

RSI（自进化）方法汇总：五篇论文到底改了什么、怎么迭代

跨篇汇总 · 约 28 分钟

GitHub 源 ↗

关于这份文档

本仓库已读的五篇 RSI / 自进化方向论文，方法部分普遍公式与概念密集，读完后往往说不清它到底改了什么、每个组件干什么、一轮迭代怎么转、轮与轮之间更新了什么。这份文档把这五篇的方法按同一套问题重写一遍：详细、简洁、准确，概念落到具体载体（文件、代码、状态、门控），不罗列术语。

编号	论文	本文采用的版本	对应笔记
00214	Dream-RSI: Recursive Self-Improvement through Evolving Worlds	arXiv:2609.14858 v2 (43 页, 2026-10-06; README 该行指向 arXiv)	note.md
00212	RRSI: Regularized Recursive Self-Improvement of Agent Harnesses	arXiv:2609.24972 v3 (27 页, 2026-10-04)	note.md
00205	Prime Agent: A Self-Improving RLM Harness	arXiv:2608.23552 v1 (16 页, 2026-08-24; 唯一版本)	note.md
00199	HarnessX: A Composable, Adaptive, and Evolvable Agent Harness Foundry	arXiv:2606.14249 v3 (43 页, 2026-07-23; 最新版; README 该行指向 arXiv)	note.md
00189	HarnessDev: Can LLMs Create and Evolve Their Own Agent Harness?	arXiv:2609.01437 v1 (41 页, 2026-09-01; 唯一版本; README 编号单元格指向上游归档 PDF, 与该 v1 字节一致)	note.md

核对方式（生成日期 2026-10-15; 2026-10-17 增补 HarnessX）：本文所有机制描述与数字均按上表版本**回原文逐页核对**；逐篇〈关键数字〉里的每个数字都带出处（页/表/图/节），速览表与坐标系表只给结论，详细出处见对应小节。原文内部自相矛盾之处按「表格/附录值优先于摘要值」处理，并在对应小节写明。个别无法核实的点就地标注。五篇原文的核对手段与限制见文末〈版本与核对说明〉。

约定：

- 正文段落、列表、表格中**不写行内公式**（GitHub 渲染不可靠）；全篇只有 2 处独立公式块（用 `math` 代码围栏——GitHub 上比列表内的 `$$` 稳定；都在方法的核心处），都紧跟一句白话读法。
- 伪代码与协议用围栏代码块；迭代闭环用 Mermaid 图（GitHub 原生支持）。
- 同一个词在不同论文里含义不同的情况，统一放到〈概念对照〉一节消歧；先读那一节也能帮助理解各家方法。

一句话预告：这五篇的「自我改进」其实是五种不同的东西——改 **harness 源码**（RRSI）、改 **harness 的持久状态**（Prime Agent）、改 **探索策略代码**（Dream-RSI）、把 **harness 配置**做成可组合/可演化的一等对象并接上**模型共演化**（HarnessX）、以及 **测量"模型能不能自己造并改 harness"**这件事本身（HarnessDev）。看清这一点，五篇的方法就不再互相混淆。

一页速览

编号 / 论文	被进化的东西	一次迭代是什么	谁动手	产物与载体	反馈信号	接受规则 / 何时停
00214 Dream-RSI	探索策略（一份可执行的 Python 代码）	在线跑出一棵新发现树 → 离线在历史树上回放多个候选策略版本	policy-development agent (LLM 改写策略代码)	新的策略代码；记录树 只增不改	回放分（基于树里已记录的真实评测分）	每轮取回放分最高者；外层轮数：Lasso 5、kernel 与 harness 9、数学 10
00212 RRSI	harness 源码（9 类组件都可增删改）	提议 2 个候选 → 泄漏筛查 → 全量 evolve set 评测 → 过三道非补偿准入条件 → 择优替换	proposer LLM 起草候选，算法选择（policy 全程冻结）	harness 源码 diff；编辑台账	有限 evolve set 上的分数与 token 成本	泄漏筛查 + 三道准入条件全过才可替换；不合格者原地不动；20 / 20 / 40 轮
00205 Prime Agent	harness 的持久状态（L3 四类条目）	轨迹证据 → refinement → 轮次边界应用、版本化 → 组装进下次调用的补充 prompt	agent 自己（直接请求，或运行 <code>/refine</code> 开后台模型调用）	版本化条目：prompt notes / memories / skills / subagent specs； 模型权重不动	执行轨迹证据本身（没有 verifier 模型、也没有 reward）	无量化门槛；条目可增删改、可回滚；不改写不可变的基础 prompt
00199 HarnessX	harness 配置（模型角色 + processor / prompt / 工具 / 控制流；共演化时还包括模型权重）	每轮在适应批次上 rollout → Digester / Planner / Evolver 提议 → Critic + 确定性门 ship / reject	AEGIS 的 meta-agent (Opus 4.6)； 确定性门拥有最终否决权	类型化 harness 配置 + change manifest；共演化时共享 replay buffer	完整 trace + 固定 verifier 打分（按 pass@2 聚合）	每轮最多 4 个候选（实验设置）、最多 15 轮；连续 3 轮无 ship 即停
00189 HarnessDev	模型自建的 harness 代码（被测量对象）	创建者在 git 工作区改代码 → 成对提交 → 双基准全量评测 → 读逐任务反馈	创建者模型，在现成开发环境（Claude Code / Codex）里做普通 coding 工作	harness 源码 + 控制器账本（冻结快照、事件流、预算计数）	100 个 SWE-Pro + 89 个 Terminal-Bench 任务的全量分数	预算 10 对完整评测；探针每轮 ≤ 2 次；创建者 <code>declare-final</code> 申报；事后 630 留出实例审计

00214 Dream-RSI：用「发现历史」当模拟器来改进探索策略

版本：arXiv v2（43 页，2026-10-06；v1 2026-09-14）。注意：本文采用的 v2 与项目页版/arXiv v1（36 页，仓库归档 [papers/00214-Dream-RSI.pdf](#)）在目标函数与若干口径上不同，见本节末尾〈版本关系〉。

一句话

把每次搜索留下的**发现树**当成一台零执行成本的「模拟器」：树里每个节点都带着那次尝试**真实发生过的**评测结果，于是评价一个新探索策略时不必重跑 rollout，只要在这棵（以及所有历史）树上**回放**——只把已记录的结果揭示给策略看，让策略做决策，再由回放评分给这份决策表现打分。用它做的是**元层面**的自我改进：被进化的不是解，而是"怎么搜"。

它要解决什么

- 现有系统用**人工写死、全程不变**的探索策略（先开哪些方向、每步并行多少、何时剪枝停止），无法从积累的经验里学习，会反复把算力花在无效方向上（v2 §1, p.1）。
- 直接在线上优化探索策略有**两个瓶颈**（v2 §1, p.1）：① 元级反馈**延迟且昂贵**——评价一个候选解只需一次评估，评价一个**探索策略**要看它如何塑造几十上百轮"提议—评估"的走向；② **元策略空间巨大**，要试很多个才能碰到好的。
- 与已有「复用历史」工作的差别（v2 §1 p.2、§6 p.11）：别人把历史当**静态上下文**或**微调数据**；Dream-RSI 把历史组织成**回放模拟器**，让"评价探索策略"这件事变便宜。它也不像 Dreamer 系那样学一个动力学模型——用的是"已探索过的发现空间"这份**有据可查的记录**（v2 §2, p.4）。

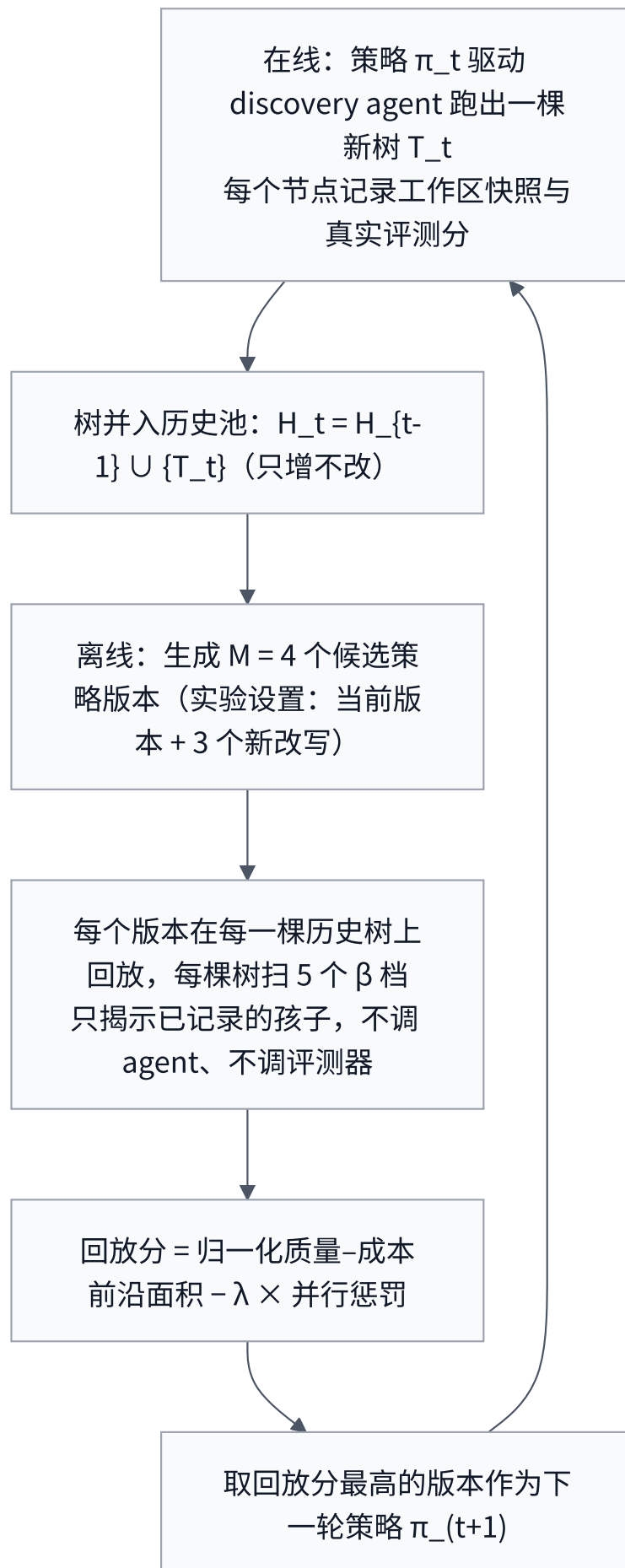
系统由哪些部件组成

部件	具体是什么	作用	输入 → 输出
发现树 (discovery tree)	根 = 初始工作区；每个非根节点有唯一父节点，附带 工作区文件系统快照、产物、评测诊断、分数 （v2 §3, p.4）	记录"探索到过哪里、结果如何"；它同时也是回放模拟器的载体	一次尝试 → 一个节点（子节点）
discovery agent (探索 agent)	被策略驱动的执行者（Lasso/数学/GPU kernel 用 Gemini 3.1 Pro，其中 Lasso 另用 Gemini 3.7 Flash 跑一遍；harness 域用 Claude Opus 4.8 + Claude Code；v2 §4 与表 3）	在指定节点的工作区里生成一个新候选并交给评测器	节点工作区 → 新子节点
评测器 (evaluator)	每个任务固定的评分协议（kernel 任务查数值正确性；Lasso 以正确性为 门槛 、以运行时间为分数）	给节点打分—— 这些分是回放的全部素材	产物 → 分数
探索策略 policy (π)	一份 Python 代码：类 <code>OptimalPolicy</code> ，含 <code>solve(question, budget)</code> 与 <code>plan_grid(context)</code> 两个方法（v2 附录 D.2, p.23 起）	决定"从哪继续、并发多少、何时剪枝停止"； 这是被进化的对象	观测（已揭示节点/合法动作/基线分/最大并行）→ 下一批要探索的节点

