i) = \arg \max_{e \in \mathcal{N}(m_i)} |\mathcal{V}(e)|, \qquad \mathrm{Hue}(e^{\star}) = \big(\mathrm{ID}(e^{\star})\varphi\big) \bmod 1, \tag{3}$$

where $\mathcal{N}(m_i)$ contains incident nets, $\mathcal{V}(e)$ contains incident pins or nodes, and $\varphi = (\sqrt{5}-1)/2$. High fanout is a proxy for global topological influence. Saturation and value are kept fixed to maintain contrast, and macros without valid incident nets are rendered in gray.

Same-dominant-net macros share a color, exposing possible spatial dispersion. This is a partial encoding: a macro can belong to several nets, so differently colored macros can still share a non-dominant net. Color therefore provides a connectivity cue, not a complete representation of the hypergraph. The structured map $\mathcal{S}_t = \{i \mapsto [x_i, y_i, w_i, h_i]\}$ links image regions to macro IDs. The VLM returns at most K IDs rather than coordinates, leaving coordinate selection and legality to the insertion operator. Appendix A preserves the prompt and an actual response, with an explanation of overlapping net membership.

4.3 Dual-Engine Selection and Reinsertion

Most iterations use a scalar-driven heuristic that samples macros without replacement according to normalized HPWL, RUDY congestion, and overlap indicators. Every τ iterations, the VLM instead diagnoses the current parent's net-aware representation and proposes $\mathcal{R}_t$, with $|\mathcal{R}_t| \le K$. Invalid or empty VLM selections fall back to the heuristic. This periodic schedule reserves VLM calls for structural guidance while retaining frequent inexpensive updates.

The selected macros are removed, sorted by descending physical area, and reinserted using Equation (2). Occupancy and net bounding boxes are updated after each insertion. Both branches use this same operator: the replace-set controls which earlier decisions are reconsidered, while the masks determine feasible coordinates. Scoring and insertion details are in Appendix C.3.

Algorithm 1: EAPlace-VLM (compact procedure).

Input: Instance, population size N, initialization rounds T_{init}, VLM period τ, relocation budget K, and search budget $\mathcal{T}$.
Output: Final physical placement $\mathbf{P}^{c\star}$.

$\mathcal{Q} \leftarrow$ initialize and prune T_{init} random greedy placements to size N
$t \leftarrow 0$
while *elapsed search time* $< \mathcal{T}$ **do**
  $\mathbf{P}_t \leftarrow \textsc{SoftmaxSelect}(\mathcal{Q})$
  if $t \bmod \tau = 0$ **then**
    $\mathcal{R}_t \leftarrow \textsc{VLMSelect}(\mathbf{P}_t, K)$, with heuristic fallback on failure
  else
    $\mathcal{R}_t \leftarrow \textsc{HeuristicSelect}(\mathbf{P}_t)$
  $\mathbf{P}_{t+1} \leftarrow \textsc{GreedyInsert}(\mathbf{P}_t|_{\mathcal{M}\setminus\mathcal{R}_t}, \textsc{SortByArea}(\mathcal{R}_t))$
  $\mathcal{Q} \leftarrow \textsc{Prune}(\mathcal{Q} \cup \{\mathbf{P}_{t+1}\}, N)$
  $t \leftarrow t+1$
$\mathbf{P}^\star \leftarrow \arg\min_{\mathbf{P}\in\mathcal{Q}} \mathrm{HPWL}(\mathbf{P})$
return $\textsc{ContinuousRefine}(\Pi(\mathbf{P}^\star))$

4.4 Grid-Free Continuous Physical Refinement

Starting from $\mathbf{P}^{c0} = \Pi(\mathbf{P}^\star)$, the refiner adjusts physical coordinates to reduce quantization-induced whitespace (Figure 2). It searches displacement candidates within a shrinking radius, using a sub-grid sampling step. Here, "grid-free" means that coordinates are no longer restricted to the original placement grid, although the local search still evaluates a finite displacement lattice. The projected macro dimensions remain their true physical sizes.

Let $\Omega(\mathbf{p}_i^c)$ combine overlap area and boundary violation, and let $\Delta\,\mathrm{HPWL}_i^c$ be the wirelength change on affected nets. A candidate $\mathbf{p}_i^{c+}$ is accepted under the legality-first rule

$$\mathrm{Accept}(\mathbf{p}_i^{c+}) = \begin{cases} \text{True}, & \Omega(\mathbf{p}_i^{c+}) < \Omega(\mathbf{p}_i^c) - \varepsilon, \\ \text{True}, & \Omega(\mathbf{p}_i^c), \Omega(\mathbf{p}_i^{c+}) \le \varepsilon \text{ and } \Delta\,\mathrm{HPWL}_i^c < -\varepsilon, \\ \text{False}, & \text{otherwise.} \end{cases} \tag{4}$$

Thus a legal placement accepts only legality-preserving HPWL improvements. Refinement sweeps over all macros, reduces the search radius after each pass, and stops when a full pass accepts no displacement. Appendix C.4 gives the sampling schedule, penalty definitions, and per-pass complexity.

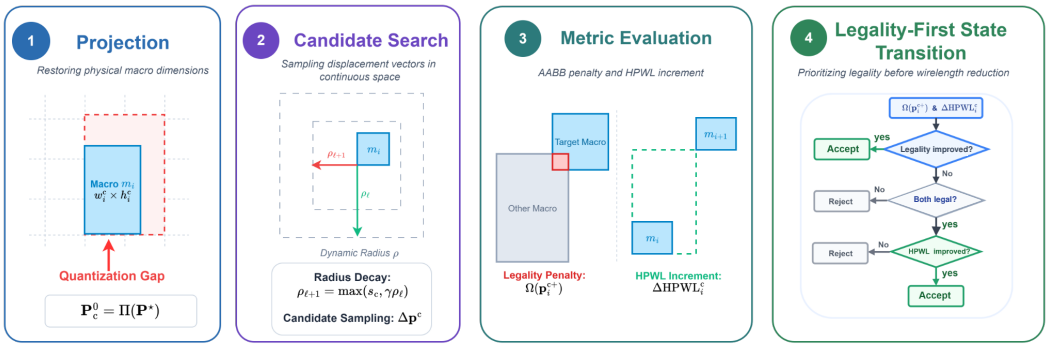

Figure 2: Four-step physical refinement. The acceptance branch first checks whether legality improves. Otherwise, acceptance requires both positions to be legal and HPWL to decrease.

5 Experiments

5.1 Experimental Setup

Following prior work (Deng et al., 2025; Lai et al., 2022; Geng et al., 2024), we evaluate on ISPD2005 (Nam et al., 2005). Baselines cover simulated annealing (SP-SA), analytical placement (NTUPlace3, RePlace, DREAMPlace), learning-based placement (GraphPlace, DeepPR, MaskPlace, ChiPFormer), and constructive/evolutionary search (WireMask-EA, EfficientPlace, EGPlace). Experiments use two Intel Xeon Platinum 8374C CPUs at 2.70 GHz and an NVIDIA RTX A6000 GPU. Defaults are $K = 6$, $N = 5$, $T_{\text{init}} = 40$, and a 1000-minute evolutionary search budget per instance. For the main HPWL comparisons, we report means and standard deviations over five independent seeds for EAPlace-VLM, WireMask-EA, and EGPlace, whereas results for the other methods follow BBOPlace-Bench (Xue et al., 2025) and the cited papers. The full baseline references are provided in Appendix D.1, and parameter sensitivity and runtime details are reported in Appendix D.

Method naming and evaluation stages. EAPlace-VLM denotes the complete framework, including continuous physical refinement. Experiments that omit this stage are explicitly identified as evaluations without continuous refinement. For mixed-size placement, EAPlace-VLM + DP denotes EAPlace-VLM followed by DREAMPlace (DP) for standard-cell placement.

Reporting conventions. HPWL ranks and relative reductions are calculated from the displayed values (means where reported). Methods are ranked separately on each benchmark within a table, with average ranks assigned to equal displayed values. An average relative reduction is the unweighted arithmetic mean of $100(H_{\text{base}} - H_{\text{method}})/H_{\text{base}}$ over the stated benchmark set. These summaries are descriptive and do not imply statistical significance.

5.2 Main Placement Results

Table 1 compares macro-only HPWL on six benchmarks with common baseline coverage. EAPlace-VLM has the lowest mean HPWL on five and the best average rank, 1.33. Relative to EGPlace, its per-benchmark reductions range from 4.3% to 25.1%, averaging approximately 12.2% from the displayed means. DREAMPlace and RePlace remain lower on adaptec4. These reductions evaluate EAPlace-VLM as a whole rather than isolating the contribution of VLM-guided selection. When EGPlace is equipped with the same continuous refinement, EAPlace-VLM has lower mean HPWL on six of eight benchmarks, with an average per-benchmark reduction of approximately 3.1% (Table 4).

For the mixed-size comparison, DREAMPlace places standard cells after macro placement. In EAPlace-VLM + DP, it is run after EAPlace-VLM has completed both evolutionary search and continuous physical refinement. EAPlace-VLM + DP has the lowest mean HPWL on four of six

Table 1: Macro-only HPWL ($\times 10^5$) on six benchmarks. Values are mean $\pm$ std where available, with a single value for NTUPlace3. Baselines follow EGPlace and BBOPlace-Bench. $+/-/=$ counts lower/higher/equal displayed HPWL values relative to EAPlace-VLM, not statistical significance. Average ranks use the convention in Section 5. Bold marks the lowest displayed value per benchmark. The same-refiner comparison is reported in Table 4.

