自进化智能体综述

A Survey of Self-Evolving Agents: On Path to Artificial Super Intelligence

Huan-ang Gao[γ†], Jiayi Geng[α†], Wenyue Hua[ϵ†], Mengkang Hu[ω†], Xinzhe Juan[σμ†], Hongzhang Liu[ξ†], Shilong Liu[α†], Jiahao Qiu[αδ†], Xuan Qi[γ†], Yiran Wu[ρ†], Hongru Wang[τα†✉], Han Xiao[τ†], Yuhang Zhou[λ†], Shaokun Zhang[ρ†], Jiayi Zhang[π], Jinyu Xiang, Yixiong Fang[θ], Qiwen Zhao[ζ], Dongrui Liu[σ], Qihan Ren[σ], Cheng Qian[β], Zhenhailong Wang[β], Minda Hu[τ], Huazheng Wang[η], Qingyun Wu[β], Heng Ji[β], Mengdi Wang[α♭✉]

[α]Princeton University, [δ]Princeton AI Lab, [γ]Tsinghua University, [ϵ]Carnegie Mellon University, [ξ]University of Sydney, [σ]Shanghai Jiao Tong University, [ρ]Pennsylvania State University, [μ]University of Michigan, [η]Oregon State University, [τ]The Chinese University of Hong Kong, [λ]Fudan University, [π]The Hong Kong University of Science and Technology (Guangzhou), [ω]The University of Hong Kong, [ϵ]University of California, Santa Barbara, [ζ]University of California San Diego, [β]University of Illinois Urbana-Champaign,

Github Repo: https://github.com/CharlesQ9/Self-Evolving-Agents

[†]Equal contribution and the order is determined alphabetically, [✉]Corresponding Author

Mengdi Wang (Princeton): 强化学习领域，DeepMind 兼职研究员。

Heng Ji (UIUC): 信息抽取与知识图谱专家。

Qingyun Wu (Penn State): 自动机器学习 (AutoML) 与多智能体系统专家 (AutoGen 作者)。

刘宇阳

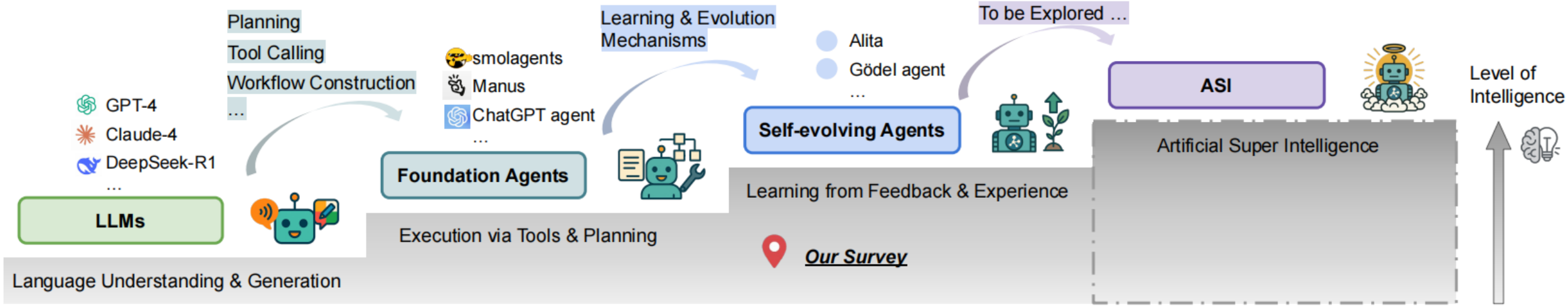
2025.12.10

持续学习 vs Self-Evolving Agents

北京大学 PEKING UNIVERSITY

□ 静态 LLM vs. 动态环境：

- ✓ 参数冻结的 LLM 无法适应**新任务、新领域知识或动态环境 (Open-ended Environments)**。
- ✓ 传统 CL 主要关注参数更新中的“灾难性遗忘”与“可塑性”的平衡（Stability-Plasticity Dilemma）。
- ✓ 现在的 进化 不再局限于 **Gradient-based 的参数更新**，扩展到了**非参数化的组件更新（Memory, Tools, Context）**。



> *"It is not the most intellectual of the species that survives; it is not the strongest that survives; but the species that survives is the one that is able best to adapt and adjust to the changing environment in which it finds itself."*
>
> *— Charles Darwin*[1]

> *“能够生存下来的物种，并非智力最高的，也并非力量最强的，而是最能适应和调整自身所处环境的物种。”*
>
> *——查尔斯·达尔文*[1]

定义与基础：范式定位

自进化代理 (Self-Evolving Agents):指能够通过与环境的持续交互，自主地改进其内部组件（包括模型参数、记忆、提示策略、工具库及自身架构）的智能系统。核心在于打破静态限制，实现开放域中的持续适应。

□ Self-Evolving Agents 定义：

✓ 智能体在开放世界中，通过与环境交互（**Trajectory τ**）和接收反馈（**Feedback r**），自主更新其组件（$\Pi \rightarrow \Pi'$）以最大化（**Utility U**）。

- **环境 (Environment) POMDP：** $E = (\mathcal{G}, \mathcal{S}, \mathcal{A}, T, R, \Omega, O, \gamma)$。
- **智能体系统 (Agent System)：** $\Pi = (\Gamma, \{\psi_i\}, \{C_i\}, \{\mathcal{W}_i\})$。
  - Γ: 架构拓扑 (Architecture/Topology)。
  - ψ_i: 模型参数 (Model Weights)。
  - C_i: 上下文 (Context: Prompt & Memory)。
  - $\mathcal{W}_i$: 工具集 (Tools)。