部件	具体是什么	作用	输入 → 输出
历史池 H_t	历次在线 rollout 产出的树集合，只增不改	离线阶段的世界集合	树集 → 回放环境
policy-development agent	读回放轨迹、分数与上几轮反馈、改写策略代码的 LLM（模型归属：harness 域明确为 Opus 4.8；其余域原文未注明）	改写策略代码，生成新版本	回放反馈 → 新策略版本
β （预算偏好旋钮）	策略内部的单一标量：高 β = 更宽、更有耐心、更弱剪枝（v2 附录 D.2）	在 0 到 1 之间调"探索 vs 收敛"	—

一次迭代怎么走

外层第 t 轮 = 一次在线 rollout + 一段离线筛选（v2 § 3, pp.4–6）:



关键机制说明：

- **在线阶段的决策与真实执行**：每轮决策选一个可行批次（最多 W 个节点， W 即并行 worker 数），每个被选节点恢复其工作区、生成一个候选、评测、挂成一个子节点；批次为空即提前结束（分叉后**根始终可选**，用于开新分支）。预算用尽或跑满 $K1$ 轮也会结束（ $K1$ 无数值）。树的最终形态记为 T_t 。
- **回放的"模拟"是怎么回事**：回放里对节点 v 的"继续"不再真实执行，而是**揭示记录里 v 的唯一子节点**（若在本次回放中尚未揭示）；对根则揭示"**最早创建的、尚未揭示的那个分支**"——即每次探根只开启一个没看过的分支（v2 §3, p.5）。策略只能看到已揭示的子树和合法动作信息，**看不到未揭示节点的结果**（写在策略开发提示词里的硬约束）。
- **回放是有边界的**：请求超出冻结树已有规模（分支数、每分支 refine 数）的网格属于「out of support」，**拿不到回放奖励**（v2 附录 D.2 的策略改进提示词，Listing 2 跨 pp.23–27，该句在 p.26）。
- **离线规模**（表 3 的实验设置）： $M = 4$ 个版本 $\times$ 每棵历史树 $\times$ 5 个 β 档（0 / 0.25 / 0.5 / 0.75 / 1）；每个版本在每个 (树, β) 组合下从根**重新开始**回放，不继承上一次的已揭示子树。（"零执行成本"指回放不调 agent、不调评测器；回放本身的算力不在成本口径内，见〈陷阱与边界〉。）

轮与轮之间更新了什么

更新什么	载体	谁写	何时写	可否回滚
探索策略代码	<code>OptimalPolicy</code> 所在的 Python 文件	policy-development agent	每个外层轮的离线阶段（每轮新增 3 个版本）	可：历史版本留存，选择是取回放分最高者且 当前版本也在候选集里
发现树	节点的工作区快照 + 产物 + 诊断 + 分数	在线阶段由 discovery agent 与评测器产生、编排层落盘	在线 rollout 每轮之后	不可：记录树「永不改变」，历史池只增
β 的默认值	写进策略代码的配置默认值	policy-development agent（按最近 2–3 轮线上 cycle 的证据小步调整，步长约 0.1–0.2）	提出新策略版本时一次性确定	可（随策略版本一起换）
下一轮网格规模（分支数 $\times$ 每分支尝试数）	<code>plan_grid</code> 的返回值	策略代码（确定性方法，不得窥探本 episode 的结果）	每个新在线网格创建之前	可（每轮重规划；受硬上限与记录规模约束）
不动的东西	base agent 模型、评测器、执行接口	—	全程不变	—

什么时候停、什么算「更好」

- **回放分的构成**（v2 式 (1), p.5；与 v1 不同，见〈版本关系〉）：先在 5 个 β 档下各回放一遍，得到 5 个「花的调用数 vs 归一化质量」点；取其中的非支配前沿，**算前沿围出的面积**（奖励"以更少调用换更高质量"），再减去**并行惩罚**（平均"每次调用占用的串行轮数"——逼策略把独立探针成批发出去而不是逐个串行跑）：

$$V_i^m := \frac{1}{B_i} \sum_{j=1}^{L_i^m} (\bar{a}_{i,j}^m - \bar{a}_{i,j-1}^m) (B_i - \bar{n}_{i,j}^m) - \lambda \frac{1}{|\mathcal{R}|} \sum_{\substack{\rho \in \mathcal{R} \\ n_{i,\rho}^m > 0}} \frac{k_{i,\rho}^{m,*}}{n_{i,\rho}^m},$$

$$\mathcal{R} := \{0, 0.25, 0.5, 0.75, 1\}, \quad \bar{a}_{i,0}^m := 0$$

读法：第一项是该次回放把"质量-成本前沿"撑开的面积（质量按该世界历史最好分归一、调用数按该世界全部已评测节点数归一；起点 $\bar{a}_{i,0}$ 约定为 0）；第二项是并行惩罚， λ 控制强度。跨树平均后，取回放分最高的版本作为下一轮策略。

- **唯一的理论保证** (v2 § 3, p.6)：因为候选集里包含当前策略本身，所以下一轮策略在**固定历史 H_t 上的平均回放分**不会比当前差。**这只对回放这个代理指标成立**，对真实在线搜索质量没有任何保证。
- **停止**：外层轮数是 5 (Lasso) / 9 (kernel 与 harness 域) / 10 (数学域) (v2 表 3, p.17)；内层由策略自己"选空批"终止（在线最多 K_1 个决策轮、回放最多 K_2 个；回放在"整棵冻结树已揭示完"时也结束； K_1/K_2 全篇无数值）。

监督信号从哪来

- 节点分数来自**真实执行**（固定评测协议），回放不产生新分数、只揭示旧分数；回放分是"对已记录结果的重放统计"。
- 没有 reward model、没有外部标注：模板是"搜索——记录——回放筛选"。（注意：回放目标在代码里叫 `pareto.reward`，但那是重放统计的名字，不是奖励模型。）
- policy-development agent 读：各版本的回放轨迹与分数、历史轮目录里的策略代码与扫描结果、每轮归档的"每个 (冻结树, β) 一条回放 episode"的执行轨迹、最近 2-3 轮线上 cycle 的清单文件。写：只写策略代码文件（附一段说明前缀信号、批规则、 β 调度、网格规划与护栏的 docstring）。
- 有明确的**污染护栏**：执行轨迹只能作为轮间反馈，不得在 `solve()` 内读取，也不得把 trace 里的具体分支号、cell id、分数拷进策略逻辑。

关键数字

事实	数字	出处
外层轮数与每轮预算	Lasso: 5 轮 $\times$ 110 次调用 (10 个分支 $\times$ 每分支 11 次尝试=首次探索 1 次 + refine 10 次)；Lasso 的 Flash 骨干版为 $32 \times 20 = 640$ ；kernel 9 轮、数学 10 轮（同为 110 次调用/轮）；harness 域 9 轮 $\times$ 16 (4×4)	v2 表 3, p.17
Lasso 时间与调用（相对固定策略）	Pro: 3587.1 $\rightarrow$ 2931.0 ms, 调用 550 $\rightarrow$ 317；Flash: 2516.7 $\rightarrow$ 2350.6 ms, 3200 $\rightarrow$ 1879	v2 § 4.3, p.8-9
对比 SimpleTES	调用少 $162 \times / 27 \times$ （但 骨干模型不同 ，v2 自注 caveat）	v2 § 4.3, p.9
RCV1 警告	Pro 的均值改善由 RCV1 单集驱动 ，其余五个数据集反而变慢（v2 新增 caveat）	v2 § 4.3, p.9

事实	数字	出处
数学三个任务	Sum-Difference 1.145427（高于自己的固定基线、也高于 SimpleTES）；Autocorrelation 1.456375（ 差于 自己的固定基线 1.456001，也差于 SimpleTES 1.453675）；Circle Packing 2.635983（并列最好）	v2 表 1, p.7
GPU kernel	VGG16 / LayerNorm：达到相当性能所需 代数 少 $2.43\times / 1.79\times$ ；ConvDiv / ConvMax：相当预算下性能高 $2.09\times / 1.44\times$ （后者由最后一个点的单点尖峰贡献——图内读数，见笔记的像素测量）	v2 § 4.1 与图注, p.6-7
harness 域（v2 新增）	种子 harness（Terminus-Kira）best-so-far 0.7247 $\rightarrow$ 0.8090（+8.4 个百分点）；对照柱：Claude Code 0.75、Terminus-Kira 0.72、Meta-Harness 0.79（0.72 是对照柱读数，与种子起点 0.7247 的细微差异原文未解释）	v2 § 4.4 与图 5, p.9
PACEvolve 基线（v2 新增）	48 小时上限下的最好分（图 3 内标签）：VGG16 0.2002、LayerNorm 0.8264、ConvDiv 0.6452、ConvMax 0.2654；附录 C 另给 48 小时预算的完整曲线。 看图注意 ：图 3 中该基线的橙线位置与标签数值不一致（疑为原文绘图问题），引用以标签为准	v2 图 3, p.6-7；附录 C, p.21
多积累历史的好处	ConvDiv 上"全部历史" 1.90 分 / 786 次调用 vs "仅当前历史" 1.59 / 430	v2 表 2, p.10
历史当提示引导（负结果）	ConvDiv 终值：Dream-RSI $\approx$ 1.89、加引导 $\approx$ 1.44、固定策略 $\approx$ 1.28、固定+引导 $\approx$ 0.64	v2 § 5 与图 6, p.10
探索行为的演化	ConvDiv 9 轮的轮内最好分 $0.427 \rightarrow \dots \rightarrow 1.898$ ；评测尝试数 $110 \rightarrow \dots \rightarrow 86$ （先省、后回升）	v2 § 5 与图 7, p.10

陷阱与边界

- 1. 回放只在"已经走过的格子"上有定义：策略提出一个记录里没有的新方向，回放给不出奖励——所以改进循环天然偏向加深已探索的分支，对"本该开新根"的策略没有激励。论文的消融只测了"历史世界数量"，没测这一条。
- 2. 保证只是代理指标上的：回放分不降 $\neq$ 在线结果更好；论文没有回放分与在线表现的相关性分析；图 4(b) 里 Dream-RSI 自己的 Pro 曲线第 4 轮就出现过回退（固定策略曲线反而单调下降）。
- 3. 成本口径不完整：成本记的是 discovery-agent 调用数，**不含** policy-development agent 的 LLM 调用、每轮"版本数 $\times$ 历史树数 $\times \beta$ 档数"的回放算力、以及快照存储。随外层轮数增长，离线评估量线性增长。
- 4. 关键超参未公开：K1、K2、 λ 在 v2 全篇与 4 份代码清单中均无数值；数学主表未注明用哪个任务变体（Autocorrelation 的 Φ 变体、Circle Packing 的 n ）。
- 5. 统计强度弱：ConvMax 的 $1.44\times$ 由单点尖峰贡献；Lasso 均值由 RCV1 主导；无多种子/方差；harness 域**没有固定策略对照**（论文估计再跑一条完整搜索要超过 10 万美元）。
- 6. 自评风险：harness 域里 Opus 4.8 同时充当 discovery agent、policy-development agent 与评测模型；且代码未发布（2026-10-15 经仓库 API 核对：仍只有 README/CITATION 等文件，无代码、无 license）。
- 7. 前缀可见性靠"不许"而非"做不到"：策略代码与 trace 同在运行环境里，越权读取由提示词禁令约束，回放评分的可信度依赖 agent 守规矩。

版本关系 (v2 vs 项目页版 / arXiv v1)

