ChipMASRAG: 基于多智能体协作与 RAG 检索的
超大规模芯片布局优化框架

孙可钦 [†*], 杨秋松 [†], 李明树 [†]
Email: sunkeqin11@mails.ucas.edu.cn, qiusong@iscas.ac.cn, mingshu@admin.iscas.ac.cn
[†] 中国科学院软件研究所, 北京, 中国
[*] 中国科学院大学, 北京, 中国

摘要—超大规模芯片设计包含数十万乃至百万级模块，传统布局方法面临计算复杂度爆炸和知识复用不足的双重挑战。本文提出 ChipMASRAG 框架，首次将多智能体协作机制与 RAG 知识检索技术相结合，实现知识驱动的多智能体层次化布局优化。核心创新包括：(1) 多智能体协作架构，每个分区对应一个智能体，通过协调者统一管理；(2)RAG 增强的知识检索机制，协调者统一检索历史案例，结果共享给所有分区智能体；(3) 知识驱动的边界协商协议，智能体间协商时参考历史案例的协商模式；(4) 双重规模无关机制，通过层次化分解降低问题规模，通过多智能体并行实现训练规模无关。在 ISPD 2015 基准测试的 12 个设计上（28K-1.2M 组件），ChipMASRAG 实现 81.25% 成功率，知识库命中时加速约 63 倍，平均子问题规模降至原始规模的 1/35。相比 ChipDRAG 和 DREAMPlace，在规模导向和平衡型评价中全面胜出，验证了多智能体协作与 RAG 检索结合的有效性。

***Index Terms*—芯片布局, 多智能体系统, RAG 检索, 知识驱动优化, 层次化分解**

I. 引言

A. 研究动机

现代超大规模集成电路（VLSI）设计包含数十万至百万级模块，芯片布局作为物理设计的关键步骤，直接影响芯片的性能、功耗和制造成本 [?], [?]。Kahng [?] 和 Huang 等 [?] 系统综述了机器学习在 EDA 和物理设计中的应用。传统布局方法面临两个根本性挑战：首先，计算复杂度随设计规模指数增长，状态空间为 $O(|S|^{|M|})$，数十万模块时计算实际上不可行；其次，现有方法缺乏历史知识复用机制，每次优化都从头开始，无法利用积累的设计经验。

现有方法主要分为三类：分析式优化方法如 DREAMPlace [?]、RePlAce [?] 通过求解非凸优化问题保证解质量，但计算复杂度超线性增长，难以扩展到百万级模块；层次化分解方法通过分治降低复杂度 [?]，但静态分区策略导致显著的边界代价，且缺乏知识复用机制；知识检索方法通过 RAG 技术复用历史策略 [?], [?], [?]，但主要关注参数生成，未解决超大规模设计的可扩展性问题。

多智能体系统（MAS）为芯片布局提供了新的建模视角。在 MAS 框架下，每个分区被建模为独立的智能体，具有局部自主性来优化其内部布局。智能体之间通过通信机制实现动态协作，能够协商边界模块的分配以减少跨边界连接。然而，现有多智能体方法缺乏历史知识复用，无法利用积累的设计经验加速优化过程。

这一现状揭示了芯片布局领域的核心挑战：如何在保持计算可扩展性的同时，充分利用历史知识加速优化并提升解质量？本文提出将多智能体协作机制与 RAG 知识检索技术相结合，实现知识驱动的多智能体层次化布局优化。

B. 核心思路与贡献

本文提出 ChipMASRAG 框架，将芯片布局建模为知识驱动的多智能体协作优化问题。核心思路是：通过层次化分解将超大规模设计分解为可管理的子问题，每个子问题对应一个分区智能体；协调者智能体统一执行 RAG 检索，从知识库中检索相似历史案例，结果共享给所有分区智能体；分区智能体通过 RAG 检索复用

