丘成桐中学科学奖 · 计算机奖赛道候选课题 20 则

> 本方案面向 **2027 赛季**（2026 赛季提交截止为 2026 年 9 月 15 日，从零启动已来不及）。倒排时间表以 2026 年 9 月为第 0 周、2027 年 9 月 15 日提交为终点，共约 52 周。

学生画像与资源假设：普通重点高中生，有一定编程与数学基础；计算资源为个人笔记本、纯 CPU、无 GPU 无集群；实验资源不超过中学常规实验室；全部使用公开数据；由中学教师即可指导。凡超出这一假设的路线，都在标签里标明了资源门槛。

四条必须先知道的赛制事实（取自官方公告，2020 – 2026 全部公示文章）：指导老师不得来自培训机构或任何谋利机构，违者取消资格；总决赛全程英文（研究报告、PPT、答辩）；入围总决赛的研究报告会在 11 月全网公示；提交材料含论文查重报告与实验视频。规则每年会改，报名开放后须重新核对当年《参赛规则》与本学科《参赛指南》。

这个赛道的评委在奖励什么

统计 2020 – 2025 共 64 项计算机奖获奖论文，这个赛道的供需失衡最明显，也因此存在最清晰的套利空间。

深度学习应用占了压倒性多数，但回报正在递减。 医学影像诊断、人脸识别、语义分割、视觉辅助系统、音乐风格迁移、甲骨文识别——这一族占了全部获奖的六成以上，且绝大多数停在铜奖与优胜奖。评委每年要看几百份"我拿一个模型跑了一个数据集"的报告。做同类课题，你的对手不是问题本身，而是三百份长得差不多的报告。

纯算法与理论稀缺，但命中率极高。 六年里这一族只有寥寥数篇：并行 bi-core 分解（2021 金奖）、并行密度峰值聚类（2022 银奖）、Max-Cut SDP 的最优解与秩（2020 优胜）、神经直觉与逻辑推理结合的几何定理证明、LLM 数学推理的形式化验证（2024 金奖）。**投稿密度低而获奖层级高，这是本赛道最值得注意的结构性事实。**而且算法与理论类课题天然只需 CPU，与本方案的资源约束完美契合。本方案 K01 – K10 有意重仓这一族。

LLM 相关是新兴且 CPU 友好的窗口。 2024 – 2025 出现了：LLM 数学推理的形式化验证（金奖）、历史推理 benchmark、基于 LLM 的多智能体重复博弈中"未来的阴影"对合作策略演化的影响（铜奖）、多模态机器心智理论。注意这几篇的共同点：交付物是**测量与结论**，不是训练出来的模型。评测型、探针型、行为分析型课题只需 API 调用或小模型 CPU 推理，是资源受限学生的最佳杠杆——本方案 K11 – K20 按这个思路设计。

这个赛道的头号死法是"工程演示"。 丘奖要的是研究报告，不是产品。一个能跑的系统 + 一张性能对比表，缺少可否证的科学假设，在分赛区就会被筛掉。本方案每条路线都强制要求 H1/H2 假设含数值或方向，且验收标准第一条是能否重现已发表基准。

算力是硬现实。 纯 CPU 笔记本意味着：ResNet 级别的从头训练勉强可行但耗时以天计，Transformer 预训练完全不可能，LoRA 微调只能用到很小的模型，量子态矢量模拟约 26 – 30 比特就吃满 16 GB 内存。每条路线都写明了具体上限与降规模版本——把算力约束写成课题设计的输入，而不是做到一半才发现。

本赛道 20 条路线的分族逻辑

- **K01 – K10 算法、系统与理论族**：结构性绕开算力墙——交付物是复杂度分析、收敛性、正确性证明与基准测量，不是模型规模。风险在于必须核实该问题的已知最好界，不能去撞已知难题。每条路线都做了这项核实。
- **K11 – K20 机器学习与应用族**：以评测、可解释性、鲁棒性分析为主，训练型课题一律限定在 CPU 可完成的规模并配降规模版本。涉及 LLM API 的路线写明了预算上限与可复现性要求（固定模型版本号与调用日期——模型会静默更新，不记录版本的实验一年后无法复现）。

两族独立排序：K01/K11 最稳，K10/K20 风险与上限最高。

K01 · 学习型索引在查询分布偏移下的鲁棒性：基于 SOSD 基准的双成本口径尾延迟测定

`推荐优先级：最高` · `分族：数据结构工程化` · `参赛子类：计算机–系统与数据结构` · `资源需求：纯 CPU 笔记本 / 16 GB 内存 / 需 Linux 真机` · `技能取向：C++ 工程 + 实验统计`

1 · 研究问题

在 SOSD（Search On Sorted Data）标准数据集上，学习型索引（learned index：RMI、RadixSpline、PGM-Index）相对传统索引（ART、STX B+ 树、二分查找）的查找延迟优势，在查询键分布由均匀随机偏移为偏斜分布（Zipf 热点、范围局部性）时是否仍然成立？若优势衰减，衰减幅度能否由各结构的"最后一英里"（last-mile）搜索区间宽度定量预测？

2 · 研究背景与空白

技术背景。 学习型索引把"有序数组上的键 → 位置"映射视为累积分布函数（CDF）的回归问题：先用一个小模型（分段线性、样条、两层递归模型）预测目标位置，再在预测位置附近的一个有界区间内做局部搜索定位精确下标。这个"预测 + 最后一英里搜索"的两阶段结构，把传统 B 树的 O(log n) 次指针追逐（pointer chasing，每一次都可能是一次缓存缺失）换成了若干次算术运算加一次局部扫描。因此它的性能优势本质上是**内存局部性优势**，而不是渐进复杂度优势——纯 CPU 环境下，一次末级缓存（LLC）缺失约合 200 – 300 个时钟周期，这才是真正的成本主项。这一点使该课题天然绕开算力墙：它测的是缓存行为，不是浮点吞吐，一台普通笔记本与一台服务器在"每次查找的缺失次数"这个口径上是可比的。

已有工作到哪一步。 Kipf 等人的 SOSD 基准（arXiv:1911.13014，及 Marcus 等人 VLDB 2021《Benchmarking Learned Indexes》）建立了公开的数据集与实现框架，覆盖 books / fb / osm / wiki 等真实与合成键集（每个 200M 条 64 位键，另有 32 位变体），统一比较 RMI、PGM、RadixSpline 与 STX B+ 树、插值 B 树（IBTree）、ART。已发表的定量结论包括：RMI 的构建时间显著慢于 PGM 与 RadixSpline，且没有任何学习型结构的构建速度能追上面向插入优化的传统结构；在 32 位数据集上 ART 表现极好，在 face32 上甚至优于全部对手；RMI 与 RadixSpline 都需要针对数据集手工调参。SOSD 还内置了分支误预测与指令数的自动分析。Crotty 的 Hist-Tree（CIDR 2021）进一步指出，一个朴素的直方图结构在同样基准上就能逼近学习型索引。

空白在于：SOSD 及其后续工作的查询负载几乎全部是"从键集中均匀随机抽取"的查找。而真实系统里的查询分布是重度偏斜的（热点键、时间局部性、范围扫描起点聚集）。偏斜会让传统结构的上层节点常驻缓存（ART 的前几层、B+ 树的内节点），而学习型索引的模型本来就小、本来就常驻，其增益空间反而可能被压缩——但这个方向没有被系统测量过。这个空白属于"评价维度的转换"，可靠性高：数据、代码、基线数值全部公开，实验成本是笔记本级的，且**结论正负都成立**（优势保持是对学习型索引普适性的一次独立支持；优势消失则是一条对系统设计有用的负面结论）。

3 · 可检验假设

- H1：在 Zipf($s \approx 1.1$) 偏斜查询负载下，RMI 相对 ART 的中位查找延迟优势，比均匀随机负载下缩小 $\geqslant 30\%$（相对量）；在极端偏斜（前 1% 键承担 90% 查询）下优势完全消失或反转。
- H2：各结构在同一负载下的 p99/p50 延迟比，与其"最后一英里"搜索区间宽度的对数 $\log_2(\varepsilon)$ 呈单调正相关，跨 4 个数据集 × 5 个结构的 20 个数据点上线性拟合决定系数 $R^2 \geqslant 0.7$。若 $R^2 < 0.7$，说明尾延迟主要由分支误预测而非区间宽度决定——这同样是有效结论。

4 · 量化验收标准

1. **方法学校验（硬门槛）**：用自行编译的 SOSD 管线，在 books / fb / osm / wiki 四个数据集（若内存不足则用官方 200M 集的前 100M 切片，并同时报告切片对结论的影响）上、对至少 5 种索引结构，重现原基准的均匀随机查找结果。因跨硬件绝对延迟不可比，判据定为**微架构计数器口径**：每次查找的末级缓存缺失数与退休指令数，与 SOSD 公开数值的偏差 $\leqslant 15\%$；同时要求各结构的**相对排序**与原文完全一致。这一步不过关，后续全部结论无效。
2. **基准测试方法学**：每个（结构 × 数据集 × 负载）组合至少重复 30 次独立运行，每次 10M 次查找；运行前固定 CPU 频率（Linux `cpupower frequency-set --governor performance`，禁用 Turbo 与 SMT）、每次运行前显式冲刷缓存（遍历一块 $\geqslant$ LLC 两倍的哑数组）、绑核（`taskset`）、关闭后台进程。**报中位数与四分位距（IQR），不报均值与标准差**，并给出全部 30 次的分布图；跨组比较用 Mann – Whitney U 检验而非 t 检验。
3. **双成本口径**：与已发表实现的对照必须同时报两种口径——**等工作量**（相同查询条数下的墙钟时间与缺失数）与**等墙钟时间预算**（固定 5 秒内完成的查询条数）。两种口径的结论若不一致，必须在正文中明确讨论。
4. **构建成本单列**：报告每种结构的构建时间与内存占用（含峰值），并给出"构建成本需多少次查询摊销回本"的交叉点，按偏斜与均匀两种负载分别给出。
5. **负载覆盖**：至少 4 种查询分布（均匀、Zipf s=0.8、Zipf s=1.1、范围局部性），每种给出明确、可复现的生成脚本与固定随机种子。
6. **可复现性**：全部代码、负载生成器、原始计时数据（CSV）与绘图脚本开源；提供 `make reproduce` 一键重跑脚本；随机种子写死在配置文件中。

5 · 数据与工具

用途	来源 / 工具
基准框架与索引实现	SOSD 开源仓库（GitHub `learnedsystems/SOSD`），含 RMI、RadixSpline、PGM、STX B+ 树、ART、二分/插值搜索的统一封装。**仅用于校验与对比，不计入本项目的数据贡献。**
键数据集	SOSD 官方数据集（books / fb / osm / wiki，200M × 64 位；另有 32 位变体），由仓库脚本自动下载（托管站点为 Harvard Dataverse，**需核实当前链接是否仍有效**；若失效则用仓库提供的合成生成器 + 公开 OSM 导出自建等价集，并声明差异）
RMI 生成器	独立的 Rust 工具 `learnedsystems/RMI`，需安装 Rust 工具链后代码生成再编译。**依赖链较重，列为风险项**
微架构计数器	Linux `perf stat`（需真机，虚拟机/WSL2 通常无 PMU 访问权限，**需核实**）；退路为 Valgrind `cachegrind`（模拟计数，非真实计数，须在正文标注口径）
计时	`rdtsc` / `clock_gettime(CLOCK_MONOTONIC)`；校准 TSC 频率
统计与作图	Python + NumPy / SciPy（Mann–Whitney U、自助法置信区间）+ Matplotlib
算力	纯 CPU；200M × 8 字节键 = 1.6 GB，加索引结构峰值约 4–6 GB，16 GB 笔记本可容纳；单次完整扫（5 结构 × 4 数据集 × 4 负载 × 30 重复）估计 6–12 小时，可夜间批跑

6 · 方法路径

1. 装 Linux 真机环境与工具链，编译 SOSD，跑通仓库自带算例，确认 `perf` 可读 PMU；完成验收标准第 1 条的重现校验并归档原始数据。
2. 核实工具能力边界：RMI 生成器是否能在本机成功产出并编译；PGM 与 RadixSpline 的误差参数 ε 是否可外部配置；SOSD 的计时循环是否已做防编译器优化（`DoNotOptimize` 类屏障）——若没有，须自行加入并在正文说明。
3. 实现查询负载生成器：给定键集与分布参数产出查询序列，写明每种分布的精确定义与种子；对"范围局部性"给出明确、可复现的判据（例如按滑动窗口从键空间的 0.1% 区间内抽取）。
4. 实现基准测试骨架：频率锁定、绑核、缓存冲刷、30 次重复、原始逐次结果落盘为 CSV（不做在线聚合，保留全部分布信息）。
5. 执行主扫描（结构 × 数据集 × 负载 × 重复），同步采集 `perf` 计数器；对每个结构额外记录"最后一英里"搜索区间宽度的经验分布（需在各实现中插桩）。
6. 分析：按第 4 块口径出中位数/IQR、Mann – Whitney 检验、双成本口径对照表、构建成本摊销交叉点；拟合 H2 的 $\log_2(\varepsilon)$ 关系并给自助法置信区间。
7. 独立交叉校验：用 `cachegrind` 模拟计数复算至少一个（结构 × 数据集）组合的缓存缺失，与 `perf` 真实计数比对；再用一台不同微架构的机器（同学的笔记本即可）重跑关键组合，验证结论的排序不随硬件改变。

7 · 新颖性边界

> **本课题不声称**：不提出新的索引结构，不声称学习型索引在任何意义上"更好"或"更差"，不声称任何新的复杂度界。SOSD 框架、全部索引实现与数据集均为他人工作，明确标注为对照基准，不计入本项目贡献。
>
> **已有工作完成了什么**：Kipf 等的 SOSD（arXiv:1911.13014）与 Marcus 等（VLDB 2021）在**均匀随机查找**负载下，对 RMI / PGM / RadixSpline / STX B+ 树 / IBTree / ART 给出了查找延迟、构建时间、内存占用与分支误预测的系统比较；Crotty（CIDR 2021）用 Hist-Tree 说明简单直方图即可逼近学习型索引。这些结论本项目不重复声称。
>
> **本项目的贡献**：把评价维度从"平均查找延迟"转换为"**查询分布偏移下的延迟分布（含尾部）**"，并给出偏斜负载下的双成本口径对照与构建成本摊销交叉点。这是主结论，不是附加内容。
>
> **为什么有价值**：真实数据库负载是重度偏斜的，而现有基准只测均匀负载。若优势保持，学习型索引的实用价值获得一次独立支持；若优势在偏斜下消失，则说明现有基准系统性高估了这类结构——两个方向都是可发表的判断。
>
> **风险提示**：若各结构在偏斜负载下差异不显著，"差异不显著"同样是有效结论，但必须给出 30 次重复的 IQR 误差棒，证明本实验有能力分辨题设中 30% 这一量级的差异。

8 · 决策门槛（go / no-go）

- **第 6 周**：环境与编译门槛。若 SOSD 无法在本机编译通过、或数据集下载链接失效，**立即降级路径 A**：放弃 RMI（Rust 依赖最重），只保留全部 header-only 的 RadixSpline、PGM、STX B+ 树、ART、二分与插值搜索五种结构；数据集改用仓库合成生成器 + 公开 OSM 节点 ID 导出自建。主结论框架（偏斜负载下的延迟分布对照）完全不受影响。
- **第 10 周**：方法学校验硬门槛。若微架构计数器偏差 > 15% 或相对排序与原文不一致，先排查是否为频率漂移/SMT/编译器优化屏障问题；到第 12 周仍不过关则**降级路径 B**：把口径整体改为 `cachegrind` 模拟计数

（确定性、跨机器可比），全文统一声明为"模拟计数口径"，放弃绝对延迟的跨文献对照，保留组间相对比较。主结论不变。

- **第 14 周**：`perf` PMU 可用性必须已核实清楚——这是需要提前确认而非边做边发现的事项。若学生只有 macOS 机器，须在第 4 周前解决 Linux 真机（双系统或旧机器重装），否则整条路线的计数器口径不成立。
- **第 26 周**：主扫描完成且 H1 有明确结论（成立或否证）。若此时仍在调试基准骨架，**降级路径 C**：把数据集从 4 个减为 2 个（books + fb，一真实一合成）、负载从 4 种减为 3 种，用省下的时间保证重复次数与统计口径完整。宁可少扫参数点，不可减少重复次数——统计口径是本课题的核心卖点。
- **预算裁剪顺序**：数据集数量 → 结构数量 → 负载种类 → （绝不裁剪）重复次数与频率控制。
- **选择前提**：适合已能独立写 C++、愿意花时间处理构建系统与 Linux 环境的学生。若学生只会 Python，这条路线的第 1 步就会卡死，应改选 K02 或 K06。

K02 · 图重排序收益的结构归因：以摊销成本口径检验社区结构强度能否预测 PageRank/BFS 的加速比

推荐优先级：高 · 分族：图算法与网络分析 · 参赛子类：计算机–算法与性能 · 资源需求：纯 CPU 笔记本 / 16 GB 内存 · 技能取向：C++ 或 Python+C 混合 + 图论

1 · 研究问题

对 20 个以上公开真实图，四种顶点重排序（graph reordering）方法——Gorder、Rabbit Order、按度排序（DegSort）、逆 Cuthill – McKee（RCM）——给 PageRank 与 BFS 带来的加速比，能否由一个**可先验计算的图结构量**（Louvain 模块度 Q、度分布幂律指数、平均度、有效直径）预测？在把重排序自身开销计入的摊销口径下，每种方法需要执行多少次图算法才能回本？

2 · 研究背景与空白

技术背景。 图算法（PageRank、BFS、连通分量）在压缩稀疏行（CSR）格式上的主要瓶颈不是算术，而是访问邻居顶点属性数组时的随机内存访问。顶点编号顺序直接决定这些访问的空间局部性：如果相邻顶点的编号也相近，它们的属性就落在同一缓存行或同一页上。顶点重排序就是在跑算法之前先把顶点重新编号一遍，用一次预处理成本换取后续每一轮迭代的缓存命中率。这类课题对本项目的资源约束是理想的：图数据是公开的、算法是纯 CPU 整数与访存密集型的、衡量指标（缓存缺失率、每边处理时间）在笔记本上就能稳定测出——**算力墙在这里根本不构成约束，反而是"小内存 + 真实缓存层级"让局部性效应更容易观测。**

已有工作到哪一步。 Wei 等的 Gorder（SIGMOD 2016）通过最大化滑动窗口内的共同邻居数来排序，报告了对多种图算法的加速；Arai 等的 Rabbit Order（IPDPS 2016）利用真实网络的层次社区结构做并行重排序；Balaji 与 Lucia 的《When is Graph Reordering an Optimization?》（IISWC 2018）已经明确提出了"重排序何时值得"的问题，并指出轻量方法（如 Hub Sorting、Hub Clustering）在摊销后往往优于重量方法。近期工作继续在这条线上推进：有研究报告 RCM 虽然摊销成本低于 Gorder，但仍需约 20 次算法执行才能抵消其重排序时间；另有面向稀疏矩阵向量乘（SpMV）与图神经网络训练的重排序实验研究，覆盖 bfs / degsort / dfs / gorder / hubcluster / hubsort / ldg / metis / minla / rabbit / rcm / slashburn 等十余种方法。

空白在于：现有实验研究几乎都停留在"哪个方法更快"的横向排名，而没有把加速比当作因变量、把可先验计算的图结构量当作自变量，去建立并检验一个预测模型。换句话说，从业者拿到一张新图时，仍然没有一条"先算个 Q 值就知道该不该重排序"的可操作判据。这个空白属于"问题方向的转换"（从"哪个更快"转为"什么能预测更快"），可靠性高：所有输入都是公开图与公开算法，实验规模可控，且**正负结论都成立**——若结构量能预测，得到一条实用判据；若不能预测，则说明重排序收益依赖更细的微架构因素，这本身是对现有直觉的一次证伪。

3 · 可检验假设

- H1：PageRank 的重排序加速比与 Louvain 模块度 Q 正相关，跨 $\geqslant 20$ 个图的 Spearman 秩相关 $\rho \geqslant 0.6$（$p < 0.05$，自助法置信区间不跨 0）。若 $\rho < 0.3$，则 H1 被否证。
- H2：摊销回本次数（break-even executions，含排序时间的总成本等于不排序时的总成本所需的算法执行次数）在 Gorder 上中位数 $\geqslant 10$，在 DegSort 上中位数 $\leqslant 3$；即"轻量方法在单次或少次执行场景下严格占优"这一判断在本样本上成立。

- **备择机制假设**：若 Q 的预测力不足，则改由"度分布幂律指数 γ"或"最大 k-core 的顶点占比"承担预测变量；模型选择用留一交叉验证的预测均方误差决定，而非事后挑选最好看的相关系数。

4 · 量化验收标准

1. **方法学校验（硬门槛）**：用自建管线在文献已报告过的至少 3 个公共图（例如 soc-LiveJournal1、com-Orkut、web-Google 或其在已发表实验中出现的等价图）上重现 Gorder 与 RCM 对 PageRank 的加速比，**与已发表数值的相对偏差 ⩽ 25%**（跨硬件不可能更严，故同时要求方法间的相对排序完全一致，且缓存缺失率的变化方向一致）。这一步不过关，后续全部结论无效。
2. **基准测试方法学**：每个（图 × 排序 × 算法）组合重复 ⩾ 20 次；固定 CPU 频率与线程绑定，禁用 Turbo；每次运行前冲刷缓存与页缓存状态；**报中位数与 IQR，不报均值**；组间比较用 Mann – Whitney U 检验。PageRank 固定为 20 次幂迭代（不用收敛判据，避免迭代数随排序变化污染计时），BFS 固定从 64 个随机但**种子写死**的源点出发。
3. **双成本口径**：与已发表实现对照时同时报**等工作量**（相同迭代数的墙钟时间）与**等墙钟时间预算**（固定 10 秒内完成的迭代数）；摊销口径单独作为第三张表，明确写出排序时间的测量方式（单线程还是并行、是否含 I/O）。
4. **样本量与统计**：图样本 ⩾ 20 个，覆盖至少 4 个类别（社交网络、网页图、道路网、合成幂律图），顶点数跨 3 个数量级；相关系数给自助法（bootstrap, ⩾ 2000 次重采样）95% 置信区间；**不报 R^2 单一数字**，必须同时给散点图与置信带。
5. **结构量计算的可复现性**：模块度用固定分辨率参数与固定随机种子的 Louvain 实现，报告 10 次独立运行的 Q 值中位数与极差；幂律指数用 Clauset – Shalizi – Newman 的最大似然 + KS 距离方法，而非双对数最小二乘，并报告拟合的 KS 检验 p 值。
6. **可复现性**：全部代码、图清单与下载脚本、原始计时 CSV、结构量表格与绘图脚本开源；一键重跑脚本；全部随机种子写死。

5 · 数据与工具

用途	来源 / 工具
真实图数据	SNAP 数据集（snap.stanford.edu/data，社交/网页/道路/协作网络，含 com-Orkut、soc-LiveJournal1、web-Google、roadNet-CA 等）；SuiteSparse Matrix Collection（sparse.tamu.edu，可按顶点数/非零数在线筛选后单图下载，有效控制下载量）
合成图	NetworkX 的 `powerlaw_cluster_graph` / LFR 基准生成器（`LFR_benchmark_graph`，内置于 NetworkX，可控社区强度 μ——这一条对 H1 至关重要，因为它允许**主动扫描**模块度而不是被动依赖真实图的分布）
图算法内核	GAP Benchmark Suite（GitHub `sbeamer/gapbs`，C++，含 PageRank、BFS、CC、BC 的高质量参考实现，支持 CSR 与 OpenMP）。**仅用于校验与对比，不计入本项目的数据贡献。**
重排序实现	Rabbit Order（作者开源，依赖 Boost 与 libnuma，**需核实在无 NUMA 的笔记本上能否编译，这是已知风险点**）；Gorder（作者开源 C++）；DegSort 与 RCM 可自行实现（RCM 亦有 SciPy `scipy.sparse.csgraph.reverse_cuthill_mckee` 内置版本，可作交叉校验）
社区检测与结构量	NetworkX（`community.louvain_communities` 自 3.x 起内置）或 igraph（`community_multilevel`，C 实现、快得多，大图上首选）；幂律拟合用 `powerlaw` 包（Alstott 等实现 CSN 方法）
计数器与计时	Linux `perf stat`（LLC 缺失、DTLB 缺失）；退路 `cachegrind`
算力	纯 CPU。com-Orkut（3M 顶点 / 117M 边）CSR 约 1 GB，Gorder 预处理是本课题最重的一步（文献报告在大图上可达分钟级），16 GB 笔记本可覆盖顶点数 ≤ 5M 的图；**超出范围**：顶点数 ≥ 50M 的图（如 web-ClueWeb）不在本课题范围内，需在正文明确声明样本上界

6 · 方法路径

1. 装环境，编译 GAP Benchmark Suite，用其自带算例与官方公布的参考结果确认安装正确；完成验收标准第 1 条的三图重现校验。
2. 核实工具能力边界：Rabbit Order 在无 NUMA 环境下能否编译运行（若不能，用其算法描述自行实现基于 Louvain 层次的重排序，并用制造样例验证——**若需自行实现，须用小图上的暴力最优解或已知性质做验证，这本身可作为方法学贡献写入**）；确认 Gorder 的窗口参数与文献一致。
3. 组建图样本：从 SNAP / SuiteSparse 选 ⩾ 15 个真实图 + ⩾ 5 个 LFR 合成图（合成图的 μ 参数扫描覆盖模块度 0.2 – 0.8），登记每图的顶点数、边数、方向性、是否去自环与重边；预处理判据写死并公开。
4. 计算结构量：对每图算 Louvain 模块度（10 次取中位数）、幂律指数（CSN 方法 + KS 检验）、平均度、有效直径（用采样估计并给误差）、最大 k-core 占比。
5. 执行主扫描：4 种排序 × 2 种算法 × 20+ 图 × 20 次重复，同步采集 `perf` 计数器与排序自身耗时；全部逐次结果落盘。
6. 分析：出中位数/IQR 表、加速比 vs 结构量的散点与 Spearman ρ（自助法置信区间）、摊销回本次数表、双成本口径对照表；用留一交叉验证选预测变量，报告预测均方误差而非仅相关系数。
7. 独立交叉校验：在 LFR 合成图上做一次**受控实验**——固定顶点数与度分布、只扫描社区强度 μ，看加速比是否随 μ 单调变化。这条是对 H1 的因果性最强的检验，与真实图上的相关性分析互为独立证据。

7 · 新颖性边界

本课题不声称：不提出新的重排序算法，不声称任何重排序方法优于另一种（这已由前人给出），不声称对图算法性能的新的理论界。GAP 内核、Gorder 与 Rabbit Order 的实现均为他人工作，标注为对照基准，不计入本项目贡献。