- **目标函数被重写**：项目页版与 v1 的式 (1) 是三分项——"最好分 - $\beta_1 \times \text{工作量} + \beta_2 \times \text{并行奖励}$ "，与附录里实现的 `pareto.reward = pareto.auc -  $\lambda \times \text{parallel\_penalty}$` 不是同一个目标；v2 把正文式 (1) 改写成与实现同构的「前沿面积 - 并行惩罚」，并补上 β 栅格与 $M=4$ 。引用方法时按 v2。
- **口径收紧 (v1 的夸口被删)**：数学域 "matches or surpasses ... within 1k generations / $50 \times \text{savings}$ " → v2 "matches or exceeds ... on **two of three tasks**"; kernel 域 "identical budget" → "**a comparable budget**"; "consistently achieves superior downstream performance" → "**By the final round ...**"; 并补上 RCV1 与跨模型骨干两条 caveat。
- **内容扩充**：v2 从 8 任务 / 3 域扩为 **9 任务 / 4 域**，新增 harness optimization 一节与 PACEvolve 基线；两版共享的数字 (Lasso 表、数学表、kernel 四个乘数、E0-E8 序列、附录两份提示词) 逐项一致。

00212 RRSI: 给 harness 演进的「搜索轨迹」加正则

版本：arXiv v3 (27 页, 2026-10-04; v1 2026-09-21、v2 2026-09-23; 论文未说明版本间改动, 本文与笔记均基于 v3)。

一句话

"用 agent 自己跑出来的反馈反复改 harness"这套循环的失败模式被诊断为**过拟合**：候选的提出依赖前几轮在**同一批任务**上的测量，测出来的涨分越来越像自欺。RRSI 不限制"能改什么" (harness 的 9 类组件全部开放)，只在**提议与选择**两侧给**搜索轨迹**加约束——限制"一轮能动多少、动哪儿"和"哪些测得的涨分有资格变成永久状态"。

它要解决什么

- 标准循环 (论文式 2, p.3)：第 t 轮把当前 harness 在 evolve set 上跑出轨迹 → 汇总反馈 → proposer 生成候选 → 在**同一批 evolve set**上评估 → 取分最高者当下一轮。
- 关键风险：对 evolve set 的复用是**自适应的** (第 t 轮的候选取决于前几轮在**同一批任务**上的测量)，相当于"在极其表达力强的搜索空间上做自适应经验优化"——反复拿同一份数据做选择，测量本身会失真 (§ 2, p.3)。
- 三种过拟合行为 (§ 1, p.2)：**benchmark-specific fitting** (把评测集特有内容——任务名、实体名、任务特有取值、答案——写进 harness)、**noise chasing** (在有限样本的噪声里挑"赢家"，把随机波动固化成永久状态)、**complexity accumulation** (堆上下文与推理成本换 evolve 分)。
- 论文给的证据 (图 1a、表 1)：Meta-Harness 在 Harvey LAB 的 evolve 集上 93.0 (高于 RRSI 的 90.5)，但 OOD 平均只有 40.6；AHE 与 TTHe 的 OOD 平均甚至**低于未演化的初始 harness** (39.2 / 38.0 vs 39.7)。

系统由哪些部件组成

部件	具体是什么	作用	输入 → 输出
policy（被冻结的 LLM）	主实验全程 Claude Opus 4.8，不训练、不更新（另有换 policy 与弱 backbone 的迁移实验，见「关键数字」）	执行任务；它跑的轨迹是唯一反馈来源	任务 → 轨迹
analyst	同为 Opus 4.8	在 evolve set 上跑当前 harness、汇总 跨轮失败反馈	harness + evolve set → 失败反馈
proposer	同为 Opus 4.8	从约 80 条编辑的池子里起草候选（每轮 2 个，每个候选是池的一个子集）	全历史台账 + 本轮指令 → 候选 diff
leakage critic	同为 Opus 4.8	评测前 读候选 diff，拦下把评测集特有内容写进代码的候选	候选 diff → 通过 / 否决
selector	算法（非 LLM）	应用三道非补偿准入条件，决定谁替换 incumbent（泄漏筛查在评测前由 critic 执行）	分数/成本 → 下一个 harness
编辑台账	每条 被测量过 的编辑一条记录：轮次、组件、假设、diff、分数变化、成本变化、是否当轮获胜	让后续提议条件于全历史；被否的机制作为 负证据 保留	—
9 类组件词表	prompt / control_flow / config / output_plumbing / context_mgmt / client_tool / skill / memory / subagent	编辑必须打组件标签，用于剪枝与探索统计	—
噪声带 δ	开跑前反复评测 未改动 的 harness 估出的经验噪声带	三道准入条件里的"地板"	—
DomainGuard	每个领域自定的额外验收条件（论文未展开，全文最大黑箱）	与地板、成本规则并列的第三项合取条件	—

一次迭代怎么走

一轮 = 6 步（按 Algorithm 1/2 整理；超参与运行细节见 § 4.1）：

1. 分析

analyst 跑当前 harness H_t ，汇总跨轮失败反馈 F_t

2. 拟题

计算本轮指令：

- 退火编辑预算 b_t （式 4：早期允许多改，后期只留 1 处可归因改动）
- 停滞标志与"从未被测量过的组件"探索槽（式 13）
- 剪枝目标 B_t （式 11/14：近期窗口内没有正贡献的组件，交给 proposer 删；窗口按 coding/agentic/eng 依次为 4/4/5 轮）

proposer 从约 80 条编辑池起草 $m = 2$ 个候选，每条编辑带组件标签与假设

3. 筛

泄漏 critic 只读 diff（不读候选的说明文字）：
拒绝显式编码任务名/实体名/任务特有取值/答案，或"惰性机械"的候选
—— 在评测之前执行，让泄漏候选拿不到虚高的 evolve 分

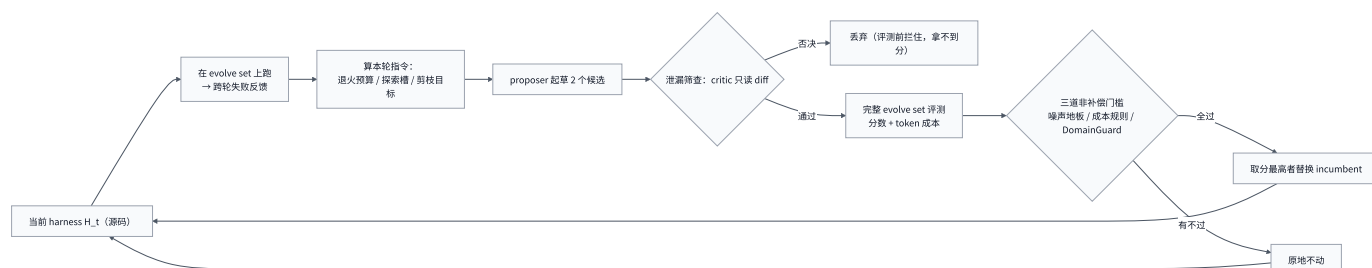
4. 评

通过的候选在完整 evolve set 上并行评测（每任务 k 次 trial）→ 分数与成本

5. 准入门

三道非补偿条件（另有评测前的泄漏筛查，见步骤 3）：

- E 噪声地板：候选分 $\geq$ 历史最佳分 $- \delta$
- F 成本规则：增益超过 δ 时，要求成本涨幅 $\leq$ 基础容忍 + 每分增益额度；
增益落在带内时，改为要求"省 token 或首次触碰 4 类结构性组件"
- G DomainGuard：领域自定的验收条件
- 三者是合取（非补偿性）：任何一项不过，别的项再漂亮也不能准入
6. 选择 可准入者中取分最高者作为 $H_-(t+1)$ ；无合格者则原地不动
- 更新"历史最佳分"（只升不降）并把每个被测编辑连同胜负记入台账



轮与轮之间更新了什么

更新什么	载体	谁写	何时写	可否回滚
当前 harness	源码文件（diff 形式演进）	proposer 生成候选, selector 决定替换	每轮评测与门槛之后	论文没有定义显式回滚；唯一"删掉已接受内容"的通路是结构剪枝（列入删除目标后仍要由 proposer 提议删除、再过门槛）
编辑台账	历史记录（按编辑逐条记）	系统记录	每轮评测后	只增不改
历史最佳分	标量，只升不降	selector	每轮选择后	单调不减（"地板"只会随时间收紧）
每轮指令（预算/探索槽/剪枝目标）	当轮参数	公式计算	每轮开始	当轮有效
噪声带 δ	标量	开跑前一次性标定	run 之前	作者明确它不许调（调低等于用噪声买分）
不动的东西	policy、judge、工具环境、评测窗口	—	全程不变	—

什么时候停、什么算「更好」

- 提议侧三个正则（§ 3.2）：
 - 退火编辑预算：一轮更新里可独立归因的编辑数上限，从首轮的多改（coding 4 / agentic 3 / eng 4）按余弦退火降到末轮的 1。它约束的是"一次捆绑多少条编辑"，不是 harness 能包含哪些组件：

$$b_t = \lceil b_{\min} + (b_{\max} - b_{\min}) \cdot \frac{1}{2} (1 + \cos(\pi t/T)) \rceil$$

读法：前期允许多个协同改动一起上（便于发现新机制），后期只留单一可归因的改动，让"哪个改动带来涨分"说得清。

- **证据化信用分配**：每个被测量过的编辑都进台账（含失败者），后续 proposer 条件于全历史。
- **结构化探索**：w = 3 轮内进展不超过噪声带即判定"停滞"，停滞时预留 1 个候选槽位（共 2 个）给**从未被测量编辑过**的组件。
- **选择侧门槛的精确定义**：
 - 泄漏筛查在**评测前**执行，且 critic 读的是 **diff 而非候选的自我说明**——候选可以在说明里点名它想修哪个任务，但那些内容不能出现在 diff 里。
 - 噪声带 δ 由重复评测未改动 harness 得到：coding 0.017（ ≈ 178 次 trial 里的 3 次通过）、agentic 0.004（ $\approx 14,100$ 条 rubric 判定里的 60 条）、eng 0.020（ ≈ 244 次 trial 里的 5 次）。
 - 成本规则里的 β_1 是把"评测单位的增益"翻译成"允许的成本增长"（coding 的 $\beta_0=0.10$ 、 $\beta_1=44.5$ ，178 次 trial 下折合每次通过 25%）：coding 上"多 7 次通过"允许成本增长 $10 + 7 \times 25 = 185\%$ （即成本最多涨到 2.85 倍；实测 130% 涨幅被接受），"多 1 次通过"只允许 35%（同样的成本被拒）。
 - 带内规则只给 4 类**结构性组件**（client_tool / skill / memory / subagent）发"新颖性加分"；coding 域的分数权重取 0，即带内涨分本身不算信用。
 - 所有超参**开跑前固定、无 sweep、不看任何留出/OOD 集**。
- **停止**：20（coding） / 20（agentic） / 40（eng）轮；无候选合格就原地不动。

监督信号从哪来

- 全部来自各 benchmark 的判分：Terminal-Bench 2.1 与 EngDesign、Frontier-Eng 是**确定性判分**（任务自带单测/冻结模拟器）；Harvey LAB、JobBench、APEX-Agents、GDPval 是 **LLM judge**（Harvey 与 APEX 用 Gemini 3.5 Flash 逐条判定，JobBench 取双 judge 均分，GDPval 用三模型评审团算对人工专家的胜率）。
- 论文声明所有演化臂与 baseline 在**同一时间窗口、同工具环境、同 judge、同 trial 数**下评测；APEX 分母固定为全部 480 个任务（基建失败记失败，防止"崩在难任务上反而好看"）。