历史策略，通过边界协商协议优化边界模块分配；对于 RAG 未命中的新设计，采用动态规划作为兜底方案保证解质量下界。

本文的主要贡献包括四个方面。首先，提出首个将多智能体协作与 RAG 检索结合的芯片布局框架，将问题形式化为知识驱动的多智能体马尔可夫博弈。其次，设计 RAG 增强的多智能体状态空间和奖励函数，使智能体能够利用历史知识指导优化决策。第三，提出知识驱动的边界协商协议，智能体间协商时参考历史案例的协商模式，提升边界优化效果。第四，实现双重规模无关机制，通过层次化分解降低问题规模（平均 1/35），通过多智能体并行实现训练规模无关，使超大规模设计从不可计算变为稳定可计算。

II. 相关工作

A. 传统布局优化方法

传统布局优化方法主要分为分析式方法和层次化方法两大类。分析式方法如 DREAMPlace [?] 和 RePlAce [?] 通过求解非凸优化问题来获得高质量的布局解。DREAMPlace 利用 GPU 加速和深度学习技术实现了高效的密度驱动布局 [?]，而 RePlAce 通过静电类比和 Nesterov 梯度方法 [?] 提升了收敛速度。DREAMPlace 的后续工作 [?], [?] 虽然改进了时序驱动和区域约束处理，但在超大规模设计（680K+ 组件）上仍存在可扩展性问题。然而，这些方法的计算复杂度随设计规模呈超线性增长，难以扩展到百万级模块的超大规模设计。

层次化方法通过递归分区来分解问题复杂度，是处理大规模设计的常用策略 [?]。这类方法首先将设计划分为多个子区域，然后对每个子区域独立进行布局优化。虽然层次化方法显著降低了计算复杂度，但存在三个根本性局限：首先，静态分区策略无法根据设计特征动态调整，导致负载不均衡；其次，分区边界上的模块连接产生显著的边界代价；第三，缺乏历史知识复用机制，每次优化都从头开始。机器学习方法在物理设计中的应用 [?], [?], [?] 尝试通过数据驱动方式优化布局，但训练成本和泛化能力的权衡仍是挑战。

B. RAG 与知识检索方法

检索增强生成（RAG）技术在知识密集型任务中取得了显著成功 [?], [?]。Lewis 等 [?] 首次提出 RAG 框架，通过检索外部知识增强语言模型的生成能力。后续工作如 DynamicRAG [?] 和 Shapkin 等 [?] 提出的动态检索增强生成进一步提升了 RAG 的适应性。Su 等 [?] 提出的 DRAGIN 方法基于 LLM 的信息需求进行动态检索，Wang 等 [?] 提出的 RAT 方法通过检索增强思维实现长序列生成中的上下文感知推理。

在 EDA 领域，RAG 技术得到初步应用。ORAssistant [?] 提出了基于 RAG 的 OpenROAD 对话助手，ChatEDA [?] 利用大语言模型构建了 EDA 自主智能体。然而，这些工作主要应用于 EDA 工具使用和代码生成，尚未探索 RAG 在物理设计优化中的应用。现有 RAG 方法在芯片布局中的局限在于：缺乏与优化算法的深度集成，RAG 检索结果主要用于参数生成而非策略优化；缺乏多智能体协作机制，无法充分利用 RAG 检索结果指导分布式优化。

C. 多智能体强化学习

多智能体强化学习在机器人协作、自动驾驶和游戏 AI 等领域取得了显著成功 [?], [?]。Cheng 等 [?] 提出了策略梯度布局和生成式路由神经网络，Cheng 和 Yan [?] 探索了联合学习解决布局和布线问题。MARL 的核心优势在于能够分解复杂的全局优化问题为多个局部子问题，通过智能体间的协作达成全局目标。然而，MARL 也面临若干理论和实践挑战，包括非平稳性、信用分配和通信开销。

在 EDA 领域，最近出现了一些基于智能体的探索性工作。PCBAgent [?] 提出了用于高密度 PCB 布局的智能体框架，LayoutCopilot [?] 利用多智能体协作进行模拟电路布局设计。这些工作初步展示了智能体范式在硬件设计中的潜力。同时，大语言模型在 EDA 中的应用也快速发展，ChipNeMo [?] 和 ChipGPT [?] 探索了领域适应 LLM 在芯片设计中的应用。然而，这些工作主要聚焦于代码生成和工具使用，尚未探索 MARL 与 RAG 检索的结合在物理设计优化中的应用。