自进化策略 (Strategy)： $f(\Pi, \tau, r) = \Pi'$。基于轨迹 τ 和反馈 r 将当前系统 Π 映射为新系统 Π'。

$$\max_f \sum_{j=0}^{n} U(\Pi_j, \mathcal{T}_j)$$

可以源自任务特定的反馈r，如奖励信号或文本评估，也可能与其他绩效指标（如完成时间、准确性或鲁棒性）结合使用。

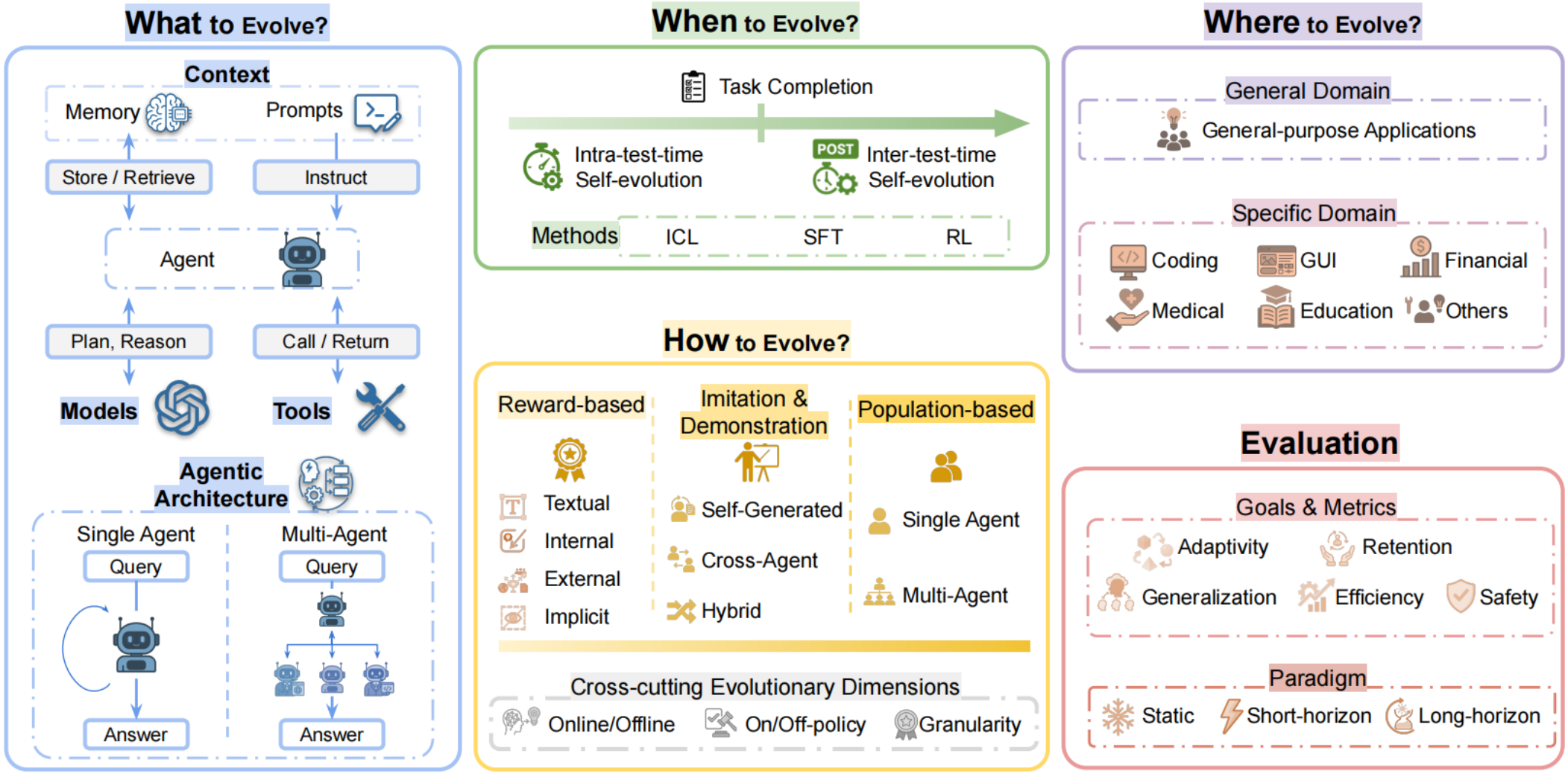
□ 与传统范式的区别：

✓ vs. Curriculum Learning： CL 侧重数据排序，Agent 侧重**自主探索**和**非参数更新**。

✓ vs. Lifelong Learning： 传统 LL 侧重被动任务流，Agent 强调**Structural Change (拓扑/工具结构变化)**和 **自我反思 (Self-Reflection)**。

Paradigm	Evolving Context	Evolving Toolset	Dynamic Tasks	Test-time Adaptation	Active Exploration	Structural Change	Self-reflect & Eval
Curriculum Learning	✗	✗	✗	✗	✗	✗	✗
Lifelong Learning	✗	✗	✓	✗	✗	✗	✗
Model Editing	✗	✗	✓	✓	✗	✗	✗
Self-evolving Agents	✓	✓	✓	✓	✓	✓	✓

综述核心分类体系



$$\Pi = (\Gamma, \{\psi_i\}, \{C_i\}, \{\mathcal{W}_i\})$$

综述核心分类体系

北京大学 PEKING UNIVERSITY

What to Evolve (进化什么):

- ✓ **Model:** Policy, Experience (Self-Generated Data).
- ✓ **Context:** Memory (Storage/Retrieval), Prompt Optimization.
- ✓ **Tool:** Creation, Mastery, Selection.
- ✓ **Architecture:** Single-Agent Optimization, Multi-Agent Collaboration.

When to Evolve (何时进化):

- ✓ **Intra-test-time:** 任务执行中的实时调整 (Task Completion).
- ✓ **Inter-test-time:** 任务间隙的复盘与升级 (Post-hoc Learning).