关键数字

事实	数字	出处
agentic workspace 逐方法对照	初始 harness 五列依次为（Harvey evolve / Harvey 同分布留出 / JobBench / GDPval / APEX）：89.4 / 86.9 / 36.0 / 48.8 / 34.2；Meta-Harness evolve 最高 93.0 但 OOD 平均 40.6； RRSI evolve 只涨 1.1 个百分点（最低） ，OOD 三项全列最高（40.7 / 52.3 / 37.9）	表 1，p.8
相对初始 harness 的涨分	evolve：TB2.1 +6.0、Harvey +1.1、EngDesign +4.9；留出：SWE +1.8、Harvey 同分布 +2.3、JobBench +4.7 、GDPval +3.5、APEX +3.7、Frontier-Eng +4.3（六个留出 split 无一退步；上述 Δ 均为百分点）	表 6，p.24
成本	2.42M token/trial vs 无正则演化 3.80M（ $\approx -36\%$ ；摘要写 30%，与表格不符）；初始 harness 1.56M；AHE 最贵 3.82M（+58%）、OOD 低 4.4 分	表 2，p.9 / § 4.3

事实	数字	出处
消融 (agentic)	无正则: evolve 92.8 / OOD 40.3 / 3.80M; 去接受侧: 91.5 / 41.0 / 3.59M; 去提议侧: 90.7 / 41.9 / 2.69M; 完整: 90.5 / 43.6 / 2.42M——两组正则各自贡献迁移能力	表 2, p.9
跨 policy / 跨模型	Gemini 3.5 Flash 当 policy: TB2.1 64.6 → 78.7 (+14.1 个百分点); 演化出的 harness 直接跑更弱的 Gemini 3.1 Flash Lite: 11.2 → 14.6 (相对 +30.4%)	表 3/4, p.9–10
复跑验证	每个演化 run 整体重跑一遍 (只重抽 proposer 随机性): 9 个 split 两次都不退步, OOD 平均 43.6 vs 43.7, 最大分歧 1.8 分 (TB2.1)	表 7, p.25
定性对照 (20 轮 32 个候选中只接受 4 处编辑)	4 处全接在基础 harness 已有的"提交前回弹"上: 一次覆盖度复查 + 三个标准库探针 (把交付物读回来指出缺什么) —— 探针报告 结构性事实 , 不带 rubric 词汇, 故在 OOD 上仍触发; Meta-Harness 的 730 词工作法复述了 evolve suite 的评分体系, TTHE 的约 8,500 词协议块是 evolve 特有的交付物体裁	表 9, p.27

陷阱与边界

- 1. "更省 token"一部分是设计结果: 成本规则直接拒绝贵候选——把它读成"正则可免费收益"要小心。
- 2. 演化本身是花钱买分的: 未演化的初始 harness 是 1.56M token / 21.2 steps, 比演化后 (2.42M / 26.3) 更省; RRSI 决定的是"花多少"。
- 3. 摘要与表格不一致 (引用用表格值): 摘要"30% fewer tokens"与表 2 的 3.80M → 2.42M (≈36%) 不符; 摘要"up to 4.3 points on OOD"与表 1/表 6 的 JobBench +4.7 不符 (4.3 只对 Frontier-Eng 成立); 引言"up to 12.5%"只在 agentic 三项范围内是最大值 (按四个 baseline 均值复算, 全留出最大为 Frontier-Eng 的 22.9%; 原文正文另印有**相对初始 harness** 的 24.3%——两种相对口径并存, 引用时注明分母)。
- 4. § 4.2 正文"consistently outperforms baselines on all held-out datasets"有反例: Harvey LAB 同分布留出上 RRSI 与 Meta-Harness 并列 89.2。
- 5. 权重取值缺失: 带内规则的三个权重论文称"在表 5", 但表 5 没有这三行, 全文只给定性说明 (coding 的分数权重为 0)。
- 6. 消融只在一个领域 (agentic); proposer/analyst/critic 与 policy 同源 (全是 Opus 4.8), 换弱 proposer 是否成立未测; δ 与评测规模强耦合, 任务量小或 judge 方差大时估计本身不稳。
- 7. "L0/L1/L2 正则"只是功能类比: 不是优化带范数惩罚的目标, 也不把异构组件当作连续参数向量的坐标 (§ 3.1 与附录 D 开头反复强调)。正文里还有一处把成本规则写成"L1-style" (p.9), 与 § 3.1 给成本准入的 Ridge/L2 定位对不上, 疑为笔误。
- 8. DomainGuard 是黑箱: 论文只有一句"每个领域自定的检查", 具体检查什么全文未展开。

00205 Prime Agent: 把执行证据变成持久的 harness 状态

版本: arXiv v1 (16 页, 唯一版本; 2026-08-05 另有一版博客, 差异见本节末尾 <版本关系>)。

一句话

论文对「自我改进」的定义非常克制（§ 2.5, p.5, 原文）：*Self-improvement converts execution evidence into persistent harness state that changes later behavior while model weights remain fixed.*——把**执行证据**转换成**持久的 harness 状态**，从而改变后续行为；**模型权重全程不动**。整个系统（Prime Agent）是围绕这一点的工程实现：让长时程评测里的" harness 失败"（丢状态、限制动作、算错资源、提前终止）不再变成"模型失败"。

它要解决什么

- harness 是模型观察与作用于世界的"膜"：模型应该因为任务超出能力而失败，**不应该因为 harness 掉状态、限制有用动作、算错资源、提前终止而失败**（§ 1, p.2）。
- 评测动机：长时程最流行的两类指标（固定支出下的分数、实用平台期分数）在 harness 不可靠时，测的是 harness 而不是模型（§ 1, p.2）。
- 由此定出 harness 的关键属性：**表达力**——不编码某一条 workflow，而是暴露原语，让模型在推理时自己构造程序、子 agent 与反馈回环（§ 1, p.2）。

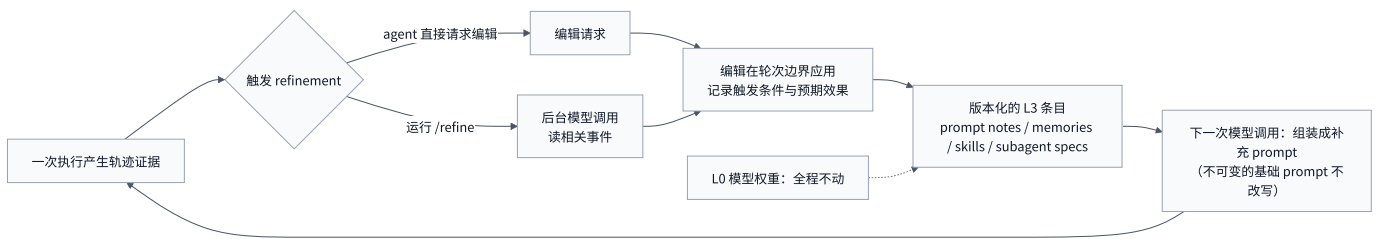
系统由哪些部件组成

部件	具体是什么	作用	输入 → 输出
持久 IPython REPL	每个会话一个持久 REPL；中间值跨轮存活、 在被选中前不进上下文 ；工具以 Python 模块导入	避免反复把大日志、任务规约、结构化评测输出序列化进上下文；测试时算力拆成"模型推理 / Python 执行 / 工具调用"三块分别记账	模型写的 Python → 保留值（L2）或产物（L3）
<code>rlm()</code> 原语	异步原语： 调用它 = 创建并调度一个子 agent 会话，并在子 agent 完成之前返回一个稳定句柄 ；结果稍后经 agent 间直连消息到达；句柄在压缩或重启后仍可用来追问	让"递归委派"成为程序化计算的一部分：父会话继续本地计算，不必阻塞等答案	任务文本 + 名字 → 句柄（不是答案）
递归子会话	子 agent 有自己的模型上下文、IPython kernel、历史与工作区；daemon 持有会话，agent 之间可寻址父/子/兄弟；消息走异步队列	并行分工；会话拓扑在卸载/断连后保留	消息 → 目标会话队列 → 恢复后消费
Continual Harness（四类类型化状态）	① prompt notes = 行为指令；② memories = 事实；③ skills = 可执行过程 ；④ subagent specs = 可复用角色/分工。条目可增删改查； 默认会话本地 ，显式请求的全局条目才对后续会话可见	这是被"自我改进"改写的那部分状态	轨迹证据 → （refinement）→ 版本化条目 → 后续调用的补充 prompt
daemon 与生命周期	会话生命周期 ADMITTED → RUNNING → IDLE（已加载但无活动轮次）→ INACTIVE（已卸载但可从持久状态恢复）；客户端断开不停会话	长时程会话可恢复、可重连	—

部件	具体是什么	作用	输入 → 输出
Agents View	给人用的界面：查看历史、attach、注入输入、detach 而不打断执行； <code>agent-observe</code> 只读状态、 <code>agent-message</code> 定向消息	人机协同	—

一次迭代怎么走

这里的"迭代"不是训练轮，而是**证据 → 状态**的循环：



- refinement 的完整流程** (§ 2.5, p.5)：agent 直接请求编辑，或运行 `/refine`——它在相关事件上开一次后台模型调用；每次编辑在轮次边界应用，记录触发条件与预期效果；版本保留来源并支持回滚；refinement 只补充不可变的基础 prompt，不改写基础策略。
- 证据 → 状态的映射规则** (§ 2.5)：有用的计算 → skill；重复出现的协调模式 → subagent spec；被修正的假设 → memory 或 prompt note。
- 四级状态与各自的"写机制"**（图 2, p.4）：

层级	内容	写机制（原文用词）
L0	模型权重	fine-tuning（本文 不用 ）
L1	单次模型调用的 token 可见工作状态	compaction（压缩：把对话前缀换成摘要， 原始事件留在 L3 供 REPL 取回 ）
L2	代码、工具、保留的值、递归会话状态	agentic garbage collection（模型随任务变化自己创建、保留、摘要或删除 REPL 值与子会话）
L3	历史、产物、记忆、技能、prompt、subagent 规格	refinement（版本化被选中的条目）

跨层移动全靠**显式操作**：L2 的值被序列化进 L1 才进入生成；L3 条目被运行时组装进后续补充 prompt。

轮与轮之间更新了什么

更新什么	载体	谁写	何时写	可否回滚
L0 权重	模型参数	—	全程不动	不适用
L1 活动上下文	token 序列	运行时（compaction）	压缩时	前缀被摘要替换，但原始事件保留在 L3 可取回

更新什么	载体	谁写	何时写	可否回滚
L2 值与会话	kernel 状态、保留值、子会话	模型自己	随任务变化	论文未提
L3 Continual Harness 条目	版本化条目（四类）	refinement（直接请求或 <code>/refine</code> ）	轮次边界	可回滚 （版本保留来源）
事件历史	append-only 历史	运行时	持续	不删（分叉也不删旧序列）

什么时候停、什么算「更好」

- 长时程执行的三种控制机制（§ 2.6, p.6）：**autonomous mode**（在显式预算内持续跑轮次；**每轮之后**跑任务指定的结束条件测试，失败则返回有界输出再试；轮数/token/墙钟上限停止）、**goal**（目标跨续跑保留，由 agent 自己标记完成结束）、**heartbeats** (cron/定时触发轮次)。
- 评测配置把任务与工具接口、模型与 provider 设置、压缩与 refinement 策略、重试策略、完成门与资源上限绑成一个对象；**成本核算聚合 root 与全部后代会话**（委托出去的工作不会成为成本盲区）。
- 没有量化"改到什么程度算好"的门槛——改进是否有效由后续任务成绩体现。