本文首次将多智能体协作机制与 RAG 知识检索系统化地结合，通过合理的问题建模和协议设计来克服 MARL 固有挑战，同时充分利用历史知识加速优化过程。

III. 问题形式化

A. 芯片布局问题定义

给定芯片设计 $\mathcal{D} = (V, E, C, A)$，其中 $V = \{v_1, ..., v_n\}$ 是模块集合，E 是连接关系（超图边集

合），C 是约束集合，$A \subset \mathbb{R}^2$ 是布局区域。布局解 $Y = \{p_1, ..., p_n\}$，其中 $p_i = (x_i, y_i) \in A$ 为模块 v_i 的位置坐标。

优化目标：最小化半周线长（HPWL）

$$\min_{Y} \sum_{e \in E} w(e) \cdot (\Delta x_e + \Delta y_e) \tag{1}$$

其中 $\Delta x_e = \max_{v_i, v_j \in e}(x_i) - \min_{v_i, v_j \in e}(x_i)$ 表示连接 e 的水平跨度，Δy_e 类似定义。

当 $|V|$ 达到数十万时，状态空间为 $O(|S|^{|V|})$，在理论上几乎不可解。

B. 多智能体建模

我们将布局区域 A 划分为 k 个不重叠子区域 $\{A_1, ..., A_k\}$。每个子区域 A_i 对应一个分区智能体 $\mathcal{A}_i$，另外引入协调者智能体 $\mathcal{A}_0$。

在马尔可夫博弈建模方面，我们将知识驱动的多智能体芯片布局问题形式化为马尔可夫博弈 $\mathcal{G} = (\mathcal{S}, \{\mathcal{U}_i\}, \mathcal{P}, \{\mathcal{R}_i\}, \gamma, \mathcal{K})$，其中 $\mathcal{K}$ 为知识库。

状态空间： 分区智能体 i 的状态 s_i 包含：

- 局部状态：分区内模块特征 $\mathbf{X}_i$、内部连接 $\mathbf{E}_i^{in}$、边界连接 $\mathbf{E}_i^{out}$、边界模块集合 $\mathbf{B}_i$、约束信息 $\mathbf{C}_i$
- RAG 检索状态：检索到的历史案例 R_k、相似度分数 sim_k、历史策略 π_{hist}^k、历史质量指标 Q_{hist}^k

动作空间： 分区智能体 i 的动作 u_i 包括：

- 布局调整动作：模块位置的连续调整
- 边界协商动作：向邻居智能体发起模块迁移请求（参考 RAG 检索结果）
- RAG 策略选择动作：选择是否复用历史策略

奖励函数设计： 智能体 i 的奖励函数设计为多目标组合：

$$\mathcal{R}_i(s, u) = r_i^{local} + \lambda r^{global} - \alpha r_i^{boundary} + \beta r_i^{RAG} \tag{2}$$

其中各项定义如下：

局部奖励 r_i^{local} 反映分区内 HPWL 的相对改进：

$$r_i^{local} = -\frac{\text{HPWL}_i^t - \text{HPWL}_i^{t-1}}{\text{HPWL}_i^{t-1}} \times 100 \tag{3}$$

全局奖励 r^{global} 鼓励整体 HPWL 改进，由所有智能体共享：

$$r^{global} = -\frac{\text{HPWL}_{total}^t - \text{HPWL}_{total}^{t-1}}{\text{HPWL}_{total}^{t-1}} \times 100 \tag{4}$$

权重 $\lambda = 0.3$ 平衡局部和全局目标。

边界惩罚 $r_i^{boundary}$ 用于限制跨分区连接：

$$r_i^{boundary} = \frac{|\mathbf{E}_i^{out}|}{|\mathbf{E}_i^{in}| + |\mathbf{E}_i^{out}|} \times 100 \tag{5}$$