How to Evolve (如何进化):

- ✓ **Reward-based** (RL/Feedback),
- ✓ **Imitation** (Demonstration),
- ✓ **Population-based** (Evolutionary Algo).

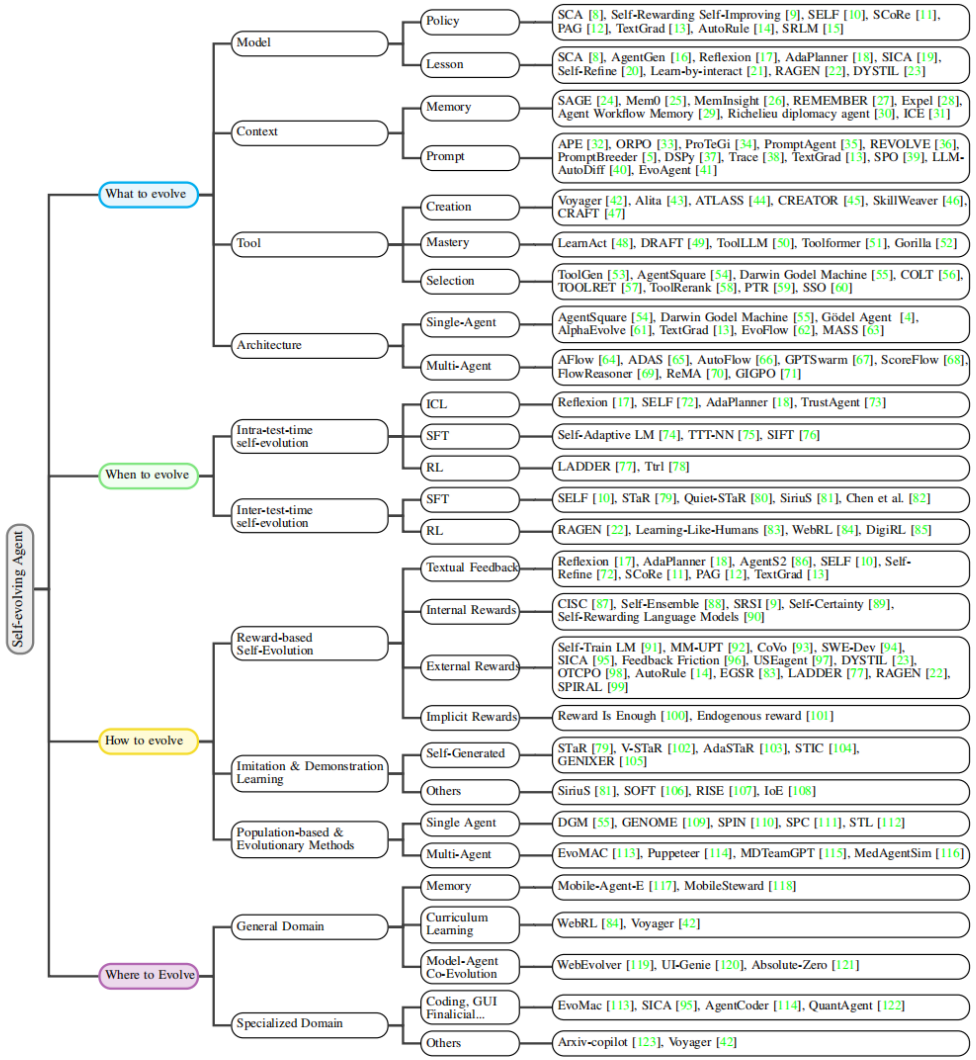


Figure 2: Taxonomy of self-evolving agents, in which agents are analyzed along the *what*, *when*, *how*, and *where* dimensions, with selected representative methods and systems annotated at each leaf node.