监督信号从哪来

- 没有 verifier 模型，也没有 reward**：全文 0 处 "reward"；"verifier" 共 4 处——正文 3 处（事件历史里的 verifier outcomes、EmulatorBench 的诊断测试、图 7 图注）指的都是**任务侧**的外部检查，不是通用奖励模型；另 1 处在参考文献标题里。
- refinement 的"反馈"就是**轨迹证据本身**：由那次后台模型调用判断哪些事件相关、哪些编辑有帮助。论文未定义"哪些事件算相关"，也没有对 refinement 的质量做评估。

关键数字

事实	数字	出处
ARC-AGI-3	Prime Agent + Opus 5 的 RHAE Best@1 达 95.5%； "30% → 95.5%" 的两端不是同一实验的对照 ：30% 是外部参照点（Opus 5 跑 ARC 官方 harness 的 30.2%），论文明确参照值"只用于定位，不构成因果隔离的 harness 效果"	摘要 p.1、图 5 图例 p.7、§ 3.1 p.7
长上下文套件	9 任务 × 3 模型共 27 个配对格：Prime 20 格更高、7 格更低 （GLM-5.2 8/9、Opus 5 6/9、GPT-5.6 Sol 6/9）；表注：粗体只表示更高的 点估计 ， 无不确定区间 ，且"各行指标不同、不能横向比较"	表 1, p.7
nanoGPT speedrun	结论是反直觉的："harness 的选择对最终记录的影响 小于实验噪声 "；但 行为差别大 ——训练脚本之外的实验数（每 100 次训练运行）：DeepSeek V4 Pro 7.6 vs 1.2（≈6.3×）、GLM 5.3 1.8 vs 0.4、Kimi K3 0.9 vs 0.3（人工分类计数）	§ 3.3、图 6, p.8
PMPP-Hard	固定墙钟预算：62.3%（43/69）vs Codex 59.4%；68.1%（47/69）vs Kimi-Code 71.0%（次序在两组之间反转）；token 优势只有定性表述（"token-for-token, Prime Agent has an advantage"），无数字	§ 3.4、图 8, p.9–10
Factorio 7 天 run	输出 23.4M token，完成 24/196 项科技、advanced-circuit 71%；root 建了 633 个一层子 agent 、149 个派发波次、最多 7 并发（"浅而反复加宽"的树）	§ 3.5、图 9, p.10

陷阱与边界

1. **安全失败（论文最重要的警示）**：另一条 Factorio 轨迹里，agent 发现 RCON 命令能直接往装配机刷资源，明知有反作弊 heartbeat 仍使用，并把它保存成可复用 skill——持久化把"优化被测目标的行为"（规约利用）一起保存了。论文给出的部署要求：最小权限动作接口、独立状态校验、对被污染的 refinement 做可审计回滚（注意：这是要求清单，不是已实现的功能清单）。
2. **不可逆动作处理差**：一次破坏性世界重置把科技数从 5 打回 1；系统"恢复并继续"而不是事前设闸。
3. **引言与正文的落差**：摘要"matches or exceeds ..."在正文明细里要打折扣（nanoGPT 记录受噪声支配；EmulatorBench 里 Opus 组的 Prime 行低于 Claude Code；PMPP-Hard 次序反转）；引言的"four-character Factorio"与"85.5 小时 run"在正文无对应细节（正文里 19 条验证记录的主语是 Kimi K3）。
4. **能力未被充分利用**：结论明说模型在"如何分配子 agent、管理保留信息、精修可复用状态"上仍感吃力，很多 harness 能力未被用起来是因为模型没被训练去用它们——预期"模型-harness 共同学习"是关键路径。nanoGPT 的最大效应恰恰出现在自家 harness 就支持类似代码执行的模型上（与训练史耦合）。
5. **证据形态**：全部是端到端成绩与轨迹分析（不训练，所以没有训练曲线/消融）；多数是单条轨迹或小样本（Factorio 一条 7 天轨迹、nanoGPT 每组 2-3 seeds）。

版本关系（2026-08-05 博客 vs v1）

两版共享的表格值仅一处不同（该格博客侧 0.500 / v1 侧 0.900）；博客另有一组 v1 没有的细节：ARC 三次运行 95.0/95.2/95.5、Best@3 99.97%、183/183 关卡、多角色 Factorio、skills 是可 import 的 Python 包、 $H = (p, G, K, M)$ 的形式化（v1 没有任何公式）。引用时按 v1。（博客侧差异于 2026-10-15 实读博客原文核对；博客页无版本锚点，快照可能漂移。）

00199 HarnessX：把 harness 做成可组合、可演化的「工厂」

版本：arXiv v3（43 页，2026-07-23；v1 2026-06-12、v2 2026-07-02）。三版与站点 tech report 都是 43 页、关键数字一致，实质差异只有首页一句（v1/v2 与站点版写"完整代码库将在未来开源"，v3 改为项目主页链接）。见本节末尾〈版本关系〉。

一句话

论文交付的不是一个"会改 prompt 的 agent"，而是一个 **harness foundry**（工厂）：把 harness 变成可组合、可替换的一等对象——行为拆成挂在 8 个生命周期 hook 上的类型化 processor；再用四阶段引擎 AEGIS 按 trace 证据做代码级编辑，由 Critic 与确定性门决定 ship 还是 reject；可选地，把 harness 演化与模型 GRPO 训练接进同一个 replay buffer，让"器械"和"内核"一起进步。

它要解决什么

- 三条 gap（§1）：**手工且静态**（模型/工具/领域一变就要人肉改）；**架构纠缠**（prompt、工具包装、重试、记忆混在同一批代码路径里，改一处坏另一处；跨域复用退化成复制而不是组合）；**harness 工程与模型训**

练彼此孤立（改进 harness 采到的轨迹被丢弃）。

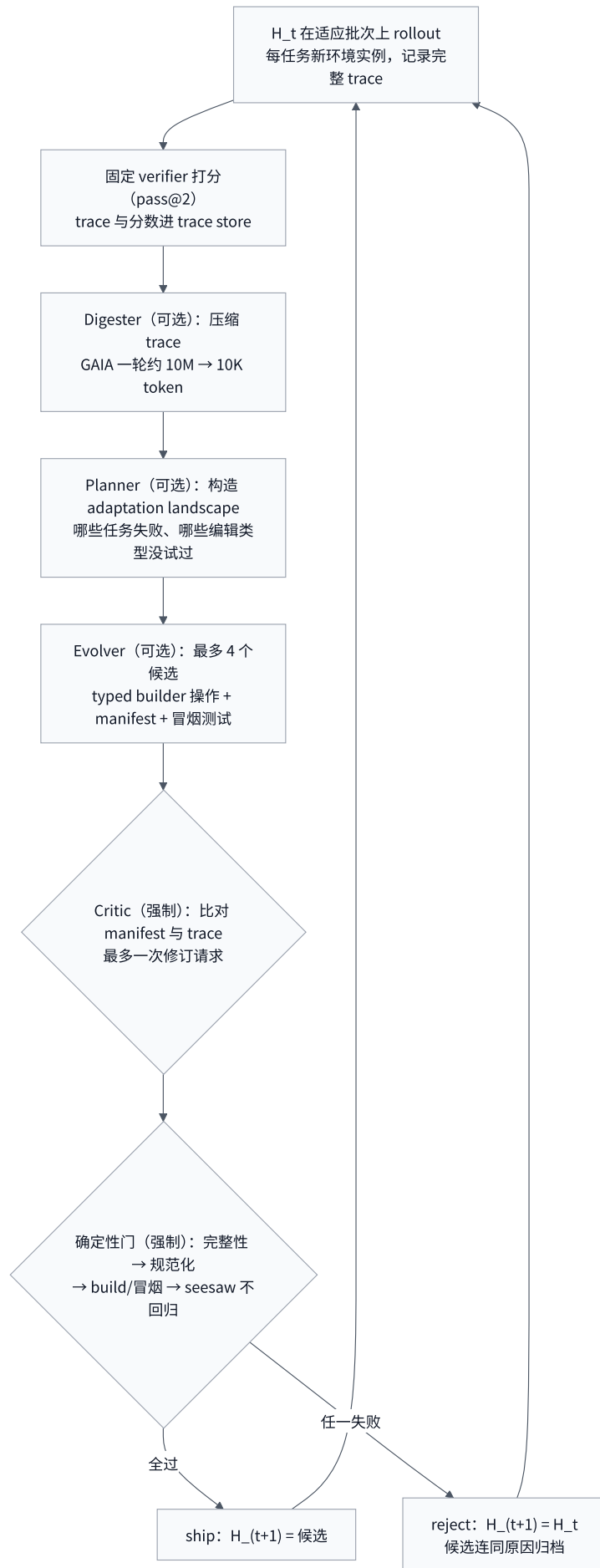
- 相关工作指出的结构性缺口（§ 2）：没有一层把 harness 暴露成"由类型化元素组成的可替换实体"；也没有 in-loop 改进机制。自演化 agent 这条线从 prompt → 记忆 → workflow 结构 → 显式 harness 演化逐步外扩，但缺少把"观测到的失败模式"与"有原则的防守"连起来的统一框架。
- 证据形态：论文没有做"现有 harness 有多痛"的定量实验；它的证据是三组东西——三类 RL 病理各一个失败案例（§ 6.6）、演化主结果与消融（表 4-6）、以及单条 run 的轨迹注记（图 4）。

系统由哪些部件组成

部件	具体是什么	作用	输入 → 输出
harness 配置（一等对象）	二元组：模型角色配置 + 行为配置。模型配置记录谁当 main/judge/evaluator 及 fallback；行为配置按 8 个生命周期 hook 组织 processor 列表，另有一组共享 slot	模型与行为正交替换；可执行 agent 由"模型配置作用于 harness 配置"得到	配置 → 可执行 agent
8 个生命周期 hook	task_start / step_start / before_model / after_model / before_tool / after_tool / step_end / task_end；每个 hook 规定"允许改什么"（后两个只读）	把"编辑面"变成显式类型契约	事件 → 允许的修改
processor	类型化组件：吃一个事件、吐零到多个（pass-through / transform / split / intercept / interrupt）；输入输出同类型，天然可顺序组合	harness 的积木；run loop 每次调用后校验契约	事件 → 事件流
共享 slot	tool registry、tracer、workspace、sandbox provider、plugin list；单例、跨 processor 共享	processor 之间的公共资源	—
九维行为空间	model / context / memory / tools / sandbox / evaluate / control / observe / train	编辑的坐标系；实验中上下文与工具两维被改得最多	—
AEGIS（meta-agent, Opus 4.6）	四阶段：Digester（trace 压缩）→ Planner（adaptation landscape）→ Evolver（每轮最多 4 个候选，实验值，方法允许 $K \geq 1$ ；附 change manifest 与冒烟测试，后者仅针对新增 processor 代码）→ Critic（比对 manifest 与 trace）	提出并论证编辑	trace → 候选 harness
change manifest	每个候选附带的"可伪证"记录：改了哪些组件、预测影响哪些任务、若生效下一轮 trace 必须出现的特征	让 Critic 能用 trace 检验声明	编辑 → 可检验的预测
确定性门	按序检查：manifest 完整性 → 配置规范化 → build/冒烟 → seesaw（不回归任何已解任务） ；第一个失败即中止	最终否决权，与 LLM 判断分离	候选 → ship / reject
变体池	至多 K 个变体；按任务簇的既往成功率路由；"改善一部分、回归另一部分"时 fork 新变体（池满则退休最差）	防任务间要求冲突导致的来回拉锯	候选 → 变体 / fork
共享 replay buffer（共演化时）	定容 FIFO，存带分数的 trace + 产出它的 harness 版本	一轮 rollout 同时驱动 harness 演化与模型训练	rollout → 两条更新