已有工作完成了什么：Wei 等（Gorder, SIGMOD 2016）与 Arai 等（Rabbit Order, IPDPS 2016）给出了方法本身及其加速比；Balaji 与 Lucia（IISWC 2018）已提出"重排序何时是优化"并给出轻量方法在摊销下常优于重量方法的结论；近期 SpMV 与 GNN 训练场景的实验研究把方法清单扩展到十余种，并报告了 RCM 约需 20 次执行回本这一量级的数字。这些结论本项目直接引用，不重复声称为发现。**历届丘奖对照**：计算机赛道 2021 年金奖《Efficient Algorithm for Parallel Bi-core Decomposition》（Phillips Academy）与 2022 年银奖《Efficient Parallel Density-Peak Clustering》（Phillips Academy / MIT PRIMES）是最相邻的两项——它们的共同形态是"为某个图/聚类问题设计一个更快的并行算法"。本课题的形态与之相反：**不设计任何算法，而是把已有方法的收益当作被解释的因变量**，判断依据是结构量的预测力而非加速比本身。这一差异必须在引言写明。

本项目的贡献：把问题方向从"哪种排序更快"转为"**能否用可先验计算的结构量预测收益**"，并用 LFR 合成图上的受控 μ 扫描把相关性证据升级为准因果证据。产出物是一条可操作的判据（例如"Q < 0.4 时任何重排序的摊销收益都为负"）及其置信区间。这是主结论。

为什么有价值：从业者面对新图时需要的是决策规则而不是排行榜。若判据成立，它可以直接写进图处理框架的启发式；若不成立（结构量无预测力），则证明收益由更细的微架构因素主导，这对后续研究是明确的方向排除。

风险提示：ρ 落在 0.3 – 0.6 之间的"弱相关"是最可能的结果。此时必须给出自助法置信区间证明本样本量（≥ 20 图）确有分辨 ρ=0.6 与 ρ=0 的统计功效；功效不足则须扩大图样本至 30 个而非改口径。

8 · 决策门槛（go / no-go）

- **第 5 周**：数据与编译门槛。若 GAP 无法编译或 SNAP 大图下载受阻，**降级路径 A**：改用 igraph（Python 绑定，pip 可装）+ 自行实现的 CSR PageRank/BFS 作为内核，图样本改以 SuiteSparse 的中等规模图（顶点 10^5 – 10^6）为主。加速比的绝对值会下降，但结构量—加速比的关系框架完全保留。
- **第 9 周**：Rabbit Order 可用性必须核实完毕——这是需要提前确认而非边做边发现的事项。若无法编译且自行实现超时，**降级路径 B**：把方法集缩为 Gorder + DegSort + RCM + HubSort 四种（后三种均可在 200 行内自行实现或由 SciPy 提供），主结论框架不变，仅在正文声明 Rabbit Order 未纳入及原因。
- **第 12 周**：方法学校验硬门槛。若三图重现的偏差 > 25% 或方法排序与文献不一致，排查频率控制、迭代数定义（是否用了收敛判据）、图预处理（是否去了重边）三个最常见原因；第 15 周仍不过关则**降级路径 C**：放弃跨文献绝对对照，改为纯内部对照设计（同一管线内比较"排序前 vs 排序后"），并把校验硬门槛替换为"在 LFR 合成图上验证 CSR 局部性指标（平均邻居编号距离）随排序单调下降"这一可解析验证的替代门槛。
- **第 30 周**：H1 必须有明确结论。若 ρ 的置信区间过宽无法判定，**降级路径 D**：把课题重心正式移到 H2（摊销回本次数的系统测定），这部分不依赖相关性显著性，且本身就是文献中缺少系统数字的方向；H1 降为次要分析并如实报告统计功效不足。主结论框架（摊销口径下的重排序决策）保留。
- **预算裁剪顺序**：结构量种类 → 图样本中的合成图数量 → 排序方法数量 → （绝不裁剪）重复次数与真实图样本量。

- **选择前提**：适合对图论有兴趣、能接受"结果可能是负相关或无相关"的学生。若学生期待"做出一个更快的算法"，这条路线不合适。

K03 · 基数约束 CNF 编码的求解成本敏感性：已发表编码排名在新求解器与新实例族上的独立检验

`推荐优先级：高` · `分族：算法设计与复杂度` · `参赛子类：计算机–算法与形式方法` · `资源需求：纯 CPU 笔记本 / 8 GB 内存即可` · `技能取向：Python 建模 + 组合数学 + 重尾统计`

1 · 研究问题

把组合问题编码为命题可满足性（SAT）实例时，"至多一个"（at-most-one, AMO）与基数约束（cardinality constraint）的六种经典 CNF 编码（pairwise、sequential/Sinz 计数器、commander、bimander、product、totalizer）之间的求解成本排名，在换用 2020 年后的现代 CDCL（conflict-driven clause learning）求解器 CaDiCaL / Kissat 并换用一组独立的实例族后，是否仍与文献报告的排名一致？若排名发生翻转，翻转能否由编码引入的辅助变量数与子句数这两个可先验计算的量解释？

2 · 研究背景与空白

技术背景。 大量组合问题（图着色、排课、拉丁方补全、Golomb 尺、装箱）在转成 SAT 之前必须把"这组布尔变量中至多有一个为真"这类计数条件翻译成 CNF 子句。最朴素的 pairwise 编码写出全部 C(n,2) 条互斥子句，子句数是平方级但不引入辅助变量；Sinz 的顺序计数器用 O(n) 个辅助变量把子句数降到线性；commander、bimander、product 编码在两者之间取折中；totalizer 与排序网络编码则支持一般的 $\leqslant k$ 约束。这里存在一个非平凡的权衡：子句数少不等于求解快，因为辅助变量会改变求解器的决策堆与子句学习行为，而**单位传播（unit propagation）的强度**——即某个编码能否在赋值后立刻推出更多蕴涵——往往才是决定性因素。这类课题的资源特性极其友好：SAT 求解是纯 CPU、单线程、内存需求通常在几百 MB 内的整数运算任务，**笔记本与服务器的唯一差别只是能跑多少个实例，而不是能不能跑**。学生只需要把时间预算换成实例数量。

已有工作到哪一步。 Nguyen 的《Empirical Study on SAT-Encodings of the At-Most-One Constraint》（SMA 2020）在三个当时主流求解器上系统比较了主要 AMO 编码；该研究在 Golomb 尺实例上用 Lingeling 求解器报告：commander 编码最快，binary 与 pairwise 编码也表现良好，而 product、bimander 与 sequential 编码表现很差。Wynn 的《A comparison of encodings for cardinality constraints in a SAT solver》（arXiv:1810.12975）比较了 Sinz 顺序计数器、Bailleux – Boufkhad 树形与 Abío 等的排序网络三类方法，结论是顺序计数器在一系列组合测试用例上最快；另有对照研究报告在成功案例中顺序计数器编码在 113 个实例上取得最优运行时、totalizer 在 92 个实例上最优。这些结论彼此并不完全一致，且**所用求解器多为 Lingeling、Clasp、Riss3G 等 2010 年代中期的版本**。

空白在于：现代 CDCL 求解器（CaDiCaL、Kissat，2019 年后多次夺得 SAT 竞赛冠军）引入了内联处理（inprocessing）、变量消去与子句消减等预处理，而这些技术恰恰会主动消掉编码引入的辅助变量——这意味着"哪种编码更好"的旧结论有可能已经被求解器的进步抹平或反转，但没有公开工作在新求解器上做过这项独立检验。这个空白属于最可靠的类型——**换样本的独立检验**（新求解器 + 新实例族）：方法完全公开、实例可自行生成、算力门槛极低、且结论正负都成立（排名稳定则说明编码选择是求解器无关的结构性因素；排名翻转则说明既有的编码选择建议已过时）。

3 · 可检验假设

- H1：在 CaDiCaL 与 Kissat 上，六种 AMO 编码之间的 PAR-2 分数（惩罚平均运行时，超时按 2 倍时限计）差异，比在 MiniSat（无内联处理的经典基线）上显著缩小——具体判据为：最优与最差编码的 PAR-2 比值，从 MiniSat 上的 $\geqslant 3$ 倍降至现代求解器上的 $\leqslant 1.8$ 倍。

- **H2**：若 H1 不成立（差异仍然巨大），则编码排名可由"辅助变量数 / 原变量数"这一比值单调预测：跨 4 个实例族 × 6 种编码的 24 个点上，PAR-2 排名与该比值的 Kendall $\tau \geqslant 0.5$。若两者都不成立，则说明编码优劣由实例族特性主导而无通用规律——这同样是有效结论，且直接否证了文献中"某编码普遍最好"的表述。

4 · 量化验收标准

1. **方法学校验（硬门槛）**：用自建的编码生成器 + 求解流水线，在文献使用过的实例族（Golomb 尺，阶数覆盖文献报告范围）上、用文献所用的同一代求解器（MiniSat 或 Glucose，可从官方仓库编译）重现已发表的编码排名。判据：**编码之间的相对排名（按中位求解时间）与文献报告完全一致**，且绝对时间在同一数量级（跨硬件允许 5 倍以内偏差）。这一步不过关，说明编码生成器本身有误，后续全部结论无效。
2. **编码正确性校验（第二道硬门槛）**：每种编码在 $n \leqslant 12$ 时必须通过**穷举等价性检验**——枚举全部 2^n 个原变量赋值，验证"存在辅助变量的延拓使公式为真"当且仅当该赋值满足原约束。这是编码类课题唯一能排除"跑得快是因为编错了"的证据，必须写进正文。
3. **重尾统计口径**：SAT 求解时间是重尾分布，**明确写明不报均值**。统计量固定为：中位数与 IQR、PAR-2 分数、仙人掌图（cactus plot，横轴已解实例数 / 纵轴累计时间）、以及每实例的成对比较（同一实例上编码 A 是否快于 B）用 Wilcoxon 符号秩检验。超时统一设为 300 秒并在正文说明该阈值对结论的敏感性（额外用 600 秒重跑最难的 20% 实例）。
4. **随机性控制**：现代求解器含随机化成分，每个（实例 × 编码 × 求解器）组合用 5 个不同的固定求解器种子各跑一次，报告 5 次的中位数；同时报告种子内方差与编码间差异的量级对比——若种子方差大于编码差异，则任何编码排名都不成立，这一判断必须在正文中给出。
5. **样本量**：$\geqslant 4$ 个实例族（Golomb 尺、图着色、拉丁方补全、鸽巢/装箱各一），每族 $\geqslant 30$ 个规模递增实例，总实例数 $\geqslant 120$；实例生成脚本与种子公开。
6. **双成本口径**：与文献对照时报**等工作量**（相同实例集上的求解时间）与**等墙钟时间预算**（固定 1 小时总预算下各编码能解出的实例数）两种口径，两者结论不一致时必须讨论。
7. **可复现性**：编码生成器、实例生成器、全部 CNF 文件的生成脚本（不必上传 CNF 本身）、原始计时 CSV、绘图脚本全部开源；一键重跑脚本；全部种子写死。

5 · 数据与工具

用途	来源 / 工具
现代 CDCL 求解器	CaDiCaL（GitHub `arminbiere/cadical`，C++，无外部依赖，`./configure && make` 即可）；Kissat（GitHub `arminbiere/kissat`，C，同样零依赖）。**仅用于校验与对比，不计入本项目贡献**
历史基线求解器	MiniSat 2.2 / Glucose（用于复现文献排名；老代码在新编译器上可能需小改，**列为风险项，需核实**）
编码生成	自行实现六种编码（Python，每种 30–80 行）。可用 PySAT（`python-sat` 包）的 `pysat.card` 模块做**独立交叉校验**——PySAT 内置 seqcounter、sortnetwrk、cardnetwrk、totalizer、mtotalizer、kmtotalizer 等编码，但 **commander / bimander / product 等 AMO 变体是否全部内置需核实**，未内置的必须自行实现
实例族	Golomb 尺（自行生成，规则明确）；图着色实例用 DIMACS 图着色基准（公开，mat.tepper.cmu.edu / DIMACS 官方镜像）；拉丁方补全与鸽巢实例自行生成（参数化脚本）；另可从 SAT Competition 官方基准归档中挑选组合类实例（satcompetition.org，实例文件较大，按需下载）
穷举等价性检验	自行实现（$n \leqslant 12$ 时 $2^n \leqslant 4096$，毫秒级）
统计与作图	Python + SciPy（Wilcoxon 符号秩）+ Matplotlib（仙人掌图）
算力	纯 CPU 单线程。单实例内存通常 < 500 MB；总工作量 = 120 实例 × 6 编码 × 3 求解器 × 5 种子 × ≤ 300 秒 ≈ 最坏 900 小时，**必须通过"先跑小规模筛掉平凡实例、只对非平凡区间做全扫描"控制**，实际目标控制在 60–100 机时（夜间批跑两周内完成）。这一算力规划本身要写进方法路径

6 · 方法路径

1. 装环境，编译 CaDiCaL / Kissat / MiniSat，用各自仓库自带的算例确认安装正确（求解结果与官方期望一致）；跑通 PySAT 并对比其内置编码与自实现编码在小 n 上的等价性。
2. 实现六种 AMO/基数编码，逐一通过 $n \leqslant 12$ 的穷举等价性检验（验收第 2 条）；此步不过不进入下一步。
3. 核实工具能力边界：PySAT 内置了哪些编码、缺哪些需自行实现；CaDiCaL/Kissat 的内联处理开关是否可关闭（这决定 H1 能否做"开/关内联处理"的受控对照——若可关闭，这是 H1 最直接的因果检验，须优先确认）。
4. 生成实例族并做**难度定标**：对每族先用二分搜索找到"中位求解时间落在 1 – 100 秒"的规模区间，只在该区间内取 30 个实例。这是控制总算力的关键步骤，判据须写死并公开（避免选择性挑实例的嫌疑）。
5. 完成方法学校验（验收第 1 条）：在 MiniSat/Glucose 上重现文献排名并归档。
6. 执行主扫描并做重尾统计分析：PAR-2、仙人掌图、成对 Wilcoxon 检验、种子方差与编码差异的量级对比；输出 H1/H2 的判定。
7. 独立交叉校验：（a）用 PySAT 内置编码复算至少两种编码的结果，验证自实现无误；（b）若第 3 步确认可关闭内联处理，在 CaDiCaL 上做"内联处理开 vs 关"的受控对照，直接检验"求解器进步抹平编码差异"这一机制假设。

7 · 新颖性边界

本课题不声称：不提出新的编码方案，不声称任何编码"最优"，不改进求解器，不给出任何新的复杂度下界（AMO 编码的 CNF 大小下界已有专门工作，见 arXiv:1704.08934，本项目不涉足）。CaDiCaL、Kissat、PySAT、DIMACS 基准均为他人工作，标注为对照基准，不计入本项目贡献。

已有工作完成了什么：Nguyen（SMA 2020）在 Lingeling 等三个求解器上给出了 AMO 编码的经验排名，包括 Golomb 尺上 commander 最优、sequential 与 product 很差这一具体结论；Wynn（arXiv:1810.12975）比较了三类基数约束编码并报告顺序计数器最快；另有工作报告顺序计数器与 totalizer 在不同实例上各占优（113 vs 92 例）。这些结论本项目引用而不重复声称。

本项目的贡献：把这些排名**放到 2020 年后的求解器（CaDiCaL / Kissat）与一组独立实例族上做重现检验**，并用"内联处理开/关"的受控对照给出机制解释。主结论是"既有编码选择建议在现代求解器上是否仍然有效"这一判断本身，而不是任何新编码。

为什么有价值：编码选择是每一个把组合问题交给 SAT 求解器的人都要做的决定，现行教科书与工具默认值大多沿用 2010 年代的经验结论。若排名稳定，这些建议获得一次独立支持；若被现代求解器抹平，则"编码调优"这一整类工程努力的边际收益需要重新评估。

风险提示："六种编码差异不显著"是完全可能且有价值的结果，但必须先通过第 4 条的种子方差对比证明本实验有能力分辨题设中 1.8 倍这一量级的差异；若种子方差本身就超过 1.8 倍，则必须扩大种子数或实例数，不得直接下"无差异"结论。

8 · 决策门槛（go / no-go）

- **第 4 周**：编码正确性门槛。若六种编码中有任何一种无法通过 $n \leqslant 12$ 的穷举等价性检验且两周内修不好，**降级路径 A**：把该编码替换为 PySAT 内置的等价编码并在正文声明来源，编码集缩为 5 种自实现 + 1 种引用。主结论框架不变。
- **第 8 周**：求解器可用性与内联处理开关必须核实完毕——这是需要提前确认而非边做边发现的事项。若 MiniSat/Glucose 无法在现代编译器上构建，**降级路径 B**：历史基线改用 CaDiCaL 的早期版本（仓库有 tag，可回退编译）或 Glucose 的社区维护分支；若都失败，则把"新旧求解器对照"这一轴替换为"内联处理开/关"对照——后者在同一份代码内完成，机制解释力反而更强，主结论框架完全保留。
- **第 14 周**：难度定标必须完成，且总算力预算必须落在 100 机时以内。若定标显示某实例族在可行规模内全部秒解或全部超时，**降级路径 C**：替换该族（备选：装箱、幻方补全、图的支配集），实例族数量下限保持 4 个。
- **第 32 周**：主扫描完成且 H1 有明确判定。若此时扫描进度不足 60%，**降级路径 D**：把求解器从 3 个减为 2 个（CaDiCaL + MiniSat，保留"新 vs 旧"这一核心对照），种子数从 5 减为 3，实例族保持 4 个。宁可减求解器数量，不可减实例族数量——跨族一致性是本课题反驳"某编码普遍最好"的核心证据。
- **预算裁剪顺序**：求解器数量 → 种子数量 → 每族实例数（下限 20）→ （绝不裁剪）实例族数量与穷举等价性检验。
- **选择前提**：适合喜欢离散数学与组合问题、能接受"主要产出是一张严谨的否证性表格而非一个新算法"的学生。本路线代码量最小、算力需求最低，是全部十条中最不容易因工程问题失败的一条，但对统计严谨性要求最高。

K04 · 开源 C 静态分析器的误报—漏报前沿：以匹配判据敏感性分析重测 Juliet 基准上的已发表评测结论

`推荐优先级：高` · `分族：编译与程序分析` · `参赛子类：计算机–软件工程与安全` · `资源需求：纯 CPU 笔记本 / 16 GB 内存 / 大量磁盘 I/O` · `技能取向：脚本工程 + 静态分析概念 + 评测方法学`

1 · 研究问题

已发表评测报告的开源 C/C++ 静态分析工具（Clang Static Analyzer、cppcheck、Infer、CodeChecker）在 Juliet 测试集上的精确率与召回率数字，对"工具告警与基准 manifest 的匹配判据"（严格同行 / ±k 行 / 同函数 / 同文件）有多敏感？改变匹配判据是否足以改变工具之间的排名，从而使既有的"哪个工具更好"的结论不成立？

2 · 研究背景与空白

技术背景。 静态应用安全测试（SAST）工具在不运行程序的前提下，用抽象解释、符号执行或模式匹配寻找缓冲区溢出、空指针解引用、未初始化变量等缺陷。评价这类工具需要"标注了缺陷位置的代码"，业界标准是 NIST 的 Juliet 测试集（隶属 SARD，Software Assurance Reference Dataset）：Juliet C/C++ 含约 64,099 个测试用例、分布在约十万个文件中，每个用例按 CWE 分类，且**成对给出 `bad`（含缺陷）与 `good`（已修复）两个函数**，并附 manifest 标明缺陷的精确文件与行号。评测时把工具告警与 manifest 比对：位置匹配算真阳性，其他算假阳性。这里的关键——也是本课题的切入点——是"位置匹配"这个判据本身没有统一定义：告警报在缺陷行的上一行算不算命中？报在同一函数的另一行算不算？不同论文的选择不同，而这个选择直接决定了最终数字。**这类课题完全绕开算力墙**：静态分析是 CPU 上的符号推理，慢在分析器本身而不在硬件，且 Juliet 的单个测试用例都是几十行的小文件。

已有工作到哪一步。 已有多项系统评测：一项研究以 manifest 的精确文件与行号为判据，报告 Clang Static Analyzer 精确率极高（约 87%）但召回率很低（约 9%），且其覆盖的 CWE 类别极少（对某些测试集只支持 CWE-457 未初始化变量）；cppcheck 呈现"高精确率、中等召回率"；cppcheck、CodeChecker 与 Infer 属于检出漏洞最少的一组；SonarQube 的假阳性数约为 CodeQL 的 2.7 倍。综述性结论是"C 语言不存在单一最优的开源 SAST 工具"。近期还出现了 CASTLE 基准（arXiv:2503.09433），专门为静态分析器与大语言模型的 CWE 检测比较构造新数据集。同时，Juliet 作为基准的局限已被指出：合成代码模式单一、易被模式匹配"猜中"。

空白在于：全部已发表评测都固定了一种匹配判据，没有任何工作把"匹配判据"本身当作自变量做敏感性分析，因此没人知道这些被广泛引用的精确率/召回率数字有多脆弱。这个空白属于"评价维度的转换"，可靠性高，而且它有一个别的方向少有的优势：Juliet 的 good/bad 配对结构提供了一个**判据无关的强证据**——工具在 `good`（已修复）函数上发出的告警必定是真误报，与行号匹配方式无关。用这个不可争议的子集做锚点，就能定量测出各种匹配判据引入的偏置。学生课题做这件事的门槛很低（全部是脚本与批处理），且结论正负都成立。

3 · 可检验假设

- H1：把匹配判据从"严格同行"放宽到"同函数"，各工具的召回率提升幅度显著不同（提升幅度的极差 ⩾ 15 个百分点），且至少发生一次工具排名翻转（按 F1 或精确率-召回率前沿上的支配关系判定）。
- H2：在判据无关的 `good` 函数子集上测得的真误报率，与用严格同行判据算出的假阳性率之间存在系统性偏差，偏差方向对所有工具一致（即严格判据系统性高估误报），偏差中位数 ⩾ 10 个百分点。若偏差方向

不一致，说明匹配判据的影响是工具特异的，这对"跨论文引用精确率数字"是一个更强的警告——同样是有效结论。

4 · 量化验收标准

1. **方法学校验（硬门槛）**：用自建评测管线，在与已发表研究相同的 Juliet 子集与相同的匹配判据（严格文件+行号）下，重现至少两个工具（Clang Static Analyzer 与 cppcheck）的精确率与召回率，**与已发表数值的绝对偏差 ⩽ 5 个百分点**；工具版本、编译数据库生成方式、启用的检查器清单必须与原文一致（原文未写明处须列为不确定项并做双版本对照）。这一步不过关，后续全部结论无效。
2. **判据敏感性扫描**：匹配判据至少 5 档（严格同行 / ±2 行 / ±5 行 / 同函数 / 同文件），每档给出全部工具的精确率、召回率、F1 与前沿图；每档的差异必须给自助法（⩾ 2000 次重采样，按测试用例重采样而非按告警重采样）95% 置信区间。
3. **分层报告**：结果必须**按 CWE 类别分层报告**，不得只给汇总数字。理由写进正文：Juliet 各 CWE 的用例数量极不均衡，汇总数字实际上由用例最多的几类主导。同时报告每个工具"完全不支持"的 CWE 类别清单——覆盖面差异是排名的主要来源之一。
4. **不平衡口径**：缺陷检测是极不平衡问题（Juliet 的 good/bad 配对使正负比例接近 1:1，但真实代码远非如此）。因此**同时报告两套数字**：Juliet 原生配对下的精确率/召回率，以及按 1:20、1:100 的正负比重采样后的 PR-AUC。**明确写明不报 accuracy 及原因**（配对结构下全部答"无缺陷"即可得 50%）。
5. **运行成本记录**：记录每个工具在完整 Juliet 上的墙钟时间、峰值内存与超时/崩溃用例数（这也是从业者的实际选型依据），报中位数与 IQR，重复 ⩾ 5 次。
6. **可复现性**：全部批处理脚本、告警解析器（各工具输出格式不同，需统一为 SARIF 或自定义 JSON）、匹配判据实现、原始告警数据与统计脚本开源；一键重跑脚本；工具版本用容器镜像或明确的版本号锁定。

5 · 数据与工具

用途	来源 / 工具
基准数据集	NIST Juliet Test Suite for C/C++（v1.3，从 NIST SARD 站点 samate.nist.gov 免费下载，约 64,099 个用例 / 十万级文件 / 解压后数 GB），含 manifest.xml 标注缺陷位置与 CWE 编号。**仅用于校验与对比，不计入本项目的数据贡献**
补充基准	CASTLE 基准（arXiv:2503.09433 附带数据，**开源许可与获取方式需核实**）可作为第二数据集验证结论的可迁移性
静态分析器	Clang Static Analyzer（LLVM 官方，`scan-build` 或 `clang --analyze`，apt/brew 可装）；cppcheck（开源，含 `--enable=all`）；Infer（Meta 开源，二进制发行版）；CodeChecker（Ericsson 开源，Clang SA/tidy 的驱动与结果管理，可直接产出 SARIF）。**全部为他人工具，标注为对照基准**
编译数据库	`bear` 或 CMake 的 `CMAKE_EXPORT_COMPILE_COMMANDS` 生成 `compile_commands.json`（Juliet 的构建方式非标准，**需核实是否需自行编写编译数据库生成脚本，这是本课题的已知工程难点**）
告警格式统一	SARIF（OASIS 标准，Clang/CodeChecker 原生支持；cppcheck 有 XML 需自行转换；Infer 输出 JSON 需自行转换）——**转换器需自行实现，须用人工核对的样本验证**
统计与作图	Python + SciPy + scikit-learn（PR-AUC）+ Matplotlib
算力	纯 CPU。瓶颈是 I/O 与分析器本身：完整 Juliet 上 cppcheck 约数十分钟至数小时、Clang SA 与 Infer 可能达十余小时（**具体数字需在第 1 阶段实测并记录，不得凭估计**）。磁盘需 ⩾ 30 GB 空闲。全部可夜间批跑，**超出范围**：需要构建整个真实开源项目（如 Linux 内核）的评测不在本课题范围内