综述核心分类体系

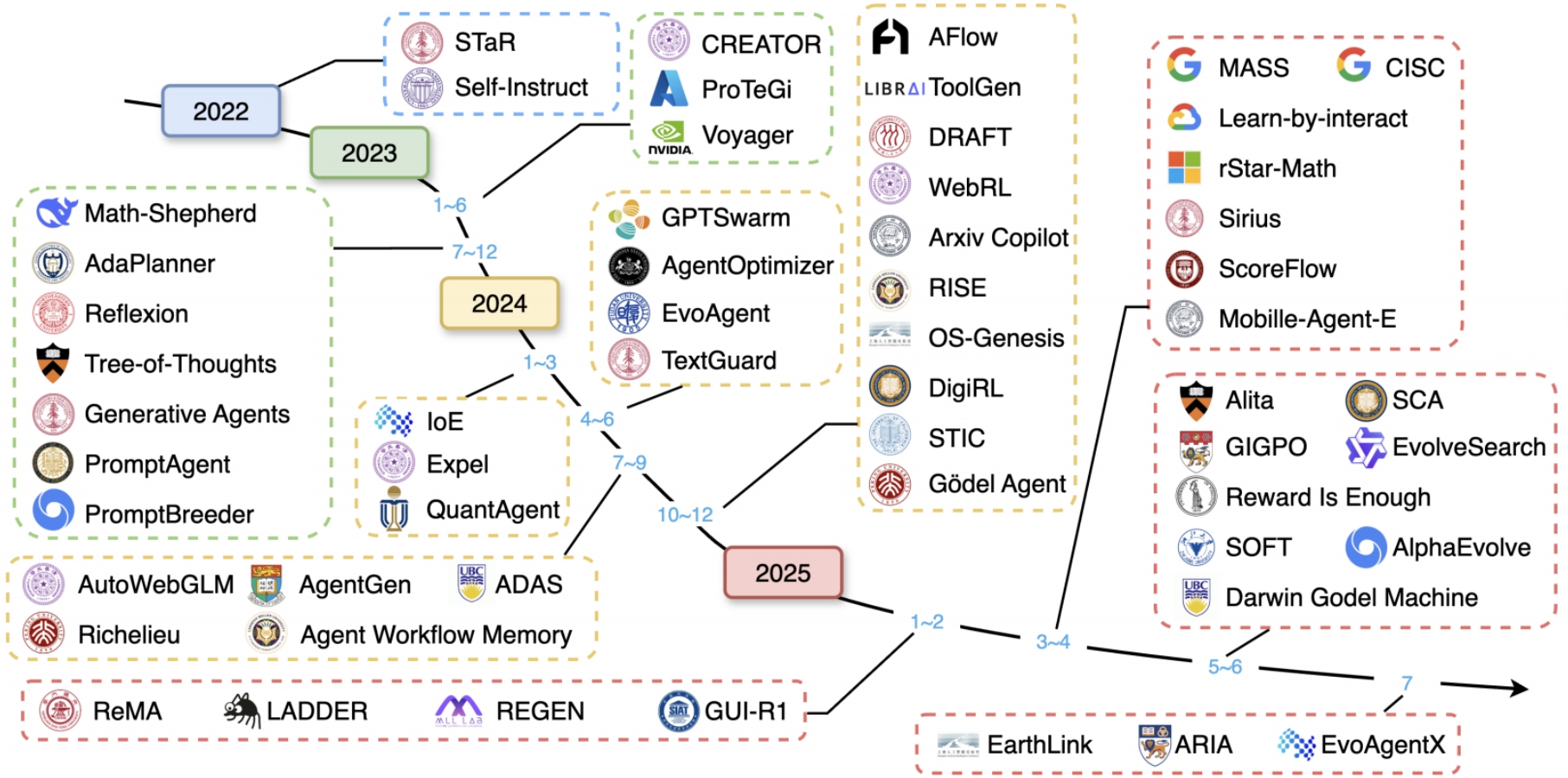


Figure 4: An evolutionary landscape of several representative self-evolving agent frameworks from 2022 to 2025. The figure chronologically organizes major research milestones in the development of self-evolving agents with capabilities such as autonomous planning, tool use, and continual self-improvement.

What to Evolve

01. 模型 (Models)

核心认知载体的参数更新。

- 自生成数据微调 (SFT)
- 强化学习 (RL) 策略优化
- 在线/离线经验吸收

02. 上下文 (Context)

动态知识的存取与指令优化。

- 记忆演化: ADD/MERGE/DELETE
- Prompt 优化: 离散搜索 vs 可微梯度
- 工作流记忆归纳

03. 工具 (Tools)

行动能力的边界扩展。

- 自主发现与创造 (Voyager, Alita)
- 技能掌握与参数精炼
- 工具编码与检索 (ToolGen)

04. 架构 (Architecture)

系统拓扑与协作模式的重构。

- 单代理: 异构模型选择与节点优化
- 多代理: 角色/通信拓扑搜索
- 自我重编程与元学习

Table 2: Representative self-evolving agent methods positioned along four evolutionary pillars; a filled bullet (●) marks dimensions where the approach actively evolves.

Method	Model		Context		Tool			Architecture	
	Policy	Experience	Prompt	Memory	Creation	Mastery	Selection	Single	Multi
SCA [8]	●	●	○	○	●	○	○	○	○
RAGEN [22]	●	●	●	○	○	○	○	●	○
AgentGen [16]	○	●	●	●	●	○	○	●	○
Promptbreeder [5]	○	○	●	○	○	○	○	●	○
Expel [28]	○	●	○	●	○	○	○	○	○
Agent Workflow Memory [29]	○	○	○	●	○	○	●	○	○
Mem0 [25]	○	○	○	●	○	○	○	○	○
MAS-Zero [159]	○	○	●	○	○	○	○	○	●
Multi-Agent Design [63]	○	○	●	○	○	○	●	○	●
SPO [39]	○	○	●	○	○	○	○	○	○
Alita [43]	○	○	○	○	●	○	●	○	○
TextGrad [13]	○	○	●	○	○	●	●	●	○
DGM [55]	○	○	●	○	○	○	○	●	○
AlphaEvolve [61]	○	○	●	○	●	●	○	●	○
ADAS [65]	○	○	●	○	●	○	○	●	●
AFlow [64]	○	○	●	○	●	○	●	●	●
ReMA [70]	○	○	○	○	○	○	○	○	●
SkillWeaver [46]	○	○	○	●	●	●	●	○	○
LearnAct [48]	○	○	○	●	●	○	●	○	○
DRAFT [49]	○	○	●	○	●	●	○	○	○
ToolGen [53]	○	○	○	●	●	●	○	○	○
CRAFT [47]	○	○	○	○	●	●	●	○	○
CREATOR [45]	○	○	○	○	●	●	○	○	○
Voyager [42]	○	○	●	●	●	●	●	○	●

What to Evolve (I) - 核心认知能力的进化 (Model & Context)