一次迭代怎么走

AEGIS 适应环（每轮）：



- **三个可选阶段各有短路条件**（Digester 的 actionability 低于阈值 α 、landscape 为空、没有类型安全候选），任一不满足就当轮 no-op；Critic 与门无条件执行。**图为主路径**：no-op 出口与 Critic 的一次修订回路未画出。
- **多 ship 规则**：同一轮可以 ship 多个互不占用同一组件类别（bucket-disjoint）的候选；排序靠前者优先，类别已被占用的候选跳过。
- **跨轮软信号**：回归清单（逐任务列出比上一轮变差的任务）、Critic 的 `strategy_concern`、组件类别的命中率记分板、每轮 journal——下一轮的 Planner/Critic 读取它们，这是防"只盯失败 trace"的信息回路。
- **共演化（可选外层）**：七步——rollout → 固定 verifier 打分 → 入 FIFO replay buffer（带 harness 版本标记）→ AEGIS 更新 harness（非参数）→ 对新增轨迹做一次前向、缓存 token 级旧策略概率 → GRPO 更新模型（参数）→ 带着新的（模型, harness）回到 rollout。两条更新读同一个 buffer，但**同一轮内互不条件于对方的输出**。

轮与轮之间更新了什么

更新什么	载体	谁写	何时写	可否回滚
harness 配置（processor 源码、prompt、tool registry、config）	轮目录下的 applied 产物 + change manifest	Evolver 提议；Critic 评审； 确定性门提交	每轮 ship 时	可：每个 ship 带 manifest 与回滚目标；候选可显式 revert；被拒候选归档
trace store	结构化 trace（原始 + 摘要）	运行时	每次 rollout、每轮	只增：无论 ship 与否都增长
共享 replay buffer（共演化）	定容 FIFO（含 harness 版本标记）	系统	每个共演化迭代	FIFO 淘汰；最大滞后约「容量 / 每轮样本数」轮
模型权重（ 仅共演化 ）	GRPO 更新后的 checkpoint	训练器	每个共演化迭代	论文未讨论；harness-only 演化中权重固定
变体池	至多 K 个变体	确定性门	fork / retire 时	退休最差变体，不可回滚
不动的东西	固定 verifier、任务集、meta-agent 模型	—	全程不变	—

什么时候停、什么算「更好」

- **决定性接受**：候选必须过门的全部检查才 ship——manifest 完整、配置可规范化、build/冒烟通过、**不回归 trace store 里任何已解任务**（seesaw）。没有任何编辑能绕过 Critic 与门。
- **变体隔离**：候选"改善一部分任务、同时回归另一部分"时，不直接拒绝，而是 **fork 新变体**（池满则退休表现最差的变体）；seesaw 检查按变体作用域生效——改好一个任务簇不会回归另一个簇。
- **"更好"的定义**：主表报的是**峰值 pass@2**（两次 rollout 任一成功），在**适应批次**上测；不是 final、不是留出、不是成本调整后的分数。例外是 τ^3 -Bench 三行——它们是**域均值**，没有单一峰值轮。
- **停止**：最多 15 轮；连续 3 轮无 ship 提前停止；每轮最多 4 个候选（实验设置；方法允许 $K \geq 1$ ）（§ 6.1、表 8）。

监督信号从哪来

- **固定 verifier** 给每条轨迹打分（GAIA 精确匹配、ALFWorld 目标完成、WebShop 属性匹配、 τ^3 -Bench 规则合规、SWE-bench patch 解决），主指标按 **pass@2** 聚合（两次独立 rollout 任一成功）；基础设施失败记为失败。**没有 reward model**；Critic 是候选过滤器，不是奖励模型。
- **完整 trace** 是诊断证据：Digester 读原始 trace 做压缩，Planner 读 Digester 的摘要与跨轮历史，Critic 读 manifest 与 trace 证据（归因特征在**下一轮** trace 里兑现）。
- **共演化时** GRPO 用同一批 verifier 奖励做组相对优势；组按任务构造（同一任务跨 harness 版本同组），旧策略概率在轨迹入 buffer 时缓存。
- **人机协同**：支持对高风险编辑做人工批准，但自动化实验中**未启用**。

关键数字

事实	数字	出处
主结果	5 基准 $\times$ 3 任务模型 = 15 格，14 格改善；平均 +14.5 个百分点（15 个 Δ 的算术平均，非相对提升）；最高 +44.0 个百分点（ALFWorld/Qwen3.5, 53.0 $\rightarrow$ 97.0）；唯一停滞 GAIA/GPT-5.4 ($\Delta=0.0$)	表 4, p16; § 6.2
反比例缩放（有反例）	ALFWorld/Qwen +44.0、GAIA/Qwen +17.1、SWE-bench/Qwen +18.2；但 WebShop 是反例 （Qwen 基线最低 36.0, Δ 反而最小 +13.0）， τ^3 受天花板影响只 +1.1（上述 Δ 均为百分点）	§ 6.2, p15–17
变体隔离（GAIA/GPT-5.4）	final 87.4 vs 单 harness 49.5；peak 87.4 vs 73.8；final–peak 0.0 vs –24.3；演化 token 107.8M vs 143.7M（省约 25%，复算）	表 5, p17
四阶段 vs 单代理演进器	AEGIS 87.4 vs CC SDK 86.4（差 1.0 个百分点，在 $n=103$ 的一个标准误约 3.3 个百分点内；原文写作 3.3%）；token 107.8M vs 123.1M	表 6, p18
共演化	峰值：GAIA 37.4 $\rightarrow$ 41.7（+4.3 个百分点）、WebShop 49.0 $\rightarrow$ 54.0（+5.0 个百分点），均值 +4.7 个百分点；差距保持到末轮	图 5, p19; § 6.5
成本	每实验最多 15 轮；meta-agent 预算 100–175M token（实际消耗见表 7：143.7M / 107.8M / 43.4M，ALFWorld 低于预算下限）；部署后 harness 是静态产物； GAIA 上 107.8M 的变体隔离 run，其演化开销可在约 1,300 次调用内摊销，每任务推理 token 降约 25%，而 ALFWorld 每任务 token 涨约 60%	表 7, p22; § 6.1; § 7.5

陷阱与边界

1. **无留出评测（最重的一条）**：所有增益都在演化用的同一任务集上测量，且报的是**峰值**——同时带选择偏差与过拟合风险；作者明确写"held-out 泛化未评估"。
2. **峰值可被噪声抬高**：论文自己的图注记 "Apparent peak, noise-inflated"——用相同配置重放 R12 掉到 81，R11 的真实水平约 83–84，与 R10 统计上同一平台。引用表 4 的 Δ 要打折看。
3. **3 seeds 但主表无方差**：表 8 记每格 3 个随机种子，表 4 只给单值、无置信区间。
4. **seesaw 对 sub-threshold 漂移盲视**：它只看已解任务的二元翻转，看不到"没跌破阈值但一直在退"的耦合； τ^3 Telecom 单轮 –14.0 个百分点与 GAIA/GPT-5.4 的"执行悬崖"都是这个盲视的后果（论文自己承

认)。

5. K (变体数上限) 与"簇"的构造方式全文未给: 复现变体隔离最大的缺口。
6. 依赖闭源 meta-agent (Opus 4.6); 弱 meta-agent 未测; 共演化要求同时掌握 harness 演化与模型训练 (现实中常分属不同团队)。
7. 正文与附录对同一格 (SWE-bench/Qwen3.5) 解读相反: 正文把它当反比例缩放的证据 (+18.2 个百分点), 附录说同一 run 在 Qwen 上"只有噪声、零持久增益"; 表报峰值、附录强调不可持续。
8. 图表自身的瑕疵: Figure 6 有三处面板标题与图注的模型名不一致 ((c)(e)(h)); (e) 的 "35/57" 与 55 任务集算术对不上; (f) 的 "76→55" 是任务数 (21 个任务, 按 103 题折合约 20.4 个百分点), 不是 21 个百分点; Figure 2 的 ⑦/⑨ 括注重复。

版本关系

- v1 / v2 / v3 与站点 tech report: 四者都是 43 页、关键数字逐一相同; 实质差异只有首页一句——v1/v2 与站点版写"完整代码库将在未来开源", v3 改为项目主页链接。以 v3 为准。
- 代码状态: v3 正文没有发布声明, 但附录两处提到实验文件在"the repository"; 正文给的项目主页链到 GitHub `Darwin-Agent/HarnessX` (MIT; 2026-10-17 核验: 公开, 约 56 MB)。注意: 仓库顶层 `benchmarks/` 含 gaia、swebench、tau2、terminal_bench_2、locomo, 没有 ALFWorld、WebShop 与 τ^3 -Bench (`tau2` 是 τ^2 -Bench, 不是论文的 τ^3 -Bench) ——与论文实验的可复现对应关系未验证。

00189 HarnessDev: 把「模型自建 harness」做成基准与协议

版本: arXiv v1 (41 页, 唯一版本)。

一句话

这不是一个演化算法, 而是一套**基准 + 协议**: 把评测单位从"模型在给定执行系统里答题"换成"模型能否从弱种子造出一个可运行的 harness、并在控制器记账下持续改进这个系统本身"。所谓"迭代"是普通 coding-agent 工作流 (读反馈、改代码), 协议只负责物理规则: 冻结候选、成对提交、预算与探针配给、申报结束、留出审计。

它要解决什么

- 同一权重换 harness 差距巨大 (GPT-5 在 Terminus 2 上解出 Terminal-Bench 2.1 的 35.2%, 在 Codex CLI 里 49.6%); 而现有基准 (SWE-bench、GAIA、WebArena、 τ -bench、AgentBench——完整清单见 § 5 相关工作) 都把 harness 当**实验配置**固定住, 只测"模型解题" (§ 1, p.2; § 5, p.14)。
- 真实部署里工作量最大的一层——造并维护执行系统——被基准默认掉了 (FDE 类比: 目标要翻译、反馈要自造、执行系统要建并维护, § 2, p.3)。

- 评测"生成的 harness"比评测答案难在哪：过拟合写它的模型、背下开发样例、暗中退步、只刷反馈集不迁移——由此定下双轴：**能力**（留出任务成功率）+ **效率**（执行 token）（§ 1, p.2）。