6 · 方法路径

1. 装齐四个分析器与 Juliet，跑通每个工具在 10 个手挑用例上的分析，人工核对告警位置与 manifest，确认解析器无误；实测并记录各工具的完整运行成本。
2. 解决编译数据库问题（本课题的第一道工程关卡）：为 Juliet 的目录结构编写批量编译数据库生成脚本，并核实 Clang SA 与 Infer 是否能正确处理 Juliet 中的 `#ifdef` 变体（Juliet 用宏切换 good/bad 路径，**若处理不当会系统性影响所有结果，必须在此步核实清楚**）。
3. 实现统一的告警解析与 SARIF 转换，并用至少 50 条人工标注的告警验证转换正确性（给出明确、可复现的核对判据与核对记录）。
4. 完成方法学校验（验收第 1 条）：在严格同行判据下重现两个工具的已发表数字并归档差异分析。
5. 实现 5 档匹配判据并执行敏感性扫描，按 CWE 分层输出精确率/召回率/F1 与前沿图，全部带自助法置信区间。
6. 构造判据无关的锚点分析：只统计 `good` 函数上的告警（必为真误报），据此定量给出各判据的偏置方向与幅度，检验 H2。
7. 独立交叉校验：在 CASTLE 或 SARD 的另一子集上重跑判据敏感性扫描，验证结论是否跨数据集成立；并对随机抽取的 100 条告警做人工判读，给出"自动匹配 vs 人工判读"的一致率（Cohen's κ），量化自动评测本身的可靠性上界。

7 · 新颖性边界

本课题不声称：不开发新的静态分析器，不改进任何工具的算法，不声称哪个工具"最好"，不构造新的漏洞基准。Juliet、SARD、CASTLE 与全部四个分析器均为他人工作，明确标注为对照基准，不计入本项目贡献。

已有工作完成了什么：多项系统评测已在固定匹配判据下给出各工具在 Juliet 上的精确率/召回率，包括 Clang SA 精确率约 87%、召回率约 9%，cppcheck 高精确率中等召回，Infer/cppcheck/CodeChecker 检出数最少，SonarQube 误报数约为 CodeQL 的 2.7 倍，以及"C 语言无单一最优开源 SAST 工具"这一综述结论；CASTLE（2025）构造了新的 CWE 检测基准。这些数字本项目直接引用，不重复声称为发现。

本项目的贡献：把**匹配判据本身**作为自变量，测量既有精确率/召回率数字对它的敏感性，并用 Juliet 的 good/bad 配对结构构造一个判据无关的锚点来定量各判据的偏置。主结论是"这些被广泛引用的评测数字有多可信、排名是否稳健"，而不是任何新工具或新数据。

为什么有价值：静态分析工具的选型在工业界直接依赖这些公开数字。如果排名会随一个从未被讨论的方法学选择而翻转，那么整类评测都需要在报告结果时同时报告判据敏感性——这是一个可以立刻被后续研究采纳的方法学建议。

风险提示：若各判据下排名完全稳定，"结论稳健"同样是有价值的正面结论（它为既有文献提供了一次方法学背书），但必须给出置信区间证明本实验有能力分辨题设中 15 个百分点这一量级的差异。

8 · 决策门槛（go / no-go）

- **第 6 周**：编译数据库与 `#ifdef` 处理必须核实完毕——这是需要提前确认而非边做边发现的头号风险项。若 Clang SA / Infer 无法正确分析 Juliet 的宏变体结构，**降级路径 A**：把 Juliet 用例预处理为"每个用例只保留 bad 或 good 一个变体"的展开版本（脚本可实现），并在正文声明该预处理及其对可比性的影响。主结论框架不变。
- **第 10 周**：方法学校验硬门槛。若两个工具的精确率/召回率与文献偏差 > 5 个百分点，先排查工具版本、检查器启用清单、Juliet 版本三项；到第 13 周仍不过关则**降级路径 B**：放弃与外部文献的绝对数值对照，改为**内部一致性校验**——用人工标注的 200 条告警金标准（学生自建，须双人独立标注并报 κ）作为校验锚点，硬门槛替换为"自动管线与人工金标准的一致率 $\geqslant$ 90%"。主结论（判据敏感性）完全保留。
- **第 16 周**：工具集必须锁定。若 Infer 或 CodeChecker 安装/运行失败，**降级路径 C**：工具集缩为 Clang SA + cppcheck + clang-tidy（三者均随 LLVM 发行，安装最稳），下限为 3 个工具——少于 3 个则"排名翻转"这一核心命题无从检验，此时须整体改投 K03。
- **第 30 周**：判据敏感性扫描完成且 H1 有明确判定。若运行成本超预算（分析器跑不完），**降级路径 D**：把 Juliet 按 CWE 分层抽样为 30% 子集（分层抽样脚本与种子公开），保证每个 CWE 类别的用例数下限；抽样对精确率/召回率的估计误差用自助法量化并在正文报告。
- **预算裁剪顺序**：补充数据集（CASTLE）→ 工具数量（下限 3）→ Juliet 抽样比例（下限 30%）→ （绝不裁剪）匹配判据档数与人工核对样本量。
- **选择前提**：适合耐心好、擅长脚本与数据清洗、能忍受"大部分时间在处理格式与构建系统"的学生。本路线的科学内容清晰但工程杂活多，不适合期待算法推导的学生。

K05 · 无锁并发队列的尾延迟归因：以缓存行隔离受控实验测定伪共享在异构多核笔记本上对 p99.9 延迟的贡献份额

`推荐优先级：中高` · `分族：并发与系统性能测量` · `参赛子类：计算机–系统与性能` · `资源需求：纯 CPU 多核笔记本（异构核心更佳）/ 8 GB 内存` · `技能取向：C++ 并发 + 微架构 + 重尾统计`

1 · 研究问题

在现代**异构多核**（heterogeneous / hybrid core，如性能核 P-core 与能效核 E-core 混合）笔记本 CPU 上，若干开源无锁（lock-free）并发队列实现的入队 – 出队延迟尾部（p99、p99.9），有多大比例可归因于伪共享（false sharing）？把生产者索引与消费者索引做缓存行隔离（padding）这一单一改动，对尾延迟的改善幅度能否由"生产者与消费者所在核之间的拓扑距离"（同一物理核的两个超线程 / 同一 L2 簇内的两核 / 跨簇 / 跨核类型）单调预测？

2 · 研究背景与空白

技术背景。 无锁队列用原子读改写指令（compare-and-swap、fetch-and-add）代替互斥锁，让多个线程并发入队出队而不阻塞。它的性能上限不由指令数决定，而由**缓存一致性协议的消息往返**决定：当生产者写头指针、消费者写尾指针，若两者落在同一条 64 字节缓存行内，每次写都会使对方核上的这条行失效，触发一次跨核的所有权转移——这就是伪共享。伪共享不改变平均吞吐太多，但会把延迟分布的尾部拉长若干倍，而尾延迟才是实时与交易类系统真正关心的量。已有工程报告显示：在严格核绑定条件下，一个生产级无锁队列因微架构伪共享出现约 3.1 倍的性能塌陷，用 128 字节空间隔离加影子变量批处理可以打破该瓶颈；同核双线程通信延迟最低，跨核后延迟增加 3 倍以上，跨 CCX（core complex）或跨插槽进一步增加。**这类课题对纯 CPU 笔记本是最优匹配**：它测的是多核之间的缓存一致性行为，一台八核笔记本与一台六十四核服务器在"两核之间的一次往返要多少周期"这个基本量上是同等可测的；而且笔记本上的**异构核心结构反而提供了服务器上没有的自变量维度**。

已有工作到哪一步。 学术侧有针对多核通信 API 的无锁算法性能影响研究（arXiv:1401.6100）与近期的免协调无锁队列工作（arXiv:2511.09410）；工程侧有成体系的无锁队列基准（`max0x7ba/atomic_queue` 提供吞吐与延迟基准与公开数据、`psy-lob-saw` 系列的队列分析、多篇对阻塞与无锁队列的对比测评）。这些工作的共同特征是：（a）主要在**同构服务器 CPU**上测量；（b）主要报告吞吐量与平均延迟，尾延迟即使给出也很少做分布形态分析；（c）把伪共享当作已知的设计注意事项（"要加 padding"），而不是当作一个被定量归因的自变量。

空白在于：没有公开工作把"缓存行隔离"作为受控变量、在**异构核心**拓扑上系统测量它对延迟分布尾部的贡献份额。这个空白同时占了两个可靠类型——**换样本**（异构核心是 2021 年后才普及的新硬件类别）与**评价维度转换**（从吞吐/均值转到尾部分位数与分布形态）。它对学生课题极友好：受控实验设计简单（改一个 `alignas` 就是全部处理组差异）、算力零门槛、且结论正负都成立。

3 · 可检验假设

- H1：移除缓存行隔离后，p99.9 延迟的中位增幅 $\geqslant 2.0$ 倍，而中位延迟（p50）的增幅 $\leqslant 1.3$ 倍——即伪共享的代价**主要落在分布尾部而非中心**。若 p50 与 p99.9 的增幅比值接近 1，H1 被否证，说明伪共享是均匀抬升而非尾部效应。

- H2：隔离带来的 p99.9 改善幅度随生产者 – 消费者核对的拓扑距离单调不减：同物理核超线程对 < 同簇不同核 < 跨簇同类型核 < 跨核类型（P↔E）。判据为 4 档拓扑上的 Jonckheere – Terpstra 趋势检验 $p < 0.05$。若在跨 P/E 核这一档出现反转，则说明异构调度的频率差异盖过了一致性效应——这是对异构 CPU 上并发设计的一条独立发现，同样有效。

4 · 量化验收标准

1. **方法学校验（硬门槛）**：用自建基准骨架重现 `atomic_queue` 项目公开发布的吞吐与延迟基准结果。因跨硬件绝对值不可比，判据定为两条同时满足：（a）在本机上各队列实现的**相对排序**与官方公布结果一致；（b）"同物理核两超线程之间一次乒乓往返（ping-pong round trip）的延迟"这一硬件基本量，与用独立的最小化 `std::atomic` 乒乓程序测得的值偏差 $\leqslant 10\%$。这一步不过关，说明计时或绑核有系统误差，后续全部结论无效。
2. **正确性门槛（第二道硬门槛）**：任何被测队列（含自行修改 padding 的版本）必须先通过**线性一致性抽查**——记录带时间戳的入队/出队操作历史，用暴力搜索验证存在一个合法的顺序化重排（历史长度 $\leqslant 12$ 时可穷举），至少 1000 条随机历史全部通过。修改过的实现若未通过此检验，其性能数据一律作废。
3. **基准测试方法学**：每个（队列 × padding 条件 × 核对 × 负载强度）组合独立重复 $\geqslant 30$ 次，每次记录 $\geqslant 10^6$ 次操作的**逐次延迟**（不做在线聚合，保留全分布）；固定 CPU 频率与调度策略（`taskset` 绑核 + `chrt` 实时优先级，**需核实笔记本 OS 上是否可用**）、禁用节能与 Turbo、每次运行前预热 $\geqslant 2$ 秒并丢弃预热段；报 p50 / p90 / p99 / p99.9 与 IQR，**明确写明不报均值与标准差**（延迟分布重尾，均值无代表性）；组间比较用 Mann – Whitney U 与分位数自助法置信区间。
4. **温度与频率漂移控制**：全程记录 CPU 温度与实际频率（`powermetrics` / `turbostat` / `/proc/cpuinfo`），每轮运行之间强制冷却间隔；若某轮的频率中位数偏离基线 > 5%，该轮数据标记并在敏感性分析中剔除，剔除比例须报告。这一条是笔记本平台特有的必需项，必须写进正文。
5. **双成本口径**：与已发表实现对照时同时报**等工作量**（相同操作条数下的延迟分布）与**等墙钟时间预算**（固定 10 秒内完成的操作数及其延迟分布），两种口径都要给出。
6. **可复现性**：全部队列源码（含改动 diff）、基准骨架、绑核脚本、原始逐次延迟数据（压缩后 CSV/二进制）、分析与绘图脚本开源；一键重跑脚本；随机种子与核对映射写死在配置中；硬件与 OS 版本、微码版本完整披露。

5 · 数据与工具

用途	来源 / 工具
队列实现（对照基准）	`max0x7ba/atomic_queue`（C++14，header-only，含官方基准与公开结果）；`cameron314/concurrentqueue`（moodycamel MPMC）；`boost::lockfree::queue`（Boost 提供）；一个自行实现的 Michael–Scott 队列与一个 Lamport SPSC 环形缓冲（作为教科书基线）。**外部实现仅用于校验与对比，不计入本项目贡献**
拓扑发现	`hwloc` / `lstopo`（识别物理核、超线程兄弟、L2/L3 共享关系）；Linux `/sys/devices/system/cpu/*/topology`；异构核心的类型识别需查 OS 特定接口（**Apple Silicon 上的 P/E 核绑定 API 与 Linux 上的 `cpu_capacity` 均需核实**）
计时	`rdtsc` / `rdtscp`（x86）或 `cntvct_el0`（ARM），须先校准计数器频率并验证跨核一致性（**非常关键：跨核 TSC 是否同步需实测确认，不同步则单向延迟不可测，只能测往返延迟**）
微架构计数器	Linux `perf`（`cache-misses`、`mem_load_l3_miss`、HITM 事件是伪共享的直接证据，**HITM 事件在消费级笔记本上是否可用需核实**）；`perf c2c`（专为伪共享设计的分析工具，若可用则是最强证据）
线性一致性检验	自行实现的小规模穷举检验器（历史长度 ⩽ 12）；可参考 Herlihy–Wing 线性一致性定义
统计与作图	Python + SciPy（Mann–Whitney、Jonckheere–Terpstra 趋势检验）+ Matplotlib（延迟分布的互补累积分布函数 CCDF 图，对数纵轴）
算力	纯 CPU，内存需求 < 1 GB。总工作量约 5 队列 × 2 padding × 4 核对 × 3 负载 × 30 重复 × 数秒 ≈ 10–20 机时，可分夜跑。**超出范围**：NUMA 效应、跨插槽通信、大规模线程数（> 物理核数）不在本课题范围，须在正文明确声明

6 · 方法路径

1. 装环境与工具链，编译各队列实现，跑通 `atomic_queue` 自带基准；用 `lstopo` 画出本机拓扑图并写入正文；实测 TSC 跨核同步性与频率校准。
2. 核实工具能力边界（本课题风险最集中的一步）：`perf c2c` 与 HITM 事件是否可用；实时调度优先级是否可设；异构核心的显式绑定 API 是否存在。任一不可用都要在此步写明替代方案，不得边做边发现。
3. 实现基准骨架：绑核、预热、逐次延迟记录（用预分配数组避免测量本身引入分配）、温度/频率监控、防编译器优化屏障；用最小化乒乓程序完成验收第 1 条的硬件基本量校验。
4. 制备处理组：对每个队列产出"有缓存行隔离"与"无缓存行隔离"两个版本（改动限于 `alignas`/padding 字段，diff 必须最小且公开），逐一通过线性一致性抽查。
5. 执行主扫描（队列 × padding × 4 档核对 × 3 档负载强度 × 30 重复），同步采集 perf 计数器与温度/频率日志。
6. 分析：CCDF 图、分位数表（p50/p90/p99/p99.9 + 自助法置信区间）、H1 的尾/中心增幅比、H2 的拓扑趋势检验、双成本口径对照表；剔除频率漂移轮次并报告敏感性。
7. 独立交叉校验：（a）用 `perf c2c` 或 HITM 计数直接验证"无隔离版本的缓存行竞争确实更高"，把性能证据与微架构证据对上；（b）在另一台不同架构的机器（同学的 Intel/AMD/Apple Silicon 笔记本）上重跑关键组合，检验拓扑趋势是否跨架构成立。

7 · 新颖性边界

> **本课题不声称**：不提出新的无锁队列算法，不声称任何队列实现更优，不声称"应该加 padding"这一工程常识是本项目发现（它是众所周知的）。`atomic_queue`、moodycamel、Boost.Lockfree 与全部微架构工具均为他人工作，标注为对照基准，不计入本项目贡献。
>
> **已有工作完成了什么**：学术侧已给出无锁算法对多核通信 API 性能的影响（arXiv:1401.6100）与新的免协调无锁队列设计（arXiv:2511.09410）；工程侧的公开基准给出了各队列在同构服务器 CPU 上的吞吐与平均延迟排名，并有报告指出严格 NUMA 绑核下伪共享可造成约 3.1 倍的性能塌陷、128 字节隔离可打破该瓶颈，以及跨核通信延迟比同核高 3 倍以上这一量级。这些结论本项目引用而不重复声称。
>
> **本项目的贡献**：把伪共享从"设计注意事项"变成**受控自变量**，在**异构核心笔记本 CPU**这一新硬件类别上，定量给出它对延迟分布**尾部**（p99.9）的贡献份额，并检验该份额是否随核间拓扑距离单调变化。主结论是这份"尾延迟归因表"与拓扑趋势判定，不是任何新实现。
>
> **为什么有价值**：异构核心已是消费级与移动端 CPU 的标配，而并发数据结构的性能经验几乎全部来自同构服务器。若拓扑趋势成立，它给出一条可操作的线程放置建议；若在 P↔E 核这一档反转，则说明异构平台上的并发调优需要与服务器完全不同的直觉。
>
> **风险提示**：笔记本平台的噪声（温度节流、后台进程、OS 调度）是本课题最大的效度威胁。若剔除频率漂移后剩余样本不足或组间差异落在噪声内，"无法分辨"必须如实报告，并给出分位数自助法置信区间证明所需样本量——不得用增加重复次数以外的方式凑显著性。

8 · 决策门槛（go / no-go）

- **第 5 周**：计时基础设施门槛。若 TSC 跨核不同步或频率无法锁定，**降级路径 A**：全部改测**往返延迟**（ping-pong，只需单核时钟）而非单向延迟，并把负载模型从"生产者持续推、消费者持续拉"改为"配对请求 – 应答"。主结论框架（伪共享对尾部的贡献 × 拓扑距离）完全保留，仅测量语义从单向改为往返，须在正文声明。
- **第 8 周**：微架构证据可用性必须核实完毕。若 `perf c2c` /HITM 不可用，**降级路径 B**：改用"缓存缺失总数 + 受控消融"作为间接证据——即用一个刻意制造伪共享的合成微基准（两个线程反复写同一/不同缓存行）标定本机上伪共享的延迟代价上界，再以该标定值解释队列上的观测。此路径的证据强度略低但完整自洽，主结论不变。
- **第 12 周**：正确性门槛。若自行修改 padding 的版本无法通过线性一致性抽查，**降级路径 C**：放弃修改第三方实现，改为只在**自行实现的 Michael – Scott 队列与 SPSC 环形缓冲**上做 padding 消融（自己的代码可控），第三方实现只作为"当前工程实践的参照点"以原样参与横向比较。主结论框架保留。
- **第 24 周**：主扫描完成且 H1 有明确判定。若噪声导致数据大量作废（作废率 > 40%），**降级路径 D**：缩减到 2 个队列 × 2 档拓扑（同核超线程 vs 跨核类型，即差异最大的两档），把节省的时间全部投入重复次数（提到 100 次）与冷却间隔。**宁可减少条件数也不减少重复次数**——本课题的全部价值建立在能否从噪声中分辨出效应。
- **预算裁剪顺序**：负载强度档数 → 队列实现数量（下限 2）→ 拓扑档数（下限 2）→ （绝不裁剪）重复次数、频率控制与线性一致性检验。
- **选择前提**：仅在学生已能写 C++ 多线程代码、且明确接受"笔记本噪声可能导致效应无法分辨"这一风险时才启动。若学生的机器是单一同构核心的旧款 CPU，H2 无从检验，应改选 K01。

K06 · 近似中介中心性算法的误差—成本前沿：从 ε-绝对误差保证转向 Top-k 排名保真度的独立检验

推荐优先级：中高 · 分族：图算法与网络分析 · 参赛子类：计算机-算法与网络科学 · 资源需求：纯 CPU 笔记本 / 16 GB 内存 · 技能取向：Python/C++ 图算法 + 概率论 + 统计

1 · 研究问题

近似中介中心性（betweenness centrality, BC）算法 KADABRA、RK（Riondato – Kornaropoulos）与 ABRA 所提供的理论保证是"所有顶点的 BC 值绝对误差 $\leqslant \varepsilon$（置信度 $1-\delta$）"，但实际应用几乎只用**排名**。在给定的墙钟时间预算下，这三种算法给出的 **Top-k 顶点集合保真度**（与精确 Brandes 结果的交集率、Kendall τ、最大排名反转距离）孰优孰劣？其排序是否与文献中按"达到 ε 保证所需时间"给出的排序一致？

2 · 研究背景与空白

技术背景。 顶点 v 的中介中心性定义为经过 v 的最短路径对数占全部顶点对最短路径的比例，是网络科学中识别"桥接节点"的核心指标（用于交通枢纽、蛋白互作网络的关键蛋白、社交网络的信息中介）。Brandes 算法把精确计算降到 O(nm)（无权图），但对百万边规模的图仍需数小时。近似算法通过随机采样最短路径来估计：RK 依据网络直径确定所需样本量；ABRA 用 Rademacher 平均等经验界自适应决定何时停止；KADABRA 改造广度优先搜索的方式并采用不同的采样策略。这类课题在资源上完全可控：**图算法是纯 CPU 整数运算，且本课题的"精确基准真值"可以通过限制图规模（顶点数 $\leqslant 5\times10^4$ ）在笔记本上数小时内算出**，不需要任何 GPU 或集群。

已有工作到哪一步。 KADABRA（Borassi – Natale，ACM JEA 2018 / arXiv:1604.08553）报告：在真实复杂网络上显著优于此前全部方法，无向图上平均比 RK 快约 100 倍、有向图上约 70 倍，比 ABRA 快 1000 倍以上，且在误差容限趋于 0 时相对 RK 的优势进一步扩大；ABRA（Riondato – Upfal）报告其在运行时与样本数上均大幅优于同等质量保证的既有算法，并支持全动态图。另有专门比较近似方法速度与精度的实验综述（Computational Social Networks, 2019）、有向图上精确与近似算法的比较（arXiv:1708.08739），以及若干自适应估计与图神经网络排名方法（arXiv:1810.10094、arXiv:1905.10418）。

空白在于：这些比较几乎全部以"达到给定 ε 的绝对误差保证"为终点来衡量成本，而使用者真正需要的"在固定时间预算内谁的 Top-k 排名最准"这一等成本口径，缺少系统的公开测量。两者不等价：ε-保证是对**所有**顶点的一致界，而 Top-k 只关心分布右尾的少数顶点；一个算法可能为了保证尾部之外的精度而浪费大量样本。这个空白属于"评价维度转换"，可靠性高，且**结论正负都成立**（若排序不变，说明 ε-保证是排名质量的良好代理；若排序翻转，说明现有比较误导了应用侧的算法选择）。

3 · 可检验假设

- H1：在等墙钟时间预算口径下，跨 $\geqslant$ 15 个图的 Top-100 交集率排序与文献按 ε-保证成本给出的排序**不完全一致**：至少在 1/3 的图上发生排名翻转（即某个在 ε 口径下更慢的算法，在等时间口径下给出更高的 Top-100 交集率）。若翻转比例 < 10%，H1 被否证，ε-保证被确立为排名质量的良好代理。
- H2：Top-k 保真度的差异幅度随 k 减小而增大（k = 10 时的算法间差距 $\geqslant$ k = 1000 时的两倍），因为小 k 更依赖对分布极右尾的采样效率。若差距随 k 无系统变化，说明三种采样策略在尾部的行为等价。

4 · 量化验收标准

1. **方法学校验（硬门槛）**：（a）用自建管线在文献使用过的至少 3 个公共图上重现 KADABRA 相对 RK 的加速比，**与已发表数值同数量级且相对排序一致**（跨硬件允许 3 倍以内偏差）；（b）在 $\geqslant 5$ 个顶点数 $\leqslant 5\times10^4$ 的图上，用自建/调用的精确 Brandes 实现与 NetworkX 的 `betweenness_centrality` 交叉比对，**全部顶点的 BC 值相对偏差 $\leqslant 1\times10^{-6}$**（同一定义与归一化下应完全一致，差异只能来自浮点）。这两条不过关，后续全部结论无效。
2. **真值口径写死**：Top-k 保真度的参照必须是**精确 Brandes 结果**，因此主分析限制在顶点数 $\leqslant 5\times10^4$ 且精确计算可在 $\leqslant 6$ 机时完成的图上；对更大的图，只做"高样本量近似作为伪真值"的补充分析，并**明确标注伪真值本身的误差棒**，不与主分析混报。
3. **等成本口径**：主口径为**等墙钟时间预算**（每图设 3 档预算：精确计算耗时的 1%、5%、20%），辅口径为**等样本数**；两种都要报。这是本课题"双成本口径"的具体形式，两者结论不一致时必须讨论。
4. **随机性与统计**：每个（图 × 算法 × 预算）组合用 $\geqslant 20$ 个固定随机种子独立运行；报 Top-k 交集率与 Kendall τ 的**中位数与 IQR**，给自助法 95% 置信区间；跨算法比较用配对 Wilcoxon 符号秩检验（配对单位为"图 × 种子"）。k 取 $\{10, 100, 1000\}$ 三档。
5. **样本量**：$\geqslant 15$ 个图，覆盖社交、网页、道路、生物（蛋白互作）与合成幂律图五类；顶点数跨 2 个数量级；并同时报告每图的直径与度分布参数（RK 的样本量依赖直径，这是解释结果的必要协变量）。
6. **基准测试方法学**：所有计时在固定频率、绑核、无后台负载的条件下进行，重复 $\geqslant 5$ 次取中位数与 IQR，**不报均值**；线程数固定为 1（KADABRA 等实现支持并行，但并行会引入不可比的调度噪声，须在正文声明这一选择并单独用一组并行实验说明其影响）。
7. **可复现性**：全部脚本、图清单与下载脚本、原始结果 CSV、统计与绘图脚本开源；一键重跑脚本；全部种子写死。

5 · 数据与工具

用途	来源 / 工具
近似 BC 算法实现	NetworKit（`networkit` pip 可装，C++ 内核 + Python 绑定，含 `KadabraBetweenness` 与 `EstimateBetweenness`/RK 类实现；**ABRA 是否内置需核实，若无须自行实现或使用作者原始代码）。仅用于校验与对比，不计入本项目贡献**
精确 BC	NetworKit 的 `Betweenness`（C++ Brandes，快）；NetworkX 的 `betweenness_centrality`（纯 Python，慢但作为独立交叉校验）；igraph 的 `betweenness`（第三方独立实现，用于三方比对）
图数据	SNAP（社交/网页/道路网）；SuiteSparse Matrix Collection；BioGRID 或 STRING 的蛋白互作网络（公开下载，需按物种筛选控制规模）；NetworkX/NetworKit 的合成生成器（幂律、LFR）
统计与作图	Python + SciPy（Wilcoxon、Kendall τ）+ Matplotlib
算力	纯 CPU。精确 Brandes 的复杂度 $O(nm)$：$n = 5\times10^4$、$m = 5\times10^5$ 的图约 2.5×10^{10} 次基本操作，NetworKit 的 C++ 实现单线程估计数小时（**须在第 1 阶段实测标定，不得凭估计**）。内存需求以邻接表为主，16 GB 充裕。**超出范围**：顶点数 $\geqslant 10^6$ 的图无法给出精确真值，不进入主分析