北京大学 PEKING UNIVERSITY

- **Model Evolution: 参数与经验的内化**:
  - ✓ **Policy (策略更新): 智能体自主生成数据微调自身。**
  - ✓ SCA (Self-Challenging Agent): 角色扮演 提出Code-as-Task的问题者，并解决问题。
  - ✓ TextGrad: 将文本反馈视为“梯度”，反向传播优化参数或 Prompt。
  - ✓ AutoRule: 将 Reasoning Trace 转化为显式基于规则的训练奖励。
  - ✓ **Experience (经验累积): 从交互中学习，不一定改参数，但改行为分布**:
  - ✓ *(捕捉经验并将其转化为推动迭代改进的学习信号)*
  - ✓ Reflexion: 语言形式的强化学习 (Verbal RL)，记录 critique 到短期记忆，指导未来的行为以避免重复错误。
  - ✓ RAGEN: 在多轮工具使用中进行在线 RL 训练。
- **Context Evolution: 上下文与记忆的动态管理**:
  - ✓ **Memory Evolution (CL 关键): 解决“学什么”和“忘什么”**:
  - ✓ Mem0: 两阶段pipeline，支持 ADD/MERGE/DELETE 操作，维护一致性。
  - ✓ MemInsight: 从轨迹中提取 Insight/Rules，而非存储原始 Raw Data (Experience Distillation)。。
  - ✓ **Prompt Optimization (PO): 优化指令**:
  - ✓ OPRO/DSPy: 将 Prompt 视为可优化参数，通过搜索或编译优化。
  - ✓ PromptBreeder: 维持一个群体，以发现越来越有效的指令。

What to Evolve (II) - 外在能力的扩展 (Tool & Architecture)

北京大学 PEKING UNIVERSITY

- **Tool Evolution: 从使用者到创造者**:
  - ✓ **Creation (自主创造): 应对未知任务。**
  - ✓ Voyager: 在 Minecraft 中利用 LLM 写代码生成新 Skill 并存入库。
  - ✓ CREATOR: 解耦抽象工具创建与具体工具使用。
  - ✓ **Mastery (熟练度): 解决新工具不可靠问题**:
  - ✓ LearnAct: 通过试错 (Trial-and-error) 修正工具代码或文档。
  - ✓ **Selection & Management (管理): 应对工具爆炸。**
  - ✓ ToolGen: 将工具检索建模为 Token 生成 (Virtual Token)。

- **Architecture Evolution: 系统拓扑的自适应**:
  - ✓ **Single-Agent Optimization:**
  - ✓ AgentSquare: 模块化搜索空间 (Planner/Memory/Tool 模块组合搜索)。
  - ✓ Darwin Gödel Machine: 允许 Agent 修改自身的 Python 代码，实现开放式进化。
  - ✓ **Multi-Agent Optimization:**
  - ✓ AFlow / ADAS: 将工作流构建定义为搜索问题 (MCTS)，自动发现最优 Agent 协作图。
  - ✓ EvoAgent: 进化智能体角色与 Prompt 组合。

When to Evolve - 进化的时机窗口

北京大学 PEKING UNIVERSITY

Intra-test-time (任务内)

“在做题时学习”：即时自适应。

Agent在执行当前任务的过程中，通过反思、尝试、错误修正来动态调整策略。这种演化通常是短时的、上下文相关的，旨在解决眼前的具体困难。

► 典型机制: 反思 (Reflexion), 动态规划修正 (AdaPlanner)

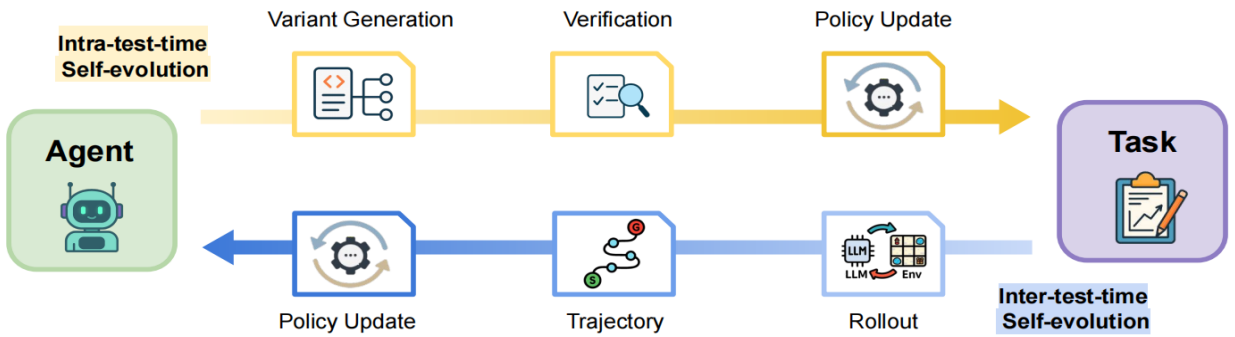
Inter-test-time (任务间)

“做完题后总结”：回顾性进化。

Agent在完成一系列任务后，利用积累的轨迹数据进行离线或在线更新。这种演化旨在提炼通用能力，更新模型参数或长期记忆，服务于未来的任务。

► 典型机制: 离线 SFT, 在线 RL (WebRL), 记忆归纳

When to Evolve - 进化的时机窗口



- **Intra-test-time Self-Evolution (测试时内)**：**目标是解决当前Specific Task**
  - ✓ **ICL-based:** 通过维护短期反思记忆缓冲，实现自我纠错。例如 AdaPlanner 根据环境反馈动态重写计划，而非死板执行；Self-Refine 通过多轮自我对话优化输出质量。
  - ✓ **即刻参数更新 (Instant SFT):** 利用元学习生成的"自编辑指令"，在推理阶段触发临时的权重更新，使模型快速适应当前上下文风格或约束。
  - ✓ **Test-time RL**: 遇到难题，生成该问题的多个变体进行集中训练 (LADDER)，在部署态临时提升特定领域的解题能力。
- **Inter-test-time Self-Evolution (测试时外)**：**目标是提升泛化能力，服务于未来任务**
  - ✓ **跨任务 ICL:** 将过往成功的轨迹、工具使用模式归纳为Prompt上下文。每一次成功的执行都为未来任务的少样本示例。
  - ✓ **自监督微调 (SFT):** SELF与STaR范式：对未标注数据生成推理链，通过结果过滤或自我评价筛选高质量数据进行微调。Quiet-STaR将这种推理内化为隐式思考。
  - ✓ **持续强化学习 (RL):** 在开放环境（如 WebRL、DigiRL）中，通过课程学习逐步提升任务难度，利用在线交互数据不断优化策略网络，实现"越用越强"。

