arXiv:2601.07641v1 [cs.AI] 12 Jan 2026

# Beyond 静态工具：面向科学推理的测试时工具演化

**Jiaxuan Lu[1,†], Ziyu Kong[2,†], Yemin Wang[3,†], Rong Fu[4], Haiyuan Wan[1,5], Cheng Yang[6], Wenjie Lou[1], Haoran Sun[1], Lilong Wang[1], Yankai Jiang[1], Xiaosong Wang[1], Xiao Sun[1], Dongzhan Zhou[1,*]**

[1]上海人工智能实验室 [2]复旦大学[3]厦门大学 [4]澳门大学 [5]清华大学[6]杭州电子科技大学

## 摘要

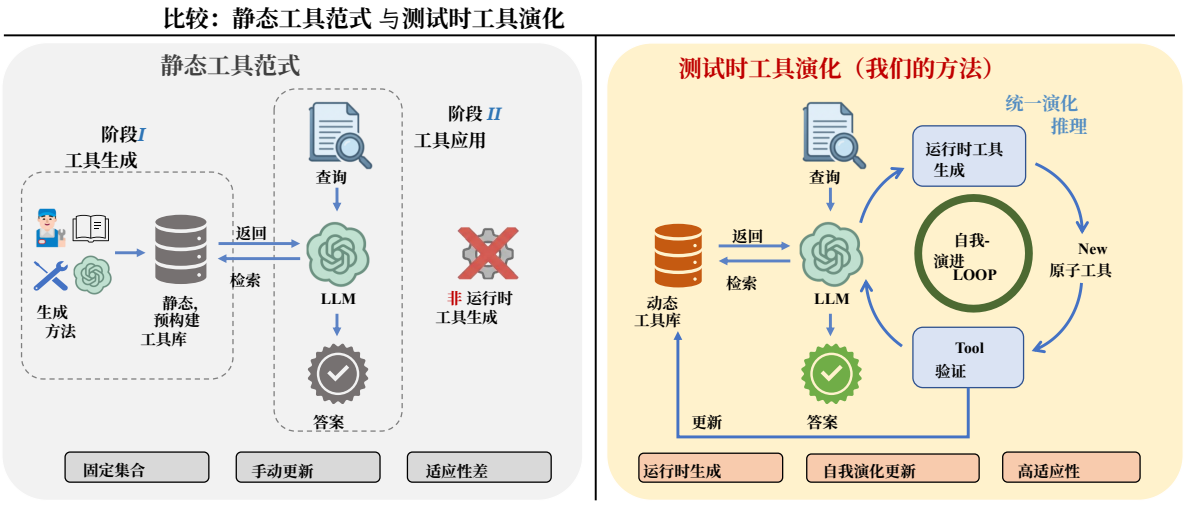
科学人工智能的核心挑战不只是推理，更在于能否在开放式的科学世界中创造计算方法。现有的基于大语言模型的智能体依赖静态、预定义的工具库；在工具稀缺、异构且本质上并不完备的科学领域中，这种范式从根本上难以奏效。本文提出测试时工具演化（TTE），这是一种新范式，使智能体能够在推理过程中合成、验证并演化可执行工具。通过将工具从固定资源转变为由问题驱动的产物，TTE 克服了静态工具库的僵化和长尾局限。为促进严格评估，我们推出了 SciEvo，这是一个包含 1,590 项科学推理任务、并由 925 个自动演化工具提供支持的基准测试。大量实验表明，TTE 在准确率和工具效率方面均达到最先进水平，同时能够有效实现计算工具的跨域适配。代码和基准测试已发布于 https://github.com/lujiaxuan0520/Test-Time-Tool-Evol。

## 1 引言

“科学人工智能”的终极目标，是构建能够驾驭物理世界无边复杂性的自主智能体：既能发现新颖的候选药物，也能推导物质的控制方程。虽然大语言模型（LLM）是强大的推理引擎（Brown 等人，2020年），但科学研究要求精确、可执行的严谨性，这本质上超出了大语言模型的概率性本质（Miret 和 Krishnan，2024年）。若没有计算工具的辅助，科学大语言模型的表现会受到显著限制（Yu 等人，2025年），在需要严格忠实性的任务中还会产生幻觉（Chen，2021年；Nijkamp 等人，2022）。