6 · 方法路径

1. 装 NetworKit / NetworkX / igraph，用小图（顶点数 $\leqslant 200$）三方比对精确 BC 值，确认定义与归一化一致；完成验收第 1 条的两项校验并归档。

2. 核实工具能力边界：NetworKit 内置了 KADABRA 与哪种 RK 变体、ABRA 是否可得、各实现的 ε/δ 参数语义是否一致（**不同实现对 ε 的定义可能相差一个归一化因子，这是本课题最容易出错的地方，必须在此步核实清楚**）；确认各实现能否强制单线程。
3. 实测精确 Brandes 的运行成本随 (n, m) 的标度，据此**确定主分析的图规模上界**并锁定图样本清单（⩾ 15 图，五类覆盖）。
4. 计算并归档全部主分析图的精确 BC 真值与协变量（直径、度分布参数、连通性）。
5. 执行主扫描：3 算法 × 3 档时间预算 × 15+ 图 × 20 种子，记录 Top-k 交集率、Kendall τ、最大排名反转距离与实际耗时。
6. 分析：等时间预算与等样本数两种口径的对照表、H1 的翻转比例统计与配对 Wilcoxon 检验、H2 的 k 依赖趋势；以直径与度分布为协变量解释算法间差异。
7. 独立交叉校验：（a）用 igraph 的独立实现复算至少 3 个图的精确 BC，验证真值无误；（b）在 LFR 合成图上做受控实验——固定 n、m 只改变直径（通过调节社区结构与随机重连），检验 RK 相对 KADABRA 的劣势是否确实随直径增大而扩大（这是对 RK 样本量依赖直径这一机制的直接检验）。

7 · 新颖性边界

> **本课题不声称**：不提出新的近似算法，不改进任何采样策略，不给出新的概率界或复杂度界，不声称哪个算法"更好"这一笼统判断。KADABRA、RK、ABRA 与 NetworKit / NetworkX / igraph 的全部实现均为他人工作，标注为对照基准，不计入本项目贡献；全部图数据来自公开数据库，不计入本项目的数据贡献。
>
> **已有工作完成了什么**：Borassi 与 Natale（KADABRA, ACM JEA 2018）给出了 KADABRA 相对 RK 约 100 倍（无向）/ 70 倍（有向）、相对 ABRA 逾 1000 倍的加速，并指出误差容限趋零时优势扩大；Riondato 与 Upfal（ABRA）给出了自适应采样的质量保证与动态图支持；另有专门的速度 – 精度实验综述与有向图上的精确/近似比较。这些结论本项目引用而不重复声称。**历届丘奖对照**：2021 年金奖《Efficient Algorithm for Parallel Bi-core Decomposition》同属"图上的中心性/分解计算"，但其贡献是一个新的并行算法；本课题不提出算法，只做等成本口径下的评价维度转换，两者在贡献类型上不重叠。
>
> **本项目的贡献**：把评价维度从"达到 ε 绝对误差保证的成本"转换为"**等墙钟时间预算下的 Top-k 排名保真度**"，并检验既有排序在这一应用侧口径下是否保持。主结论是这份"等成本前沿 + 排名翻转清单"，不是任何新算法。
>
> **为什么有价值**：应用者（网络科学、生物信息）几乎只使用 BC 排名而非绝对值，却依据 ε-保证的比较来选算法。若排序保持，ε-保证被确立为排名质量的合理代理，这是一条有用的正面结论；若翻转，则说明应用侧的算法选择建议需要重写。
>
> **风险提示**：三种算法在 Top-k 上"实际上差不多"是可能结果。此时必须用 20 种子 × 15 图的配对设计给出置信区间，证明本实验有能力分辨交集率 5 个百分点量级的差异；若功效不足，须扩大种子数或图数，不得直接下"等价"结论。

8 · 决策门槛（go / no-go）

- **第 4 周**：参数语义门槛。若无法确认三种实现的 ε/δ 定义一致，**必须在此周解决**——这是需要提前核实而非边做边发现的事项。若确实不一致，**降级路径 A**：整体放弃 ε 参数口径，改为**纯样本数/时间预算驱动**（直接指定采样条数或运行时长，绕开 ε 的定义分歧）。主结论框架（等成本下的排名保真度）完全不受影响，反而更干净。

- **第 8 周**：ABRA 可得性核实完毕。若无内置实现且作者代码无法运行，**降级路径 B**：算法集缩为 KADABRA + RK + 一个自行实现的朴素均匀最短路采样基线（后者代码量约 150 行，且作为"教科书方法"参照有独立价值）。三算法的下限保持，主结论不变。
- **第 12 周**：精确真值的算力标定完成，图样本清单锁定。若精确 Brandes 在目标规模上超时，**降级路径 C**：把主分析图规模上界下调至 $n \leqslant 2\times10^4$ ，图数量补到 20 个以维持统计功效。**宁可用更多更小的图，不可用更少更大的图**——本课题的统计单位是"图"，样本量比单图规模重要。
- **第 28 周**：主扫描完成且 H1 有明确判定。若进度不足，**降级路径 D**：时间预算档从 3 档减为 2 档（1% 与 10%），k 从 3 档减为 2 档（10 与 100，即差异最可能出现的两档），种子数保持 20 不变。
- **预算裁剪顺序**：图类别数（下限 3 类）→ 时间预算档数 → k 档数 → （绝不裁剪）种子数与精确真值的正确性校验。
- **选择前提**：适合对网络科学有兴趣、能读懂采样算法的概率保证（不必会证）、并能接受"结论可能是三者等价"的学生。本路线全程 Python 可完成，工程门槛低于 K01/K05，是编程基础较弱学生的首选之一。

K07 · 顶点覆盖数据规约规则的边际效力：以逐规则消融实验测定各规则对核大小的独立贡献及其结构预测因子

推荐优先级：中 · 分族：算法设计与复杂度 · 参赛子类：计算机–算法与组合优化 · 资源需求：纯 CPU 笔记本 / 16 GB 内存 · 技能取向：图论 + 算法实现 + 消融实验设计

1 · 研究问题

在最小顶点覆盖（minimum vertex cover）的参数化预处理中，六条经典数据规约规则（度-1 规则、度-2 折叠、支配规则、线性规划/Nemhauser – Trotter 约简、皇冠约简 crown reduction、unconfined 规则）各自对最终核（kernel）大小的**边际贡献**是多少——即在保留其他全部规则的前提下单独移除某一条，核的顶点数与边数增加多少？这些边际贡献能否由可先验计算的图结构量（度分布、局部聚类系数、二部性程度）预测，从而给出"对某类图该先跑哪几条规则"的可操作判据？

2 · 研究背景与空白

技术背景。 核化（kernelization）是参数化算法的核心技术：在多项式时间内把实例 (G, k) 化简为等价实例 (G', k')，其中 G' 的规模仅由 k' 有界，且 G 有大小 $\leqslant k$ 的顶点覆盖当且仅当 G' 有大小 $\leqslant k'$ 的顶点覆盖。顶点覆盖是核化在实践中效果最好的少数问题之一。各条规则的机制不同：度-1 规则直接把度为 1 的顶点的邻居放进覆盖；LP/Nemhauser – Trotter 约简解一个可用二部图最大匹配求解的线性松弛，把取值为 1 和 0 的顶点分别确定下来；皇冠约简寻找一种特殊的二部结构（"皇冠"）并整体消去。**关键的资源特性是：全部规约规则都是多项式时间的，且本课题完全不需要求解 NP 难的顶点覆盖本身**——交付物是核的大小，不是最优解。这使课题彻底绕开了算力墙，也绕开了"让学生去挑战已知难题"的陷阱：我们测量的是预处理的效力，而不是试图打败 PACE 冠军求解器。

已有工作到哪一步。 Abu-Khzam 等的《Kernelization Algorithms for the Vertex Cover Problem: Theory and Experiments》系统实验了多条规则，给出的具体建议是：皇冠约简在实践中比线性规划快得多，有时化简效果同样好（尽管其最坏情况核大小界更差），最佳流程似乎是先做预处理与高度数方法，再做皇冠约简。PACE 2019 顶点覆盖赛道的冠军求解器 WeGotYouCovered 采用了组合策略：激进的核化 + 局部搜索 + 分支约简 + 最先进的分支定界求解器，并公布了完整实现与实例集。另有针对偏好连接模型的核大小分析、顶点覆盖结构参数化的理论工作，以及基于快速证据验证的紧凑整数规划设计（arXiv:2509.25445）等新近方向。

空白在于：现有实验工作报告的是"规则组合的总效果"与"先跑哪个更快"的经验流程，而没有做过逐规则的消融实验——因此没人知道在现代规则集里，哪几条规则实际上是冗余的（它们的效果已被其他规则完全覆盖），也没人给出"边际贡献取决于图的什么性质"的预测。这个空白属于"评价维度转换 + 问题方向转换"，可靠性高：实例与实现全部公开、算力门槛极低、正负结论都成立（若某条规则边际贡献接近零，那是一条对求解器工程有直接价值的简化建议；若每条都不可或缺，则现有规则集被证明是精简的）。

3 · 可检验假设

- H1：在同时保留 LP/Nemhauser – Trotter 约简的规则集中，移除皇冠约简后核顶点数的中位增幅 $\leqslant 5\%$；而移除 LP 约简后的中位增幅 $\geqslant 30\%$。即"皇冠约简的化简效力在有 LP 约简时基本被覆盖，其价值主要在速度而非效力"。若移除皇冠后增幅 $> 15\%$，H1 被否证，说明两条规则捕获的是不同结构。
- H2：各规则的边际贡献可由图结构量预测：度-1/度-2 规则的贡献与"度 $\leqslant 2$ 顶点的比例"的 Spearman $\rho \geqslant 0.7$；皇冠约简的贡献与图的局部二部性度量（例如最大二部子图的边占比估计或三角形密度的倒数）$\rho \geqslant$

0.5。若两者都不成立，说明边际贡献由更全局的结构决定，这本身是对"规则可按图类型选择"这一直觉的否证。

4 · 量化验收标准

1. **方法学校验（硬门槛）**：用自建核化管线，在文献与 PACE 使用过的公共实例上重现已发表的核大小数字，**相对偏差 ⩽ 5%**（核大小是确定性量，不存在硬件差异，因此判据可以严格；若某规则实现有多种变体导致差异，须逐一列出变体并说明所选版本）。同时要求：在至少 3 个已发表实例上，本管线得到的规则应用顺序建议与 Abu-Khzam 等的结论方向一致。这一步不过关，后续全部结论无效。
2. **正确性门槛（第二道硬门槛）**：核化必须保持解等价。判据：对 ⩾ 200 个顶点数 ⩽ 30 的随机小图，用暴力枚举求出精确最小顶点覆盖，验证 **VC(G) = VC(G′) + （规约过程中确定进入覆盖的顶点数）**，要求 100% 通过，任何一例失败即视为实现有误。这是消融实验唯一能排除"核变小是因为规约写错了"的证据，必须写进正文。
3. **消融设计写死**：采用"留一移除"（leave-one-out）与"逐一加入"（greedy add-one）双向消融，两套结果都报。规约过程必须迭代至不动点（fixpoint），且规则应用顺序在同一实验内固定并公开；额外做一次顺序随机化实验（⩾ 10 个随机顺序）量化顺序对核大小的影响幅度——若顺序效应大于规则效应，全部消融结论无效，必须在正文明确报告这一检查。
4. **统计口径**：核缩减比例是有界比值，**报中位数与 IQR，不报均值**；跨实例比较用配对 Wilcoxon 符号秩检验（配对单位为实例）；相关性给自助法（⩾ 2000 次重采样）95% 置信区间与散点图，不报单一 R^2。
5. **样本量**：实例 ⩾ 60 个，覆盖至少 4 类（PACE 2019 顶点覆盖赛道公开实例、DIMACS 团/独立集基准的补图、SNAP 真实网络、随机与幂律合成图）；顶点数跨 3 个数量级；每类下限 10 个。
6. **运行成本单列**：报告每条规则的单次应用耗时与在整个不动点迭代中的累计耗时（中位数与 IQR，⩾ 5 次重复，固定频率），因为文献中"皇冠约简比 LP 快得多"这一结论正是速度侧的，必须在同一口径下独立复核。
7. **可复现性**：全部规约规则实现、实例清单与下载脚本、原始核大小与计时 CSV、统计脚本开源；一键重跑脚本；随机种子写死。

5 · 数据与工具

用途	来源 / 工具
实例集	PACE 2019 挑战赛顶点覆盖赛道公开实例（pacechallenge.org 归档，含公开与隐藏测试集的公开部分）；DIMACS 团问题基准（可取补图转为顶点覆盖实例）；SNAP 真实网络；自建随机图与幂律图生成器。**外部实例仅用于校验与对比，不计入本项目的数据贡献**
参考求解器/核化器	WeGotYouCovered（PACE 2019 冠军，开源）或 KaMIS 的规约模块，作为核大小的**独立交叉校验**参照。**仅用于校验与对比，不计入本项目贡献**
规则实现	自行实现六条规则（Python + NumPy 原型 → C++ 或 Numba 加速版本；每条 50–150 行）。LP/Nemhauser–Trotter 约简可归约为二部图最大匹配，用 NetworkX 的 `hopcroft_karp_matching` 或 SciPy 的 `maximum_bipartite_matching` 实现（**须核实 SciPy 版本的接口与复杂度**）；皇冠约简同样依赖最大匹配
精确求解（仅用于正确性门槛的小图）	自行实现的暴力枚举（$n \leqslant 30$）；可选用 PuLP + CBC 或 Google OR-Tools 的整数规划做二次确认（**仅在 $n \leqslant 200$ 的小图上使用，不用于主分析**）
结构量计算	NetworkX / igraph（度分布、聚类系数、三角形计数、k-core）
统计与作图	Python + SciPy + Matplotlib
算力	纯 CPU。全部规约规则为多项式时间：最大匹配 $O(m\sqrt{n})$，$n = 10^6$ / $m = 10^7$ 的实例在数分钟内可完成（**须实测标定**）。内存以邻接表为主，16 GB 可覆盖 PACE 公开实例的绝大部分。**超出范围**：求解最小顶点覆盖的最优解（NP 难）不在本课题范围内，正文必须明确声明本项目不求解、不与求解器竞速

6 · 方法路径

1. 装环境，下载 PACE 与 DIMACS 实例，跑通参考核化器（WeGotYouCovered 或 KaMIS）在几个实例上的输出，作为后续对照的锚点；实测规约的运行成本标度。
2. 逐条实现六条规约规则，每条实现后立刻通过 $n \leqslant 30$ 的暴力等价性检验（验收第 2 条），不通过不进入下一条。
3. 核实工具能力边界：SciPy/NetworkX 的最大匹配实现在百万边规模上是否可用（若不可用，须自行实现 Hopcroft–Karp，**若需自行实现，须用制造样例与 NetworkX 小图结果双重验证，这本身可作为方法学贡献写入**）；确认参考核化器公布的核大小定义与本项目一致（是否把已确定顶点计入）。
4. 完成方法学校验（验收第 1 条），归档差异分析。
5. 组建实例样本（$\geqslant 60$ 个、4 类），计算全部结构量协变量。
6. 执行双向消融扫描（留一移除 + 逐一加入）与顺序随机化检查，记录核顶点数、核边数、每规则耗时；输出边际贡献分布。
7. 分析与独立交叉校验：（a）出边际贡献表、配对 Wilcoxon 检验、结构量相关性与置信区间；（b）用参考核化器对同一批实例产出核大小，验证本管线的全规则核与之在同一量级（差异须能由规则集差异解释并逐条说明）；（c）在合成图上做受控实验——固定顶点数与边数、只扫描"度 $\leqslant 2$ 顶点比例"，检验 H2 的预测关系是否在受控条件下成立。

7 · 新颖性边界

本课题不声称：不提出新的规约规则，不改进任何核大小的理论上界，不求解最小顶点覆盖，不与 PACE 求解器比赛求解速度或解质量，不声称任何新的参数化复杂度结果。PACE 实例、DIMACS 基准、WeGotYouCovered/KaMIS 与全部六条规则均为他人工作，标注为对照基准，不计入本项目贡献。

已有工作完成了什么：Abu-Khzam 等给出了多条规则的实验比较与流程建议（皇冠约简比线性规划快得多、有时化简同样好、建议先预处理与高度数方法再做皇冠约简）；PACE 2019 冠军 WeGotYouCovered 展示了激进核化 + 局部搜索 + 分支约简 + 分支定界的组合在竞赛实例上的效力；理论侧有偏好连接模型下的核大小分析与结构参数化结果。这些结论本项目引用而不重复声称。

本项目的贡献：首次以**逐规则消融**的方式给出各规约规则的**边际**核缩减贡献分布，并检验该贡献能否由可先验计算的图结构量预测。主结论是这份"边际贡献表 + 冗余规则清单 + 结构预测判据"，不是任何新规则或新求解器。

为什么有价值：核化的实现成本主要花在规则数量上，而现有建议只告诉工程师"都跑一遍、按某个顺序"。若某条规则在现代规则集中边际贡献接近零，实现者可以直接省掉它；若边际贡献可由图性质预测，则可以做成自适应的规则选择。两者都能直接落到求解器工程里。

风险提示：规则之间存在强交互（A 的贡献取决于 B 是否在场），因此"留一移除"的边际贡献不是可加分解。正文必须明确这一点，并用"逐一加入"的第二套结果作为交互强度的度量；若两套结果差异巨大，"规则贡献高度依赖组合、不存在稳定的边际排序"同样是有效且重要的结论。

8 · 决策门槛（go / no-go）

- **第 6 周**：正确性门槛。若任何一条规则无法通过 $n \leqslant 30$ 的暴力等价性检验，**降级路径 A**：把该规则移出规则集并在正文声明，规则数下限为 4 条（度-1、度-2 折叠、LP 约简、支配规则——这四条实现最简单、正确性最易验证）。消融框架完全保留。
- **第 10 周**：最大匹配的规模可行性必须核实完毕——这是需要提前确认而非边做边发现的事项。若 SciPy/NetworKit 的实现无法处理百万边实例且自行实现超时，**降级路径 B**：把实例规模上界下调至 $m \leqslant 10^6$ ，并把实例数量从 60 补到 80 以维持统计功效。主结论不变（边际贡献是相对量，不依赖绝对规模）。
- **第 14 周**：方法学校验硬门槛。若核大小与文献偏差 > 5%，先排查规则变体定义、不动点迭代是否跑满、核大小是否把已确定顶点计入三项；第 17 周仍不过关则**降级路径 C**：放弃与外部文献的绝对核大小对照，改用参考核化器（WeGotYouCovered/KaMIS）在本机上的实际输出作为对照锚点，硬门槛替换为"本管线全规则核与参考核化器核的顶点数比值落在 [0.9, 1.3] 且差异可逐规则解释"。
- **第 22 周**：顺序随机化检查必须完成。若顺序效应的幅度超过规则效应（即换个顺序核大小的变化比移除一条规则还大），**降级路径 D**：把课题重心正式转为"**规约规则应用顺序对核大小与耗时的影响**"这一主题——这依然是本领域缺少系统数据的方向，消融框架、实例集、统计口径全部复用，主结论框架保留。
- **第 32 周**：消融扫描完成且 H1 有明确判定。若进度不足，**降级路径 E**：实例类别从 4 类减为 3 类（保留 PACE + SNAP + 合成），每类实例数不低于 15。
- **预算裁剪顺序**：结构量种类 → 实例类别数（下限 3）→ 规则数（下限 4）→ （绝不裁剪）正确性门槛与顺序随机化检查。
- **选择前提**：适合喜欢图论证明与算法细节、能耐心把六条规则各自的边界情形写对的学生。本路线的最大隐患是规则实现的正确性，正确性门槛必须严格执行，不可为赶进度跳过。

K08 · 常数时间性质统计检验的功效标定：dudect 方法在消费级 CPU 上对可控泄漏的检出曲线与假阳性率

推荐优先级：中 · 分族：密码学（防御向） · 参赛子类：计算机–信息安全与测量 · 资源需求：纯 CPU 笔记本 / 8 GB 内存 · 技能取向：C 语言 + 统计检验 + 实验设计

1 · 研究问题

以时序泄漏检测工具 dudect 为代表的"黑盒统计检验"方法，在噪声远高于服务器的**消费级笔记本 CPU** 上，对**可控幅度的人工植入泄漏**的检出功效（statistical power）随测量样本量如何变化？在已知常数时间的参考实现上，该方法在常用阈值下的假阳性率是多少——换言之，公开报告中"某实现未通过常数时间检验"的结论，需要多少样本量才能被认为可靠？

伦理与合规声明（写在方案首位）：本课题**只做防御向测量**，全部对象为学生本机上编译的开源密码库的公开 API；不针对任何在线服务、不采集任何第三方数据、不尝试恢复任何真实密钥、不开发任何攻击工具。产出物是检测方法的功效曲线与使用建议，属于安全测试方法学。

2 · 研究背景与空白

技术背景。 密码实现若其执行时间依赖于秘密数据（密钥、明文），攻击者就可能通过反复计时统计恢复秘密——这就是时序侧信道。防御手段是写"常数时间"代码：避免依赖秘密的分支与内存访问。验证一段代码是否常数时间有两条路：静态/符号分析（对二进制或中间表示做推理）与黑盒统计测量。Reparaz 等的 dudect（《Dude, is my code constant time?》, DATE 2017, ePrint 2016/1123）属于后者：对两类输入（例如全零向量 vs 随机向量）各测大量执行时间样本，用 Welch t 检验判断两个时间分布是否有统计显著差异；原型仅约 300 行 C 代码，开源于 `oreparaz/dudect`。它的优势是不依赖任何静态分析、直接反映目标平台的真实行为；**公开文献同时明确指出它的弱点：对现代计算机系统中普遍存在的噪声高度敏感**。近期还出现了基于低层代码轨迹动态分析的常数时间验证工作（arXiv:2604.16832）与单轨迹侧信道漏洞自动检测方向（arXiv:2304.02102），CRoCS 维护的 ct-tools 站点收录了这一类工具的横向清单。

空白在于：dudect 类方法被广泛使用并被用来给出"某实现不是常数时间"的判断，但公开文献里没有系统的**功效标定**——即"泄漏幅度多大、样本量多少时，这个检验有多大概率检出"，以及"在确定常数时间的代码上，它多久会误报一次"。没有功效曲线，一个"未检出泄漏"的结果无法区分"确实没有泄漏"与"样本量不足"。这个空白属于"评价维度转换"，且它具有学生课题少见的优点：**可以人工制造已知真值**——把一段常数时间代码改成"泄漏 N 个时钟周期"的可控版本，真值完全已知，因此功效与假阳性率都能被精确测量，而不需要任何真实漏洞。算力上完全无门槛。

3 · 可检验假设

- H1：在消费级笔记本上，dudect 的检出功效随植入泄漏幅度 Δ（以时钟周期计）与样本量 N 呈可标定的关系：达到 80% 检出功效所需的样本量 N_{80} 与 Δ 的平方成反比（即 $N_{80} \cdot \Delta^2 \approx$ 常数，符合 t 检验的理论标度）。判据为跨 ⩾ 6 个 Δ 档位的双对数拟合斜率落在 [−2.4, −1.6]。若偏离该区间，说明噪声结构不满足检验假设，需要改用非参数或分位数方法——这同样是重要结论。
- H2：在已知常数时间的参考实现上，dudect 默认判据（$|t| > 4.5$ 判为非常数时间）的假阳性率在笔记本上显著高于名义水平：⩾ 20 次独立完整运行中至少 15% 误判为非常数时间。若假阳性率 ⩽ 5%，H2 被否证，方法在噪声平台上的稳健性获得一次独立支持。

4 · 量化验收标准

1. **方法学校验（硬门槛）**：用自建测量框架重现 dudect 原论文/仓库中给出的示例结果——即在其自带的"确定泄漏"示例（如朴素的非常数时间内存比较）上检出泄漏、在其自带的常数时间示例上不检出，且 **t 统计量随样本量增长的轨迹形态与官方示例一致**（趋势方向与量级一致，绝对值跨平台不可比）。同时用一个独立实现的 Welch t 检验（Python/SciPy）对同一批原始时间样本复算 t 值，与 dudect 内部计算的偏差 $\leqslant 1\%$。这一步不过关，后续全部结论无效。
2. **真值构造的可控性**：人工泄漏必须通过一个**参数化的延迟注入**实现（例如依据秘密位数决定执行的空转循环次数），且注入幅度须用独立的直接计时实测标定（报中位数与 IQR），标定值与设定值的偏差 $\leqslant 10\%$。未经标定的"泄漏幅度"不得用于功效曲线。
3. **功效与假阳性的统计口径**：功效曲线的每个点（$\Delta \times N$ 组合）用 $\geqslant 30$ 次**独立完整实验**（每次重新采样、重新计算）估计，报检出比例与 Clopper – Pearson 精确二项置信区间；假阳性率用 $\geqslant 50$ 次在常数时间参考实现上的独立运行估计，同样给精确置信区间。**明确写明不用单次运行的 t 值下结论**——单次 t 值正是本课题要质疑的做法。
4. **测量环境控制**：固定 CPU 频率、绑核、禁用 Turbo 与 SMT、记录温度与频率日志、每次运行前预热并丢弃预热段；同时报告"未控制环境"（默认设置、允许后台负载）下的对照结果——两种环境的功效差异本身就是本课题的核心交付物之一。全部延迟数据报中位数与 IQR，**不报均值**。
5. **覆盖面**：至少 3 个开源密码库的至少 5 个原语（例如常数时间内存比较、AES 软件实现、Curve25519 标量乘、RSA 模幂的常数时间与非常数时间变体）；每个原语同时测其"库自带版本"与"人工注入泄漏版本"，前者用于假阳性/漏检评估，后者用于功效曲线。
6. **方法对照**：把 dudect 的结果与至少一种**独立机制**的检测方法对照——首选基于动态污点/未初始化内存追踪的 ctgrind / Valgrind-MemSan 方案或 TIMECOP（**可用性需核实**）。两种方法给出不同结论的原语要逐一分析原因。
7. **可复现性**：全部测量代码、泄漏注入补丁、原始时间样本（压缩存档）、统计脚本开源；一键重跑脚本；固定种子；完整披露硬件、OS、编译器版本与编译选项（**编译选项极其关键：优化级别会改变是否常数时间，必须逐一记录**）。