How to Evolve (I) - 奖励驱动机制 (Reward-based)

核心逻辑： 优化目标 $max \sum U(\Pi, \tau)$ 。关键在于 Feedback r 的设计。

- **Textual Feedback (文本反馈)**：
  - ✓ **利用 LLM 理解自然语言 Critic。**
  - ✓ Reflexion (Verbal RL), TextGrad (文本梯度).
  - ✓ 信息丰富，可解释性强；但难以数值化，不稳定。
- **External & Implicit Rewards (显式与隐式)**：
  - ✓ **External:** 编译器报错、游戏得分、Rule-based check (AutoRule).
  - ✓ **Implicit:** 即使在未明确标记为奖励的反馈信号下也能学习。Reward Is Enough (In-context scalar signals)。
- **Internal Rewards (内在奖励)**：
  - ✓ **利用模型自身的 Logits 或 Consistency。**
  - ✓ Self-Consistency / CISC , Self-Rewarding LM.
  - ✓ 减少对人工标注的依赖，实现 Self-Supervised。
- **Reward Granularity (奖励粒度)**：
  - ✓ **结果导向** (DPO, Rejection Sampling) 。
  - ✓ **过程导向** (Step-level, e.g., Math-Shepherd , AlphaMath)，解决稀疏奖励问题。

How to Evolve (II) - 模仿学习与种群进化

北京大学 PEKING UNIVERSITY

◘ Imitation & Demonstration Learning：

- ✓ **Self-Generated Demos:** STaR（Bootstrap Reasoning). 自举式生成推理链，过滤正确样本微调。
- ✓ **Cross-Agent:** SiriuS（Multi-Agent Bootstrapping). 建立 Experience Library，新手 Agent 模仿专家 Agent。

◘ Population-based & Evolutionary Methods：

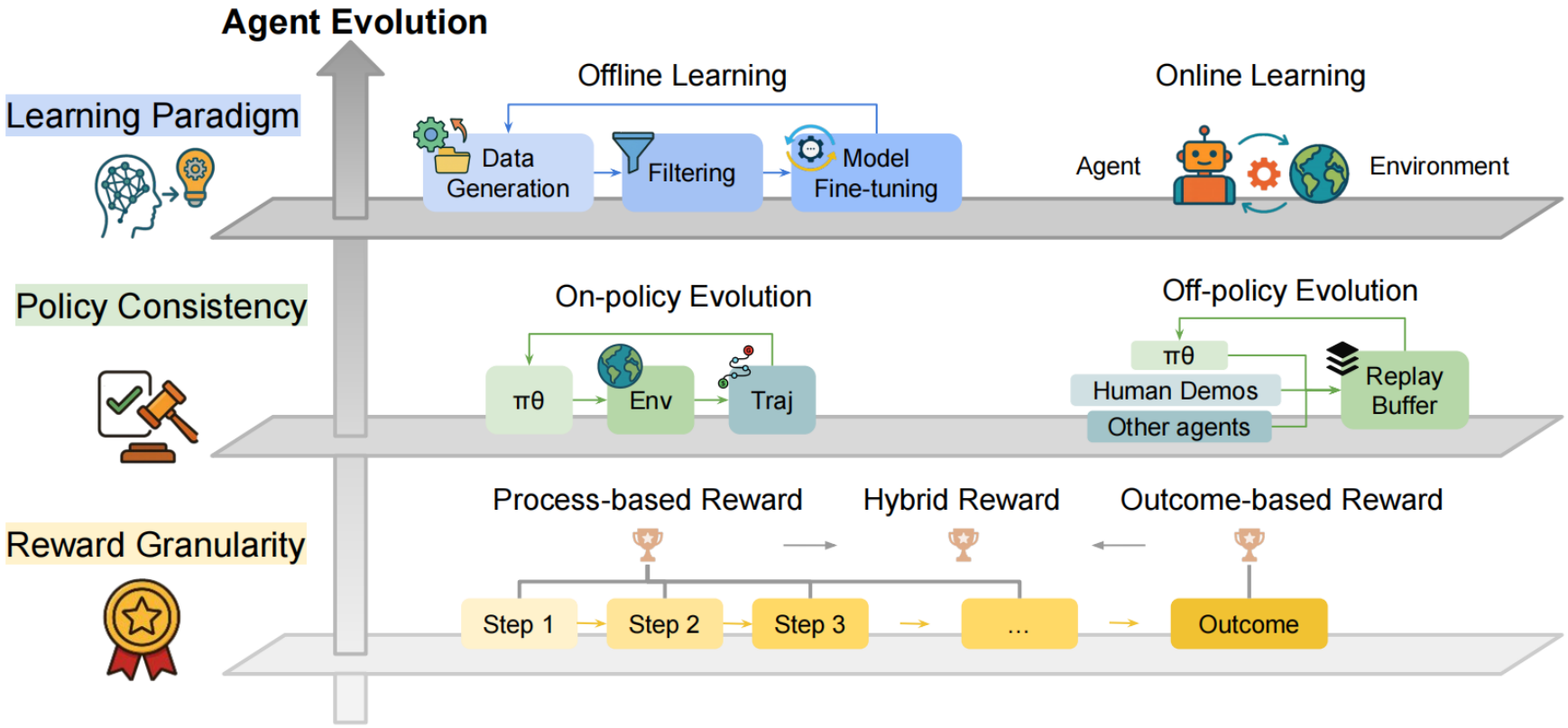
- ✓ **借鉴生物进化，维护 Agent 种群，进行 Selection, Mutation, Crossover。**
- ✓ **Single Agent Evolution:** Darwin Gödel Machine (DGM): 开放式进化，修改自身代码。GENOME: 模型权重的遗传算法优化。
- ✓ **Multi-Agent Evolution:** EvoMAC: 类似神经网络训练，将 Error 作为 Loss，反向传播修改 Agent 角色和 Prompt。Puppeteer: 进化“指挥官”策略，动态调度 Agent。

Table 4: Comparison of self-evolution method families along key dimensions.