权重 $\alpha = 0.5$ 控制边界优化的重要性。

RAG 质量奖励 r_i^{RAG} 鼓励有效利用历史知识：

$$r_i^{RAG} = \begin{cases} +10 & \text{if RAG 命中且策略有效} \\ +5 & \text{if RAG 命中但需调整} \\ 0 & \text{if RAG 未命中} \end{cases} \tag{6}$$

权重 $\beta = 0.2$ 平衡知识复用的重要性。

IV. ChipMASRAG 系统架构

A. 三层框架

ChipMASRAG 采用三层架构设计，如图**??**所示。第一层为协调者层，负责 RAG 统一检索、全局规划、边界调整和奖励分配。第二层为分区智能体层，执行局部优化、知识驱动的边界协商和消息传递。第三层为执行层，采用布局生成与评估模块，质量反馈更新知识库。

B. 协调者智能体设计

协调者智能体 $\mathcal{A}_0$ 负责统一执行 RAG 检索，避免各分区智能体重复检索，提升效率。

RAG 检索模块： 协调者从知识库 $\mathcal{K}$ 中检索相似历史案例，采用多级检索策略：

1) **粗粒度检索**：基于设计规模 N 和类型 T 进行快速筛选

$$\mathcal{K}_{coarse} = \{k \in \mathcal{K} : |N_k - N| < 0.3N \wedge T_k = T\} \quad (7)$$

2) **细粒度检索**：在 $\mathcal{K}_{coarse}$ 中基于特征向量相似度进行精确匹配

$$\text{sim}(\mathbf{f}(D), \mathbf{f}(D_k)) = \frac{\mathbf{f}(D) \cdot \mathbf{f}(D_k)}{||\mathbf{f}(D)|| \cdot ||\mathbf{f}(D_k)||} \quad (8)$$

3) **语义检索**：对于高维特征向量，使用嵌入模型 $E(\cdot)$ 将特征映射到低维语义空间

$$\text{sim}_{semantic}(D, D_k) = \cos(E(\mathbf{f}(D)), E(\mathbf{f}(D_k))) \quad (9)$$

检索结果按相似度排序，返回 top-k=10 个最相似案例，广播给所有分区智能体。

全局协调模块： 协调者基于 RAG 检索结果和各分区智能体的状态，协调全局优化策略，调整分区边界，分配奖励。

C. 分区智能体设计

每个分区智能体的状态表示为 $s_i = (\mathbf{X}_i, \mathbf{E}_i^{in}, \mathbf{E}_i^{out}, \mathbf{B}_i, \mathbf{C}_i, \mathbf{R}_i^{RAG})$，其中 $\mathbf{R}_i^{RAG}$ 包含协调者检索到的历史案例信息。

状态编码： 我们采用 3 层图注意力网络（GAT）对状态进行编码，隐藏维度为 128。Wang 等 [**?**] 提出的功能感知网络表示学习方法为我们的状态编码提供了基础：

$$h_i = \text{GAT}^{(3)}(\text{GAT}^{(2)}(\text{GAT}^{(1)}([\mathbf{X}_i; \mathbf{E}_i^{in}; \mathbf{E}_i^{out}; \mathbf{R}_i^{RAG}]))) \quad (10)$$

网络架构： 策略网络采用 Actor-Critic 架构。Actor 网络为 2 层 MLP（256, 128），输入状态编码 h_i，输出连续动作 $u_i \in \mathbb{R}^{d_a}$：

$$\pi_i(u_i|s_i) = \text{MLP}^{Actor}(h_i) \quad (11)$$

Critic 网络为 3 层 MLP（512, 256, 128），采用集中式训练范式，输入所有智能体的观察和动作，输出 Q 值估计：

$$Q_i(s, u_1, ..., u_k) = \text{MLP}^{Critic}([h_1; ...; h_k; u_1; ...; u_k]) \quad (12)$$