Method	adaptec1	adaptec2	adaptec3	adaptec4	bigblue1	bigblue3	$+/-/=$	Avg. rank
SP-SA	18.84±4.62	117.36±8.73	115.48±7.56	120.03±4.25	5.12±1.43	164.70±19.55	0/6/0	8.50
NTUPlace3	26.62	321.17	328.44	462.93	22.85	455.53	0/6/0	11.17
RePlace	16.19±2.10	153.26±29.01	111.21±11.69	37.64±1.05	2.45±0.06	119.84±34.43	1/5/0	6.67
DREAMPlace	15.81±1.64	140.79±26.73	121.94±25.05	**37.41±0.87**	2.44±0.06	107.19±29.91	1/5/0	6.17
GraphPlace	30.10±2.98	351.71±38.20	358.18±13.95	151.42±9.72	10.58±1.29	357.48±47.83	0/6/0	11.00
DeepPR	19.91±2.13	203.51±6.27	347.16±4.32	311.86±56.74	23.33±3.65	430.48±12.18	0/6/0	10.83
MaskPlace	7.62±0.67	75.16±4.97	100.24±13.54	87.99±3.25	3.04±0.06	90.04±4.83	0/6/0	6.67
ChiPFormer	6.62±0.05	67.10±5.46	76.70±1.15	68.80±1.59	2.95±0.04	72.92±2.56	0/6/0	5.67
WireMask-EA	5.86±0.06	52.96±0.61	57.10±2.08	60.10±1.66	2.12±0.01	64.41±3.88	0/6/0	3.58
EfficientPlace	5.94±0.04	46.79±1.60	56.35±0.99	58.47±1.61	2.14±0.01	58.38±0.54	0/6/0	3.33
EGPlace	5.86±0.14	39.92±1.59	60.13±1.83	56.21±0.43	2.26±0.01	53.03±5.43	0/6/0	3.08
EAPlace-VLM	**5.61±0.03**	**33.67±1.80**	**55.57±0.31**	49.64±0.50	**2.06±0.01**	**39.74±3.66**		**1.33**

Table 2: Mixed-size HPWL ($\times 10^7$), with DREAMPlace (DP) placing standard cells after macro placement. EAPlace-VLM + DP runs DREAMPlace after the complete EAPlace-VLM procedure, including continuous refinement. Compared results follow BBOPlace-Bench and EGPlace. Bold marks the lowest displayed mean per benchmark, and average ranks use the reporting convention in Section 5.

Benchmark	DREAMPlace	MaskPlace + DP	WireMask-EA + DP	EfficientPlace + DP	EGPlace + DP	EAPlace-VLM + DP
adaptec1	9.62 ± 0.78	10.86 ± 0.18	8.93 ± 0.01	7.20 ± 0.12	7.53 ± 0.11	$\mathbf{7.09 \pm 0.29}$
adaptec2	12.45 ± 3.31	12.98 ± 0.58	9.20 ± 0.05	9.20 ± 0.61	$\mathbf{9.06 \pm 0.36}$	9.22 ± 0.40
adaptec3	17.18 ± 0.51	26.14 ± 0.07	21.72 ± 0.01	16.49 ± 1.07	14.15 ± 0.19	$\mathbf{14.10 \pm 0.13}$
adaptec4	40.04 ± 3.87	26.14 ± 0.07	20.51 ± 0.01	14.70 ± 0.25	15.69 ± 0.09	$\mathbf{14.21 \pm 0.13}$
bigblue1	8.30 ± 0.06	10.64 ± 0.01	10.35 ± 0.02	8.67 ± 0.10	8.99 ± 0.06	$\mathbf{8.17 \pm 0.00}$
bigblue3	38.22 ± 1.48	54.98 ± 1.06	42.52 ± 0.11	$\mathbf{28.48 \pm 0.96}$	29.08 ± 0.23	29.81 ± 1.75
Avg. rank	4.33	5.83	4.25	2.25	2.50	**1.83**

benchmarks and the best average rank, 1.83 (Table 2). However, EGPlace + DP has lower mean HPWL on adaptec2, and both EfficientPlace + DP and EGPlace + DP are lower on bigblue3. Thus, the favorable average ranking does not imply uniform downstream improvements. This comparison also does not isolate the effect of continuous refinement on mixed-size placement.

5.3 Effect of VLM-Guided Diagnosis

Table 3: VLM backbone comparison for EAPlace-VLM with continuous refinement disabled (HPWL $\times 10^5$). Bold marks the lowest displayed mean among the two VLM variants.

Benchmark	WireMask-EA	EGPlace	EAPlace-VLM (GPT-4o)	EAPlace-VLM (Qwen)
adaptec1	5.86 ± 0.06	5.86 ± 0.14	5.78 ± 0.06	$\mathbf{5.77 \pm 0.03}$
adaptec2	52.96 ± 0.61	39.92 ± 1.59	37.00 ± 0.35	$\mathbf{36.25 \pm 1.84}$
adaptec3	57.10 ± 2.08	60.13 ± 1.83	$\mathbf{58.39 \pm 0.67}$	58.63 ± 0.26
adaptec4	60.10 ± 1.66	56.21 ± 0.43	54.16 ± 0.68	$\mathbf{53.83 \pm 0.37}$
bigblue1	2.12 ± 0.01	2.26 ± 0.01	$\mathbf{2.18 \pm 0.01}$	$\mathbf{2.18 \pm 0.01}$
bigblue2	194.91 ± 1.43	10.96 ± 0.57	$\mathbf{10.64 \pm 0.35}$	10.91 ± 1.33
bigblue3	64.41 ± 3.88	53.03 ± 5.43	47.77 ± 4.93	$\mathbf{46.02 \pm 3.59}$
bigblue4	130.49 ± 2.32	63.10 ± 1.76	62.55 ± 1.63	$\mathbf{61.75 \pm 1.84}$

With continuous refinement disabled, both EAPlace-VLM backbone variants have lower mean HPWL than EGPlace on all eight benchmarks (Table 3). Qwen has a lower displayed mean on five benchmarks, GPT-4o on two, and their displayed means are equal on bigblue1. Figure 3 shows their search trajectories. These comparisons show search-stage improvements with two VLM backbones, but they do not separately isolate the contributions of visual input, topology-to-color encoding, or VLM-guided selection from the rest of the search framework.

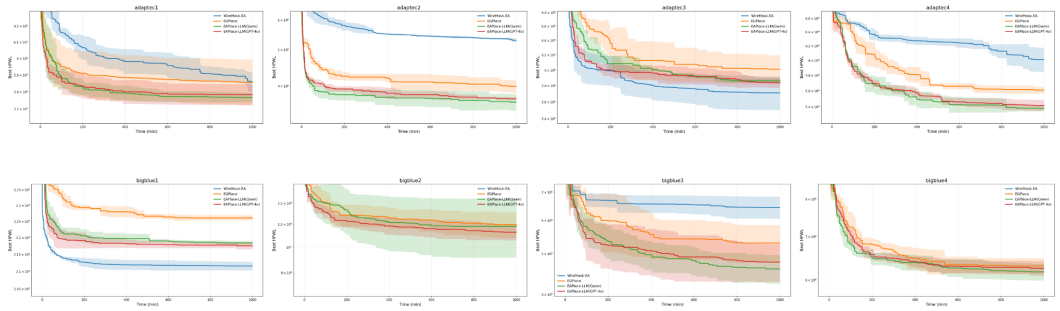

Figure 3: Grid-level search convergence without continuous refinement. The curves show the best HPWL over 1000 minutes, while shaded regions indicate variation across independent runs. All eight original trajectories are retained, with enlarged panels provided in Appendix **??**.

5.4 Effect of Continuous Refinement

Table 4: Continuous-refinement ablation (HPWL $\times 10^5$). The "Without refinement" group disables EAPlace-VLM's final refinement stage. The "With refinement" group compares EAPlace-VLM with EGPlace equipped with the same continuous refiner. Bold marks the lower displayed mean within each group, rather than a statistically significant difference.

	Without refinement		With refinement	
Benchmark	EGPlace	EAPlace-VLM	EGPlace	EAPlace-VLM
adaptec1	5.86 ± 0.14	$\mathbf{5.77 \pm 0.03}$	5.68 ± 0.14	$\mathbf{5.61 \pm 0.03}$
adaptec2	39.92 ± 1.59	$\mathbf{36.25 \pm 1.84}$	37.19 ± 1.34	$\mathbf{33.67 \pm 1.80}$
adaptec3	60.13 ± 1.83	$\mathbf{58.63 \pm 0.26}$	$\mathbf{55.39 \pm 0.62}$	55.57 ± 0.31
adaptec4	56.21 ± 0.43	$\mathbf{53.83 \pm 0.37}$	50.08 ± 0.59	$\mathbf{49.64 \pm 0.50}$
bigblue1	2.26 ± 0.01	$\mathbf{2.18 \pm 0.01}$	2.13 ± 0.01	$\mathbf{2.06 \pm 0.01}$
bigblue2	10.96 ± 0.57	$\mathbf{10.91 \pm 1.33}$	$\mathbf{9.72 \pm 0.40}$	10.10 ± 1.27
bigblue3	53.03 ± 5.43	$\mathbf{46.02 \pm 3.59}$	46.33 ± 5.34	$\mathbf{39.74 \pm 3.66}$
bigblue4	63.10 ± 1.76	$\mathbf{61.75 \pm 1.84}$	55.94 ± 1.17	$\mathbf{55.79 \pm 1.49}$

Table 4 evaluates the final refinement stage of EAPlace-VLM and applies the same module to EGPlace. Relative to their respective pre-refinement placements, mean HPWL decreases on every benchmark, with average per-benchmark reductions of 8.7% for EGPlace and 7.4% for EAPlace-VLM. With continuous refinement disabled, EAPlace-VLM has an average per-benchmark reduction of approximately 4.6% relative to EGPlace. When EGPlace is equipped with the same refinement, EAPlace-VLM has an average reduction of approximately 3.1% and lower mean HPWL on six of eight benchmarks. EGPlace with refinement has lower means on adaptec3 (55.39 versus 55.57) and bigblue2 (9.72 versus 10.10).