Dimension	Reward-based	Imitation/Demonstration	Population-based
Feedback Type	Scalar reward, natural language, confidence, external signals	Demonstration trajectories, exemplars, rationales	Fitness scores, task success, competitive signals
Data Source	Self-generated, environment, external rules	Self-generated or other agents, humans	Population generations, multi-agent systems
Reward Granularity	Outcome/process/hybrid (flexible)	Usually outcome/process (via demo steps)	Often outcome-level, sometimes process via competition
Online/Offline	Both (reward learning, RL, DPO, SFT)	Typically offline, sometimes online demo mining	Online evolution or batch population updates
On/Off-policy	Both (DPO, Reflexion, GRPO)	Primarily off-policy, but online variants can be on-policy	Off-policy (population); self-play is on-policy
Sample Efficiency	Moderate (depends on reward sparsity)	High (if demo quality is high)	Usually low (needs many trials)
Stability	Sensitive to reward design	Sensitive to demo quality/diversity	Sensitive to population size/diversity
Scalability	Good with automation	Limited by demo collection	High but resource-intensive

How to Evolve (III) - 跨领域的演化维度

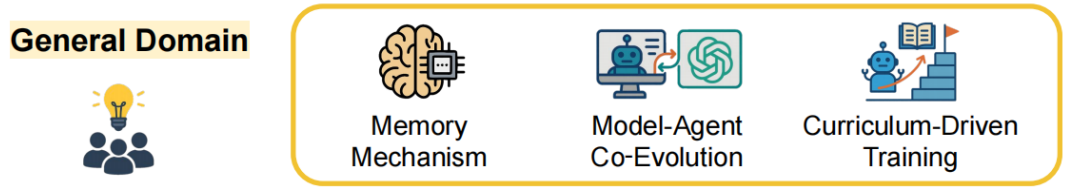
- **进化的交叉维度**：
  - ✓ **Online vs. Offline:** 实时交互学习 vs. 离线数据集学习。
  - ✓ **On-policy vs. Off-policy:** Reflexion (On-policy, 学自己) vs. DPO (Off-policy, 学历史/专家)。



Where to Evolve - 通用与垂直领域

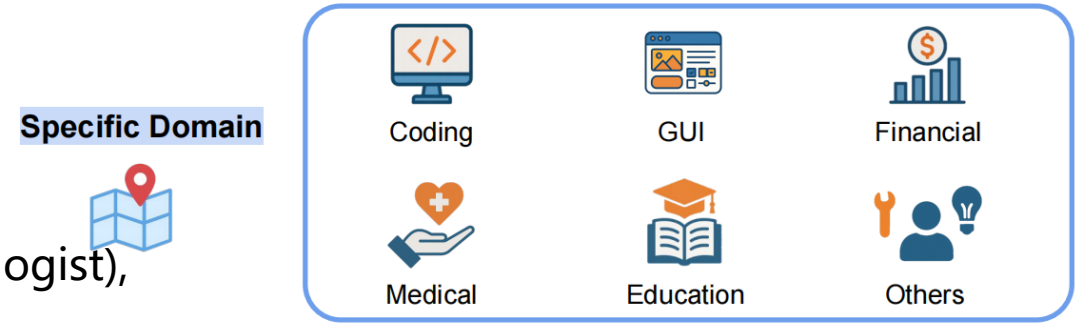
◘ General Domain (通用领域):

- ✓ **Memory Mechanism:** Mobile-Agent-E（Tips & Shortcuts).
- ✓ **Curriculum-Driven**: WebRL（自动生成由易到难的任务**).**
- ✓ **Model-Agent Co-Evolution:** UI-Genie（同时进化 Agent 和 Reward Model).



◘ Specialized Domain (垂直领域):

- ✓ **Coding:** SICA（自修补代码), EvoMAC .
- ✓ **GUI/Web:** Voyager , WindowsAgentArena .
- ✓ **Science (AI4S):** Arxiv Copilot , OriGene（Virtual Biologist), STELLA .
- ✓ **Medical:** Agent Hospital（进化医生 Agent).



评测与基准：动态框架

目标与指标

- 适应性 (Adaptability): 面对新规则的调整速度。
- 鲁棒性 (Robustness): 面对干扰或对抗攻击的稳定性。
- 长时学习: 终身学习过程中的知识累积与抗遗忘。