5 · 数据与工具

用途	来源 / 工具
时序泄漏检测工具	dudect（GitHub `oreparaz/dudect`，约 300 行 C，零依赖）。**仅用于校验与对比，不计入本项目贡献**
对照检测方法	ctgrind（Langley，基于 Valgrind）/ Valgrind MemorySanitizer 方案；TIMECOP（SUPERCOP 工具链的一部分）；CRoCS `ct-tools` 站点（crocs-muni.github.io/ct-tools）提供工具清单与状态。**具体可用性与安装难度需核实，列为风险项**
被测密码实现	libsodium（常数时间为设计目标，含 `sodium_memcmp`、Curve25519）；mbedTLS（含常数时间与传统实现的对照配置）；BearSSL（明确的常数时间设计文档）。全部开源、可本地编译。**仅作为测量对象，不评价其安全性质，不声称发现任何漏洞**
泄漏注入	自行实现的参数化延迟注入补丁（依赖秘密位的空转循环 / 依赖秘密的查表访问两种机制各一套）
计时与环境控制	`rdtscp` / `clock_gettime`；`taskset`、`cpupower`、`turbostat`（**笔记本平台上的可用性需核实**）
统计	Python + SciPy（Welch t 检验、Clopper–Pearson 区间）+ statsmodels（功效分析）+ Matplotlib
算力	纯 CPU，内存 < 1 GB。单次 dudect 运行数十秒到数分钟；总工作量约 6 Δ 档 × 6 N 档 × 30 重复 × 5 原语 ≈ 数十机时，可夜间批跑。**超出范围**：硬件功耗/电磁侧信道测量（需示波器与探头）不在本课题范围内，须在正文明确声明

6 · 方法路径

1. 装环境，编译 dudect 与其自带示例，复现"泄漏示例检出 / 常数时间示例不检出"这一基本行为；用 SciPy 独立复算 t 值完成验收第 1 条。
2. 核实工具能力边界（本课题的关键风险步）：对照方法（ctgrind / TIMECOP）能否在本机安装运行；`cpupower` / `turbostat` 在笔记本上是否可用；编译器优化级别对被测函数是否引入或消除分支（须用反汇编逐一确认，这一步不能靠猜）。
3. 实现并标定参数化泄漏注入：用直接计时实测每个 Δ 档位的真实幅度（中位数与 IQR），确认与设定值偏差 $\leqslant 10\%$；未达标的档位重新设计。
4. 建立测量框架：环境控制脚本、独立完整实验的自动重复、原始时间样本落盘（保留全分布，不做在线聚合）、温度与频率日志。
5. 执行功效扫描（$\Delta \times N \times 30$ 次独立实验）与假阳性扫描（常数时间实现 $\times \geqslant 50$ 次独立运行），在"受控环境"与"默认环境"两种条件下各做一遍。
6. 分析：功效曲线与 N_{80} 的 Δ 标度拟合（H1）、假阳性率与精确置信区间（H2）、受控 vs 默认环境的对照、双成本口径（等样本数 / 等墙钟时间预算下的检出率）。
7. 独立交叉校验：（a）用对照检测方法（ctgrind 类）对同一批原语给出判定，与 dudect 结果做一致性分析（Cohen's κ）并逐一解释分歧；（b）在另一台不同微架构的机器上重跑功效曲线的关键档位，检验 $N_{80} \cdot \Delta^2$ 标度是否跨平台成立。

7 · 新颖性边界

> **本课题不声称**：不提出新的泄漏检测方法，不实施任何攻击，不恢复任何密钥，不声称发现任何开源库的安全漏洞（若测量中出现异常，正确做法是按各项目的安全披露流程报告，而不是写进论文作为发现），不做硬件侧信道测量。dudect、ctgrind/TIMECOP 与全部被测密码库均为他人工作，标注为对照基准，不计入本项目贡献。
>
> **已有工作完成了什么**：Reparaz 等（DATE 2017 / ePrint 2016/1123）提出并开源了 dudect，用 Welch t 检验对两类输入的执行时间分布做统计比较，明确说明该方法不依赖静态分析而依赖目标平台上的实测，同时指出其对现代系统噪声高度敏感；近期工作把常数时间验证推进到低层代码轨迹的动态分析（arXiv:2604.16832）与单轨迹漏洞自动检测（arXiv:2304.02102）；CRoCS 的 ct-tools 汇总了这一类工具。这些工作本项目引用而不重复声称。
>
> **本项目的贡献**：给出 dudect 类黑盒方法在**消费级噪声平台**上的**功效曲线与假阳性率标定**——用人工可控泄漏构造已知真值，量化"检出需要多少样本"与"不检出意味着什么"。主结论是这条标定曲线与由它导出的使用建议（例如"报告未检出泄漏时必须同时报告该样本量下可检出的最小泄漏幅度"），不是任何新工具。
>
> **为什么有价值**：目前大量安全实践直接引用 dudect 的单次运行结论。若功效标定显示常用样本量只能检出 ≫ 实际攻击所需的泄漏幅度，那么"通过 dudect 检验"这一说法本身需要附带条件；若方法比预想稳健，则为其广泛使用提供一次独立背书。
>
> **风险提示**：笔记本平台的噪声可能大到使功效曲线在可行样本量内根本不出现拐点。这一结果本身即是重要结论（"该方法在此类平台上不可用"），但必须给出精确二项置信区间证明本实验设计确有分辨力，且必须在受控与非受控两种环境下都做，避免把"没配置好环境"误报为"方法无效"。

8 · 决策门槛（go / no-go）

- **第 5 周**：合规与选题边界确认。指导教师须书面确认本课题的防御向定位与"不实施攻击、不做漏洞披露式结论"的边界；同时确认参赛材料中不含任何可直接用于攻击的产物。此项不通过则整条路线不启动（应改选 K03 或 K07）。
- **第 8 周**：泄漏注入标定门槛。若参数化延迟注入无法被编译器保留（优化掉空转循环）或标定偏差 > 10%，**降级路径 A**：改用"依据秘密位选择不同缓存行的查表访问"作为泄漏机制（不可被优化消除，且幅度由缓存缺失代价自然标定）。功效曲线框架完全保留，仅泄漏机制从计算型改为访存型，须在正文声明并讨论两种机制的差异。
- **第 12 周**：对照检测方法的可用性必须核实完毕——这是需要提前确认而非边做边发现的事项。若 ctgrind/TIMECOP 均无法运行，**降级路径 B**：对照方法改为"**源码级人工审计**"——由学生对每个被测原语逐行判定是否存在依赖秘密的分支/访存，并给出明确、可复现的判定规则与判定记录（双人独立判定并报 Cohen's κ）。证据强度略降但完全自洽，主结论不变。
- **第 20 周**：功效曲线必须出现可用拐点。若在最大可行样本量下所有 Δ 档位的检出率都接近 0 或都接近 1，**降级路径 C**：重新设计 Δ 档位（用第 8 周标定的实测幅度做对数等距重排），并把重心从"功效曲线"转向"**环境控制对检出能力的贡献**"——即受控 vs 默认环境的对照，这部分不依赖拐点位置且本身缺少公开数据。主结论框架（方法的可信度标定）保留。
- **第 30 周**：假阳性扫描完成且 H2 有明确判定。若进度不足，**降级路径 D**：被测原语从 5 个减为 3 个（常数时间内存比较、AES 软件实现、Curve25519 标量乘），密码库从 3 个减为 2 个，重复次数保持不变。
- **预算裁剪顺序**：密码库数量（下限 2）→ 原语数量（下限 3）→ Δ 档位数（下限 4）→ （绝不裁剪）独立完整实验的重复次数与环境控制。

- **选择前提**：仅在学生能读写 C、理解假设检验的功效概念、且指导教师认可防御向边界时才启动。若学生对"我要做黑客项目"有期待，这条路线应当明确劝退——本课题的全部内容是测量方法学。

K09 · 纯 CPU 量子线路模拟的可行边界：态矢量法与矩阵乘积态法在 QAOA 线路上的成本交叉点与比特上限标定

`推荐优先级：中低（风险偏高）` · `分族：量子计算模拟` · `参赛子类：计算机–量子计算与数值方法` · `资源需求：纯 CPU 笔记本 / 16 GB 内存（内存是唯一硬约束）` · `技能取向：Python + 线性代数 + 数值实验`

1 · 研究问题

在一台 16 GB 内存的纯 CPU 笔记本上，态矢量（statevector）模拟与矩阵乘积态（matrix product state, MPS）模拟在量子近似优化算法（QAOA）求解 MaxCut 的线路上，墙钟成本的交叉点位于哪一组参数（比特数 n、层数 p、图的平均度 d）？态矢量法的**实际**可达比特上限比理论内存上限（$2^n \times 16$ 字节）低多少，缺口分别由哪些具体开销（解释器与库的常驻内存、保存态矢量时的副本、门作用时的临时缓冲）贡献？

2 · 研究背景与空白

技术背景。 经典计算机模拟量子线路有两条主流路径。**态矢量法**显式存储 2^n 个复振幅，每个双精度复数 16 字节，因此内存需求随比特数指数增长，但对任意线路都成立、无近似。**矩阵乘积态/张量网络法**只存储纠缠结构，成本由"键维"（bond dimension）决定；对低纠缠线路可以远超态矢量法的比特数，但一旦纠缠增长，键维指数上升，反而更慢。QAOA 是这两种方法的天然试验场：$p = 1$ 时纠缠很浅，p 增大时纠缠迅速铺开。**这个课题的内存边界必须写清楚且必须自己核实**，因为它是整条路线的物理约束：

比特数 n	complex128 态矢量（16 B/振幅）	complex64（8 B/振幅）	在 16 GiB 机器上的判断
20	16 MB	8 MB	充裕
24	268 MB	134 MB	充裕
26	1.07 GB	0.54 GB	舒适（可保存副本、可并行多组）
27	2.15 GB	1.07 GB	舒适
28	4.29 GB	2.15 GB	可行（保存态矢量的副本会到 8.6 GB，接近上限）
29	8.59 GB	4.29 GB	勉强（不得保存副本，须原地操作）
30	17.18 GB（= 恰好 16 GiB）	8.59 GB	双精度**不可行**；单精度勉强
31	34.4 GB	17.18 GB	**超出范围**

即：**双精度态矢量在 16 GiB 机器上的绝对上限是 29 比特，实际舒适区是 26 – 28 比特**；改用单精度可上推约 1 比特，但须付出精度代价并在验收标准里量化。这一表格中的每一格都要在项目第 1 阶段用实测内存占用验证，不得直接引用。**需核实**：Qiskit Aer 在应用多比特门时是否分配临时缓冲、`save_statevector` 是否产生完整副本——这两项直接决定实际上限比理论值低几比特，也正是本课题研究问题的一部分。

已有工作到哪一步。 Qiskit Aer 文档明确给出"n 比特态矢量占用 2^n 个 16 字节复数"这一公式，并提供 `max_memory_mb` 参数在超限时报错；Aer 内置了 `statevector`、`density_matrix`、`matrix_product_state`、`stabilizer`、`extended_stabilizer` 等多种模拟方法（**`tensor_network` 方法是否要求 GPU 需核实**）。QAOA 与 MaxCut 本身是被研究得极为充分的方向，大量工作报告了不同 p 下的近似比与参数优化策略；模拟侧也有大量在 GPU 与 HPC 集群上的性能工作。

空白在于：几乎所有模拟性能研究都在服务器或 GPU 上给出标度，而"一台普通笔记本上这两类方法各自的可行边界与交叉点在哪"这一对学生与教学最有用的问题，没有系统的公开标定。这个空白属于"方法层面的可复现性贡献"——物理与算法结论均已发表，本项目的贡献是一份**跨方法的成本前沿与资源判据**。这类贡献在计算领域是被承认的，但**定位必须讲清楚，否则会被误读为重复工作**：本项目不研究 QAOA 好不好用，只研究"在给定资源下能模拟到哪里、用哪种方法"。

3 · 可检验假设

- **H1**：在固定截断保真度（$\geqslant 1-10^{-6}$）下，MPS 法与态矢量法的墙钟成本交叉点比特数 $n^*(p)$ 随 QAOA 层数 p 单调递减：$p=1$ 时 $n^*(1)\geqslant 26$；$p=3$ 时 $n^*(3)\leqslant 20$。若 $n^*(3)>24$，H1 被否证，说明 QAOA 在中等 p 下的纠缠增长比预期慢得多——这对模拟方法选择是更有利的结论。
- **H2**：态矢量法的实际可达比特上限比表中理论值低 1 – 2 比特，且缺口的主导项是"保存/读取态矢量时的完整副本"而非常驻内存；关闭态矢量保存（只取测量统计）可回收 $\geqslant 1$ 比特。若关闭保存后上限不变，说明主导项是门作用的临时缓冲，这是一条对模拟器使用者同样有用的结论。

4 · 量化验收标准

1. **方法学校验（硬门槛）**：三重校验，全部通过方可继续。（a）**与自建模拟器逐振幅比对**：自行用 NumPy 实现一个直接的态矢量模拟器（约 100 – 150 行，门作用用张量重塑实现），在 $n\leqslant 20$ 的 QAOA 线路上与 Qiskit Aer 的输出逐振幅比较，最大绝对偏差 $\leqslant 10^{-10}$。（b）**与解析结果比对**：在 $n\leqslant 16$ 时用暴力枚举求出 MaxCut 目标函数在 QAOA 输出态下的精确期望值，与模拟测得值比较，偏差 $\leqslant 10^{-12}$（$p=1$ 时还可与已发表的解析表达式核对）。（c）**MPS 与态矢量的保真度比对**：同一线路的 MPS 结果与态矢量结果的态保真度 $\geqslant 1-10^{-6}$。三条任一不过，后续全部结论无效。
2. **数值收敛检查**：MPS 侧必须做**键维收敛扫描**——对每个 (n, p) 组合扫描至少 5 个键维档位，报告目标函数期望值随键维的收敛曲线与截断误差累计值；**明确报告在给定键维下是否真正达到设定的保真度**，不得只报最大键维的结果。这是数值模拟类课题的必备项。
3. **内存标定**：用实测（`resource.getrusage` 峰值 RSS + `/proc/self/status`，或平台等价接口）逐比特测量实际内存占用，与理论表逐格比对，报告缺口及其归因；每个配置重复 $\geqslant 5$ 次报中位数与 IQR。
4. **基准测试方法学**：每个（方法 × n × p × 图）组合重复 $\geqslant 10$ 次；固定 CPU 频率、绑核、禁用 Turbo；**报中位数与 IQR，不报均值**；线程数固定并在正文声明（Aer 默认多线程，须显式设定 `max_parallel_threads` 以保证可比）。
5. **双成本口径**：交叉点必须同时用**等工作量**（相同线路、相同精度目标下的墙钟时间）与**等墙钟时间预算**（固定 60 秒内两种方法各能达到的最大 n 或最小截断误差）两种口径给出，两者不一致时必须讨论。
6. **样本量**：图实例 $\geqslant 30$ 个，覆盖至少 3 类（3-正则图、Erdős – Rényi 图两档密度、几何/网格图），每类 $\geqslant 8$ 个；比特数覆盖 $\geqslant 8$ 个档位；$p\in\{1,2,3,4\}$。
7. **可复现性**：全部线路生成、模拟、内存标定与统计脚本开源；一键重跑脚本；随机种子写死；QAOA 参数固定为公开的解析或预优化值（**不做参数优化**——避免把优化器的随机性混入成本测量，这一选择须在正文说明）。

5 · 数据与工具

用途	来源 / 工具
量子模拟器	Qiskit + Qiskit Aer（pip 可装，纯 CPU 版本无 GPU 依赖）；模拟方法用 `AerSimulator(method="statevector")` 与 `method="matrix_product_state"`。`method="tensor_network"` **需核实是否要求 GPU/cuQuantum，若是则不纳入**
独立对照模拟器	自行实现的 NumPy 态矢量模拟器（校验用）；可选 `qsim` / `Cirq`（Google，CPU 可用）或 `quimb`（张量网络，纯 Python/NumPy，CPU 友好）作为第三方交叉校验。**仅用于校验与对比，不计入本项目贡献**
线路与问题实例	QAOA MaxCut 线路自行构造（结构完全公开）；图实例用 NetworkX 生成器（3–正则、Erdős–Rényi、网格）与固定种子；可另取 SNAP 小图的诱导子图作为真实图样本
精确基准	$n \leqslant 24$ 时 MaxCut 的精确最优值可由暴力枚举或 ILP（OR-Tools/CBC）求得，用于把模拟结果换算为近似比。**仅作参照，不计入贡献**
内存与计时	`resource.getrusage`、`psutil`、`tracemalloc`；`time.perf_counter`；`taskset`、`cpupower`
统计与作图	Python + SciPy + Matplotlib
算力	纯 CPU。硬约束见第 2 块的比特上限表。单次 $n = 28$、$p = 3$ 的态矢量模拟估计数十秒至数分钟（**须实测标定**）；总工作量约 2 方法 × 8 比特档 × 4 层数 × 30 图 × 10 重复，须靠"先标定单次成本再裁剪参数网格"控制在 60–100 机时内。**超出范围**：$n \geqslant 30$ 的双精度态矢量模拟、密度矩阵（density matrix）方法（内存为 4^n，$n = 15$ 即达 8.6 GB）、任何含噪声模型的多轨迹模拟——三者均须在正文明确声明为超出范围

6 · 方法路径

1. 装 Qiskit/Aer，跑通官方教程算例；自行实现 NumPy 态矢量模拟器；完成验收第 1 条的三重校验并归档。
2. 核实工具能力边界（本课题第一优先级）：Aer 的 `tensor_network` 方法是否需要 GPU；`save_statevector` 是否产生副本；多比特门是否分配临时缓冲；单精度模式如何开启及其精度代价；`max_parallel_threads` 的语义。全部以实测内存与文档双重确认，写成一张能力边界表。
3. 实测标定比特上限：从 $n = 20$ 起逐比特上推，直到出现内存错误或严重换页，记录每一步的峰值 RSS；分别在"保存态矢量"与"仅取测量统计"两种模式下各做一遍（这是 H2 的直接检验）。
4. 构建实例集与线路生成器，固定 QAOA 参数（使用公开的解析值或文献报告的预优化值，不自行优化）。
5. 执行 MPS 侧的键维收敛扫描，为每个 (n, p) 确定达到保真度目标的最小键维，记录截断误差累计。
6. 执行主扫描（两种方法 × 参数网格 × 重复），出成本曲线、交叉点 $n^*(p)$、双成本口径对照表。
7. 独立交叉校验：（a）用 `quimb` 或 `qsim` 复算至少 3 个 (n, p) 组合，验证成本量级与交叉点位置；（b）在另一台内存不同的机器（如 8 GB 或 32 GB）上复测比特上限表，验证"实际上限 = 理论上限 − 缺口"这一关系是否随内存规模成立。

7 · 新颖性边界

本课题不声称：不提出新的模拟算法，不改进 QAOA，不声称任何关于 QAOA 近似比或参数优化的新结论（这一方向文献极多，本项目一律引用不声称），不涉及任何真实量子硬件，不声称"量子优势"相关的任何判断。Qiskit、Aer、quimb、NetworkX 与全部 QAOA 理论均为他人工作，标注为对照基准，不计入本项目贡献。

已有工作完成了什么：Qiskit Aer 文档给出了态矢量的内存公式与多种模拟方法；QAOA 求解 MaxCut 的性能、参数优化与理论分析已有大量公开结果；模拟器的性能标度在 GPU 与 HPC 平台上已被系统研究。**丘奖计算机赛道 2023 年铜奖《MNIST Handwritten Digit Classification with Quantum Neural Network》（北京市第一零一中学）**是历届获奖中与量子计算最相邻的一项，但它做的是量子神经网络在图像分类上的应用，与本课题的模拟方法学成本标定在对象、方法与判断依据上均不重叠。

本项目的贡献：给出**纯 CPU 消费级硬件上的模拟可行边界与方法交叉点的定量标定**——包括实测的比特上限及其与理论值的缺口归因、MPS 与态矢量的成本交叉点 $n^*(p)$、以及一份可直接复用的资源判据。这是主结论，属于"方法层面的可复现性贡献"，定位必须在论文摘要第一句就写明，避免被误读为 QAOA 研究。

为什么有价值：量子算法的教学与入门研究绝大多数在个人电脑上进行，而"我这台机器能模拟到多少比特、该用哪种方法"目前只能靠试错。一份带误差棒的可行边界表与交叉点判据，对这一大批使用者是可直接采纳的结果。

风险提示：本路线的最大风险是**被误读为"又一个 QAOA 项目"**。缓解方式是把 QAOA 严格降格为"提供可控纠缠强度的线路族"，正文中不出现任何关于 QAOA 优劣的论断，全部结论只关于模拟成本。

8 · 决策门槛（go / no-go）

- **第 4 周**：内存上限实测必须完成，且实测表与第 2 块的理论表逐格对上。若实测上限比理论值低 3 比特以上且原因不明，**这不是失败而是研究内容**——把归因分析正式列为主结论之一（H2 的加强版），但必须在本周确定分析手段（内存剖析工具）。
- **第 8 周**：工具能力边界表必须完成，`tensor_network` 方法的 GPU 依赖必须核实清楚——这是需要提前确认而非边做边发现的事项。若 Aer 的 MPS 方法在本机不可用或性能异常，**降级路径 A**：MPS 侧改用 `quimb`（纯 Python + NumPy，CPU 原生，键维完全可控），主结论框架（两类方法的成本交叉点）完全保留，仅实现来源改变。
- **第 14 周**：三重方法学校验硬门槛。若逐振幅比对偏差 $> 10^{-10}$，排查门序约定（比特序 little/big endian）、参数符号约定、随机种子三项最常见原因；第 17 周仍不过关则**降级路径 B**：放弃与 Aer 的逐振幅比对，改以自建 NumPy 模拟器为唯一真值来源（$n \leqslant 24$ 内它足够快），Aer 降为被测对象之一而非参照。校验硬门槛替换为"自建模拟器在 $n \leqslant 16$ 上与暴力解析结果偏差 $\leqslant 10^{-12}$"。
- **第 22 周**：键维收敛扫描必须完成。若 MPS 在 $p \geqslant 2$ 时键维就爆炸到无法在可行时间内达到保真度目标，**降级路径 C**：把 p 的范围缩为 $\{1, 2\}$，并把研究问题重心从"交叉点"转为"**MPS 可用区的边界刻画**"——即在 (n, p, d) 空间中标出 MPS 仍优于态矢量的区域及其边界形状。主结论框架保留，且这个刻画本身信息量不低。
- **第 30 周**：主扫描完成且 H1 有明确判定。若进度不足，**降级路径 D**：图类别从 3 类减为 2 类（3-正则 + Erdős – Rényi），比特档位从 8 个减为 6 个，重复次数保持 10 次不变。
- **预算裁剪顺序**：图类别数 → 比特档位数 → p 档位数（下限 2）→ （绝不裁剪）三重校验、键维收敛扫描与重复次数。
- **选择前提**：仅在学生已具备线性代数基础（张量积、酉算符）、且明确接受"本课题不产出量子算法结论、只产出模拟方法学结论"这一定位时才启动。若学生或家长期待"做量子计算研究"的叙事，须提前说清本课题的实际内容，否则中途改题的风险很高。

K10 · 图算法正确性的 Lean 4 机器检验：最大流—最小割定理形式化的构建与形式化工作量剖析

`推荐优先级：最低（风险最高，需最强学生）` · `分族：形式化验证与定理证明助手` · `参赛子类：计算机–形式方法与逻辑` · `资源需求：纯 CPU 笔记本 / 16 GB 内存（编译内存是唯一瓶颈）` · `技能取向：数理逻辑 + 函数式编程 + 极强的坚持力`

1 · 研究问题

在 Lean 4 与 mathlib 生态中，一名高中生能否在 12 个月内完成有限有向图上**最大流 – 最小割定理**（max-flow min-cut theorem）及 Ford – Fulkerson 方法在整数容量下**终止性**的、零 `sorry` 的机器检验证明？在这一过程中，纸笔证明里被视作"显然"的哪些步骤消耗了不成比例的形式化工作量（以证明行数、新增定义数、编译时间计），mathlib 的现有基础设施覆盖了其中多大比例？

2 · 研究背景与空白

技术背景。 交互式定理证明助手（interactive theorem prover）把数学证明写成机器可完全检查的形式对象：每一步推理都必须由内核逐条验证，任何跳步都会被拒绝。Lean 4 及其社区数学库 mathlib 是当前发展最快的体系，mathlib 已包含逾 11.5 万条定义与 23.2 万条定理，图论部分（`SimpleGraph` 及其配套）近年增长迅速，2025 年已有 Tutte 定理作为教学性形式化项目完成的公开记录，也有把超立方体最优 pebbling 数完整形式化的工作（20 个 Lean 源文件、约 12,071 行）。**这类课题的算力特性极为特殊：它几乎不需要计算，只需要编译**——Lean 的编译内存需求较高但完全在 16 GB 笔记本能力内，一台没有 GPU 的机器与一台服务器在形式化能力上没有差别。这是全部十条路线中"结构性绕开算力墙"最彻底的一条。

已有工作到哪一步。 mathlib 的图论部分已有匹配理论（Hall 婚配定理、Tutte 定理）等成果；形式化算法正确性（排序、树的插入删除）在 Lean 4 中被明确列为可行的应用方向；有面向 Lean 4 的综述（arXiv:2501.18639）与自动形式化基准工作。**必须核实的关键事实：mathlib 当前是否已包含最大流 – 最小割定理或其等价形式（例如通过 Menger 定理或线性规划对偶）。** 这一项无法凭记忆断定，必须在项目启动的头四周内用 mathlib 官方文档搜索与 `loogle` / `Moogle` 检索工具逐一确认，**并以官方仓库当前状态为准**。若已存在，整条路线必须立即换目标定理（见第 8 块降级路径）。

空白在于：形式化数学社区里"某定理是否已被形式化"有公开记录，但"形式化它花了多少代价、代价花在哪里"几乎没有系统的定量剖析；而这恰恰是决定形式化方法能否规模化的关键数据。本课题因此设计成**双交付物**：一份可编译的形式化，加一份形式化工作量剖析数据集。后者的价值不依赖前者是否完全完成——即使定理只形式化到一半，工作量剖析仍然成立。这一设计是本路线在高风险下仍可交付的根本保障。

3 · 可检验假设

- **H1（工作量放大比）**：完成的形式化的 Lean 代码行数与对应纸笔证明行数之比 $\geqslant 30:1$，且其中 $\geqslant 60\%$ 的行数消耗在纸笔证明中**不显式出现**的基础设施上（有限性论证、可判定性实例、图与流的表示转换、类型强制转换）。若比值 $< 15:1$，说明 mathlib 的覆盖已远超预期，这本身是对形式化成熟度的一次正面测量。
- **H2（mathlib 覆盖率）**：证明所需的引理中，能直接由 mathlib 现有定理满足的比例 $\geqslant 50\%$，而需自行证明的部分集中在"图上的归纳构造"这一类。若覆盖率 $< 30\%$，则说明图算法方向的形式化基础设施仍然薄弱，本项目的中间引理本身具有向 mathlib 贡献的价值。