On bigblue3, EAPlace-VLM without continuous refinement reduces HPWL by 13.2% relative to EGPlace (53.03 to 46.02). EAPlace-VLM reduces HPWL by 25.1% relative to EGPlace (53.03 to 39.74) and by 14.2% relative to EGPlace equipped with the same refinement (46.33 to 39.74). These percentages use different baselines and are not additive contributions of separate modules. The results show that refinement benefits both search methods, while its gains depend on the input placement.

5.5 Efficiency and Trade-offs

The continuous refinement stage of EAPlace-VLM takes 3.25–24.24 minutes on average across benchmarks, below 2.5% of the 1000-minute search budget. The reported VLM latency is 5.36–8.08 seconds per invocation, but this per-call measurement alone does not determine the total VLM overhead (Appendix D.6). In the reported congestion comparison, peak RUDY for EAPlace-VLM is 1.00–1.32 times the value of its variant without continuous refinement (Appendix D.2). These entries illustrate that lower HPWL can coexist with higher peak congestion, but they do not establish the cause of the mixed-size outcomes in Table 2. Results for $K = 1$–8 and $N = 3$–7 appear in Appendices D.3 and D.4. Because the evaluation is limited to ISPD2005 and these placement metrics, it does not establish timing or detailed-routing gains.

6 Conclusion

EAPlace-VLM combines periodic VLM-guided macro selection with heuristic evolutionary search and legal greedy reinsertion. A topology-to-color representation supplies partial connectivity cues, while continuous physical refinement reduces whitespace induced by conservative grid quantization. EAPlace-VLM and its downstream DREAMPlace flow attain the best average ranks in the six-benchmark macro-only and mixed-size comparisons, respectively. Compared with EGPlace equipped with the same continuous refinement, EAPlace-VLM has lower mean HPWL on six of eight benchmarks, with an average per-benchmark reduction of approximately 3.1%. These results distinguish EAPlace-VLM's overall gains from matched-refinement comparisons, but they do not validate every VLM rationale or isolate the causal contribution of its visual input. Macro-level HPWL gains do not guarantee uniform downstream improvements, and refinement can increase peak congestion. Future work will examine richer connectivity representations and extend evaluation to timing-driven placement, 3D-IC floorplanning, and multi-die integration.

AI USE STATEMENT

During manuscript preparation and revision, generative AI tools assisted with language polishing, grammar correction and LaTeX formatting. They were not used to fabricate experimental measurements or replace the recorded VLM response. The authors are responsible for verifying the final technical content, analyses, and claims.

REPRODUCIBILITY STATEMENT

The main text describes the representation, selection–insertion interface, refinement acceptance rule, and evaluation protocol. Appendices A and C provide the prompt, a recorded response, formal definitions, full pseudocode, and refinement details. Appendix D reports sensitivity, congestion, and invocation/runtime measurements.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*, 2023.

Oguzhan Alagoz, Heather Hsu, Andrew J Schaefer, and Mark S Roberts. Markov Decision Processes: A Tool for Sequential Decision Making under Uncertainty. *Medical Decision Making*, 30 (4):474–483, 2010.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-VL: A Versatile Vision-Language Model for Understanding, Localization, Text Reading, and Beyond. *arXiv preprint arXiv:2308.12966*, 2023.

Tung-Chieh Chen, Zhe-Wei Jiang, Tien-Chang Hsu, Hsin-Chen Chen, and Yao-Wen Chang. NTUplace3: An Analytical Placer for Large-Scale Mixed-Size Designs with Preplaced Blocks and Density Constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(7):1228–1240, 2008.

Chung-Kuan Cheng, Andrew B. Kahng, Ilgweon Kang, and Lutong Wang. RePlAce: Advancing Solution Quality and Routability Validation in Global Placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 38(9):1717–1730, 2018.

Ruoyu Cheng and Junchi Yan. On Joint Learning for Solving Placement and Routing in Chip Design. *Advances in Neural Information Processing Systems*, 34:16508–16519, 2021.

James P. Cohoon and William D. Paris. Genetic Placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 6(6):956–964, 1987.

Ji Deng, Zhao Li, Ji Zhang, and Jun Gao. EGPlace: An Efficient Macro Placement Method via Evolutionary Search with Greedy Repositioning Guided Mutation. In *International Conference on Machine Learning*, pp. 13237–13255, 2025.

Yonggan Fu, Yongan Zhang, Zhongzhi Yu, Sixu Li, Zhifan Ye, Chaojian Li, Cheng Wan, and Yingyan Celine Lin. GPT4AIGChip: Towards Next-Generation AI Accelerator Design Automation via Large Language Models. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 1–9, 2023.

Zijie Geng, Jie Wang, Ziyan Liu, Siyuan Xu, Zhentao Tang, Mingxuan Yuan, Jianye Hao, Yongdong Zhang, and Feng Wu. Reinforcement Learning within Tree Search for Fast Macro Placement. In *International Conference on Machine Learning*, 2024.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *ACM Transactions on Information Systems*, 43(2):1–55, 2025.

Andrew B. Kahng, Jens Lienig, Igor L. Markov, and Jin Hu. *VLSI Physical Design: From Graph Partitioning to Timing Closure*, volume 312. Springer, 2011.

Myung-Chul Kim and Igor L. Markov. ComPLx: A Competitive Primal-Dual Lagrange Optimization for Global Placement. In *Proceedings of the 49th Annual Design Automation Conference*, pp. 747–752, 2012.

Yao Lai, Yao Mu, and Ping Luo. MaskPlace: Fast Chip Placement via Reinforced Visual Representation Learning. *Advances in Neural Information Processing Systems*, 35:24019–24030, 2022.

Yao Lai, Jinxin Liu, Zhentao Tang, Bin Wang, Jianye Hao, and Ping Luo. ChiPFormer: Transferable Chip Placement via Offline Decision Transformer. In *International Conference on Machine Learning*, pp. 18346–18364, 2023.

Yibo Lin, Shounak Dhar, Wuxi Li, Haoxing Ren, Brucek Khailany, and David Z. Pan. DREAMPlace: Deep Learning Toolkit-Enabled GPU Acceleration for Modern VLSI Placement. In *Proceedings of the 56th Annual Design Automation Conference*, pp. 1–6, 2019.

Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of Heuristics: Towards Efficient Automatic Algorithm Design Using Large Language Model. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32201–32223, 2024a.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual Instruction Tuning. *Advances in Neural Information Processing Systems*, 36:34892–34916, 2023a.

Jiaxi Liu, Xufeng Yao, Yibo Lin, and Bei Yu. Large Language Models for Electronic Design Automation: A Comprehensive Survey. *arXiv preprint arXiv:2312.10200*, 2023b.

Jing Liu, Weicai Zhong, Licheng Jiao, and Xue Li. Moving Block Sequence and Organizational Evolutionary Algorithm for General Floorplanning with Arbitrarily Shaped Rectilinear Blocks. *IEEE Transactions on Evolutionary Computation*, 12(5):630–646, 2008.

Mingjie Liu, Teodor-Dumitru Ene, Robert Kirby, Chris Cheng, Nathaniel Pinckney, Rongjian Liang, Jonah Alben, Himyanshu Anand, Sanmitra Banerjee, Ismet Bayraktaroglu, et al. ChipNeMo: Domain-Adapted LLMs for Chip Design. *arXiv preprint arXiv:2311.00176*, 2023c.

Mingjie Liu, Nathaniel Pinckney, Brucek Khailany, and Haoxing Ren. VerilogEval: Evaluating Large Language Models for Verilog Code Generation. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 1–8, 2023d.

Shang Liu, Wenji Fang, Yao Lu, Qijun Zhang, Hongce Zhang, and Zhiyao Xie. RTLcoder: Outperforming GPT-3.5 in Design RTL Generation with Our Open-Source Dataset and Lightweight Solution. In *2024 IEEE LLM Aided Design Workshop (LAD)*, pp. 1–5, 2024b.

Tianyi Liu, Yaxin Xu, Lin Geng, Ningzhong Liu, Han Sun, and Yu Wang. RSPlace: Rotation sensing macro placement via bidirectional tree expansion. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(28):23837–23845, 2026a. doi: 10.1609/aaai.v40i28.39559.

Tianyi Liu, Yaxin Xu, Ningzhong Liu, Han Sun, and Shanding Xu. Rotation Mutation-Driven Evolutionary Algorithm for Efficient Macro Placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 45(1):232–245, 2026b. URL https://ieeexplore.ieee.org/document/11037516.

Jingwei Lu, Pengwen Chen, Chin-Chih Chang, Lu Sha, Dennis Jen-Hsin Huang, Chin-Chi Teng, and Chung-Kuan Cheng. ePlace: Electrostatics-Based Placement Using Fast Fourier Transform and Nesterov's Method. *ACM Transactions on Design Automation of Electronic Systems*, 20(2): 1–34, 2015.

Don MacMillen, Raul Camposano, Dwight Hill, and Thomas W. Williams. An Industrial View of Electronic Design Automation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(12):1428–1448, 2000.

Somdeb Majumdar, Uday Mallappa, and Hesham Mostafa. AI Alone Isn't Ready for Chip Design: A Combination of Classical Search and Machine Learning May Be the Way Forward. *IEEE Spectrum*, 61(12):38–43, 2024.

Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nova, et al. A Graph Placement Methodology for Fast Chip Design. *Nature*, 594(7862):207–212, 2021.

Shibing Mo, Jing Liu, Jianchu Xu, and Ruilin Wu. AutoFloorplan: Evolving Heuristics for Chip Floorplanning with Large Language Models and Textual Gradient-Guided Repair, 2025a.

Shibing Mo, Kai Wu, Qixuan Gao, Xiangyi Teng, and Jing Liu. AutoSGNN: Automatic Propagation Mechanism Discovery for Spectral Graph Neural Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 19493–19502, 2025b.