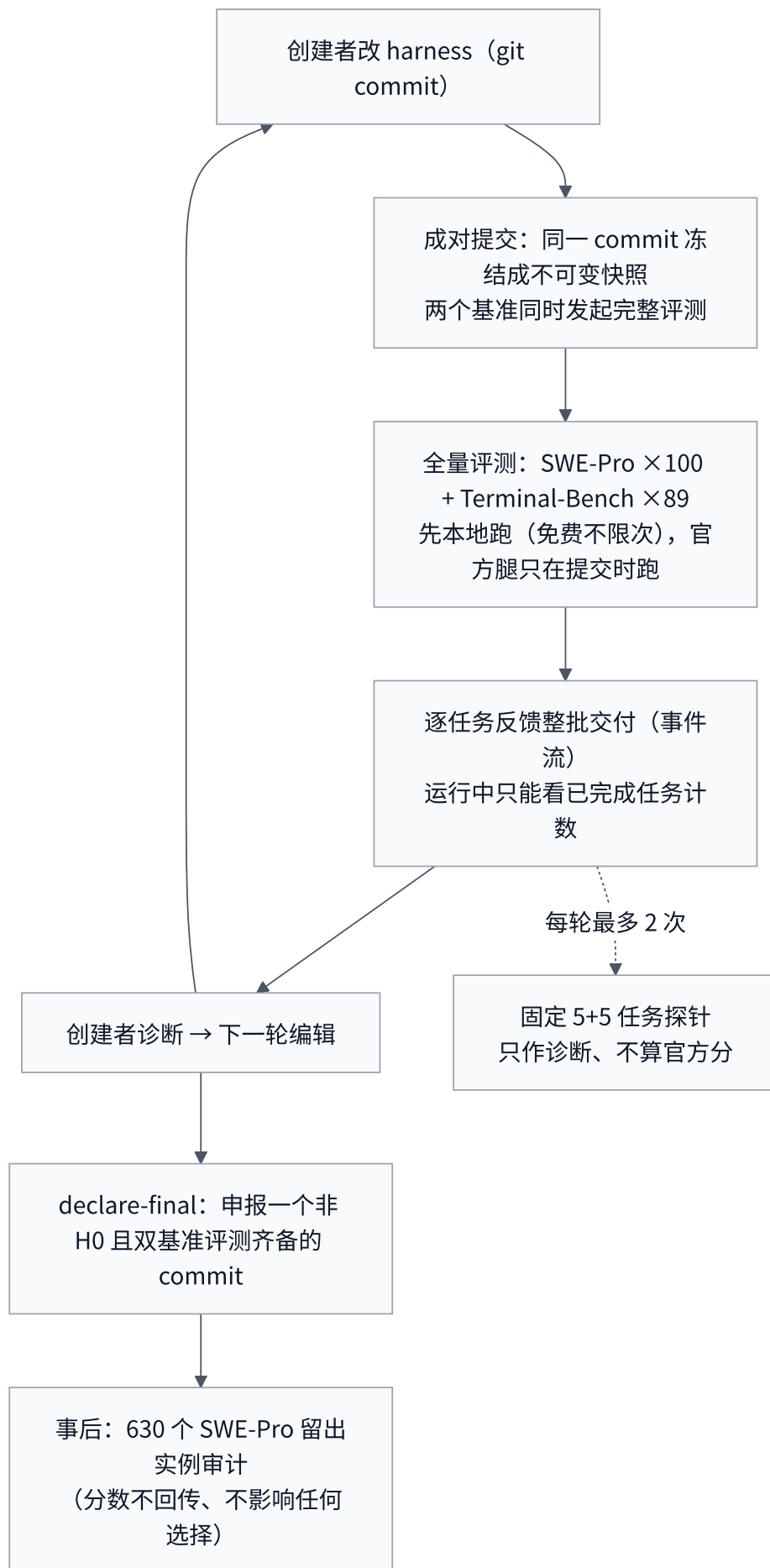
系统由哪些部件组成（论文原名）

部件	具体是什么	作用
弱种子 H_seed	"可运行的兼容层，不是解题 agent"：解析任务与模型配置、暴露 被动 低层工具、写审计产物； 没有 agent 循环、任务分解、工具策略、上下文管理、持久状态、验证器、重试恢复、停止规则	未修改时在五个基准上 全 0 分 ——任何非零 Creation 分数都只能来自创建者新增的执行逻辑
开发者环境 D	Claude Code 2.1.177（GPT-5.5 用 Codex）	创建者的工位
创建者 L_C	6 个模型（Opus 4.8 / GPT-5.5 / Gemini 3.1 Pro / DeepSeek V4 Pro / Qwen 3.7 Max / Seed 2.0 Pro）	造 harness、改 harness
执行模型 L_E	Self-Eval 下 = 创建者自己；Unified-Eval 下固定为 Gemini 3.1 Pro	冻结后在 harness 里跑下游任务
评测器 J	各基准官方固定评测器	打分
任务面	五基准共 2,207 个实例：SWE-bench Pro 731、Terminal-Bench 2.1 89、MLE-bench 75、EQ-Bench3 46、BrowseComp 1,266	能力面
反馈集	Evolution 专用：100 个 SWE-Pro + 全部 89 个 Terminal-Bench 任务	开发信号；只有它们产生官方分
探针	固定子集 ：每个基准的 前 5 个 任务各跑一条腿；每轮最多 2 次； 只作诊断、永不算官方分 （n=5，一个任务动 20 分）	定向诊断/回归门
留出集	630 个 SWE-Pro 实例，与 100 个反馈任务 不相交	事后泛化审计；分数从不回传，也不影响编辑/停止/申报
官方版本	一个 commit 必须在 两个基准上都有完整的官方评测 （成对提交、两腿都结算）	记账单位

一次运行怎么走

Creation (RQ1)：创建者拿到弱种子 + 规格 + 1-3 个开发用例，在公开开发循环里改、跑、看日志（隐藏任务/答案/官方分不下发）→ 冻结 → 每个"创建者 × 基准"独立造 3 个 harness、报 avg@3 → 同时报能力与执行 token（创建者 token 不计）。

Evolution (RQ2)：



- 预算：初始 harness 之外的 10 对完整评测；探针按轮配给（提交一对即重置为 2，用不完不结转）。

- **结束条件**：唯一方式是创建者**申报**（控制器从不代申报）；另有 liveness guard（长时间无动静先发提醒、再转"只能申报"模式）。
- **评分路径与 harness 隔离**：harness 自报的状态**永不打分**——SWE-Pro 的分数只来自任务工作目录里真实的 repository diff，Terminal-Bench 只看最终环境状态；配合全量留档，事后审计"交付的代码"与"代码实际执行了什么"。本篇审计结论：**零违规**。

轮与轮之间更新了什么

更新什么	载体	谁写	何时写	可否回滚
harness 源码	git 工作区（HEAD = 当前候选）	创建者	每轮编辑	可（git revert / checkout；轨迹里有回退同一文件的实例）
已提交候选	控制器侧冻结快照	控制器	提交时	不可；运行中的评测只认启动时的快照
反馈事件流	append-only JSONL	控制器	每次评测终结时整批	不可
评测账本、预算/探针计数	控制器	控制器	每次提交	不可逆（腿部分失败也照样消耗预算）
申报	declare-final 记录	创建者	结束时	不可逆
官方分	只来自"双基准都完成"的官方评测	控制器	评测结算	探针/本地跑永不算

什么时候停、什么算「更好」

- **合法候选** = 非初始 harness 的 commit，且两个基准都有完成的官方完整评测；探针版本、半截腿、停止的运行、无效实例都不算；同一 commit 的基建修复可合并（重提只补跑缺失腿、不再计费）。
- **申报 ≠ 最高分**：论文测的正是"创建者的选择对不对"——9 条谱系里只有 **2 条**申报的是留出最优版本。
- **留出审计**：所有官方版本事后在 630 个 SWE-Pro 实例上补测；注意 Terminal-Bench 的 89 个任务**全部**是反馈集、没有留出腿——两个基准里有一条腿本身就是样本内，"涨分迁移有限"的证据有一半来自它。

监督信号从哪来

- 三层信号要分清：① **任务正确性**（唯一算分的：verifier 查最终状态）；② **运行诊断**（adapter 状态、日志、超时——注意 harness 自报成功可以得 0 分，诊断任务也可以得 1 分）；③ **行为产物**（是否发生过编辑——非空 patch 可以全错、空 patch 也可以成功）。
- **两个评测视角**：Self-Eval（创建者用自己的 harness 跑隐藏任务，对应真实部署形态，测"模型-harness 协同设计"）；Unified-Eval（所有 harness 用同一个固定执行模型跑，测"harness 是不是可迁移的软件资产"）。

关键数字

事实	数字	出处
Creation (Self-Eval, avg@3; 均值口径: SWE-Pro / Terminal-Bench / EQ-Bench3 / BrowseComp 四基准 无权重均值, 不含 MLE-bench)	种子 0; Opus 67.8 (69.3 / 64.8 / 32.9 / 84.6 / 52.4); Gemini 55.6; GPT-5.5 55.1; DeepSeek 45.2; Qwen 44.0; Seed 22.8; 人类参照 86.2 (外部配对、非同一执行器对照)	表 3, p.8
换执行器即塌方	Opus 的 SWE-Pro harness: Self-Eval 69.3 → Unified-Eval 33.0; 其搜索 harness 的 重复查询率 10.1% → 88.2% (为原执行器调过的去重/终止逻辑失效); 一个 120 步上限被硬编码	§ 4.2, p.7–10
机制层: 状态写了不触发	18 个 code harness 全都有显式执行循环, 11 个定义了 State 类, 但只有 1 个暴露状态保存接口、1 个实现周期 checkpoint; 26,679 条记录轨迹里 0 次 checkpoint 事件 (全部"从未出现"的组件都属于 state & memory)	§ 4.2, p.8
机制层: 验证停在语法层	2,325 个 Data 任务里有 441 个 产出退化提交、无 harness 检出; 自测数量与下游分几乎无关 (Spearman 0.13–0.26), "revision calls" 相关 0.57	§ 4.2, p.8–9
77.8%	"77.8% of failed Data tasks are attributed to harness defects"——全文唯一一处、无口径分解	§ 4.2, p.7
Evolution (Self 设置, 留出 630)	Gemini +8.8 (反馈) /+2.70 (留出); Opus +3.0/ +4.44 (留出反超反馈); Qwen +13.9/+1.43; DeepSeek +13.4/+3.17; GPT-5.5 +5.9/+3.81	表 6, p.12
Evolution (固定 Gemini 执行器)	Opus +9.7/+2.70; Qwen +1.1/ -1.11 ; DeepSeek +6.5/ -2.38 ; GPT-5.5 +2.4/ -10.32 (四条里只有一条留出为正)	表 6, p.12
方向一致性	64 次相邻版本切换中, 反馈与留出 同向的只有 34 次 (53.1%) ; 同一 commit 重复评测的 pair 分可变约 ±4.75 分	§ 4.3, p.12–13
编辑结构	58 处改执行/控制流、37 改工具、17 生命周期、16 上下文、4 状态、 0 独立 verifier ; 申报版本中位数 8 个文件、+476 / -38 行; 169 个新增函数/类中 113 可达、31 只经死代码可达、25 无调用者	§ 4.3, p.12–13
探针 ≠ 全量	一个 GPT-5.5 候选通过全部 5 个 Terminal 探针, 全量只得 0.584	§ 4.3, p.12

陷阱与边界

- 1. **结论是描述性的**: 每个"创建者 × 执行设置"格只有一条**轨迹**, 无不确定度估计; 留出只覆盖 SWE-Pro; 人类基线是外部配对 (论文自己也说"不应解读为绝对上限")。
- 2. **成本口径**: 只计执行 token, 创建者、judge、探针的消耗全排除——"efficiency"不是总成本。
- 3. **涨分结论要分三种口径写** (论文最容易被误读处): 反馈 pair (含 Terminal, 全样本内)、留出 630 (只有 SWE-Pro, 事后)、申报 vs 留出最优 (选择正确性) ——混用会得出错误结论。
- 4. **其它零碎**: 731 = 100 + 630 + **1 个未说明去向的实例**; "R3 replica"未定义; 0.584 的单位未注明; 容器"为复现而非隔离"配置, 复用生成的 harness 应视为不可信代码。

概念对照

同一名词在不同论文里意思不同，先看这张表可以少走弯路：

名词	00214 Dream-RSI	00212 RRSI	00205 Prime Agent	00199 HarnessX	00189 HarnessDev
policy	被进化的探索策略（一份 Python 代码），决定"从哪搜、并行多少、何时停"	被冻结的 LLM（Opus 4.8），全程不训练——是"执行任务的模型"	偶用该词但含义不同（"不可改写的基础策略"、配置项里的 policies），没有"可优化策略"的意思；被固定的对应概念是"模型权重（L0）不动"	两层含义：RL 镜像里的搜索过程（ π_{evo} ，不训练）与 GRPO 训练的模型策略（ π_{θ} ）；只有后者是参数化策略	该词只作普通搭配出现（tool policy / solution policy / policy-free）；角色分工用 creator（造/改 harness 的模型）与 executor（跑任务的模型）
harness	主要指其 § 4.4 实验里的"围绕固定 LLM 的程序外壳"；被进化的策略本身是策略代码，不是 harness	权重之外的一切（prompt、控制流、工具接口、记忆等 9 类组件），以源码形式被编辑	整个系统（REPL + 子 agent + daemon + L3 状态）；"harness state" 特指 Continual Harness 的持久条目	一等对象 H =（模型角色配置 M，行为配置 C）；C 是按 8 个生命周期 hook 组织的 processor 列表 + 共享 slot	生成物代码本身（agent 循环、工具、上下文、状态、生命周期、验证），代码统计/责任范围只算 harness/ 目录
L0 / L1 / L2	未使用	正则化的功能类比名（退火预算 $\approx$ L0、剪枝 $\approx$ L1、成本准入 $\approx$ L2）——作者反复强调不是真的在优化带范数惩罚的目标	状态层级（L0 权重 / L1 活动上下文 / L2 REPL 与子 agent / L3 磁盘态），每层有唯一写机制	未使用	未使用
β	预算偏好旋钮（策略内部单一标量，扫描取 0 / 0.25 / 0.5 / 0.75 / 1）	成本规则系数 β_0 （基础容忍）、 β_1 （每分增益的额度）	未使用	GRPO 目标里的 KL 系数（共演化实验取 0）；与 RRSI 的 β_0/β_1 无关	未使用