4 · 量化验收标准

1. **方法学校验（硬门槛）**：在启动主定理之前，学生必须先独立完成一个**已有公开形式化可供对照**的小目标——例如插入排序的正确性与保序性、或 mathlib 中已有定理的一条独立重证——要求：`lake build` 通过、零 **`sorry`**、零 **`axiom`**（除 mathlib 标准公理外）、零 **`native_decide`**，并把自己的行数与结构同公开形式化对照，说明差异来源。这一步不过关（尤其是零 `sorry` 这一条），说明学生尚不具备完成主定理的能力，后续全部工作无效。**"有 `sorry` 的证明不算证明"必须写死为项目纪律**。
2. **主交付物的完成度判据**：主定理的形式化按预先声明的引理清单逐条计分，每条引理有明确的"完成/未完成"二值状态（完成 = 零 `sorry` 且被主定理实际引用）。最终报告必须给出完整的引理清单与状态表，**不得用"基本完成"这类表述**。
3. **工作量剖析数据集（第二交付物，与主交付物同等重要）**：对每条引理记录——Lean 行数、定义数、所用 mathlib 引理清单、首次编译通过前的提交次数、累计工时（学生自记，须每日记录而非事后回忆）、单独编译时间（中位数与 IQR，$\geqslant 5$ 次重复）。这份数据集本身开源发布。
4. **统计口径**：行数比与工时比是有界正量且分布高度偏斜，**报中位数与 IQR，不报均值**；覆盖率给 Clopper – Pearson 精确二项置信区间；分类（"基础设施" vs "核心论证"）须有预先写死的、可复现的判据，并由两人独立分类后报 Cohen's κ（$\kappa < 0.6$ 则判据需重写）。
5. **可编译性与可复现性**：整个项目作为一个 Lean 4 `lake` 包发布，锁定 mathlib 版本（`lake-manifest.json` 提交），提供在干净环境中的一键构建脚本与 CI 配置；第三方 clone 后 `lake build` 必须成功。
6. **诚实性条款**：论文中出现的每一条形式化结论都必须对应仓库中一个零 `sorry` 的定理，并给出文件名与行号；任何未完成部分必须在正文中显式列出，不得省略。

5 · 数据与工具

用途	来源 / 工具
定理证明助手	Lean 4（leanprover/lean4，官方 `elan` 工具链管理器安装）+ mathlib4（社区数学库，通过 `lake` 依赖引入，版本须锁定）。**纯 CPU，无 GPU 需求；mathlib 完整构建需数 GB 磁盘与较高编译内存，首次构建建议使用官方缓存 `lake exe cache get` 而非本地全量编译——这一点必须在第 1 周核实**
定理检索	mathlib4 官方文档站（leanprover-community.github.io/mathlib4_docs）、`loogle`、`Moogle`、`exact?` / `apply?` 策略。**用于核实"目标定理是否已存在"，这是本路线的头号前置任务**
参照形式化	已公开的 Lean 4 教学性形式化项目（如 Tutte 定理形式化、超立方体 pebbling 形式化的 12,071 行代码）作为工作量对照点。**仅用于校验与对比，不计入本项目贡献**
纸笔证明底本	标准教科书证明（如 CLRS 的最大流最小割一节、Diestel《Graph Theory》），用于计算"纸笔行数"这一分母；底本须在正文指明版本与页码
备选目标定理	König 定理（二部图最大匹配 = 最小顶点覆盖）；Ford–Fulkerson 在整数容量下的终止性（独立于最大流最小割）；Dilworth 定理；Edmonds–Karp 的 $O(VE^2)$ 界。**每一条都须先核实 mathlib 中是否已存在**
工时与提交记录	Git 提交历史 + 学生每日工时表（模板固定，每日填写）
算力	纯 CPU。瓶颈是 Lean 编译的内存与时间：单文件编译通常秒级到分钟级，mathlib 缓存下载数 GB。16 GB 内存充裕。**超出范围**：任何需要大规模自动化证明搜索或训练模型的方案不在本课题范围内

6 · 方法路径

1. 装 `elan` + Lean 4 + mathlib（用官方缓存），跑通 mathlib 的示例与 `lake build`；建立项目骨架与 CI。
2. **核实目标定理的现状**（本路线第一优先级、不可后延）：用官方文档、`loogle`、`exact?` 逐一检索最大流最小割及其等价形式、Menger 定理、线性规划对偶在 mathlib 中的状态，形成书面结论并附检索记录。结论决定主目标是保留还是切换。
3. 完成方法学校验（验收第 1 条）的小目标形式化，零 `sorry` 通过，并与公开形式化做行数与结构对照。
4. 把主定理拆成预先声明的引理清单（建议 15 – 30 条），为每条写出纸笔证明与 Lean 语句签名（`theorem ... : ... := sorry`），此清单在项目中期后**不得再增删**（增删须记录并说明理由）——这是使工作量剖析有意义的前提。
5. 自最基础的引理起逐条消去 `sorry`，每条完成即记录工作量剖析的全部字段；每周提交并保持 CI 绿。
6. 中期（第 26 周）盘点：按引理清单计分，判断主定理能否在剩余时间内完成，据此触发第 8 块的门槛决策。
7. 独立交叉校验与收尾：（a）请第三方（指导教师或社区）在干净环境中 clone 并 `lake build`，确认可复现；（b）对工作量分类做双人独立标注并报 κ；（c）把可复用的中间引理整理为向 mathlib 提交的候选（是否被接受不作为验收条件，但提交本身作为附录证据）。

7 · 新颖性边界

本课题不声称：不声称证明了任何新数学定理（最大流最小割定理是 1956 年的经典结果），不声称发明了新的证明方法，不开发新的证明助手或自动化策略，不涉及任何机器学习辅助证明。Lean 4、mathlib 与全部参照形式化均为他人工作，标注为对照基准，不计入本项目贡献。

已有工作完成了什么：mathlib 已含逾 11.5 万定义、23.2 万定理，图论部分已有 Hall 与 Tutte 等匹配理论成果，2025 年公开了 Tutte 定理的教学性形式化项目，另有超立方体最优 pebbling 数的完整 Lean 4 形式化（20 文件、12,071 行）；Lean 4 形式化算法正确性（排序、树操作）已被明确列为成熟应用方向。**丘奖计算机赛道有两项高度相邻的历届获奖工作必须点名：2024 年金奖《LLM Mathematical Reasoning Grounded with Formal Verification》与 2023 年优胜奖《Close the Loop of Neural Intuition and Logical Reasoning for Geometry Theorem Proving》**。两者的共同点是"用形式化工具为语言模型或神经直觉提供验证"，重心在 AI 一侧；本课题**完全不涉及任何模型**，重心在形式化工程本身与其工作量的定量剖析，方法与判断依据均不重叠。这一差异必须在论文引言中明确写出，否则极易被评委误认为同类。

本项目的贡献：（1）一份零 `sorry` 的、可被第三方一键复现的图论定理形式化（若 mathlib 中确不存在，则该形式化本身即为对公共基础设施的增量）；（2）一份**形式化工作量剖析数据集**——逐引理的行数、mathlib 依赖、工时与编译成本，以及"基础设施 vs 核心论证"的成本分解。第二项是主结论，也是本课题在主定理未能完成时仍然成立的保障。

为什么有价值：形式化数学的规模化瓶颈不在于"能不能形式化"，而在于"代价花在哪里"。逐引理的成本分解数据在公开文献中极为稀缺，而它直接指向"应该优先建设哪些基础设施"这一决策。

风险提示：若第 2 步核实发现 mathlib 已包含目标定理，本课题**不因此失败**——主目标切换到备选清单中的下一条，工作量剖析框架完全不变。这一点必须在方案里写死，因为它是本路线最可能触发的意外。

8 · 决策门槛（go / no-go）

- **第 4 周**：目标定理现状核实完毕（不可后延）。若最大流最小割已在 mathlib 中，**降级路径 A**：按备选清单顺序切换目标（König 定理 → Ford – Fulkerson 整数终止性 → Dilworth 定理 → Edmonds – Karp 复杂度界），每次切换前同样先核实。工作量剖析框架与全部验收标准保持不变。
- **第 10 周**：方法学校验硬门槛。若学生无法独立完成小目标形式化并零 `sorry` 通过，**这是唯一一条应当整体放弃的门槛**——本路线对形式化能力的要求无法通过投入时间弥补。此时立即整体改投 K03（同属离散/逻辑方向，SAT 编码课题对同类兴趣的学生适配度最高），已完成的 Lean 学习不浪费（SAT 与逻辑同源）。这一条必须提前告知学生与家长。
- **第 16 周**：引理清单必须锁定，且已有 $\geqslant 20\%$ 的引理零 `sorry` 完成。若完成比例 $< 10\%$，**降级路径 B**：把主目标从"完整定理"缩为"**该定理的一个受限版本**"（例如限制在整数容量且图为有限简单有向图、或只形式化"最大流 $\leqslant$ 最小割"这一较易的方向），并把剩余方向明确列为未完成。主交付物缩水但工作量剖析完整。
- **第 26 周**：中期盘点。若完成比例 $< 50\%$，**降级路径 C**：正式把论文重心从"我们形式化了 X"转为"**形式化 X 的工作量剖析与瓶颈识别**"，主定理的未完成部分作为"已识别的瓶颈"写入结果而非隐去。这一降级保留了全部主结论框架，且在形式化社区中是被接受的贡献形态。
- **第 40 周**：可复现性门槛。第三方干净环境 `lake build` 必须成功。若 mathlib 版本漂移导致构建失败，**降级路径 D**：把 mathlib 版本回退并锁定到最后一个可构建版本，在正文声明版本与日期。**不得以"本地能跑"代替第三方复现**。

- **预算裁剪顺序**：向 mathlib 提交贡献（可完全砍掉）→ 引理清单规模（可缩至 15 条）→ 主定理的完整性（可缩为受限版本）→ （绝不裁剪）零 `sorry` 纪律、工作量剖析数据的每日记录、第三方可复现构建。
- **已知困难的定位**：图上的归纳与有限性论证在 Lean 中的代价远高于纸笔，这是本课题预期中的主要成本来源——**这个困难本身就是研究内容**，应被测量并写入结果，而不是被当作障碍绕开。
- **选择前提**：仅在学生已经自学过 Lean 或 Coq 的基础（能独立写出十几行的证明）、数学基础扎实、且明确接受"可能只完成一半定理"这一风险时才启动。这是全部十条路线中唯一一条对学生个人能力有硬性前置要求的路线；不满足前置条件时的正确做法是选 K03 而不是硬上。

K11 · 长时序预测中「线性基线 vs Transformer」争议在中国气象与电力新样本上的独立仲裁——兼量化随机划分导致的性能高估

`优先级:最高` · `分族:时间序列预测` · `资源:纯 CPU / 零预算` · `技能:Python + 统计` · `类型:独立复检`

1 · 研究问题

> Zeng 等（2023）「单层线性模型 DLinear 在长时序预测（long-term time series forecasting, LTSF）上普遍优于 Transformer 类模型」的结论，在中国气象站点与电力负荷的独立新样本上是否成立？若在部分序列上反转，反转能否由序列的趋势/周期强度与平稳性等可预测性结构解释？

2 · 研究背景与空白

长时序预测指给定历史窗口预测未来 96 – 720 步。2020 – 2022 年 Informer、Autoformer 等 Transformer 模型在 ETT、Weather、Electricity 基准上刷榜；Zeng 等（AAAI 2023）用两个单层线性网络（DLinear）在这些基准上大幅反超，引发方向性争议；Nie 等（ICLR 2023, PatchTST）随后反驳称等规模设置下 Transformer 仍更优。争论双方给出的数字互相矛盾（同一 ETTh1-96 任务 MSE 报告差可达 10% 以上）。

空白在：正反双方都在同一批既有西方基准上互搏，尚无第三方在全新地理与时间覆盖的样本上做中立仲裁；且主流基准全部使用固定划分，随机划分（数据泄漏）造成的高估幅度在这批任务上缺少系统量化。方法全开源、纯 CPU 可跑、仲裁结论正反都成立，适合一年期学生课题。

3 · 可检验假设

- H1：在 ⩾12 条中国站点日尺度气象序列与 ⩾2 条电力负荷序列上，DLinear 在 96 步与 336 步预测的 MSE 与最优 Transformer 基线（PatchTST）差距在 ±10% 以内，即「简单基线足够」在新样本上复现；若某类序列显著反转（配对差的 95% 置信区间不含 0），反转可由该序列的季节强度指标预测（logistic 回归 AUC ⩾ 0.75）。
- H2：随机划分相对按时间划分会使所有模型的测试 MSE 系统性偏低（性能高估）⩾15%，且高估幅度与序列一阶自相关强度正相关（Spearman $\rho > 0.5$）。

4 · 量化验收标准

1. **方法学校验（硬门槛）**：自建管线在 ETTh1 上重现 DLinear 论文报告的 96/336 步 MSE，偏差 ⩽5%；同时重现 PatchTST 官方仓库结果偏差 ⩽5%。不过关则后续全部结论无效。
2. 每组实验 5 个随机种子，报均值 ± 标准差，不报单次最好结果。
3. 模型对照采用双成本口径：等参数量、等 CPU 墙钟训练时间，两组都报。
4. 主口径为按时间划分训练/测试；额外做一次随机划分对照，量化高估幅度（对应 H2）。
5. 逐序列给出胜负表，配对差异用自助抽样（bootstrap，⩾1000 次）给 95% 置信区间；「差异不显著」为有效结论。
6. 全部脚本、配置与处理后数据开源，第三方可一键重跑。

5 · 数据与工具

用途	来源 / 工具
方法学校验与对照基准	ETT / Weather / Electricity（Autoformer 官方仓库统一下载）——**仅用于校验与对比，不计入本项目数据贡献**
新样本：中国气象	NOAA ISD / GSOD（含数百个中国站点，逐日温度/风速/气压，1950s–今），按站点切片下载，单站 MB 级
新样本：电力负荷	中国省级公开负荷序列**需核实**（备选：全国电工数学建模竞赛公开负荷数据；再备选 UCI ElectricityLoad 仅作对照）
模型实现	DLinear/NLinear（官方仓库）、PatchTST（官方仓库）、季节朴素与 SARIMA 基线（statsmodels）——预训练/官方代码不计入本项目方法贡献
算力量级	DLinear 在 ETTh1 单次训练 CPU 约 1–3 分钟；PatchTST 单数据集 CPU 约 20–60 分钟（第 0–2 周实测校准；若单次 >2 小时则缩短回看窗口至 336、减少输入通道）

6 · 方法路径

1. 装环境，跑通 DLinear 与 PatchTST 官方算例，完成第 4 块第 1 条校验。
2. 实测两模型在本机 CPU 的单位墙钟，写出能力边界表（哪些配置超出范围）。
3. 下载并清洗 NOAA 中国站点序列与负荷序列，给出缺失处理的可复现判据。
4. 统一评测协议（回看 336、预测 96/336、时间划分），跑全部模型 × 序列 × 5 种子。
5. 加跑随机划分对照，计算逐序列高估幅度并与自相关强度做秩相关。
6. 计算各序列季节强度/平稳性指标，检验其对胜负的预测力（H1 后半）。
7. 交叉校验：用 statsmodels 独立实现的季节朴素基线核对自建评测代码的 MSE 计算。

7 · 新颖性边界

- 本课题不声称提出任何新预测模型，DLinear 与 PatchTST 的原始结论均已发表，不得声称为本项目发现。
- 已有工作：Zeng et al. (AAAI 2023) 与 Nie et al. (ICLR 2023) 已在 ETT/Weather/Electricity/Traffic 上给出相反结论；历届丘奖中 C-TranAD（2025 优胜，复值 Transformer 异常检测）做的是异常检测新模型，与本题「基线仲裁」问题方向不同。
- 本项目贡献（主结论）：其一，在中立的新地理样本上对该争议做独立仲裁；其二，量化随机划分在 LTSF 任务上的高估幅度并给出与序列结构的关联判据。
- 价值：争议双方都未在自身之外的样本上被检验，独立仲裁无论指向哪边都构成有效结论；泄漏高估的量化对社区实践有直接参考价值。

8 · 决策门槛（go / no-go）

- **第 2 周末**：ETTh1 校验偏差 >5% → 排查至第 4 周；仍不过则**降级**为「推理端复检」：加载官方 checkpoint 只做评测与新样本推理，保留仲裁与高估量化两条主结论框架。
- **第 6 周末**：核实中国电力负荷公开来源。拿不到 → **具名降级**：全部换为 NOAA 中国站点气象序列并扩至 ⩾20 条，主结论不变，仅应用面变窄。
- **第 20 周**：若 PatchTST 超参搜索 CPU 墙钟超 60 小时预算 → 冻结官方超参不搜索，并在论文限制一节明确声明。

- 预算裁剪顺序：先砍 720 步预测长，再砍序列条数（下限 12 条），最后砍随机划分对照的重复次数（下限 3 种子）。

K12 · SMOTE 类重采样在梯度提升分类器与阈值调优对照下的真实收益——PR-AUC 与概率校准双口径的独立复检

`优先级:高` · `分族:评测审计 / 不平衡学习` · `资源:纯 CPU / 零预算` · `技能:Python + 统计` · `类型:评价维度扩展`

1 · 研究问题

> Elor 等（2022）「对强分类器而言 SMOTE 类重采样基本无益」的结论，在极端不平衡（正类 <0.5%）的公开数据集与概率校准（calibration）这一新评价维度上是否仍然成立？重采样带来的排序指标变化与校准恶化之间是否存在系统性权衡？

2 · 研究背景与空白

SMOTE（Synthetic Minority Oversampling Technique, 2002）在少数类样本间线段上合成新样本，是不平衡分类最常被套用的预处理。Elor & Averbuch-Elor（2022, "To SMOTE, or not to SMOTE?"）在 73 个数据集上系统检验，结论：对 XGBoost/Catboost 等强分类器，重采样在排序指标上无稳定收益，对弱分类器才有益；后续 benchmark（arXiv:2303.03094, 2023）进一步支持「恰当调阈值即可替代重采样」。

空白在：这批审计集中在中度不平衡与排序/分类指标；正类占比 <0.5% 的极端区间覆盖稀疏，且重采样对**预测概率校准质量**（欺诈风控、医疗分诊真正消费的量）的破坏少有系统量化。数据全公开、单模型训练秒级，结论正负都成立。

3 · 可检验假设

- H1：在 ⩾10 个正类占比 0.02% – 2% 的公开数据集上，LightGBM + 阈值调优相对 LightGBM + SMOTE 的 PR-AUC 差异的 95% 置信区间包含 0（即极端区间上原结论复现）。
- H2：SMOTE 使预测概率系统性偏高，Brier 分数与期望校准误差（ECE）相对不重采样恶化 ⩾30%，且恶化幅度随过采样倍率单调上升。

4 · 量化验收标准

1. **方法学校验（硬门槛）**：用原论文开源代码/协议在其 3 个数据集上重现「强分类器下 SMOTE 无益」的方向性结论，且 PR-AUC 数值偏差 ⩽5%。不过关则后续全部结论无效。
2. **统计口径预先写死**：报 PR-AUC、召回、精确率、Brier、ECE 与可靠性图；**不报 accuracy**——正类 0.2% 时全预测负类即得 99.8% accuracy，该指标无信息。
3. 5×2 分层交叉验证 × 5 随机种子，报均值 ± 标准差；配对差用自助法给 95% 置信区间。
4. 不平衡率控制实验：对同一数据集降采样正类至 2%/0.5%/0.1%/0.02% 四档，画收益 – 不平衡率曲线（⩾4 个参数点）。
5. 对照方法 ⩾5 种：不处理、类权重、阈值调优、SMOTE、Borderline-SMOTE、随机欠采样。
6. 全部管线开源，第三方一键重跑；总计算量预估 <20 CPU 小时。

5 · 数据与工具

用途	来源 / 工具
极端不平衡数据	ULB Credit Card Fraud（Kaggle 公开，284,807 行，正类 0.17%）；KDD Cup 1998、Santander、保险理赔等 OpenML 公开集（openml.org 按 imbalance 检索下载）——**仅作输入，不计入本项目数据贡献**
校验对照	Elor 2022 论文开源代码与其数据清单——**仅用于校验，不计入贡献**
重采样实现	imbalanced-learn（内置 SMOTE 全家族；文档层面核实默认 k 近邻参数与原论文一致）
分类器	LightGBM / XGBoost / logistic 回归（CPU 上单数据集训练秒—分钟级）
校准度量	scikit-learn `calibration_curve`、自实现 ECE（需自行实现分箱，写明箱数判据）

6 · 方法路径

1. 装环境，复现原论文 3 个数据集结论，完成硬门槛校验。
2. 核实 imbalanced-learn 各实现与原始算法论文的参数差异，写成对照表。
3. 按不平衡率与领域分层选定 ⩾10 个公开数据集，冻结清单后不再增删。
4. 跑全因子实验：数据集 × 6 种处理 × 3 分类器 × 5 种子。
5. 做不平衡率降采样控制实验，绘收益曲线。
6. 分析排序指标与校准指标的权衡结构（H2），配可靠性图。
7. 交叉校验：用另一套独立实现（如自写朴素 SMOTE）核对合成样本分布一致。

7 · 新颖性边界

- 本课题不声称提出新重采样方法；「强分类器下 SMOTE 无益」由 Elor et al. (2022) 与 2023 年 benchmark 已发表，不得声称为本项目发现。
- 已有工作：Elor 2022 覆盖 73 个中度不平衡数据集、排序指标口径；2303.03094 加入调参对照。历届丘奖中 2025 优胜「Class-Imbalanced Semi-Supervised rPPG」是把不平衡学习应用于具体任务，与本题的方法审计方向不同。
- 本项目贡献（主结论）：把审计推进到极端不平衡区间（评价样本切片扩展）+ 校准维度（评价维度转换），并给出「排序收益 vs 校准代价」的量化权衡。
- 价值：校准恶化直接影响风控/医疗中概率的可用性，这一代价在教程与实践中普遍被忽略；若校准并不恶化（H2 否证），同样是对社区流行说法的有效纠正。

8 · 决策门槛（go / no-go）

- **第 3 周末**：原论文代码若无法运行或结果偏差 >5% → 排查至第 5 周；仍不过则**降级**：以 imbalanced-learn 标准实现 + 自建协议重建基线，校验对象改为「重现 ULB 欺诈集上已发表的 LightGBM PR-AUC 区间」，审计框架不变。
- **第 8 周末**：确认 ⩾10 个满足不平衡率区间且列可用的数据集；不足 → 用降采样构造（已在设计中），并明确标注为半合成切片。

- **第 30 周**：若 H1、H2 均呈「无显著差异」→ 不改题；以等价性检验（TOST）报告效应量上界，负结论成文。
- 预算裁剪顺序：先砍 Borderline-SMOTE 等变体，再砍分类器至 2 种；数据集下限 8 个不可再砍。

K13 · 神经推荐是否真的在进步——MovieLens 与 Amazon 2023 新类目切片上的等墙钟基线审计，加测长尾覆盖与新颖度

`优先级:高` · `分族:推荐与信息检索` · `资源:纯 CPU / 零预算` · `技能:Python + 数据工程` · `类型:评价维度扩展 + 新样本复检`

1 · 研究问题

> Ferrari Dacrema 等（2019）「多数神经推荐模型可被近邻/线性基线打败」的结论，在 Amazon Reviews 2023 新类目切片上、以纯 CPU 等墙钟成本口径衡量时是否仍成立？把评价维度从准确类指标扩展到长尾覆盖率与新颖度后，基线与神经模型的排序是否改变？

2 · 研究背景与空白

Ferrari Dacrema, Cremonesi & Jannach（RecSys 2019）审计了 18 个顶会神经推荐模型：仅 7 个可复现，其中 6 个被 ItemKNN 等简单启发式打败；Rendle 等（2020）进一步证明调好参的矩阵分解（matrix factorization, MF）仍是强基线。此后 EASE（2019）等线性模型持续刷新「简单方法上限」。

空白在：该审计（a）评价维度只覆盖 HR/NDCG 准确类指标，长尾覆盖率、新颖度、流行度集中度这些部署侧关键维度未进入胜负判定；（b）数据止于 2019 年前的基准，Amazon Reviews 2023（时间与类目全新）上无人重做；（c）成本口径按「训练到收敛」，消费级 CPU 等墙钟预算下的排序未被报告。数据公开、基线全部分钟级，适合学生课题。

3 · 可检验假设

- H1：在 MovieLens-1M 与 Amazon 2023 两个类目切片上，等 CPU 墙钟预算（每模型 2 小时）下，EASE/ItemKNN 的 NDCG@10 不低于 NeuMF 的 95%（即基线结论在新样本、新口径下复现）。
- H2：若把胜负指标换成目录覆盖率（coverage@10）与平均推荐新颖度（负 log 流行度），至少一处排序反转：神经模型或 MF 在 beyond-accuracy 维度显著优于近邻基线（配对自助 95% CI 不含 0）。

4 · 量化验收标准

1. **方法学校验（硬门槛）**：用 Dacrema 开源评测框架重现其 MovieLens-1M 上 ItemKNN 与一个神经模型的 HR@10 / NDCG@10，偏差 ⩽5%。不过关则后续全部结论无效。
2. **统计口径预先写死**：排序指标报 HR@10 / NDCG@10 / MRR；beyond-accuracy 报 coverage@10、新颖度、推荐列表基尼系数；**不报 accuracy**（top-N 推荐无意义）。
3. 划分口径双轨：留一法（与原文对齐）+ 按时间划分（防泄漏），并量化两者差异——推荐评测的划分泄漏本身是已知问题（Meng et al. 2020），本项目给新数据上的量化。
4. 每模型 5 个随机种子，报均值 ± 标准差。
5. 等墙钟口径：每模型固定 2 CPU 小时训练+调参预算（含早停），同时报「官方默认配置」口径，两种都报。
6. 全部脚本与切片构造代码开源，一键重跑。

5 · 数据与工具

用途	来源 / 工具
校验基准	MovieLens-1M / 20M（GroupLens 官网直接下载）——**仅作输入与校验，不计入本项目数据贡献**
新样本	Amazon Reviews 2023（McAuley Lab，HuggingFace 按类目切片下载）；选 2 个类目并降采样至 ≤500 万交互，5-core 过滤
评测框架	Dacrema 官方仓库（GitHub: RecSys2019_DeepLearning_Evaluation）；RecBole（内置 NeuMF/EASE/ItemKNN——需核实其指标实现与 Dacrema 框架一致，不一致则统一用自建评测层）
模型	ItemKNN、EASE、MF（implicit 库 ALS）、NeuMF；NeuMF 在 ML-1M 上 CPU 单轮约 5–15 分钟、30 轮过夜可行；EASE 为闭式解，ML-1M 上分钟级
算力边界	Amazon 全类目（数亿交互）**超出范围**；≤500 万交互切片内全部模型 CPU 可行