Shibing Mo, Jing Liu, Jianchu Xu, and Ruilin Wu. Order Matters: Unveiling the Hidden Impact of Macro Placement Sequences via Proxy-Guided LLM Evolution. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2606.08904.

Hiroshi Murata, Kunihiro Fujiyoshi, Shigetoshi Nakatake, and Yoji Kajitani. VLSI Module Placement Based on Rectangle-Packing by the Sequence-Pair. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(12):1518–1524, 1996.

Gi-Joon Nam, Charles J. Alpert, Paul Villarrubia, Bruce Winter, and Mehmet Yildiz. The ISPD2005 Placement Contest and Benchmark Suite. In *Proceedings of the 2005 International Symposium on Physical Design*, pp. 216–220, 2005.

Stéphane Ross and Drew Bagnell. Efficient Reductions for Imitation Learning. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 661–668, 2010.

Dipayan Saha, Shams Tarek, Katayoon Yahyaei, Sujan Kumar Saha, Jingbo Zhou, Mark Tehranipoor, and Farimah Farahmandi. LLM for SoC Security: A Paradigm Shift. *IEEE Access*, 12: 155498–155521, 2024.

Yunqi Shi, Ke Xue, Song Lei, and Chao Qian. Macro Placement by Wire-Mask-Guided Black-Box Optimization. *Advances in Neural Information Processing Systems*, 36:6825–6843, 2023.

Shailja Thakur, Baleegh Ahmad, Zhenxing Fan, Hammond Pearce, Benjamin Tan, Ramesh Karri, Brendan Dolan-Gavitt, and Siddharth Garg. Benchmarking Large Language Models for Automated Verilog RTL Code Generation. *arXiv preprint arXiv:2212.11140*, 2023.

Laung-Terng Wang, Yao-Wen Chang, and Kwang-Ting Tim Cheng. *Electronic Design Automation: Synthesis, Verification, and Test*. Morgan Kaufmann, 2009.

Haoyuan Wu, Zhuolun He, Xinyun Zhang, Xufeng Yao, Su Zheng, Haisheng Zheng, and Bei Yu. ChatEDA: A Large Language Model Powered Autonomous Agent for EDA. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 43(10):3184–3197, 2024.

Xingyu Wu, Sheng-hao Wu, Jibin Wu, Liang Feng, and Kay Chen Tan. Evolutionary Computation in the Era of Large Language Model: Survey and Roadmap. *IEEE Transactions on Evolutionary Computation*, 29(2):534–554, 2025.

Ke Xue, Ruo-Tong Chen, Rong-Xi Tan, Xi Lin, Yunqi Shi, Siyuan Xu, Mingxuan Yuan, and Chao Qian. BBOPlace-Bench: Benchmarking Black-Box Optimization for Chip Placement. *arXiv preprint arXiv:2510.23472*, 2025.

Xufeng Yao, Jiaxi Jiang, Yuxuan Zhao, Peiyu Liao, Yibo Lin, and Bei Yu. Evolution of Optimization Algorithms for Global Placement via Large Language Models. *arXiv preprint arXiv:2504.17801*, 2025.

He Yu and Jing Liu. Automatically Optimizing Heuristics for Robust Scale-Free Network Design via Large Language Models. *Scientific Reports*, 15(1):41030, 2025.

He Yu and Jing Liu. Exploring Automated Algorithm Design Synergizing Large Language Models and Evolutionary Algorithms: Survey and Insights. *Evolutionary Computation*, 2026.

A VLM Prompt and Response Example

This appendix provides a representative example of the VLM interaction used in the **VLM-Guided Topological Diagnosis**. The example is based on adaptec1 and preserves an actual VLM response. The VLM receives a rendered macro placement image together with structured macro information, and it returns a replace-set of macro identifiers for subsequent reinsertion.

A.1 Rendered Placement Input

Figure 4 shows an example rendered placement used as the visual input to the VLM. Each macro is drawn as a rectangle on the grid canvas. Macro colors encode dominant-net connectivity, and the grid mesh provides spatial reference for estimating relative positions and distances.

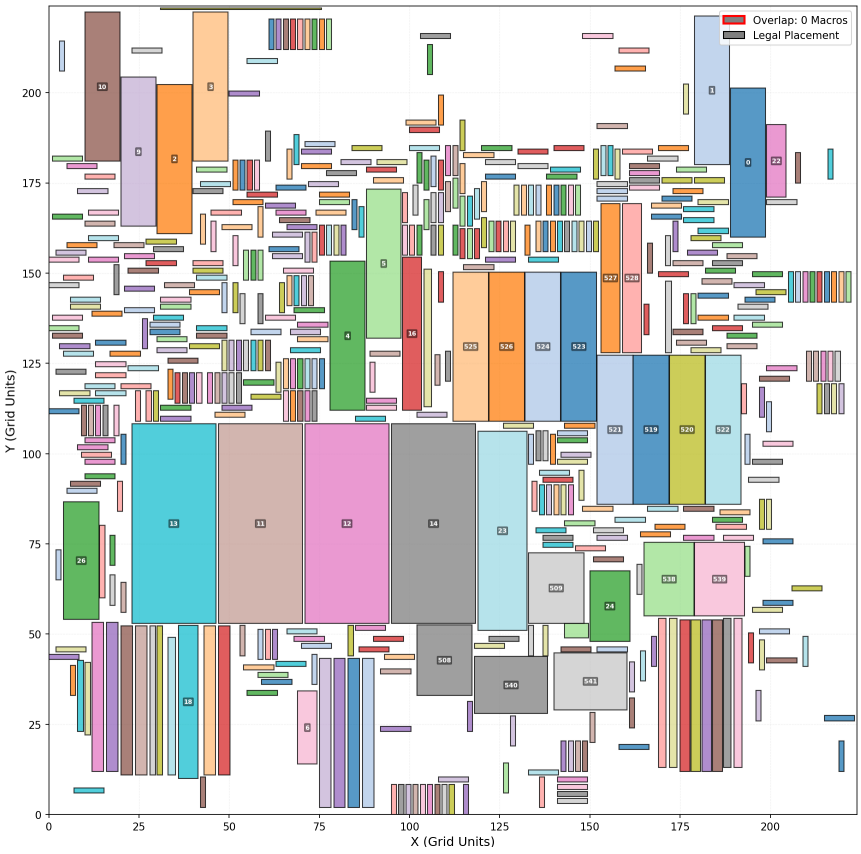

Figure 4: Rendered placement input for VLM-guided topological diagnosis on adaptec1. Macro colors are determined by dominant incident nets, while the grid mesh provides spatial reference.

A.2 Prompt Template

The prompt instructs the VLM to analyze the rendered placement and select up to K macros for replacement. The VLM is required to return valid macro identifiers rather than continuous coordinates. The template is shown below.

> You are an expert in chip physical design and macro placement. Given the current macro placement image and structured macro information, select at most K macros that should be removed and reinserted, so that the placement quality can be improved in terms of wirelength, congestion, and overlap.
>
> **Image encoding rules.** The image is a top-down grid-level macro placement. The white background indicates unoccupied placement space. Light gray grid lines provide spatial

reference. Each macro is drawn as a filled rectangle with a thin boundary. The position and size of each rectangle correspond to its grid-level coordinate and quantized macro size.

The color of a macro is determined by its dominant incident net. Macros with the same color are likely to be strongly correlated by net-level connectivity and are expected to be placed close to each other to reduce HPWL. If macros with the same or similar connectivity are spatially separated, they may contribute to large wirelength and should be considered as replacement candidates.

Structured macro information. The structured information contains the grid size, the number of placed macros, the replacement budget K, and a macro map. The macro map records each macro identifier and its grid-level rectangle:

$$i \mapsto [x_i, y_i, w_i, h_i].$$

Output requirement. Return only valid macro identifiers from the provided macro map. Do not output continuous coordinates. The number of selected macros must not exceed K.

The response should contain:

$$\texttt{macros}, \quad \texttt{prob}, \quad \texttt{rationale},$$

where `macros` is the selected replace-set, `prob` is the confidence score, and `rationale` briefly explains the selection based on color grouping, spatial dispersion, and macro positions.

A.3 Example VLM Response on Adaptec1

For the adaptec1 example in Figure 4, the replacement budget is set to $K = 6$. The VLM returns the following response.

VLM Response.

$$\texttt{prob} = 0.870$$

$$\texttt{macros} = [11, 12, 13, 14, 23, 519]$$

Rationale. Macros 11, 12, 13, 14, and 23 form a same-color macro group around the central placement region, while macro 519 is spatially separated from this group. This separation indicates potential topological dispersion among macros with strong net-level correlation. Reconsidering these macros for reinsertion may improve the compactness of the corresponding macro group and reduce HPWL.

The selected macro identifiers are then used as the replace-set

$$\mathcal{R}_t = \{m_{11}, m_{12}, m_{13}, m_{14}, m_{23}, m_{519}\}.$$

Interpreting color and overlapping net membership. The response above is reproduced without altering its macro IDs, confidence, or rationale. The selected macros share a net, but each macro may also belong to other nets. Rendering assigns each macro the color of its dominant incident net, and the shared net need not be dominant for every selected macro. Consequently, macros 11, 12, 13, 14, and 23 appear in different colors in Figure 4 despite their shared connectivity. The phrase "same-color macro group" therefore does not match the displayed colors. Their shared connectivity neither validates this color description nor establishes how the model arrived at its selection. The example illustrates an actual replace-set proposal and the information loss of dominant-net coloring, rather than error-free visual diagnosis. Deterministic reinsertion handles coordinates and geometric legality, but it does not verify the rationale or guarantee an optimal replace-set.