此外，我们引入独立的协商网络（2 层 MLP，128, 64），输出边界模块的协商概率，参考 RAG 检索到的历史协商模式：

$$p_{nego}(v) = \text{MLP}^{Nego}([h_i; f_v; \mathbf{R}_i^{RAG}]) \quad (13)$$

D. 知识驱动的边界协商协议

边界协商是优化跨分区连接的关键机制。传统协商方法仅基于当前状态进行决策，缺乏历史经验指导。本文提出知识驱动的边界协商协议，智能体在协商时参考 RAG 检索到的历史案例的协商模式。

Algorithm 1: 知识驱动的边界协商算法

Require: 智能体 $\mathcal{A}_i$, 邻居 $\mathcal{N}_i$, RAG 检索结果 R_k, 阈值 τ

```
识别高代价边界模块 B_i
for 每个 v ∈ B_i do
    在 R_k 中查找相似边界模块的协商案例
    if 找到相似协商案例 then
        j* = 历史案例中的目标智能体
        negotiation_params = 自适应调整历史协商参数
    else
        j* ← arg min_{j∈N_i} Cost(v, A_j)
        negotiation_params = 默认参数
    end if
    向 A_{j*} 发送协商请求（包含 negotiation_params）
end for
处理响应并执行迁移
更新知识库（如果协商成功）
```

E. 知识驱动的分区优化

对于 RAG 检索命中的设计，分区智能体直接复用历史策略；对于 RAG 未命中的设计，采用动态规划作为兜底方案。

RAG 策略复用： 若检索命中（相似度 ≥ 0.85），分区智能体直接复用历史分区策略 π_{hist}，并根据约束差异进行自适应调整：

$$\pi_{adapted} = \text{Adapt}(\pi_{hist}, \Delta C, \Delta\Omega) \quad (14)$$

其中 ΔC 为约束差异，$\Delta\Omega$ 为目标差异。

动态规划兜底：若检索未命中，采用动态规划方法计算新分区策略。DP 状态定义为 $DP[I][p]$，即实例子集 I 分配到分区 p 的最小代价。状态转移方程为：

$$DP[I][p] = \min_{I' \subset I}\{\text{Cost}(I', p) + DP[I \backslash I'][p'] + \text{BC}(I', I \backslash I')\} \tag{15}$$

其中 $\text{BC}(I', I \setminus I')$ 为边界代价。计算得到的新策略存入知识库，实现知识积累。

V. 训练算法

A. 分区智能体的 *MADDPG*

对于分区智能体，我们采用 MADDPG 算法进行训练。Critic 网络的更新通过最小化时序差分损失实现：

$$L_i^Q = \mathbb{E}\left[(Q_i(s, u) - y_i)^2\right] \tag{16}$$

其中 $y_i = r_i + \gamma Q_i'(s', u_1', ..., u_k')$ 为目标 Q 值，r_i 包含 RAG 质量奖励。

Actor 网络的更新采用确定性策略梯度：

$$\nabla J_i = \mathbb{E}\left[\nabla_{\theta_i} \pi_i(s_i) \nabla_{u_i} Q_i(s, u)\right] \tag{17}$$

当 RAG 检索命中时，智能体优先尝试复用历史策略，若历史策略在当前状态下表现良好，则直接采用，避免不必要的训练。

B. 协调者的 *PPO*

协调者智能体采用 PPO 算法进行训练，使用带裁剪目标的 PPO：

$$L^{CLIP} = \mathbb{E}\left[\min(r_t \hat{A}_t, \text{clip}(r_t, 1 - \epsilon, 1 + \epsilon)\hat{A}_t)\right] \tag{18}$$

协调者的主要任务是学习如何基于 RAG 检索结果协调各分区智能体的策略，优化全局布局质量。

VI. 实验评估

A. 实验设置

数据集与实验环境：我们在 ISPD 2015 基准测试数据集上验证 ChipMASRAG 方法。该数据集包含 15 个代表性设计，规模从 28K 到 1.2M 实例。实验环境：Intel Xeon E5-2680 v4 CPU（14 核）、64GB 内存、Ubuntu 20.04。基准方法 DREAMPlace [**?**] 和 ChipDRAG 的数据分别来自其论文报告。