6 · 方法路径

1. 装环境，跑通 Dacrema 框架自带算例，完成硬门槛校验。
2. 核实 RecBole 与 Dacrema 框架的负采样评测差异（已知会显著改变指标），冻结统一评测协议。
3. 构造 Amazon 2023 两个类目切片（5-core、降采样），发布切片构造脚本。
4. 跑全部模型 × 数据集 × 5 种子，双成本口径、双划分口径。
5. 计算 beyond-accuracy 指标，检验 H2 的排序反转。
6. 汇总胜负表与置信区间，分析反转与数据集长尾结构的关系。
7. 交叉校验：ItemKNN 用两套独立实现互核 NDCG（偏差 ≤1%）。

7 · 新颖性边界

- 本课题不声称提出新推荐算法；「基线打败神经模型」由 Dacrema et al. (2019)、Rendle et al. (2020) 发表，不得声称为本项目发现。
- 已有工作：Dacrema 覆盖 2015 – 2018 顶会模型、准确类指标、留一法口径；Meng et al. (2020) 指出划分方式问题但未在 2023 数据上量化。
- 本项目贡献（主结论）：其一，Amazon 2023 新切片 + 等 CPU 墙钟口径下的独立复检；其二，把胜负判定扩展到覆盖率/新颖度维度并检验排序反转。
- 价值：若基线结论在新数据继续成立，是对「进步幻觉」的又一次独立支持；若 beyond-accuracy 维度反转，则给「神经模型价值在多样性而非精度」提供量化证据——两个方向都成文。

8 · 决策门槛（go / no-go）

- **第 3 周末**：Dacrema 框架在本机跑通且校验达标；若框架依赖损坏（2019 年代码）→ 限期 2 周迁移到 RecBole 并用其复现论文数字作为替代校验，评测协议声明差异。
- **第 8 周末**：Amazon 2023 切片清洗完成且 NeuMF 在切片上单轮 ≤20 分钟；超时 → **具名降级**：切片降到 ≤100 万交互，或砍 NeuMF 改为只审计 EASE/MF/ItemKNN 三强基线（审计框架与 H2 均保留）。
- **第 26 周**：若双数据集结论方向不一致 → 不视为失败，改以「结论的数据依赖性」为主轴成文，加做第三切片仲裁。

- 预算裁剪顺序：先砍 20M 规模实验，再砍时间划分双轨（保留声明），种子下限 3。

K14 · 深度知识追踪的增益边界——DKT 相对最优逻辑回归在 ASSISTments 与 EdNet 冷启动切片上的等成本复检

`优先级:高` · `分族:教育技术 / 人机交互` · `资源:纯 CPU / 零预算` · `技能:Python + 教育数据` · `类型:新样本 + 新切片复检`

1 · 研究问题

> Gervet 等（2020）「精心构造特征的逻辑回归（Best-LR）在多数数据集上与深度知识追踪（Deep Knowledge Tracing, DKT）打平」的结论，在学生冷启动切片（每个学生的前 10/25/50 次作答）上是否仍然成立？等 CPU 墙钟成本口径下，DKT 的剩余增益是否被训练成本抵消？

2 · 研究背景与空白

知识追踪根据学生历史作答序列预测下一题对错，是自适应学习系统的核心。Piech 等（NeurIPS 2015）的 DKT（LSTM）宣称在 ASSISTments 上 AUC 从 0.69 提到 0.86；但后续修正（数据重复行问题）与 Gervet 等（JEDM 2020）的系统对比表明：带时间窗特征的逻辑回归在 9 个数据集中的多数上与 DKT 打平（AUC 差 <0.01），仅在大数据集上 DKT 略优。

空白在：现有对比全部按「全历史平均」报 AUC；而真实部署最关键的是**冷启动阶段**（新学生前几十题）的预测质量——该切片上参数多的模型可能恰恰最弱，文献几乎不报。此外 EdNet（2020 公开）时间上晚于 Gervet 基准，可作独立新样本。数据全公开、模型量级小，纯 CPU 可完成。

3 · 可检验假设

- H1：在按学生分组划分下，Best-LR 与 DKT 的全序列 AUC 差 $\leqslant 0.01$（复现已发表结论）；但在「学生前 10 次作答」切片上，Best-LR 的 AUC 反超 DKT $\geqslant 0.02$（简单模型冷启动优势）。
- H2：等 CPU 墙钟口径（每模型 1 小时）下，DKT 因训练不充分损失 AUC $\geqslant 0.015$，而 Best-LR 无损失，即等成本口径改变胜负。

4 · 量化验收标准

1. **方法学校验（硬门槛）**：用 Gervet 开源代码在 ASSISTments2009（去重版）上重现 Best-LR 与 DKT 的 AUC（约 0.77 量级），偏差 $\leqslant 2$ 个百分点。不过关则后续全部结论无效。
2. **统计口径预先写死**：主指标 AUC（领域惯例）+ 辅报 PR-AUC 与校准 ECE；**不报 accuracy**（作答正确率约 65% 且随切片漂移，accuracy 混淆先验与判别力）。划分**必须按学生分组**（同一学生不得跨训练/测试），另做一次按交互随机划分对照，量化学生泄漏的高估幅度。
3. 冷启动切片预登记：前 10 / 25 / 50 次作答三档，切片判据在跑实验前冻结。
4. 每配置 5 个随机种子，报均值 ± 标准差。
5. 双成本口径：训练到收敛（早停）与等墙钟 1 CPU 小时，两组都报。
6. 全部代码开源；ASSISTments09 使用社区去重版本并写明版本号（该数据集的重复行问题曾污染大量论文——第 2 周核实清楚）。

5 · 数据与工具

用途	来源 / 工具
校验与主数据	ASSISTments 2009（去重版）/ 2012 / 2017（官方站点公开下载，34 万–600 万交互）——**仅作输入，不计入本项目数据贡献**
独立新样本	EdNet-KT1（Riiid 公开，GitHub 下载，1.3 亿交互）→ 按学生随机降采样 1–2%（约 100–260 万交互）
基线实现	Gervet 官方仓库（GitHub: theophilee/learner-performance-prediction，含 Best-LR/DKT/SAKT）——需核实依赖可装
DKT	PyTorch LSTM（约 10 万参数）；ASSISTments09 上 CPU 单轮约 3–10 分钟，20 轮 1–3 小时，可行；EdNet 降采样切片约 3–5 倍
算力边界	EdNet 全量与 Transformer 类 KT 模型（SAINT 等）**超出范围**；SAKT 小配置可作可选加项

6 · 方法路径

1. 装环境，跑通 Gervet 仓库，完成硬门槛校验并核实 ASSISTments09 版本问题。
2. 实测 DKT 在本机的单轮墙钟，冻结等成本预算与能力边界表。
3. 构造 EdNet 降采样切片与三档冷启动切片，发布构造脚本。
4. 跑 Best-LR / DKT ×4 数据集 × 5 种子，双成本口径、双划分口径。
5. 按冷启动切片分层计算 AUC，检验 H1 交叉。
6. 分析冷启动差异随作答数增长的收敛曲线（给 95% 自助置信带）。
7. 交叉校验：AUC 计算用 scikit-learn 与自实现两套互核。

7 · 新颖性边界

- 本课题不声称提出新 KT 模型；「LR 与 DKT 打平」由 Gervet et al. (2020)、Wilson et al. (2016) 发表，不得声称为本项目发现。
- 已有工作：Gervet 覆盖 9 数据集全序列 AUC；Baker 组（EDM 2021）研究过「学生开始新技能时」的切片但未做冷启动 × 模型类别 × 等成本的交叉。
- 本项目贡献（主结论）：冷启动切片这一新评价维度上的系统对比 + EdNet 新样本独立复检 + 等墙钟口径。
- 价值：冷启动质量直接决定自适应系统对新学生的最初体验；若 DKT 在冷启动不输（H1 后半否证），则「部署仍应选深度模型」获得证据——正反都成文。

8 · 决策门槛（go / no-go）

- **第 3 周末**：Gervet 仓库跑通且校验达标；依赖损坏 → 限期 2 周用 pyKT 库替代并以其论文数字为校验对象（协议差异写明）。
- **第 7 周末**：EdNet 下载与降采样管线完成；若下载受阻 → **具名降级**：以 ASSISTments12 + 17 作为双独立样本，主结论框架不变。
- **第 24 周**：若冷启动切片上两模型差异不显著（CI 含 0 且等价界内）→ 以等价性结论成文，重心移到「等成本口径 + 学生泄漏量化」（H2 与口径 2 的对照已足够支撑正文）。
- 预算裁剪顺序：先砍 SAKT 加项，再砍 ASSISTments17；冷启动三档与按学生划分不可砍。

K15 · 小参数量 CNN 从零训练 vs 预训练特征线性探针——CIFAR-10-C 腐蚀鲁棒性与校准的等墙钟对比

`优先级:中高` · `分族:轻量计算机视觉 / 鲁棒性` · `资源:纯 CPU / 零预算` · `技能:PyTorch 基础` · `类型:评价维度转换`

1 · 研究问题

> 在消费级纯 CPU 的固定墙钟预算（8 小时）内，「从零训练小 CNN」与「冻结预训练骨干 + 线性探针（linear probe）」两条路线，谁能在 CIFAR-10-C 腐蚀鲁棒性与概率校准上取得更好的前沿？精度、鲁棒性、校准三者在 0.1M – 11M 参数区间内如何联动？

2 · 研究背景与空白

Hendrycks & Dietterich（ICLR 2019）建立 CIFAR-10-C 基准：15 种腐蚀 × 5 强度，指标为平均腐蚀误差 mCE；Ovadia 等（NeurIPS 2019）表明分布偏移下模型校准（ECE）系统性恶化。这些结论全部建立在 GPU 训练的中大型模型（WRN-28-10 等）上。

空白在：文献从未回答「无 GPU 的固定墙钟预算下，训练与探针两条路线的鲁棒性/校准前沿谁更优」——这恰是教育与边缘场景的真实约束；且 0.1M – 1M 参数的极小模型区间在 CIFAR-10-C 上的精度 – 鲁棒性 – 校准联动无系统报告。评测本身零训练成本，结构性绕开算力墙：主要交付物是测量与联动规律，而非训练出的模型。

3 · 可检验假设

- H1：等 8 CPU 小时预算下，预训练 ResNet-18 特征 + 线性探针的 mCE 比最优从零训练小 CNN 低 ⩾10 个相对百分点，但其 ECE 在高强度腐蚀（强度 4 – 5）下反而更差（过自信迁移）。
- H2：在从零训练路线内部，0.1M – 1M 参数区间的相对 mCE（相对干净误差归一）与参数量近似无关（斜率 95% CI 含 0），即「小模型更脆弱」的直觉在该区间不成立。

4 · 量化验收标准

1. **方法学校验（硬门槛）**：下载公开预训练 ResNet-18（CIFAR-10 版）checkpoint，复现其干净测试精度（约 93 – 95%，以模型卡为准）偏差 ⩽1 个百分点，并复现文献报告的 CIFAR-10-C mCE 偏差 ⩽5%。不过关则后续全部结论无效。
2. 从零训练路线门槛：自训小 CNN 干净精度 ⩾85%，否则该路线数据点无效。
3. **统计口径预先写死**：报 mCE（按 Hendrycks 原始口径以 AlexNet 归一或明确改用绝对误差并声明）、逐腐蚀类型误差、ECE（15 箱）与可靠性图；类别平衡数据**不报 PR-AUC**，报 top-1 误差。
4. 每配置 5 个随机种子报均值 ± 标准差（探针路线对特征提取种子不敏感，报 3 次数据增广重复）。
5. 等墙钟口径：两条路线各限 8 CPU 小时（含调参）；另报「不限时收敛」口径，两种都报。
6. 全部训练日志、checkpoint 与评测脚本开源。

5 · 数据与工具

用途	来源 / 工具
数据	CIFAR-10（torchvision 内置）；CIFAR-10-C（Zenodo 官方下载，约 2.9 GB）——**仅作输入与对照，不计入本项目数据贡献**
预训练骨干	公开 CIFAR-10 版 ResNet-18 checkpoint（huggingface / GitHub 社区权重,写明来源与哈希）与 ImageNet 预训练 MobileNetV3——**不计入本项目模型贡献**
训练	PyTorch CPU。实测标尺（第 1 周校准）：ResNet-18 全量 CIFAR-10 CPU 单轮约 25–60 分钟 → 100 轮**超出范围**；0.1–0.5M 参数自设计 CNN + 20k 子集单轮约 3–8 分钟 → 40 轮过夜可行
特征提取	ResNet-18 对 5 万训练图一次性推理约 1–2 CPU 小时；线性探针（logistic 回归）秒级
评测	CIFAR-10-C 全量 95 万张推理：小 CNN 约 2–4 小时/模型；预算内每条路线最多评 6 个模型点

6 · 方法路径

1. 装环境，完成预训练模型的干净精度与 mCE 双校验（硬门槛）。
2. 实测本机单轮/单张推理墙钟，写出能力边界表（哪些配置超 8 小时预算）。
3. 设计 4 – 6 个参数量档位的小 CNN（0.1M/0.3M/1M/3M），在等预算内训练 × 5 种子。
4. 构建探针路线：两种骨干 × 线性/浅 MLP 探针，等预算内完成。
5. 全部模型过 CIFAR-10-C，计算 mCE、逐腐蚀误差与 ECE。
6. 拟合参数量 – 相对 mCE 关系（H2），自助法给斜率置信区间。
7. 交叉校验：用 numpy 独立实现 ECE 与 torchmetrics 结果互核（偏差 ⩽1%）。

7 · 新颖性边界

- 本课题不声称提出新的鲁棒化方法；CIFAR-10-C 基准与「偏移下校准恶化」结论分别由 Hendrycks & Dietterich (2019)、Ovadia et al. (2019) 发表，不得声称为本项目发现。
- 已有工作：鲁棒性 – 模型规模关系在 GPU 大模型区间有报告（越大越稳）；探针 vs 微调的迁移对比（Kumar et al. 2022）关注分布偏移精度，未覆盖 CPU 等墙钟口径与校准联动。
- 本项目贡献（主结论）：消费级 CPU 等墙钟预算下两条路线的鲁棒性 – 校准前沿测绘 + 极小参数区间的联动规律。
- 价值：给「没有 GPU 的部署者该选哪条路」一个有误差棒的定量答案；H1 的任一半被否证都构成有信息量的结论。

8 · 决策门槛（go / no-go）

- **第 2 周末**：完成墙钟实测。若单轮时间比估算高 >2 倍（老旧 CPU）→ **具名降级**：数据子集降到 10k、腐蚀评测降采样每类 2000 张（分层抽样），主结论框架不变、CI 拉宽。
- **第 6 周末**：从零训练路线若最优点仍 <85% 精度 → 该路线保留为「负结果数据点」，重心全转探针路线内部对比（骨干 × 探针容量 × 校准），H1 改为探针内部版本。
- **第 20 周**：CIFAR-10-C 全量评测墙钟超支 → 固定降采样口径并对全量做一次抽检校准（偏差 ⩽1%）。
- 预算裁剪顺序：先砍浅 MLP 探针，再砍 3M 参数档；5 种子降为 3 为最后手段。

K16 · SHAP 与 LIME 特征归因分歧的决定因素——模型类别、特征相关性与维度的受控扫描及可操作判据

`优先级:中高` · `分族:可解释性` · `资源:纯 CPU / 零预算` · `技能:Python + 统计` · `类型:机制补全`

1 · 研究问题

> Krishna 等（2022）记录了 SHAP 与 LIME 等归因方法的「分歧问题」（disagreement problem），但未回答：分歧度能否由数据与模型的可测性质（特征相关性结构、维度、模型非线性度）系统预测？能否给出使用者可在拿到数据时就计算的「分歧风险判据」？

2 · 研究背景与空白

事后归因方法（post-hoc attribution）中 SHAP 基于 Shapley 值、LIME 基于局部线性代理，是表格数据可解释性的两大默认选择。Krishna 等（arXiv:2202.01602）定义了 top-k 重合率、秩相关等分歧度量，在真实数据上证明分歧普遍且实践者无所适从；后续跨域比较（2024 – 2025）发现一致性「依数据集而异」，但均为观察性描述。

空白在：没有工作用**受控合成数据**系统扫描「什么因素以多大效应量放大分歧」并在真实数据上验证判据——这是从「现象记录」到「机制与预测」的一步。全部计算为 TreeSHAP（精确、秒级）与 LIME 重复运行，纯 CPU 数十小时内完成；交付物是测量与判据，结构性绕开算力墙。

3 · 可检验假设

- **H1**：特征两两相关系数均值 ρ 从 0 升到 0.8 时，SHAP – LIME 的 top-5 重合率单调下降且降幅 $\geqslant$30 个百分点（合成数据，控制其他因素）。
- **H2**：由（ρ、维度 d、模型测试精度、非线性度代理）四个可测量构成的回归模型，能在留出的真实数据集上预测分歧度，$R^2 \geqslant 0.4$；若 $R^2 < 0.2$，则结论转为「分歧主要由方法内在随机性驱动」——负结论同样有效。

4 · 量化验收标准

1. **方法学校验（硬门槛）**：其一，在可解析求解的线性模型 + 独立特征场景下，自建管线算出的 SHAP 值与解析 Shapley 值偏差 $\leqslant$1%；其二，重现 Krishna et al. 开源代码在其一个公开数据集上的分歧度量数字，偏差 $\leqslant$5%。不过关则后续全部结论无效。
2. **统计口径预先写死**：分歧度量三件套预登记——top-k 重合率（k=3,5）、Spearman 秩相关、符号一致率；LIME 每实例重复 10 次报均值 ± 标准差（其采样随机性必须与「方法间分歧」显式分离并分别量化）。
3. 合成数据因子设计：$\rho \in \{0, 0.2, 0.4, 0.6, 0.8\} \times d \in \{10, 30, 100\} \times 3$ 类模型（logistic / 随机森林 / 梯度提升），每格 10 个数据种子。
4. 真实数据验证 $\geqslant$8 个 OpenML 公开表格数据集，判据回归用留一数据集交叉验证。
5. 效应量全部给自助 95% 置信区间。
6. 全部代码与合成数据生成器开源，一键重跑。

5 · 数据与工具

用途	来源 / 工具
合成数据	自建生成器（多元高斯 + 可控非线性响应面）——生成器本身是项目交付物之一
真实数据	OpenML CC-18 中选 ⩾8 个表格数据集（openml.org API 下载）——**仅作输入，不计入本项目数据贡献**
归因实现	shap 库（TreeSHAP 精确路径，秒级）、lime 库；**KernelSHAP 高维慢——d=100 时需核实单实例耗时，超 30 秒则该格只用 TreeSHAP**
校验对照	Krishna et al. 2022 开源代码与线性模型解析 Shapley 值——**仅用于校验，不计入贡献**
算力量级	全项目估计 <40 CPU 小时（LIME 重复是大头），过夜分批

6 · 方法路径

1. 装环境，完成解析校验与 Krishna 复现双硬门槛。
2. 实现合成数据生成器，验证生成的相关结构与设定一致（Frobenius 偏差 $\leqslant 5\%$）。
3. 跑合成因子全网格，分离「方法间分歧」与「LIME 内在方差」两个成分。
4. 对每个因素拟合分歧度响应曲线（H1），给置信带。
5. 在真实数据集上计算同套度量与四个预测量，拟合并交叉验证判据（H2）。
6. 写出可操作判据卡（给定新数据集 5 分钟内可算的分歧风险评分）。
7. 交叉校验：TreeSHAP 与暴力枚举 Shapley（$d \leqslant 12$ 的子问题）互核。

7 · 新颖性边界

- 本课题不声称提出新归因方法；分歧现象本身由 Krishna et al. (2022) 发表，不得声称为本项目发现。
- 已有工作：Krishna 2022 给出度量与实践者访谈；跨域比较（JAIC 2025 等）报告了「数据依赖性」的观察；2603.15821 从假设类角度做了理论分析但无受控实验扫描。
- 本项目贡献（主结论）：受控合成扫描给出各因素对分歧的效应量排序 + 在真实数据上验证过的分歧风险判据。
- 价值：把「方法会分歧」升级为「什么时候会分歧、可提前多少预知」，对使用者可直接操作；H2 失败的负结论（分歧不可预测）也界定了该问题的界限。

8 · 决策门槛（go / no-go）

- **第 3 周末**：双硬门槛完成；解析校验不过 → 说明对库的理解有误，禁止进入扫描阶段，排查至通过为止（此项无降级）。
- **第 10 周末**：合成扫描中若 LIME 内在方差 > 方法间分歧的 2 倍 → **具名降级**：把课题重心改为「LIME 采样方差的决定因素与稳定化代价」，度量与因子设计全部沿用，主结论框架（可测因素 → 解释不可靠性）不变。
- **第 28 周**：真实数据判据 $R^2 < 0.2$ → 按 H2 预案以负结论成文，并给出功效分析证明有能力检出 $R^2 \geqslant 0.4$。

- 预算裁剪顺序：先砍 d=100 档，再砍真实数据集至 6 个；LIME 重复次数下限 5。

K17 · LLM 评改中文议论文的偏差探针——长度、辞藻与错别字的受控扰动因果实验

`优先级:中` · `分族:LLM 评测与探针 / 教育技术` · `资源:API 预算 ≤200 元` · `技能:Python + 实验设计` · `类型:评价维度转换（scoring 型偏差 + 中文场景）`

1 · 研究问题

> 大语言模型作为作文评分者（LLM-as-a-judge）时，对与内容质量无关的表面特征——篇幅、成语/辞藻密度、模板化开头、错别字——存在多大的因果性评分偏差？这些偏差在中文议论文场景下的效应量与方向，与已发表的英文对话评测偏差是否一致？

2 · 研究背景与空白

LLM 评审已被大量用于模型对比与作文批改。已发表工作：Wang 等（2023）证明成对比较中的位置偏差（position bias）；Zheng 等（NeurIPS 2023, MT-Bench）量化长度偏差与自我偏好；OffsetBias、CALM（2024）系统整理了 6 – 12 类偏差；arXiv:2506.22316（2025）明确指出**打分型（scoring-based）偏差研究仍然稀少**，且以上工作几乎全部基于英文对话/问答场景。

空白在：中文作文评分场景的 scoring 型偏差没有受控因果证据——「同一篇作文只改长度/辞藻，分数变多少」这一教师和学生最关心的量未被测过。实验为纯 API 评测 + 统计分析，零训练，结构性绕开算力墙。

3 · 可检验假设

- H1：内容保持不变时，扩写 30% 使 LLM 评分平均提升 ⩾0.3 个标准分（标准化到人工分数的 SD），压缩 30% 使其下降幅度更大（损失厌恶不对称）；成语密度加倍提升 ⩾0.2 SD。
- H2：注入每百字 1 处错别字造成的扣分，超过人类评分者已发表扣分惯例的 2 倍（若无中文文献值则以本项目招募的教师小样本为对照），即 LLM 对表面错误过度敏感。

4 · 量化验收标准

1. **方法学校验（硬门槛）**：先在英文公开基准 ASAP-AES（带人工分）上，用本项目提示词管线使所选 LLM 的评分与人工分的二次加权 Kappa（QWK）达到已发表 LLM 评分工作的水平（偏差 ⩽0.05 QWK）。不过关说明评分管线本身不可靠，后续偏差测量全部无效。
2. 因果设计预登记：每篇原文生成 5 类扰动版本（扩写 30% / 压缩 30% / 成语加密 / 模板化开头 / 每百字 1 错别字），扰动由脚本 + 人工核对完成，核对判据（内容命题不变）写成 checklist 并开源。
3. **统计口径预先写死**：配对设计，效应量用混合效应模型（作文为随机效应）估计，报点估计 + 95% CI；显著性检验用配对置换检验；**不报 accuracy**（评分是有序回归问题）。评审模型 ⩾2 个、每个条件 3 次独立调用取均值并报调用间 SD。
4. **可复现性硬要求**：固定模型版本号（如 deepseek-chat 与 glm-4 的具体版本串）、temperature=0、记录每次调用日期；提示词全文、原文与扰动文本、全部原始返回开源。**模型版本会变——版本与日期不记录则实验不可复现，视为不合格。**
5. 预算核算：100 篇 × 6 版本 × 2 模型 × 3 重复 = 3600 次调用，每次约 2k token，共约 7M token；按主流国产 API 单价估算 30 – 100 元，上限 200 元（含返工），⩽500 元约束内。

6. 样本量功效分析先行：按 SD=1 检出 0.2 SD 效应、power 0.8，需配对 n ≈ 100，故原文下限 100 篇。

5 · 数据与工具

用途	来源 / 工具
中文作文（带参考分）	公开中文作文评分语料**需核实**（候选：HSK 动态作文语料库（需注册）、公开高考满分/范文集配教师评分）；保底方案见 go/no-go
英文校验基准	ASAP-AES（Kaggle 公开，13k 篇带人工分）——**仅用于管线校验，不计入本项目数据贡献**
评审模型	2–3 个国产 LLM API（版本冻结）；扰动生成可用 API 但**须人工逐篇核对**，AI 不得代写分析
统计	statsmodels 混合效应模型 / R lme4（CPU 秒级）
人类对照	本校语文教师 2–3 名对 30 篇子样本盲评（同校教师指导合规）

6 · 方法路径

1. 搭评分管线，在 ASAP-AES 上完成 QWK 硬门槛校验。
2. 核实中文语料可得性（第 4 周截止），冻结 100 篇原文清单。
3. 脚本生成 5 类扰动版本，人工按 checklist 逐篇核对内容不变性。
4. 全因子调用 API（盲化、随机顺序），落盘全部原始返回。
5. 拟合混合效应模型估计各扰动因果效应，检验 H1/H2。
6. 教师小样本盲评同批扰动文本，对比人机偏差方向与幅度。
7. 交叉校验：换一套改写提示词重生成 20 篇扰动文本，验证效应量稳健（差 ⩽30%）。

7 · 新颖性边界

- 本课题**不**声称首次发现 LLM 评审偏差；位置/长度偏差由 Wang et al. (2023)、Zheng et al. (2023) 发表，偏差分类由 OffsetBias/CALM 完成，不得声称为本项目发现。
- 已有工作均为英文、以对话/问答为主、以成对比较为主；scoring 型偏差被 2506.22316 指为空白。历届丘奖 Hrbench（2024 优胜）是构建 LLM 历史推理能力基准，对象是模型能力而非评分者偏差，方向不同。
- 本项目贡献（主结论）：中文议论文场景下 4 – 5 类表面特征对 LLM 打分的**因果效应量表**及其与人类评分者的对比。
- 价值：直接回答「学生投其所好写长文堆成语能骗到多少分」；效应不显著（偏差小）同样是对 LLM 批改可用性的正面有效结论，须以 CI 证明检出能力。