B BENCHMARK DISCUSSION

Although the ISPD2005 benchmark is relatively old, we believe it remains a suitable and meaningful benchmark for evaluating macro placement methods. A key reason is that ISPD2005 explicitly specifies macros in its design files, which is essential for studying macro placement in a rigorous and unambiguous manner. In contrast, many publicly available benchmarks do not provide explicit macro annotations, making it difficult to define the macro placement problem consistently. Moreover, ISPD2005 contains 543–1,329 macros per design, which still makes it one of the largest publicly available macro-level benchmarks. While more recent datasets such as ICCAD 2015/Superblue provide large-scale circuits, they do not explicitly identify macros. Existing studies therefore often rely on heuristic macro selection strategies, such as sorting cells by area and selecting the top-N cells. However, such heuristics may compromise the diversity and authenticity of the macro placement setting, and thus may not faithfully reflect the impact of macro placement order. For these reasons, ISPD2005 has continued to be widely adopted in recent studies and benchmarking efforts, including EGPlace (Deng et al., 2025), RSPlace (Liu et al., 2026a), BBOPlace-Bench (Xue et al., 2025), and EA-Rotation (Liu et al., 2026b). This continued usage suggests that, despite its age, ISPD2005 remains a relevant and valuable benchmark for macro placement research.

C ADDITIONAL DEFINITIONS AND IMPLEMENTATION DETAILS

C.1 FULL PROBLEM FORMULATION AND NOTATION

This appendix collects the complete physical and grid-domain definitions used by EAPlace-VLM. Table 5 summarizes the main mathematical notation used throughout the paper.

Table 5: Main mathematical notation.

Symbol	Description
$\mathcal{M}, m_i, N_m$	Macro set, the i-th macro, and the number of macros.
$\mathcal{E}, e, \mathcal{O}$	Net set, a single net, and the set of fixed I/O ports.
$\mathcal{V}(e), \mathcal{N}(m_i)$	Pins/nodes incident to net e, and nets incident to macro m_i.
$\mathcal{D}^{\mathrm{c}}$	Continuous physical canvas, $\mathcal{D}^{\mathrm{c}} = [0, W^{\mathrm{c}}] \times [0, H^{\mathrm{c}}]$.
$\mathcal{D}, G_x, G_y$	Discrete grid canvas and its horizontal/vertical resolutions.
r_x, r_y	Physical grid-cell sizes, $r_x = W^{\mathrm{c}}/G_x$ and $r_y = H^{\mathrm{c}}/G_y$.
$(w_i^{\mathrm{c}}, h_i^{\mathrm{c}}), (w_i, h_i)$	True physical size and quantized grid size of macro m_i.
$\mathbf{p}_i^{\mathrm{c}}, \mathbf{p}_i$	Continuous physical coordinate and discrete grid coordinate of macro m_i.
$\mathbf{P}^{\mathrm{c}}, \mathbf{P}$	Complete continuous placement and complete grid-level placement.
$\mathbf{P}^{\star}, \mathbf{P}^{\mathrm{c}0}, \mathbf{P}^{\mathrm{c}\star}$	Best grid-level placement, projected continuous placement, and final refined placement.
$R_i^{\mathrm{c}}(\cdot), R_i(\cdot)$	Continuous and grid-level rectangles occupied by macro m_i.
$\Pi(\cdot)$	Projection from a grid-level placement to a continuous physical placement.
$V_i(\mathbf{p}_i), W_i(\mathbf{p}_i)$	PositionMask and WireMask for candidate legality and HPWL evaluation.
$\mathcal{Q}, \mathcal{R}_t$	Evolutionary population and replace-set selected at iteration t.
$N, K, \tau, \mathcal{T}$	Population size, relocation budget, VLM invocation period, and search time budget.
$\Delta\mathbf{p}^{\mathrm{c}}, \mathbf{p}_i^{\mathrm{c}+}$	Continuous displacement and tentative continuous position of macro m_i.
$\rho, \rho_0, \gamma, s_{\mathrm{c}}$	Search radius, initial radius, radius decay factor, and continuous search step.
$\Omega(\mathbf{p}_i^{\mathrm{c}}), B(\mathbf{p}_i^{\mathrm{c}})$	Continuous legality penalty and boundary-violation penalty.
$\Delta\mathrm{HPWL}_i^{\mathrm{c}}, \lambda, \varepsilon$	Physical HPWL change, boundary penalty weight, and legality tolerance.

C.1.1 CONTINUOUS MACRO PLACEMENT PROBLEM

A macro placement instance is represented as $I = (\mathcal{M}, \mathcal{E}, \mathcal{D}^{\mathrm{c}}, \mathcal{O})$, where $\mathcal{M} = \{m_1, m_2, \ldots, m_{N_m}\}$ is the macro set and $\mathcal{D}^{\mathrm{c}}$ is the continuous physical chip canvas. Each macro $m_i \in \mathcal{M}$ has fixed physical dimensions, with width w_i^{c}, height h_i^{c}, and area A_i^{c}. Connectivity is modeled by a hypergraph $\mathcal{E} = \{e_1, e_2, \ldots, e_{N_e}\}$, in which each net $e \in \mathcal{E}$ connects a subset of macros and fixed I/O ports in $\mathcal{O}$. We use $\mathcal{V}(e)$ to denote the pins or nodes incident to net e, and $\mathcal{N}(m_i)$ to denote the nets incident to macro m_i. The physical locations of fixed I/O ports are treated as constants during placement optimization. A continuous placement is written as $\mathbf{P}^{\mathrm{c}} = \{\mathbf{p}_i^{\mathrm{c}}\}_{i=1}^{N_m}$, where $\mathbf{p}_i^{\mathrm{c}} = (x_i^{\mathrm{c}}, y_i^{\mathrm{c}})$ gives the lower-left coordinate of macro m_i. The corresponding axis-aligned rectangle is

$$R_i^{\mathrm{c}}(\mathbf{p}_i^{\mathrm{c}}) = [x_i^{\mathrm{c}}, x_i^{\mathrm{c}} + w_i^{\mathrm{c}}) \times [y_i^{\mathrm{c}}, y_i^{\mathrm{c}} + h_i^{\mathrm{c}}). \tag{5}$$

The continuous macro placement problem seeks a physical placement that optimizes the placement objective subject to legality constraints:

$$R_i^{\mathrm{c}}(\mathbf{p}_i^{\mathrm{c}}) \cap R_j^{\mathrm{c}}(\mathbf{p}_j^{\mathrm{c}}) = \varnothing, \quad \forall i \neq j, \tag{6}$$

$$R_i^{\mathrm{c}}(\mathbf{p}_i^{\mathrm{c}}) \subseteq \mathcal{D}^{\mathrm{c}}, \quad \forall i. \tag{7}$$

Equation (6) prevents pairwise macro overlap, whereas Equation (7) ensures that every macro lies entirely within the chip boundary. These equations define the physical domain legality requirements that the final placement must satisfy.

C.1.2 GRID-LEVEL REPRESENTATION

The original placement problem is continuous, but searching directly in $\mathbb{R}^{2N_m}$ is too expensive for large macro placement instances. EAPlace-VLM therefore performs the main combinatorial search in a discrete grid representation and later refines the selected solution in the continuous physical domain.

The physical canvas $\mathcal{D}^{\mathrm{c}} = [0, W^{\mathrm{c}}] \times [0, H^{\mathrm{c}}]$ is divided into a $G_x \times G_y$ grid. The physical size of each grid cell is

$$r_x = \frac{W^{\mathrm{c}}}{G_x}, \qquad r_y = \frac{H^{\mathrm{c}}}{G_y}. \tag{8}$$

The discrete grid canvas is

$$\mathcal{D} = \{0, \ldots, G_x - 1\} \times \{0, \ldots, G_y - 1\}. \tag{9}$$

Each macro is conservatively mapped to integer grid dimensions:

$$w_i = \left\lceil \frac{w_i^{\mathrm{c}}}{r_x} \right\rceil, \qquad h_i = \left\lceil \frac{h_i^{\mathrm{c}}}{r_y} \right\rceil. \tag{10}$$

Because of the ceiling operation, the grid footprint of a macro is never smaller than its physical footprint. Thus, a placement that is legal under the quantized grid footprint does not create macro overlap after projection because of size underestimation.

In the grid domain, the coordinate of macro m_i is

$$\mathbf{p}_i = (x_i, y_i), \qquad x_i, y_i \in \mathbb{Z}, \tag{11}$$

where $\mathbf{p}_i$ denotes the lower-left grid coordinate. The grid rectangle occupied by m_i is

$$R_i(\mathbf{p}_i) = [x_i, x_i + w_i) \times [y_i, y_i + h_i), \tag{12}$$

with half-open intervals interpreted over grid cells. A complete grid-level placement is

$$\mathbf{P} = \{\mathbf{p}_i\}_{i=1}^{N_m}. \tag{13}$$

A grid-level placement is legal when all quantized macro rectangles are non-overlapping and fully contained in the discrete canvas:

$$R_i(\mathbf{p}_i) \cap R_j(\mathbf{p}_j) = \varnothing, \quad \forall i \neq j, \tag{14}$$

$$R_i(\mathbf{p}_i) \subseteq \mathcal{D}, \quad \forall i. \tag{15}$$

Equivalently, each macro coordinate must satisfy

$$0 \le x_i \le G_x - w_i, \qquad 0 \le y_i \le G_y - h_i, \quad \forall i. \tag{16}$$

Unless stated otherwise, placement symbols without superscripts, including $\mathbf{P}$, $\mathbf{p}_i$, R_i, w_i, and h_i, refer to the grid domain. Continuous physical quantities are marked with a superscript, like $\mathbf{P}^{\mathrm{c}}$, $\mathbf{p}_i^{\mathrm{c}}$, R_i^{c}, w_i^{c}, and h_i^{c}.