评测范式

- 静态评估: 传统的固定数据集 (Benchmark)。
- 短期自适应: 任务内多轮交互的性能提升曲线。
- 长期终身: 跨越数周/数月的持续学习能力跟踪。

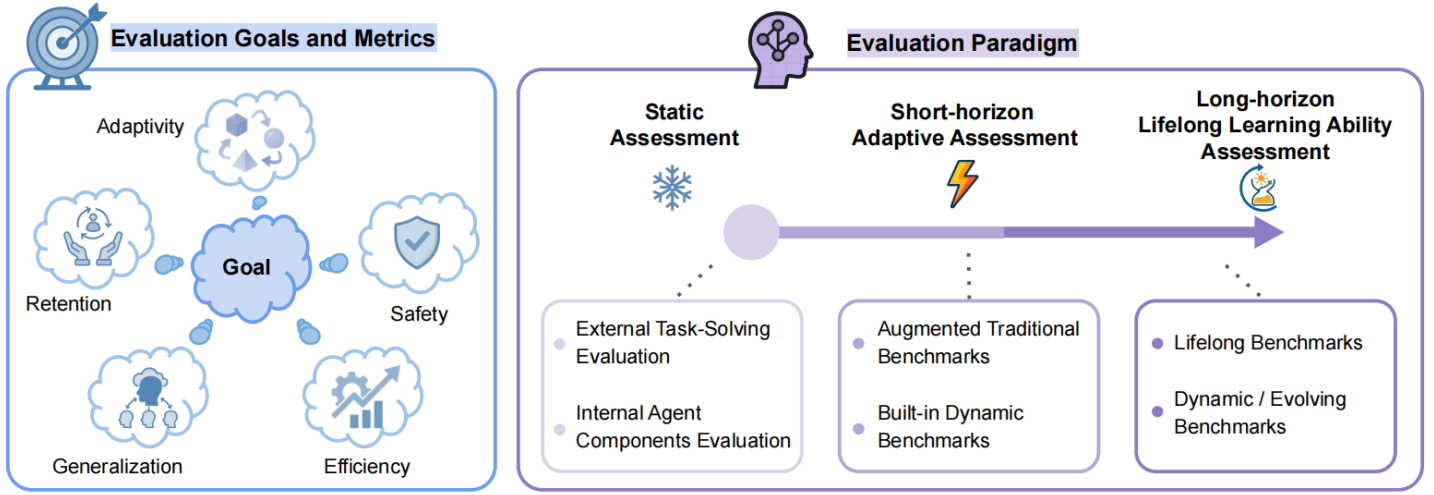
动态基准 (Self-Evolving)

为避免数据泄露与过拟合。

- 利用多代理系统重构现有样例，生成具有高置信度的新实例。
- 细粒度探测模型的子能力（如快捷方式偏见、查询多样性）。

“动态评测强调‘任务分布随时间演化’，避免系统过拟合静态基准，真实反映自进化与持续学习的长期能力。”

评估体系 (Evaluation)



1. 核心指标 (Goals & Metrics)

- **Adaptivity (适应性)**: 随迭代次数的成功率提升 (Success Rate by Iteration Steps).
- **Retention (保留能力 - CL 核心):**
  - ✓ Forgetting (FGT): 新任务学习后，旧任务性能下降幅度。$FGT_t = \frac{1}{t-1}\sum_{i=1}^{t-1}[\max_{j\in\{i\ldots t\}}(J_{j,i}) - J_{t,i}]$
  - ✓ Backward Transfer (BWT): 新任务学习对旧任务的正向增益。
- **Generalization (泛化):** OOD (Out-of-Distribution) 性能。
- **Safety & Efficiency:** Safety Score, Token Consumption.

2. 评估范式 (Paradigms)

- **Static Assessment:** 传统 Benchmark (AgentBench, GAIA).
- **Short-Horizon Adaptive:** 短期适应性 (ADAS 迭代曲线).
- **Long-Horizon Lifelong Learning (推荐关注):**
  - ✓ LifelongAgentBench80: 跨 DB/OS/KG 领域的连续任务流。
  - ✓ LTMBenchmark: 专注评估 Long-term Memory 的健壮性。

自进化智能体与持续学习 (CL) 的深度结合点

非参数化记忆 (Non-parametric Memory) 取代 Replay Buffer

- **CL 痛点：** 传统 CL 依赖 Replay Buffer 存储原始样本，受限于存储空间和隐私。
- **Agent 方案：** Memory Evolution (e.g., Expel, MemInsight) 实现了 **High-level Experience Replay**。Agent 存储的是规则 (Rules)、技能 (Skills) 和反思 (Reflections)，而非原始数据。这是一种更高级的 Knowledge Distillation，能更高效地抗遗忘。

模块化进化 (Modular Evolution) 解决可塑性-稳定性困境

- **CL 痛点：** Stability-Plasticity Dilemma。
- **Agent 方案：** Tool Creation (Voyager) 和 Architecture Search (AgentSquare) 提供了一种 **Add-on** 式的进化。通过增加外部工具库或调整模块组合来适应新任务，而冻结核心 LLM 参数，这天然实现了 Parameter Isolation，极大降低了灾难性遗忘风险。

主动课程学习 (Active Curriculum)

- **CL 痛点：** 传统 CL 无法控制任务到达顺序。
- **Agent 方案：** AgentGen 和 WebRL 展示了 Agent 可以主动生成适合当前能力的训练数据（Curriculum Learning），从而平滑学习曲线，这是被动 CL 无法做到的。

自进化范式在 AI for Science (AI4S) 的落地

实验室自动化 (Lab Automation) - Tool Evolution

- **痛点**：实验设备接口繁杂，新仪器不断引入。
- **Agent 方案**：利用 Voyager 式的 Tool Creation 机制。Agent 阅读新仪器说明书 -> 编写控制代码 (Python API) -> 在仿真/沙箱中试错 (Mastery) -> 存入 Tool Library。实现实验室能力的自主扩张。

科学假设迭代 (Hypothesis Refinement) - Reward-based Evolution

- **痛点**：科学探索空间巨大，试错成本高。
- **方案**：利用 Reflexion 式的 Verbal RL。Agent 提出假设 -> 设计实验 -> 获得湿实验/仿真反馈 (External Reward) -> 生成 Textual Critique -> 修正假设。
- **参考**：类似Coscientist，加入Explicit Self-Evolution循环，让 Agent 随着实验增多，越来越擅长提假设。

复杂方程求解与材料发现 - Evolutionary Architecture

- **痛点**：梯度优化易陷入局部最优。
- **方案**：利用 Darwin Gödel Machine 或 AlphaEvolve。在材料结构搜索或符号回归任务中，通过种群进化 (Population-based) 探索非梯度的算法空间，让 Agent 进化出针对特定物理系统的专用求解器代码。