8 · 决策门槛（go / no-go）

- **第 4 周末**：中文带分语料若不可得 → **具名降级 A**：主实验改在 ASAP-AES 英文语料上做（扰动类型改为长度/连接词密度/拼写错误），中文部分缩为教师评分的 60 篇自建小语料；偏差探针框架与统计设计完全不变。
- **第 6 周末**：QWK 校验不达标 → 迭代提示词至第 8 周；仍不过则**降级 B**：放弃「绝对分」改用同一作文扰动对的成对比较（偏差以胜率翻转度量），框架保留。
- **第 12 周末**：预算消耗若超 50%（返工过多）→ 砍评审模型至 1 个 + 重复 2 次，效应量 CI 拉宽但主结论保留。

- 已知困难即研究内容：扰动「只改表面不改内容」的边界本身难判——checklist + 教师抽检的失败率要如实报告，作为方法学结果之一。

K18 · 中文数学应用题的模板化扰动评测——数字替换、无关条件与叙述改写下大模型退化谱的测绘

`优先级:中` · `分族:LLM 评测与探针` · `资源:API ≤100 元 + 本地 CPU 推理` · `技能:Python + 数学题工程` · `类型:新样本（中文）+ 新切片`

1 · 研究问题

> Apple GSM-Symbolic（2024）在英文 GSM8K 上证明：仅替换数字或加入无关条件即可使 LLM 数学准确率大幅下降。该退化谱在**中文**小学/初中应用题上形状如何？本地可跑的 1.5B 小模型与 API 大模型的退化幅度差多少倍？

2 · 研究背景与空白

GSM-Symbolic（Mirzadeh et al. 2024）把 GSM8K 题目模板化后重新实例化，发现改数字、加无关子句可致最高 65% 的准确率下降，质疑 LLM 的「推理」是否为模式匹配；GSM-Plus（2024）细分了数字替换/位数扩展/干扰句等八类扰动。检索确认：这一族工作全部基于英文基准，**中文应用题的模板化扰动评测未见成熟发表**（已换三种检索式确认；「未找到」不等于「不存在」，第 4 周做最终文献补查）。中文有公开题库 Math23K（2.3 万题）与 Ape210K 可作模板源。评测为纯推理，API + 小模型 CPU 推理可完成，结构性绕开算力墙。

3 · 可检验假设

- H1：同位数数字替换造成的准确率下降 ≤3 个百分点，而「加入含数字的无关条件」造成 ≥10 个百分点下降（方向与 GSM-Plus 英文结论一致），即退化主要来自干扰而非数值本身。
- H2：本地 Qwen2.5-1.5B-Instruct（4-bit 量化）的平均退化幅度 ≥ API 旗舰模型的 2 倍，且其对叙述改写（同一数学结构、换情境）的敏感度最高。

4 · 量化验收标准

1. **方法学校验（硬门槛）**：自建评测管线（提示词 + 答案抽取器）先在 GSM8K 官方测试集 200 题上运行所选 API 模型，准确率与该模型公开报告值偏差 ≤5 个百分点（记录版本差异原因）。答案抽取器另在 100 题人工核对下错判率 ≤1%。不过关则后续全部结论无效。
2. 题库工程：从 Math23K 抽 150 题模板化（数字变量化 + 单位/量纲校验脚本），每题生成 8 变体（原题、同位数替换 ×2、位数扩展、无关条件 ×2、叙述改写 ×2）；每个模板实例经程序求解器验证答案正确后才入库。
3. **统计口径预先写死**：题级配对设计；退化量报准确率差 + McNemar 检验 + 配对自助 95% CI；temperature=0 贪心解码（确定性），另对 30 题子集做 temperature=0.7 × 5 采样敏感性附录；**不报未配对的整体 accuracy 对比**。
4. **可复现性硬要求**：API 模型版本号、调用日期、temperature、完整提示词开源；本地模型给 GGUF 文件哈希与 llama.cpp 版本号。
5. 预算：150 题 × 8 变体 × 3 API 模型 ≈ 3600 次 × 约 1.5k token ≈ 8M token，估 20 – 60 元，上限 100 元。
6. 本地推理墙钟：1.5B Q4 在 8 核 CPU 约 15 – 25 token/s，每题约 300 token 生成 ≈ 15 – 25 秒，3600 题次 ≈ 15 – 25 小时，过夜 3 – 4 晚（第 1 周实测校准）。

5 · 数据与工具

用途	来源 / 工具
中文题源	Math23K（公开下载，23k 中文应用题带答案）；Ape210K（GitHub，需核实当前可下载性）——**仅作模板源输入，题库模板与扰动集是本项目的数据贡献**
英文校验	GSM8K（HuggingFace 公开）——**仅用于管线校验与对照，不计入贡献**
API 模型	2–3 个国产 API（deepseek / GLM / Qwen 系列，版本冻结）
本地模型	Qwen2.5-0.5B / 1.5B-Instruct GGUF Q4_K_M + llama.cpp（纯 CPU；0.5B 约 40–60 token/s）
变体验证	SymPy 程序求解器对每个实例复算答案——**这是入库硬条件**

6 · 方法路径

1. 搭管线，完成 GSM8K 校验与答案抽取器错判率核验（硬门槛）。
2. 实测本地模型 token/s，冻结题量与变体数的墙钟预算表。
3. 模板化 150 题：变量化、约束（正整数解等）、SymPy 复算，全部脚本化。
4. 生成 8 变体 × 全模型评测，落盘原始输出。
5. 配对分析各扰动类型退化量，检验 H1；对比本地/API 模型检验 H2。
6. 错误类型标注（计算错 / 提取错 / 被干扰句带偏），判据写成可复现 codebook，双人标注一致性 $\kappa \geqslant 0.7$。
7. 交叉校验：30 题双语平行（人工翻译成英文）测同一模型，分离「语言效应」与「扰动效应」。

7 · 新颖性边界

- 本课题不声称发现「LLM 数学脆弱性」现象——GSM-Symbolic (2024)、GSM-Plus (2024) 已在英文上发表，不得声称为本项目发现；也不声称提出新推理方法。
- 历届丘奖 2024 金奖「LLM Mathematical Reasoning Grounded with Formal Verification」方向是用形式验证增强推理，与本题「扰动评测测绘」方向不同；Hrbench（2024 优胜）是历史学科基准构建，本题在扰动因果设计上更进一步。
- 本项目贡献（主结论）：首个（以第 4 周文献补查为准）中文应用题模板化扰动评测集（程序验证过的 150 模板 × 8 变体）+ 中文退化谱与英文已发表结论的定量对照 + 小模型/大模型退化倍率。
- 价值：中文教育场景直接消费该结论（AI 判题/答疑可靠性）；若中文退化谱与英文一致（H 不显著差异），普适性获独立支持——正反都成文。

8 · 决策门槛（go / no-go）

- **第 4 周末**：文献补查若发现中文模板化扰动基准已发表 → **具名转向**：保留全部管线，差异轴改为「小学 vs 初中难度分层 × 小模型量化档位」的新切片，主结论框架（退化谱测绘）不变。
- **第 8 周末**：模板化速度若 <10 题/周 → 题量降到 80 题 × 5 变体（功效分析重算，CI 拉宽约 1.4 倍），不改框架。

- **第 12 周末**：API 若涨价/停服导致预算超 100 元 → 砍至 1 个 API + 本地 0.5B/1.5B/3B 三档，H2 改为「模型规模 – 退化」曲线。
- 选择前提：适合能忍受「大量题目工程 + 脚本调试」的学生；数学结构校验本身是研究内容而非杂务。

K19 · 中国文化物象零样本识别差距的测绘——CLIP 与 Chinese-CLIP 在自建评测切片上的对比与提示语言消融

`优先级:中低` · `分族:多模态定量分析` · `资源:纯 CPU / 零预算（图片收集工作量大）` · `技能:Python + 数据集构建` · `类型:新评测切片构建`

1 · 研究问题

> 英文语料训练的 CLIP 在中国文化特定物象（节令食品、传统器物、戏曲行当、非遗工艺）上的零样本识别相对其通用类目基线退化多少？该差距中有多大比例可仅靠提示工程（英文加 "Chinese" 限定、改用中文提示 + Chinese-CLIP）收回？

2 · 研究背景与空白

对比视觉-语言模型（CLIP）零样本分类依赖训练语料的概念覆盖。已发表工作：Pouget 等（2024, "No Filter"）证明英文过滤的 CLIP 在非西方地理切片上系统性退化；Yin 等（2024）表明提示中加国家名可在亚非样本上提升最高 +2.6 top-1；Chinese-CLIP（Yang et al. 2022）提供 2 亿中文图文对训练的对照模型。现有地理多样性数据集（Dollar Street、GeoDE）以日常用品为主，类目按「物体功能」组织。

空白在：**文化特定物象**（同一功能、文化形制不同，如月饼 vs 西式糕点、二胡 vs 小提琴）的细粒度评测切片不存在，中国子类尤其粗。构建 40 – 60 类评测切片是学生可完成的数据贡献；全部推理 CPU 可行，结构性绕开算力墙。

3 · 可检验假设

- H1：CLIP ViT-B/32 在文化切片上的 top-1 准确率比其在语义难度相当的通用对照类目组上低 ⩾15 个百分点，而 Chinese-CLIP 的对应差距 ⩽5 个百分点。
- H2：英文提示加入 "Chinese/traditional Chinese" 限定词可收回 CLIP 差距的 ⩾1/3；剩余差距在错误分析中主要表现为「映射到西方功能等价物」（占错误 ⩾40%，按预登记 codebook 判定）。

4 · 量化验收标准

1. **方法学校验（硬门槛）**：自建零样本推理管线复现 CLIP ViT-B/32 在 CIFAR-100 上公开报告的 zero-shot 准确率，偏差 ⩽2 个百分点；同法复现 Chinese-CLIP 模型卡上一项基准数字偏差 ⩽2 个百分点。不过关则后续全部结论无效。
2. 评测切片规格：⩾40 类 × 每类 ⩾15 张，全部图片给出来源 URL 与许可（CC/公有领域），逐张记录；类目定义与「对照通用类目组」的配对判据（视觉粒度相当）预登记。
3. **统计口径预先写死**：报 top-1/top-5 与逐类召回；类间不平衡有限但仍**不以总体 accuracy 单指标下结论**，主口径为宏平均逐类召回 + 类级自助 95% CI（对「类」重抽样）；提示模板效应报配对差。
4. 错误分类 codebook 预登记，双人独立标注 200 个错例，一致性 $\kappa \geqslant 0.7$。
5. 提示消融 ⩾4 组：CLIP 原版 80 模板均值 / 加限定词 / 中文直译提示 / Chinese-CLIP 中文提示。
6. 评测集、脚本、逐张许可清单全部开源。

5 · 数据与工具

用途	来源 / 工具
评测切片图片	Wikimedia Commons、公有领域图库（按许可筛选）——**切片的类目设计、筛选与标注是本项目的数据贡献；原始图片版权归属原作者**
对照数据	CIFAR-100（校验用）、GeoDE 中国子集（对照）——**仅作校验与对照，不计入贡献**
模型	OpenAI CLIP ViT-B/32（open_clip 加载）、Chinese-CLIP ViT-B/16（官方仓库）——**预训练模型不计入本项目模型贡献**
算力量级	ViT-B/32 CPU 推理约 5–15 张/秒；1000 张 × 6 组提示 ≈ 每组 2–4 分钟，全项目推理 <10 CPU 小时（第 1 周实测校准）
图片查重	perceptual hash（imagehash 库）去重，阈值预登记

6 · 方法路径

1. 装环境，完成 CIFAR-100 与 Chinese-CLIP 双校验（硬门槛）。
2. 冻结类目表（40 – 60 类）与对照类目配对表，预登记判据。
3. 收集图片：每类 15 – 25 张，逐张记录来源与许可，phash 去重，第二人抽检 20% 标签。
4. 跑全部模型 × 提示组合，落盘 logits。
5. 计算宏平均召回与配对提示效应，检验 H1/H2 前半。
6. 按 codebook 做错误类型双人标注，检验 H2 后半。
7. 交叉校验：10 类子集用另一实现（HuggingFace transformers 的 CLIP）复算，logits 偏差应可忽略。

7 · 新颖性边界

- 本课题不声称发现「CLIP 有文化/地理偏差」这一现象——Pouget et al. (2024)、Yin et al. (2024) 已发表，不得声称为本项目发现；也不训练任何模型。
- 已有工作：GeoDE/Dollar Street 覆盖地理多样性但类目为功能性日用品；Chinese-CLIP 论文只在通用中文基准上评测，未做文化物象细粒度切片；历届丘奖 CelsiaNet（2024 铜奖）是多模态 VLM 框架构建，方向不同。
- 本项目贡献（主结论）：一个许可干净、逐张溯源的中国文化物象零样本评测切片 + CLIP/Chinese-CLIP 差距的量化测绘 + 提示语言可收回比例的因果分解。
- 价值：为多模态模型的文化覆盖提供可复用的测量工具；若 CLIP 差距很小（H1 否证），则「文化偏差担忧在该类目尺度不成立」同样是有效结论。

8 · 决策门槛（go / no-go）

- **第 6 周末**：图片收集速率核查。若合格图片 <8 类/月 → **具名降级**：类目降到 30 类、每类 12 张（类级 CI 拉宽），或改以 GeoDE 中国子集重标注细粒度标签为主、自采图片为补充；测绘框架不变。
- **第 10 周末**：核查许可合规抽检通过率 ⩾95%；不达 → 暂停扩量，先修筛选规则（此为合规硬项，无降级）。
- **第 24 周**：若 CLIP 与 Chinese-CLIP 在切片上差距 <5 个百分点且 CI 窄 → 按 H1 否证路径成文，重心移到提示消融与错误类型结构。

- 选择前提：适合愿意做细致数据工作的学生；图片许可与标注质量是本题的生命线，工作量占全程约 40%。

K20 · 0.5B–3B 开源模型 GGUF 量化退化的中英不对称性——困惑度与下游任务联动的全档位测绘

`优先级：探索（风险最高）` · `分族：LLM 评测 / 鲁棒性` · `资源：纯 CPU（墙钟重）/ 零 API 预算` · `技能：llama.cpp + 统计` · `类型：新参数区间 + 评价维度联动`

1 · 研究问题

> Marchisio 等（2024）在 8B – 103B 模型上证明量化对非拉丁文字语言伤害更大。在消费级 CPU 唯一可部署的 0.5B – 3B 小模型区间、GGUF Q2 – Q8 全档位上，中文相对英文的退化不对称性是否仍成立、是否更剧烈？困惑度（perplexity）退化能否作为下游任务退化的可靠代理？

2 · 研究背景与空白

训练后量化（post-training quantization）是无 GPU 部署 LLM 的唯一途径。Marchisio 等（EMNLP Findings 2024）发现：4-bit 下非拉丁文字语言退化约为拉丁文字的 2.7 倍（-1.9% vs -0.7%），自动指标还系统性低估真实伤害；2025 年机器翻译量化研究进一步表明低资源语言最多损失 10+ COMET 分。llama.cpp 社区维护的困惑度表只覆盖英文 wikitext。

空白在：（a）0.5B – 3B 区间未被上述工作覆盖——小模型权重冗余更少，退化规律可能不同（可否证的机制假设）；（b）Q2 – Q8 全档位细扫 × 中英双语 × 困惑度-下游联动这一组合没有公开测绘。全部为推理评测，无训练，绕开算力墙——但墙钟总量是本题最大风险。

3 · 可检验假设

- H1：同一量化档位下，中文任务（C-Eval 子集）的相对退化 ⩾ 英文任务（MMLU 子集）的 1.5 倍，且该不对称性随模型变小、比特数变低而放大。
- H2：困惑度退化与下游准确率退化在（模型 × 档位）网格上的 Spearman 秩相关 <0.7，即困惑度不是可靠代理——与社区默认实践相悖，两个方向都有文献张力，正反都成文。

4 · 量化验收标准

1. **方法学校验（硬门槛）**：其一，各模型 Q8/FP16 基线在 C-Eval 与 MMLU 子集上的准确率与模型卡/技术报告公开值偏差 ⩽3 个百分点；其二，wikitext-2 困惑度与 llama.cpp 社区公开表对应值偏差 ⩽5%。不过关则后续全部结论无效。
2. 测绘网格预登记：3 模型（Qwen2.5-0.5B/1.5B/3B-Instruct）× 6 档位（Q2_K/Q3_K_M/Q4_K_M/Q5_K_M/Q6_K/Q8_0）× 2 语言 × 2 指标层（困惑度、选择题准确率），共 36 个模型-档位点，缺一须声明。
3. **统计口径预先写死**：下游任务用选择题（单 token 比较，避免生成毛刺）；每个点对题目顺序做 3 次随机重排取均值 ± SD；退化量为相对基线的配对差 + 题级自助 95% CI；**不报单一汇总分**，逐学科切片保留。困惑度用同一 1M token 语料（中文用公开新闻语料切片,英文 wikitext-2），语料清单冻结。
4. **可复现性硬要求**：GGUF 文件哈希、llama.cpp 版本号（提交 commit）、全部运行命令与种子开源；量化文件用官方发布版，**不自行量化**（避免引入自变量），来源写明。

5. 墙钟预算表先行：每点评测（500 题 + 1M token 困惑度）估 2 – 4 CPU 小时，36 点 ≈ 72 – 144 小时，夜间分批 4 – 6 周完成（第 1 – 2 周实测校准并冻结预算表）。
6. 全部原始 logits/输出落盘开源。

5 · 数据与工具

用途	来源 / 工具
模型	Qwen2.5-0.5B/1.5B/3B-Instruct 官方 GGUF 全档位（HuggingFace 下载，共约 20–40 GB 磁盘）——**预训练模型与官方量化文件不计入本项目贡献**
推理	llama.cpp（版本冻结）；实测标尺：1.5B Q4 约 15–25 token/s、3B Q4 约 6–12 token/s、0.5B Q4 约 40–60 token/s（8 核笔记本，第 1 周校准）；选择题评测走单 token logits，比自由生成快一个量级
中文下游	C-Eval 验证集子集（HuggingFace 公开）——**仅作输入，不计入贡献**；抽 500 题分层覆盖学科
英文下游	MMLU 子集 500 题（HuggingFace 公开），与 C-Eval 学科尽量配对
困惑度语料	wikitext-2（英）+ 公开中文语料 1M token 切片（来源写明、冻结）

6 · 方法路径

1. 装 llama.cpp，实测三模型 token/s 与单点评测墙钟，冻结预算表与能力边界表（如 7B**超出范围**须明写）。
2. 完成 Q8/FP16 基线双校验（硬门槛）。
3. 搭选择题 logits 评测器,在 100 题人工核对下抽取错判率 ⩽1%。
4. 按网格逐点评测，夜间批处理，中途落盘断点续跑。
5. 计算逐点退化量与不对称比,检验 H1；做困惑度 – 下游秩相关，检验 H2。
6. 学科切片分析：不对称性集中在哪些学科（语言知识型 vs 推理型）。
7. 交叉校验：抽 3 个点用另一推理后端（如 candle 或 transformers+bitsandbytes CPU）复测方向一致性。

7 · 新颖性边界

- 本课题**不**声称发现「量化伤害多语言」现象——Marchisio et al. (EMNLP Findings 2024) 与 2508.20893 已发表，不得声称为本项目发现；不提出新量化方法，不自行量化模型。
- 已有工作：Marchisio 覆盖 8B/35B/103B 与多语言宽度；llama.cpp 社区表只有英文困惑度；小模型区间 × 全档位 × 中英配对 × 代理有效性这一组合未见发表（已换三种检索式确认；「未找到」不等于「不存在」，第 3 周文献补查为准）。
- 本项目贡献（主结论）：消费级区间的量化退化中英测绘图 + 困惑度代理有效性的定量判定。
- 价值：直接指导中文用户「几比特以下不该用」；若不对称性消失（H1 否证）或困惑度是好代理（H2 否证），同样是对社区实践有用的结论——但须以 CI 证明有能力分辨 1.5 倍的不对称。

8 · 决策门槛（go / no-go）

- **第 2 周末**：墙钟实测。若单点 >6 CPU 小时（机器较弱）→ **具名降级 A**：砍 3B 模型与 Q5/Q6 档，网格缩至 2 模型 × 4 档 = 16 点，主结论框架不变、分辨率变粗。

- **第 6 周末**：基线校验不达标（常见原因：提示模板与官方评测不一致）→ 排查至第 8 周；仍不过则**降级 B**：放弃与公开值对齐，改为「全部配置自洽对比」（同一管线内部配对差仍有效），并在限制一节声明绝对值不可外推。
- **第 16 周**：若各档位退化量的 CI 宽度 > 效应量本身（噪声淹没）→ 加大题量至 1000 或减档聚焦 Q2 – Q4 低比特区（效应最大处），保证至少低比特区结论有分辨力。
- 选择前提：仅适合机器可整夜运行、且能接受「效应可能很小」这一风险的学生；本题排 K20 因为墙钟总量最大、效应量最不确定。

附录 A · 倒排时间表（52 周，通用骨架）

> 终点是 **2027 年 9 月 15 日提交截止**（按 2026 赛季日程外推，须以当年官方公告为准）。第 0 周为 2026 年 9 月第一周。每个阶段都有可测的硬门槛，不是任务描述。

周次	阶段	硬门槛（可测，不达标即触发降级）
0–2	选题定稿	从本方案选定 1 条主路线 + 1 条备选路线；确认指导老师；核对当年官方参赛指南的学科研究范围
2–6	环境与方法学校验	**跑通该路线验收标准第 1 条的方法学校验**。这是全程最重要的门槛：不过关不得进入下一阶段
6–10	数据/器材到位	数据完成下载并通过完整性检查；或器材到齐且装置重复性达标（实验类 RSD ≤ 阈值）
10–14	第一次 go/no-go	按各路线写明的阈值判定。不达标立即切换到该路线的具名降级路径，**不要拖到第 20 周**
14–28	主体研究	完成主实验/主计算的全部参数点；中期产出可展示的核心图表 3 张
28–34	第二次 go/no-go	主结论方向已明确（含"差异不显著"这一有效结论）；误差棒已能证明有分辨该差异的能力
34–40	交叉校验	用第二种独立方法重算核心量，两者一致或差异有可解释来源
40–46	中文初稿	全文完成；图表定稿；代码与数据整理为可一键重跑的仓库
46–50	英文稿与查重	**完成英文研究报告与查重报告**（总决赛强制英文，且提交材料含查重报告）
50–52	提交与答辩准备	系统提交（9 月 15 日截止，逾期不可修改）；英文 PPT 与答辩问答演练

两个容易被忽略的时间陷阱：

其一，英文化不是最后一周的事。总决赛全程英文（研究报告、PPT、答辩），从第 0 周起就要用英文记录关键术语与方法描述，否则第 46 周会成为瓶颈。

其二，11 月 3 – 10 日的公示期会把研究报告全网公开。这意味着新颖性问题会暴露在公众视野，查重与文献边界必须在提交前就处理干净。

附录 B · 资源清单（全赛道通用底座）

用途	来源 / 工具	说明
文献检索	Google Scholar、arXiv、PubMed、Semantic Scholar、Connected Papers	**必须用英文检索**；中文检索会系统性漏掉近三年工作
全文获取	arXiv、bioRxiv、PMC、作者主页、通过学校图书馆或指导老师获取	不要使用非法下载渠道，公示期会暴露
代码与数据托管	GitHub（公开仓库）+ Zenodo（存档并获取 DOI）	可复现性是评审加分项，也是自证独立完成的证据
计算环境	Python 3.11+（conda/uv 管理）、Jupyter	环境须导出为 `environment.yml` 或 `requirements.txt`
统计与作图	numpy / scipy / statsmodels / matplotlib	图表须含误差棒；配色对色盲友好
实验记录	纸质或电子实验记录本，逐日、带日期、不得回填	评委可能追问原始记录；实验类还需按要求提交实验视频
写作	LaTeX（Overleaf）或 Word；参考文献用 Zotero 管理	英文稿建议请母语者或老师润色，但**内容必须学生本人撰写**
版本管理	Git，从第 0 周开始提交	连续的提交历史是"学生本人完成"最有力的旁证

附录 C · 红线清单

> 以下每一条，触碰即可能导致取消参赛/获奖资格，或在评审中被直接判负。

资格红线 1. 指导老师不得来自公司、培训机构或任何谋利机构。组委会有理由相信学生受培训机构指导完成研究报告，即保留取消资格的权利。 2. 不得跨校组队；同一学科每人只能报名一次；同一篇论文不得报两个学科。 3. 参赛信息须据实填写；报告提交截止后不得更改团队与指导老师信息。 4. 曾参加或拟参加其他竞赛、已投稿或发表的，**必须在报名时如实申报**。

学术诚信红线 5. 不得将他人（含指导老师、高校课题组）已完成的工作写成本人贡献。研究报告须明确区分"他人已发表成果""公开数据库/事件表"与"本项目贡献"。 6. 公开数据库、已发表事件表、预训练模型只能作为对照与输入，**不得计入学生的数据或方法贡献**。 7. 不得给出未标注为推测的物理/因果外推。讨论部分可以推测，但必须明示。 8. 提交材料含查重报告；公示期报告全网公开，抄袭与过度重合会被公开发现。 9. AI 使用政策以当年官方参赛规则为准（公开公告中未见明文，**须核实**）。无论政策如何，研究报告的科学内容必须由学生本人完成；若使用 AI 辅助，按当年规则要求致谢。

方法学红线（不违规，但会被评委问倒） 10. 没有方法学校验的结果不可信——所有路线的验收标准第 1 条都是这个门槛。 11. 不平衡分类报 accuracy、时间序列随机划分、幂律只做双对数最小二乘、模拟不做收敛性检查：这四项是评审最常见的追问点，方案里已逐条写死正确口径。 12. "差异不显著"是有效结论，但必须有误差棒证明有能力分辨该差异；没有误差棒的"无差异"等于没有结论。

附录 D · 本方案的使用方式

1. **先读赛道导语**，判断这个赛道的评委口味是否匹配学生的能力结构。跨赛道选题（例如数学好的学生去做经济金融建模的理论建模族）常常比在本赛道硬拼更有胜算。
2. **按可行性顺序从上往下读**，不要从最精彩的那条开始。编号越小越稳。
3. **选定一条主路线后，必须再选一条备选路线**，且两条最好落在不同的族——这样第一次 go/no-go 触发时有真正的退路。
4. **第 2 – 6 周的方法学校验是全案的生死线**。 这一步跑不通，说明学生的技能栈与该路线不匹配，此时换题的成本远低于第 20 周换题。
5. **本方案的赛制信息取自官方公告（yau-awards.com，2020 – 2026 全部公示文章），但规则每年会改**。 报名开放后必须重新核对当年的《参赛规则》与六个学科各自的《参赛指南》，尤其是研究范围、评审标准与 AI 使用政策。