C.1.3 Placement Quality Metric

We use HPWL as the main proxy for placement quality. For a grid-level placement $\mathbf{P}$, $\mathrm{HPWL}(\mathbf{P})$ is defined as

$$\begin{aligned}\mathrm{HPWL}(\mathbf{P}) = \sum_{e\in\mathcal{E}} \Big[&\big(\max_{m_i\in e} x_i - \min_{m_i\in e} x_i\big) \\ &+ \big(\max_{m_i\in e} y_i - \min_{m_i\in e} y_i\big)\Big]\end{aligned} \tag{17}$$

For a continuous placement $\mathbf{P}^{\mathrm{c}}$, $\mathrm{HPWL}(\mathbf{P}^{\mathrm{c}})$ is defined in the same way by replacing the grid coordinates (x_i, y_i) in Equation (17) with the physical coordinates $(x_i^{\mathrm{c}}, y_i^{\mathrm{c}})$. Grid-level HPWL guides the combinatorial search, while physical HPWL is used to evaluate the final placement.

C.1.4 Masks for Candidate Evaluation

EAPlace-VLM uses two grid-level masks to evaluate candidate macro locations: PositionMask and WireMask. The PositionMask V_i is a binary mask that records whether a candidate coordinate is geometrically feasible. A coordinate is feasible only when placing $R_i(\mathbf{p}_i)$ there keeps the macro inside the grid canvas and avoids overlap with all currently fixed macros.

The WireMask W_i assigns a wirelength cost to each candidate coordinate. Following MaskPlace, $W_i(\mathbf{p}_i)$ records the HPWL increment caused by placing m_i at $\mathbf{p}_i$. Since the placement of m_i can affect only its incident nets, this cost is evaluated over $\mathcal{N}(m_i)$.

The PositionMask removes infeasible locations, and the WireMask orders the feasible ones by wirelength. The selected candidate is therefore

$$\mathbf{p}_i^{\star} = \arg \min_{\mathbf{p}_i:\, V_i(\mathbf{p}_i)=1} W_i(\mathbf{p}_i). \tag{18}$$

Together, the two masks provide the grid-level evaluation primitives used by the insertion strategy in the evolutionary search.

C.1.5 Grid-to-Continuous Projection and Quantization Gap

After grid-level search, a discrete placement $\mathbf{P}$ is projected to the continuous domain to obtain an initial physical placement $\mathbf{P}^{\mathrm{c}0}$:

$$\mathbf{P}^{\mathrm{c}0} = \Pi(\mathbf{P}), \tag{19}$$

where $\Pi(\cdot)$ scales grid coordinates by the cell sizes r_x and r_y and restores the physical macro dimensions w_i^{c} and h_i^{c}.

The ceiling quantization in Equation (10) makes each macro's grid footprint at least as large as its physical footprint. For example, if two macros are horizontally adjacent in the grid domain, the physical gap after projection is

$$r_x w_i - w_i^{\mathrm{c}} \ge 0. \tag{20}$$

This gap is not required by physical design rules. It is introduced by conservative grid quantization and may enlarge net bounding boxes, thereby degrading HPWL. This observation motivates the grid-free continuous refinement described later.

C.2 Complete Evolutionary Procedure

Algorithm 2 expands the compact procedure in the main text. In each initialization round, macros are greedily inserted into an empty canvas in a random order. The resulting legal placement is then added to the population, which is pruned to at most N candidates. During search, a parent is selected

Algorithm 2: Overall procedure of EAPlace-VLM.

Input : $\mathcal{M}, \mathcal{E}, \mathcal{D}, \Pi$, population size N, initialization rounds T_{init}, VLM period τ, replace budget K, and time budget $\mathcal{T}$
Output: final continuous placement $\mathbf{P}^{\text{c}\star}$

▷ Part A: Initialization
$\mathcal{Q} \leftarrow \varnothing$
for $r = 1$ **to** T_{init} **do**
  $\sigma_r \leftarrow \textsc{RandomPermutation}(\mathcal{M})$
  $\mathbf{P}_0^{(r)} \leftarrow \textsc{GreedyInsert}(\varnothing, \sigma_r)$
  $\mathcal{Q} \leftarrow \textsc{Prune}(\mathcal{Q} \cup \{\mathbf{P}_0^{(r)}\}, N)$
end for
▷ Part B: Dual-engine macro repositioning
$t \leftarrow 0$
while $\textsc{ElapsedTime} < \mathcal{T}$ **do**
  $\mathbf{P}_t \leftarrow \textsc{SoftmaxSelect}(\mathcal{Q})$
  if $t \bmod \tau = 0$ **then**
    $\mathcal{R}_t \leftarrow \textsc{VLMSelect}(\mathbf{P}_t, K)$
    if $\mathcal{R}_t = \varnothing$ **then**
      $\mathcal{R}_t \leftarrow \textsc{HeuristicSelect}(\mathbf{P}_t)$
    end if
  else
    $\mathcal{R}_t \leftarrow \textsc{HeuristicSelect}(\mathbf{P}_t)$
  end if
  $\mathbf{P}_{\text{fix}} \leftarrow \mathbf{P}_t|_{\mathcal{M}\setminus\mathcal{R}_t}$
  $\sigma_t \leftarrow \textsc{SortByArea}(\mathcal{R}_t)$
  $\mathbf{P}_{t+1} \leftarrow \textsc{GreedyInsert}(\mathbf{P}_{\text{fix}}, \sigma_t)$
  $\mathcal{Q} \leftarrow \textsc{Prune}(\mathcal{Q} \cup \{\mathbf{P}_{t+1}\}, N)$
  $t \leftarrow t + 1$
end while
▷ Part C: Continuous physical refinement
$\mathbf{P}^{\star} \leftarrow \arg\min_{\mathbf{P}\in\mathcal{Q}} \text{HPWL}(\mathbf{P})$
$\mathbf{P}^{\text{c}0} \leftarrow \Pi(\mathbf{P}^{\star})$
$\mathbf{P}^{\text{c}\star} \leftarrow \textsc{ContinuousRefine}(\mathbf{P}^{\text{c}0})$
return $\mathbf{P}^{\text{c}\star}$

by SOFTMAXSELECT. The scheduled branch proposes the replace-set, with heuristic fallback for an invalid VLM selection. Selected macros are removed and reinserted by descending physical area, and the child enters the pruned population.

The function names GREEDYINSERT, VLMSELECT, HEURISTICSELECT, and CONTINUOUSREFINE denote the insertion, diagnosis, heuristic selection, and physical refinement modules, respectively. After search, only the best grid-level placement is projected and refined. Sections C.3 and C.4 provide the operator details.

C.3 SELECTION AND INSERTION DETAILS

The main decision in each evolutionary iteration is the choice of the replace-set $\mathcal{R}_t$, namely the macros removed from the current placement $\mathbf{P}_t$ and reconsidered for insertion. This choice strongly affects the subsequent search trajectory. A fixed heuristic is fast, but it can miss global topological patterns that arise from earlier sequential placement decisions. EAPlace-VLM therefore uses an asymmetric dual-engine mechanism, in which a lightweight policy router schedules a frequent heuristic branch and a periodic VLM branch according to a deterministic temporal rule.

In most iterations, the scalar-driven heuristic provides efficient local evolution. For each macro m_i, this branch computes a priority score from normalized numerical indicators. Let $\tilde{h}_i$, $\tilde{u}_i$, and $\tilde{o}_i$ denote the normalized HPWL contribution, RUDY congestion score, and overlap penalty of m_i, respectively. The score is

$$s_i = \tilde{h}_i + w_{\text{r}}\tilde{u}_i + w_{\text{o}}\tilde{o}_i. \tag{21}$$

The sampling probability is obtained by a stabilized softmax:

$$q_i = \frac{\exp(s_i - \max_j s_j)}{\sum_k \exp(s_k - \max_j s_j)}. \tag{22}$$

Macros are sampled without replacement according to q_i to form the replace-set. This branch is inexpensive and serves as the default evolution engine.

Every τ iterations, EAPlace-VLM invokes VLM-guided topological diagnosis for more global relocation guidance. At iterations satisfying $t \bmod \tau = 0$, the current placement $\mathbf{P}_t$ is converted into the net-aware visual representation described in Section 4.2. The VLM analyzes the rendered placement together with structured macro information and returns a replace-set

$$\mathcal{R}_t = \{m_{i_1}, m_{i_2}, \ldots, m_{i_k}\}, \quad k \le K. \tag{23}$$

This branch is intended to complement the numerical heuristic by proposing macros that may contribute to global topological dispersion, such as spatially separated macros with strong net-level correlation.

The VLM branch outputs candidate macro identifiers rather than coordinates. The subsequent mask-guided greedy insertion strategy determines coordinates and enforces geometric legality, without validating the model's verbal rationale. When the VLM branch does not produce a valid replace-set, replace-set selection is instead performed by the scalar-driven heuristic.

C.3.1 Mask-Guided Greedy Insertion Strategy

The mask-guided greedy insertion strategy is the common placement operator used in both initialization and evolutionary search. Given an ordered macro sequence and a partial grid-level placement, it inserts macros sequentially while maintaining legality.

In the initialization, the partial placement is empty, and the insertion order is the randomized macro permutation σ_r. In evolutionary search, the partial placement is obtained by removing $\mathcal{R}_t$ from the parent placement $\mathbf{P}_t$ and keeping all other macros fixed. The removed macros are sorted by descending physical area A_i^{c} before reinsertion, so larger macros are placed earlier when spatial constraints are tighter.

For each macro m_i, the strategy evaluates all candidate grid coordinates $\mathbf{p}_i$ using the PositionMask and WireMask defined in Section C.1.4. The PositionMask filters candidates that violate boundary or non-overlap constraints, and the WireMask ranks feasible candidates by induced wirelength cost. The macro is inserted at the feasible coordinate selected by Equation (2). After each insertion, the grid occupancy and net-bounding information are updated before processing the next macro.

Because each macro is placed only at a coordinate allowed by the PositionMask, the placement remains legal throughout the sequential construction.