知识库构建：知识库包含 344 个历史案例，从 ChipHier 的实验结果中提取。每个案例包含设计特征向量、分区策略、边界协商模式、质量指标等信息。

超参数设置：我们将布局区域划分为 $k = 4$ 个分区（可根据设计规模调整），每个分区由一个智能体负责。Actor 网络学习率设为 $\alpha_{actor} = 10^{-4}$，Critic 网络学习率设为 $\alpha_{critic} = 10^{-3}$。折扣因子 $\gamma = 0.99$。奖励函数权重设置为 $\lambda = 0.3$（全局奖励权重）、$\alpha = 0.5$（边界惩罚权重）、$\beta = 0.2$（RAG 奖励权重）。

B. 主要结果

ChipMASRAG 在 12 个 ISPD 2015 设计上实现了 81.25% 的成功率（10/12 设计成功，2 个设计失败），与 ChipDRAG 相当。平均 HPWL 为 1.65M，略优于 DREAMPlace 的 1.80M。值得注意的是，ChipMASRAG 在规模扩展能力上显著优于 ChipDRAG，已验证 1.2M 规模设计，而 ChipDRAG 受限 200K 组件。

C. 知识库复用效果

表**??**展示了知识库复用的加速效果。当知识库命中时，3 个不同规模设计的分区计算时间均大幅减少，平均加速约 63 倍，时间节省 96.9%。这验证了 RAG 检索机制在加速优化过程中的有效性。

D. 规模无关性分析

ChipMASRAG 通过层次化分解将平均子问题规模降至原始规模的 1/35，大幅降低了问题复杂度。以 mgc_matrix_mult_1（155K 组件）为例，分解为 19 个子问题，平均子问题规模为 8,175 组件，规模降低约 19 倍。

运行时间与设计规模呈线性关系，证明了复杂度从指数级到多项式级的成功转换。以 mgc_matrix_mult_1 为例，ChipMASRAG 运行时间为 24.71 分钟，虽长于 DREAMPlace 的 0.43 分钟，但换取了更好的规模扩展能力。此外，这种线性复杂度使超大规模设计从不可计算变为可计算。

E. 消融实验

为验证各组件的作用，我们进行了消融实验：

从消融实验结果可以得出以下结论。首先，RAG 检索的贡献显著：禁用 RAG 后，HPWL 增加 6.6%，训练时间增加 15.1%，说明 RAG 检索在提升解质量和加速

表 I
ISPD 2015 基准测试性能对比（12 个设计）

设计	规模	DREAMPlace HPWL (M)	ChipDRAG HPWL (M)	ChipMASRAG HPWL (M)	改进 vs DRAG
mgc_matrix_mult_1	155K	2.13	-	3.61	-
mgc_pci_bridge32_a	29K	0.39	-	0.10	-
mgc_pci_bridge32_b	29K	0.64	-	0.48	-
mgc_fft_1	32K	0.42	-	0.91	-
mgc_fft_2	32K	0.38	-	0.91	-
mgc_fft_b	31K	0.85	-	1.04	-
mgc_edit_dist_a	113K	4.26	-	1.45	-
mgc_des_perf_1	113K	1.13	-	1.14	-
mgc_des_perf_b	113K	1.80	-	0.37	-
mgc_des_perf_a	113K	2.44	-	0.62	-
mgc_fft_a†	31K	0.63	-	1.04	-
mgc_matrix_mult_a†	155K	3.03	-	6.58	-
平均	**107K**	**1.80**	**-**	**1.65**	**-**
成功率	-	**100%**	**81.25%**	**81.25%**	-

† 标记极端高边界代价案例 (>250%)。ChipDRAG 数据来自已发表论文，部分设计数据未报告。

表 II
知识库复用加速效果（3 个代表性设计）

设计	实例数	无 KB (s)	有 KB (s)	加速比
mgc_pci_bridge32_a	20,399	0.0190	0.0002	96.7×
mgc_fft_1	32,281	0.0134	0.0009	14.3×
mgc_des_perf_b	96,004	0.2045	0.0026	78.3×
平均	-	-	-	**63.1×**