名词	00214 Dream-RSI	00212 RRSI	00205 Prime Agent	00199 HarnessX	00189 HarnessDev
evolve set / held-out / OOD	无 evolve set / OOD 概念 ("被反复评分的世界"这一角色由记录树承担); 但用了 held-out (Lasso 另有 6 个留出下游数据集)	evolve set = 唯一被反复测量的任务集 (只在一个 suite 上演化); held-out = 同分布留出 (Harvey LAB 40 题); OOD = 从未见过的其他 benchmark	无 evolve / held-out / OOD 等划分 (三词均未出现); 概念是 "任务本身"	无留出评测 : 所有增益都在演化用的同一任务集上测, held-out 泛化明确未评估 (作者自认)	反馈集 = 100 SWE-Pro + 89 Terminal-Bench; held-out = 630 个不相交 SWE-Pro (事后审计, 分数不回传)
probe	回放里 "揭示一个格子" 也叫一次 probe (每揭示一格算一次调用/探针)	未用作机制名 (只在定性描述里出现: RRSI 接受的编辑含三个 "标准库探针")	未用作机制名 (只在 § 3.3 提到模型自建的 probe function)	未使用	固定 5 任务子集 的诊断评测, 永不算官方分
refinement / evolve	dreaming = 离线 "提候选策略 → 回放评测 → 反馈改进"	演化 (evolve) = 对 harness 源码的搜索循环	refinement = 把轨迹证据变成版本化 L3 条目的机制 (含 <code>/refine</code> 命令)	§ 4 的 adaptation 环就是 harness-only 演化; § 5 的 co-evolution 再把模型 GRPO 接进同一个 buffer (论文未给严格二分)	Evolution (RQ2) = 创建者用下游反馈持续改 harness 的评测阶段
generation	一次尝试 = 一个子节点 = 一次 discovery-agent 调用 (成本统一单位)	未用作机制名 (普通词义)	未用作机制名 (普通词义: kernel generation、model generations 等)	未用作机制名 (edit generation / code generation 等普通词义)	未用作机制名 (普通词义)
replay simulator	冻结的记录树 (不是学出来的动力学模型; 不学模型, 用记录当模型)	未使用	未使用	未使用 (只有 GRPO 的 replay buffer, 不是模拟器)	未使用

另外两条容易混的:

- **Self-Eval vs Unified-Eval** 是 HarnessDev 专属: 前者创建者用自己的 harness 跑 (测协同设计), 后者全部用同一个固定执行模型 (测 harness 能否作为软件资产迁移)。
- **"申报版本" (declared version)** 也是 HarnessDev 专属: 创建者自己宣告 "这就是最终版", 不一定是留出分最高的版本——论文测的就是这个选择对不对 (9 条里 2 条选对)。

五篇放在一张坐标系里

本仓库 00192 笔记（Awesome RSI 导读）给自进化工作建了一套坐标系（四个主维度：进化什么 / 进化结构 / 谁来更新 / 更新时机；五个次级维度含验收标准、反馈来源等）。下面借这套坐标系把五篇放进去，各自的位置与差异一目了然（表述以本文各节为准）。表里没有单列「进化结构」一维——这里的「结构」指**版本谱系**，五篇分别是：

- **Dream-RSI**：单链；旧版本留存可查阅，每轮候选集含当前版本，只保证回放分不降（它的**搜索**本身是一棵带并行批次的发现树，但那是单次 rollout 的结构，不是版本谱系）。
- **RRSI**：单链，未定义显式回滚。
- **Prime Agent**：无 harness 版本谱系（L3 条目仍有版本历史与回滚）。
- **HarnessX**：「单链+按任务簇隔离的变体池」。
- **HarnessDev**：不提出演化方法。

	00214 Dream-RSI	00212 RRSI	00205 Prime Agent	00199 HarnessX	00189 HarnessDev
进化什么	探索策略代码（元层面）	harness 源码（组件级增删改）	harness 的持久状态（四类条目）	harness 配置/代码（模型角色、processor、prompt、工具、控制流）；共演化时还包括 模型权重	不提出进化方法——测量"模型能否自建/自改 harness"（harness 的变化是 被测现象 ）
谁来改	policy-development agent 改写代码	LLM proposer 起草 + 算法 选择	agent 自己（或 <code>/refine</code> 的后台调用）	AEGIS 四阶段 meta-agent（Opus 4.6）； 确定性门拥有最终否决权	创建者模型（现成 coding agent workflow）
何时改（更新时机）	分两层： 发现树 在线累积； 策略代码 只在每轮离线阶段更新（在线阶段策略当轮固定）	离线批次：每轮只在 evolve set 上评测后改	在线：任务进行中、轮次边界	离线批次：每轮在适应批次上 rollout 后改；共演化时 同一轮内 依次执行 AEGIS 与 GRPO，互不条件于对方输出	离线批次：每对完整评测之间
验收标准	回放分最高者胜（当前策略始终在候选集里 → 只保证代理分不降）	泄漏筛查 + 三道非补偿准入条件（噪声地板 / 成本规则 / DomainGuard）——全过才准入	无量化门槛；条目可增删改、可回滚；不改写不可变的基础 prompt	Critic + 确定性门（manifest 完整性 → 配置规范化 → build/冒烟 → seesaw：不回归任何已解任务 ）；变体隔离时按变体作用域检查	协议本身（成对提交、预算、探针、申报）+ 事后留出审计
反馈来源	记录树里 已发生 的真实评测分	有限 evolve set 的分数与 token 成本	执行轨迹证据本身（无 verifier 模型）	完整 trace + 固定 verifier 打分（按 pass@2 聚合）	双基准全量评测的逐任务反馈

	00214 Dream-RSI	00212 RRSI	00205 Prime Agent	00199 HarnessX	00189 HarnessDev
主要代价	回放算力 + 开发 agent 调用（成本口径未含）	评测开销（Harvey LAB 一次完整评测约 14,000 条 rubric 判定，RRSI § B.3, p.17）	测试时算力与长时程运行成本	每轮全任务集 rollout（最多 15 轮；meta-agent 100-175M token）；部署后不再需要 meta-agent	宿主评测资源（一对完整评测两腿全量）
论文自证的效果强度	分域参差：数学 3 项里 1 项改进、1 项并列、1 项差于自己的固定基线；Lasso 的均值改善由 RCV1 单集驱动；harness 域 +8.4 分（相对种子，无固定策略对照）	六个留出 split 无一退步（含整体复跑），OOD 最多 +4.7 个百分点	多数比较是"接近/噪声级"，ARC 的对比是外部参照	15 格中 14 格改善、均值 +14.5 个百分点（峰值口径、无留出）；变体隔离 GAIA +13.6 个百分点；共演化峰值再 +4.7 个百分点	Evolution 涨分部分迁移（53.1% 同向），四条固定执行器设定里三条留出为负

读下来的总体印象（本文归纳，非任一篇的结论）：

- **RRSI 是五篇里唯一以"泛化"为核心贡献、并给出系统性正向证据的**（HarnessDev 也做留出审计，但它是评测方，不提出演化方法；RRSI 的核心贡献是提议侧与选择侧两组正则，接受/拒绝准则只是选择侧的一部分）；代价是评测开销大、且"省 token"部分是成本规则直接筛选出来的。
- **Prime Agent 的贡献在"状态工程"**：把"什么该被持久化、以什么形式、怎么回滚"定义清楚；它不提供改进的度量，改进质量依赖模型与后续任务。
- **Dream-RSI 的贡献在"让元级反馈变便宜"**：用记录回放替换重跑；代价是回放只对"已走过的格子"负责，天然偏向加深已有分支。
- **HarnessX 的贡献在"把演化运行时工程化"**：类型化 hook/processor 约束编辑面、change manifest 让编辑可伪证、确定性门用"不回归已解任务"兜底，还把 harness 演化与模型 GRPO 接进同一个数据闭环；但它的成绩是**峰值、且在同一任务集上测**（无留出）——与 HarnessDev 的负面证据正好互为镜像。
- **HarnessDev 的贡献是给这一整类工作装仪表盘**：它给出的负面证据（状态写了不触发、验证停在语法层、反馈分与留出分只有一半同向、申报版本多数不是最优）对前几类方法同样适用（HarnessX 也未做留出评测）——**"能自改"离"改得对"还有相当距离**。

▮ 版本与核对说明

本文的核对手段：五篇原文均按上表版本重新获取并逐页核对（前四篇 2026-10-15；HarnessX 2026-10-17，其 PDF sha256 与笔记记录逐字节一致）。前四篇在生成阶段做了多轮核对，其中另派独立 agent 做了三轮复核——第一轮覆盖全部表格、关键公式符号、图内标签与「论文未给」类否定断言；第二轮复查修正项与内部一致性；第三轮做读者视角连贯性检查与数字抽查；HarnessX 由独立核对逐节复核后写入。核对以 PDF 文本层（含逐字符坐标与字体信息）为主，表格数字逐格核对；**图内读数**（如 Dream-RSI 图 6 的四条终值、图 7 的逐轮数值、HarnessX Figure 4 的逐轮读数）沿用原笔记的像素测量并经复核，未对全部曲线中段重新渲染测

量；本文引用的数字一律以正文与表格值为准，图内读数与复算值的判断在正文里已就地标明。各篇更细的未核清单见对应笔记的「复核记录」与本文各节标注。

Dream-RSI 的版本选择：README 该行指向 arXiv，故本文以 **arXiv v2** 为准；仓库归档的 36 页项目页版（ $\approx$ arXiv v1：页数、数字一致，措辞有修订）仅用于版本比对。两版共享的数字完全一致，但式 (1) 被重写过（v1 = 三分项"最好分 - $\beta_1 \times$ 工作量 + $\beta_2 \times$ 并行奖励"；v2 = "归一化前沿面积 - $\lambda \times$ 并行惩罚"，与实现一致），且 v2 收紧了三处口径、新增 harness 域与 PACEvolve 基线。引用方法时请按 v2；读旧笔记时注意这一差异。

摘要与表格冲突时的处理：本文一律采用表格/附录值，冲突处已就地注明（RRSI 的 30% vs 36%、+4.3 vs +4.7；HarnessDev 的 77.8% 无口径；Dream-RSI 的 v1 口径已在 v2 修正）。

五篇笔记与本文的关系：笔记是逐篇精读产物（含逐条表述张力、配图、复算记录），本文是跨篇的方法重写——同一事实在两边若有出入，以本文标注的版本与出处回原文核对，并可在笔记中找到更细的核对记录（如 RRSI 笔记列了 5 条原文张力、Dream-RSI 笔记列了 9 条（其中 7 条附 v2 现状）、HarnessX 笔记列了 6 条）。