C.4 Continuous Refinement Details

As discussed in Section C.1.5, ceiling-based grid quantization can introduce artificial whitespace after a grid-level placement is projected to the continuous domain. Starting from $\mathbf{P}^{\mathrm{c}0} = \Pi(\mathbf{P}^{\star})$, the grid-free continuous refinement adjusts macro coordinates directly in the physical domain. Its goal is to reduce quantization-induced whitespace and physical HPWL while preserving legality.

The best grid-level placement $\mathbf{P}^{\star}$ from the evolutionary search provides the initial spatial structure. After projection, each macro recovers its physical dimensions w_i^{c} and h_i^{c}, and refinement operates on the continuous coordinates in $\mathbf{P}^{\mathrm{c}0}$. Unlike the grid-level search, the macro coordinates are no longer restricted to integer grid locations.

The refinement then applies a coarse-to-fine local search. For each macro m_i, it samples continuous displacement vectors $\Delta\mathbf{p}^{\mathrm{c}} = (\Delta x^{\mathrm{c}}, \Delta y^{\mathrm{c}})$ within a search radius ρ. A tentative position is

$$\mathbf{p}_i^{\mathrm{c}+} = \mathbf{p}_i^{\mathrm{c}} + \Delta\mathbf{p}^{\mathrm{c}}. \tag{24}$$

Displacement candidates are sampled on a sub-grid lattice with step

$$s_{\mathrm{c}} = \max\left(1, \frac{\min(r_x, r_y)}{5}\right). \tag{25}$$

The search radius is initialized as $\rho_0 = 2\max(r_x, r_y)$ and reduced after each pass:

$$\rho_{\ell+1} = \max(s_c, \gamma\rho_\ell), \qquad 0 < \gamma < 1. \tag{26}$$

This schedule first permits movements at the scale of a few grid cells and then concentrates on finer continuous adjustments.

For a tentative position $\mathbf{p}_i^c$, geometric legality is measured by a continuous penalty that combines macro overlap and boundary violation:

$$\Omega(\mathbf{p}_i^c) = \sum_{j \neq i} \left| R_i^c(\mathbf{p}_i^c) \cap R_j^c(\mathbf{p}_j^c) \right| + \lambda B(\mathbf{p}_i^c), \tag{27}$$

where the first term is the AABB intersection area with all other macros. The boundary penalty is

$$\begin{aligned} B(\mathbf{p}_i^c) =& (-x_i^c)_+ h_i^c + (-y_i^c)_+ w_i^c \\ &+ (x_i^c + w_i^c - W^c)_+ h_i^c + (y_i^c + h_i^c - H^c)_+ w_i^c. \end{aligned} \tag{28}$$

Here, $(\cdot)_+ = \max(\cdot, 0)$, and λ controls the boundary-violation penalty. For HPWL evaluation, moving m_i affects only the nets in $\mathcal{N}(m_i)$. Consistent with the grid-level WireMask, $\Delta\mathrm{HPWL}_i^c$ denotes the physical HPWL change over these affected nets after the tentative move $\mathbf{p}_i^c \to \mathbf{p}_i^{c+}$.

Candidate acceptance follows a legality-first rule. Let ε be a near-zero numerical tolerance. A tentative position $\mathbf{p}_i^{c+}$ is accepted if it strictly reduces the legality penalty, or if both the current and tentative positions are legal and the tentative move reduces HPWL over the affected nets:

$$\mathrm{Accept}(\mathbf{p}_i^{c+}) = \begin{cases} \text{True}, & \Omega(\mathbf{p}_i^{c+}) < \Omega(\mathbf{p}_i^c) - \varepsilon, \\ \text{True}, & \Omega(\mathbf{p}_i^c) \le \varepsilon, \quad \Omega(\mathbf{p}_i^{c+}) \le \varepsilon, \\ & \Delta\mathrm{HPWL}_i^c < -\varepsilon, \\ \text{False}, & \text{otherwise.} \end{cases} \tag{29}$$

Thus, if a macro is involved in residual overlap or boundary violation, refinement first seeks to improve legality. Once legality is satisfied, a move is accepted only when it preserves legality and reduces HPWL over the affected nets.

The refinement proceeds in passes over all macros. After each pass, the search radius is reduced according to Equation (26). The procedure terminates when a full pass accepts no displacement. With naive AABB overlap checking, each pass evaluates $(2\lfloor \rho/s_c \rfloor + 1)^2$ candidate displacements for each macro. The overlap evaluation cost is

$$\mathcal{O}\left(N_m^2 \left(2\left\lfloor \frac{\rho}{s_c} \right\rfloor + 1 \right)^2 \right), \tag{30}$$

and the HPWL-increment evaluation cost is

$$\mathcal{O}\left(N_m \left(2\left\lfloor \frac{\rho}{s_c} \right\rfloor + 1 \right)^2 \overline{|\mathcal{N}|}\, \overline{d}_e \right), \tag{31}$$

where $\overline{|\mathcal{N}|}$ is the average number of nets affected by moving one macro, and $\overline{d}_e$ is the average number of pins or nodes per net. Since ρ decreases geometrically, the number of candidate displacements decreases across passes. The final output is the continuous placement $\mathbf{P}^{c\star}$.

D Additional Experimental Results

D.1 Experimental Setup and Baseline References

Following prior work on macro placement (Deng et al., 2025; Lai et al., 2022; Geng et al., 2024), we evaluate EAPlace-VLM on the ISPD2005 benchmark suite (Nam et al., 2005). The baselines include the simulated-annealing method SP-SA (Murata et al., 1996), analytical placers NTUPlace3 (Chen et al., 2008), RePlace (Cheng et al., 2018), and DREAMPlace (Lin et al., 2019), learning-based placers GraphPlace (Mirhoseini et al., 2021), DeepPR (Cheng & Yan, 2021), MaskPlace (Lai et al., 2022), and ChiPFormer (Lai et al., 2023), as well as recent constructive or evolutionary methods WireMask-EA (Shi et al., 2023), EfficientPlace (Geng et al., 2024), and EGPlace (Deng et al., 2025).

All experiments were conducted on a workstation which is equipped with two Intel Xeon Platinum 8374C CPUs at 2.70 GHz and an NVIDIA RTX A6000 GPU. Unless otherwise specified, EAPlace-VLM uses a replace budget of $K = 6$, a population size of $N = 5$, and $T_{\text{init}} = 40$ randomized initialization rounds. For each benchmark instance, the evolutionary search is capped at 1000 minutes. EAPlace-VLM includes continuous physical refinement by default, and experiments that disable this stage are explicitly identified. In the mixed-size evaluation, DREAMPlace is run after EAPlace-VLM completes.

D.2 Congestion Evaluation

Congestion is also important for evaluating macro placement quality, because lower HPWL need not imply lower routing pressure. We use the original grid-based RUDY metric to evaluate peak congestion. For each benchmark, the reported peak RUDY value of EAPlace-VLM without continuous refinement is normalized to 1.00. The observations below refer to comparisons within Table 6 and are not pooled with the main HPWL summaries.

Among the entries in Table 6, EAPlace-VLM has the lowest reported HPWL on seven of eight benchmarks. Its peak RUDY values range from 1.00 to 1.32 relative to the entries for EAPlace-VLM without continuous refinement, including increases of 26% on adaptec4 and 32% on bigblue3. Thus, lower HPWL in this comparison does not imply uniformly lower congestion.

The methods exhibit different wirelength–congestion trade-offs. WireMask-EA has the lowest reported peak RUDY on four benchmarks despite higher HPWL than EAPlace-VLM on those instances. The table does not include EGPlace equipped with the same refinement, so it cannot compare congestion under matched refinement. Because the refiner's acceptance rule uses geometric legality and HPWL rather than a congestion objective, reduced whitespace is a possible explanation for higher routing pressure. However, the reported comparisons do not establish this mechanism or a causal link to the mixed-size outcomes in Table 2.

Table 6: Reported HPWL and original grid-based peak RUDY congestion. Congestion is normalized to EAPlace-VLM without continuous refinement = 1.00 for each benchmark. "Without refinement" denotes this ablation, whereas EAPlace-VLM denotes the complete method. Bold marks the lowest displayed value for each metric within this table.

	WireMask-EA		EGPlace		EAPlace-VLM (without refinement)		EAPlace-VLM	
Benchmark	HPWL	Cong.	HPWL	Cong.	HPWL	Cong.	HPWL	Cong.
adaptec1	5.94	**0.46**	5.81	1.01	5.79	1.00	**5.63**	1.04
adaptec2	53.01	**0.59**	37.80	1.44	38.82	1.00	**36.20**	1.01
adaptec3	59.79	**0.47**	60.56	1.65	58.92	1.00	**55.97**	1.13
adaptec4	59.93	1.73	56.08	1.57	54.33	**1.00**	**49.61**	1.26
bigblue1	2.13	1.53	2.26	1.06	2.19	**1.00**	**2.07**	**1.00**
bigblue2	196.84	25.55	**10.48**	37.89	12.77	**1.00**	11.87	**1.00**
bigblue3	62.70	**0.23**	45.39	1.03	45.55	1.00	**36.02**	1.32
bigblue4	133.61	12.09	62.00	5.06	62.48	**1.00**	**56.05**	1.01

D.3 Sensitivity to Relocation Budget

Table 7 shows how HPWL changes with the relocation budget when continuous refinement is disabled in EAPlace-VLM. The setting $K = 6$ gives the best result across the six benchmarks. A small K limits the amount of exploration introduced by each VLM intervention. A large K perturbs too many macros at once and may slow convergence. The relocation budget therefore needs to balance topological exploration and placement stability.

Table 7: Sensitivity to the VLM relocation budget K in EAPlace-VLM without continuous refinement (HPWL $\times 10^5$).