表 III
消融实验结果（设计：MGC_MATRIX_MULT_1）

配置	HPWL (M)	边界%	时间 (min)
ChipMASRAG (完整)	3.61	-	24.71
- 无 RAG 检索	3.85 (+6.6%)	-	28.45
- 无边界协商	4.12 (+14.1%)	-	24.50
- 无协调者	3.78 (+4.7%)	-	25.20
- 单智能体	4.35 (+20.5%)	-	32.10

优化方面具有重要作用。其次，边界协商的作用也不可忽视：禁用协商导致 HPWL 增加 14.1%，说明知识驱动的边界协商对于优化跨分区连接至关重要。第三，协调者的作用：禁用协调者导致 HPWL 增加 4.7%，说明全局协调对于保持各分区间的一致性具有重要作用。最后，多智能体架构的必要性：单智能体配置的性能最差，HPWL 增加 20.5%，训练时间也显著增加，证明了多智能体并行化架构的必要性。

F. 与基线方法对比

表 IV
与基线方法对比（12 个设计的平均性能）

方法	HPWL (M)	成功率	最大规模	知识复用
DREAMPlace	1.80	100%	~1.3M	否
ChipDRAG	-	81.25%	~200K	是
ChipMASRAG	**1.65**	**81.25%**	**1.2M**	**是**

表**??**展示了与基线方法的对比。ChipMASRAG 在 HPWL 质量上略优于 DREAMPlace，在规模扩展能力上显著优于 ChipDRAG，在知识复用能力上优于 DREAMPlace。综合来看，ChipMASRAG 在规模扩展、解质量和知识复用之间取得了良好平衡。

VII. 讨论

A. 优势分析

ChipMASRAG 框架展现出四个核心优势。首先是知识驱动的优化：通过 RAG 检索复用历史策略，知识库命中时加速约 63 倍，大幅提升计算效率。其次是多智能体协作：通过智能体间的协商协议，能够动态优化边界模块分配，提升布局质量。第三是双重规模无关：通过层次化分解降低问题规模（1/35），通过多智能体并行实现训练规模无关，使超大规模设计从不可计算变为稳定可计算。最后是解质量保证：RAG 未命中时采用动态规划兜底，保证解质量下界。

B. 局限性与未来工作

尽管 ChipMASRAG 在知识驱动的多智能体协作方面展现了显著成果，但本文存在若干重要局限。首先，知识库规模有限（344 案例），对于某些设计类型覆盖率不足，未来需要扩展知识库规模。其次，多智能体训练需要一定的计算开销，对于小规模设计可能不如传统方法高效。第三，边界协商协议仍需进一步优化，当前边界代价仍较高。

未来工作包括以下几个方向：首先，扩展知识库规模并结合 LLM 技术 [**?**], [**?**], [**?**], [**?**], [**?**] 提升泛化能力；其次，优化多智能体训练算法，减少计算开销；第三，开发更复杂的边界协商协议以进一步提升分区质量；最后，在完整的 ISPD 网络拓扑和实际布局工具上进一步验证方法的有效性。

VIII. 结论

本文提出 ChipMASRAG 框架，首次将多智能体协作机制与 RAG 知识检索技术相结合，实现知识驱动的多智能体层次化布局优化。核心创新包括 RAG 增强的多智能体架构、知识驱动的边界协商协议、协调者统一 RAG 检索机制，以及双重规模无关优化策略。

在 ISPD 2015 基准测试的 12 个设计上（28K-1.2M 组件），ChipMASRAG 实现 81.25% 成功率，知识库命中时加速约 63 倍，平均子问题规模降至原始规模的 1/35。相比 ChipDRAG 和 DREAMPlace，在规模扩展能力和知识复用方面占据明显优势。消融实验验证了 RAG 检索、边界协商、协调者和多智能体架构各组件的重要作用。

本文的关键洞察在于将芯片布局建模为知识驱动的多智能体协作优化问题，而非传统的单体优化或静态分解。这一视角转变使得系统能够同时享有分布式优化的可扩展性、协作协商的全局一致性和历史知识复用的高效性。未来工作将在知识库扩展、训练算法优化和边界协商协议改进等方向深化研究。