K	adaptec1	adaptec2	adaptec3	adaptec4	bigblue1	bigblue3
1	5.78 ± 0.02	36.51 ± 1.31	60.65 ± 0.88	56.38 ± 0.47	2.21 ± 0.01	49.79 ± 4.61
2	5.85 ± 0.12	36.45 ± 0.66	60.30 ± 1.25	55.58 ± 0.68	2.20 ± 0.00	49.57 ± 3.13
3	5.78 ± 0.06	37.67 ± 1.82	60.16 ± 1.27	56.69 ± 1.23	2.22 ± 0.01	48.65 ± 2.37
4	5.84 ± 0.10	38.45 ± 1.10	60.26 ± 0.76	55.95 ± 0.42	2.19 ± 0.01	53.42 ± 1.13
5	5.80 ± 0.05	37.59 ± 1.18	60.34 ± 2.39	55.69 ± 0.66	2.21 ± 0.00	49.11 ± 3.71
6	$\mathbf{5.77 \pm 0.03}$	$\mathbf{36.25 \pm 1.84}$	$\mathbf{58.63 \pm 0.26}$	$\mathbf{53.83 \pm 0.37}$	$\mathbf{2.18 \pm 0.01}$	$\mathbf{46.02 \pm 3.59}$
7	5.85 ± 0.12	36.78 ± 1.41	60.14 ± 1.13	56.40 ± 1.18	2.20 ± 0.01	48.94 ± 4.97
8	5.79 ± 0.03	36.56 ± 0.82	60.03 ± 2.09	55.51 ± 0.70	2.19 ± 0.02	48.33 ± 3.66

D.4 Sensitivity to Population Size

The population size controls how many candidate placements are maintained during evolutionary search. We evaluate this hyperparameter by varying the population size from 3 to 7, with the continuous refiner disabled to isolate its effect on grid-level search. As shown in Table 8, performance is relatively stable in the tested range, but an intermediate population size gives a better overall trade-off. A small population limits placement diversity and may weaken evolutionary exploration. A larger population keeps more candidates, but under a fixed time budget it can reduce the search effort spent on each trajectory. Setting $N = 5$ achieves the best or near-best HPWL on most benchmarks, whereas larger populations do not give consistent improvement. We therefore set the population size to $N = 5$ in the main experiments. This setting provides a practical balance between population diversity and search depth.

Table 8: Sensitivity to the evolutionary population size N in EAPlace-VLM without continuous refinement (HPWL $\times 10^5$).

N	adaptec1	adaptec2	adaptec3	adaptec4	bigblue1	bigblue3
3	5.76 ± 0.04	37.48 ± 0.33	59.36 ± 1.39	55.85 ± 0.12	2.20 ± 0.01	$\mathbf{45.49 \pm 3.64}$
4	5.73 ± 0.06	36.25 ± 2.06	59.01 ± 0.35	54.73 ± 0.89	2.21 ± 0.03	45.56 ± 2.71
5	5.77 ± 0.03	$\mathbf{36.25 \pm 1.84}$	$\mathbf{58.63 \pm 0.26}$	$\mathbf{53.83 \pm 0.37}$	2.18 ± 0.01	46.02 ± 3.59
6	$\mathbf{5.71 \pm 0.08}$	36.44 ± 0.80	58.99 ± 0.50	55.31 ± 1.15	2.19 ± 0.01	47.35 ± 4.32
7	5.76 ± 0.04	36.97 ± 2.01	60.20 ± 1.00	55.82 ± 0.80	$\mathbf{2.17 \pm 0.01}$	46.32 ± 3.07

D.5 Continuous Refiner Runtime

Since the continuous refinement stage of EAPlace-VLM runs only after the evolutionary search terminates, we report its runtime separately in Table 9. Under the 1000-minute evolutionary search budget, the refiner takes only a few minutes on most benchmarks and remains below 25 minutes even on the largest case.

These results show that the continuous refiner adds limited post-processing overhead compared with the main evolutionary search. The refinement stage can therefore improve final physical placement quality without becoming the dominant computational cost.

Table 9: Runtime of the continuous refinement stage of EAPlace-VLM under a 1000-minute evolutionary search budget.

Benchmark	Search budget (min)	Refinement time (min)
adaptec1	1000.00	3.86 ± 0.09
adaptec2	1000.00	3.25 ± 0.05
adaptec3	1000.00	5.39 ± 0.09
adaptec4	1000.00	16.13 ± 0.17
bigblue1	1000.00	5.12 ± 0.28
bigblue2	1000.00	9.58 ± 0.05
bigblue3	1000.00	24.24 ± 0.46
bigblue4	1000.00	16.92 ± 0.50

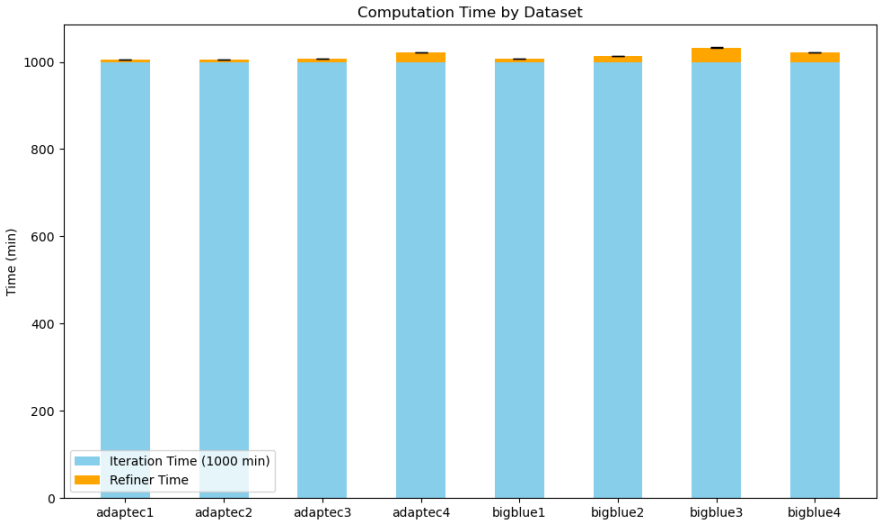

Figure 5: Distribution of computational time consumption. The stacked bars compare refinement time with the 1000-minute evolutionary search budget.

D.6 VLM Invocation Cost

Table 10 reports the per-invocation cost of VLM-guided diagnosis under the default relocation budget $K = 6$. The VLM input consists of the rendered placement and the structured macro information described in Section 4.2. The output is restricted to at most six macro identifiers and a short rationale.

The average total token usage is about 500 tokens per invocation, and the latency ranges from 5.36 to 8.08 seconds across the tested benchmarks. Because the VLM branch is invoked only every τ iterations rather than at every evolutionary step, this cost is amortized over the scalar-driven search. These measurements characterize a single invocation, whereas the total VLM overhead also depends on the number of calls and the timing protocol.

Table 10: Average per-invocation VLM token usage and latency.

Benchmark	Prompt tokens	Completion tokens	Total tokens	Latency (s)
adaptec1	337	171	508	5.36
adaptec2	337	162	499	6.82
adaptec3	337	178	515	5.75
adaptec4	337	162	499	5.62
bigblue1	337	168	505	5.96
bigblue2	337	163	500	8.08
bigblue3	337	170	507	5.44
bigblue4	337	170	507	5.63

D.7 Qualitative Diagnostic Visualization

Figure 6 preserves the qualitative diagnostic heatmaps on eight benchmarks. Highlighted regions are spatially concentrated, including areas containing compact macro arrangements. This visualization is descriptive: it is not used to measure placement quality or to establish which internal model features caused a selection. In particular, dominant-net colors encode only part of the connectivity, and different colors do not exclude a shared net (Appendix A). Quantitative comparisons of the VLM variants and refinement are reported in Tables 3 and 4.

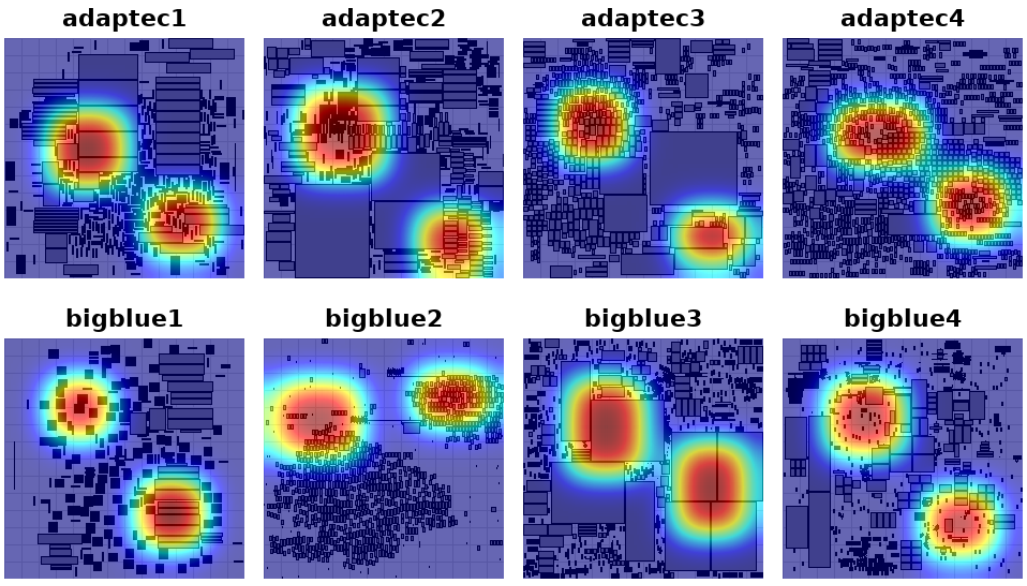

Figure 6: Qualitative diagnostic heatmaps on ISPD2005. The original visualizations are retained as descriptive illustrations, not quantitative evidence of internal model attention or causal reasoning.