当前范式试图通过静态工具库弥合这一差距，即通过人工整理或离线合成构建预定义函数。尽管这一范式对标准化任务（例如天气、预订）行之有效，但在科学推理中却难以为继。我们指出了静态方法的两个致命瓶颈。首先，科学工具极其稀疏且异质。与通用领域中丰富的生态系统不同，科学函数分散且缺乏标准化，因此人工整理出全面的工具库在计算上难以实现。其次，也是最关键的一点，静态工具库无法预见新颖探究所需的定制计算原语，这使智能体只能充当被动的选择者，而非主动的发现者，从而人为限制了它们解决未见问题的潜力（Schick 等人，2023；Wan 等人，2025年）。

我们认为，图1(a) 所示的静态工具范式从根本上不适用于科学推理。要成为真正的科学家，智能体不能只是选择工具，还必须演化工具。不同于现有方法依赖有限的预定义工具库，我们提出了测试时工具演化（TTE），这一范式转变将科学推理从静态检索转向动态演化。TTE 不依赖僵化的工具库，而是在推理阶段按需合成可执行工具。通过动态地将复杂问题分解为原子函数并实时验证，TTE 确保工具库中的每个工具都与问题空间内在契合。我们在两项基础任务中实现了 TTE：从零开始合成工具（TTE-Zero），智能体在没有先验知识的情况下从头构建工具库来解决问题；以及跨领域工具适配（TTE-Adapt），智能体动态地重新利用源工具库（例如，材料科学）来攻克新领域（例如，化学）。




（a）现有静态工具范式　　（b）提出的测试时工具演化

图1：范式比较：静态工具范式（左）与测试时工具演化（右）。静态方法需要预先收集工具库，因此覆盖范围和领域适应性受限。我们的测试时演化从空工具库开始，并在问题求解过程中按需生成工具，从而能够持续适应新的领域和问题。

我们的贡献总结如下：

1. 我们提出了**测试时工具演化**，这是一种新颖的框架，模拟了科学方法的迭代特性。通过允许工具在推理过程中生成、验证和演化，TTE 克服了静态范式固有的僵化。

2. 我们发布了**SciEvo**，这是一个全面的工具演化评测基准，包含1,590个科学评测实例，并由一个包含925个演化工具的工具库提供支持。

3. 大量评测表明，TTE 在科学推理方面达到了新的最先进水平（SOTA）。具体而言，TTE-Zero 在准确率和工具利用效率方面均显著优于现有基线方法，而 TTE-Adapt 则实现了有效的跨领域工具适应，展现了计算原语在不同科学学科间的可迁移性。

## 2 相关工作

### 2.1 静态工具范式

通过外部工具增强大语言模型的范式，使其能力超越了静态的参数化知识。奠基性研究确立了这种交互的机制，例如，ReAct（Yao等人，2022）提出将推理轨迹与工具操作交替进行，而 Toolformer（Schick等人，2023）表明，大语言模型可以通过自监督微调学会使用计算器和搜索 API。在这些执行框架的基础上，后续研究专注于扩展工具空间。Gorilla（Patil等人，2024年）和 Tool-LLM（Qin 等人，2024年）等系统采用指令微调和基于检索的机制，从海量的预定义 API 库中选择合适的工具，例如，HuggingFace 或 RapidAPI，从而使模型能够处理各种通用领域的查询。

静态工具范式已广泛应用于专业科学领域，以应对特定领域任务的复杂性。在化Bran 等人，2024年），CheMatAgent（Wu 等人，2025年）和 ChemMAS（Yang 等人，2025年）等系统整合了一系列由专家策划的固定工具，涵盖从简单计算器到复杂合成规划器，旨在实现有机合成和药物发现的自动化。其他方法则着重通过整合知识库来增强领域能力，例如 HoneyComb（Zhang 等人，2024年），或利用多智能体框架发掘潜在的跨学科关系，如 SCP（Jiang 等人，2025年）所示。最后，近期研究对这些静态工具集的影响进行了严格基准测试，例如 ChemToolAgent（Yu 等人，2025年）和 MatTools（Liu 等人，2025年）。尽管这些系统在边界明确的场景中表现有效，但它们有一个共同的

关键局限，即，它们依赖预先定义的静态工具库，无法全面覆盖开放式任务空间。

### 2.2 动态工具合成

为了解决静态工具库覆盖范围有限的问题，近期研究转向使大语言模型能够动态生成工具。CREATOR（Qian 等人，2023）和 CRAFT（Yuan 等人，2024年）等方法利用大语言模型的代码生成能力，通过抽象推理合成定制工具，以解决特定问题。然而，这些方法通常将工具生成视为一次性过程；而 LATM（Cai 等人，2024年）和 ToolMaker（Wölflein 等人，2025年）则采用解耦范式，将工具制作阶段与推理分离，从而阻碍了实时适配。

为了超越静态生成，Voyager（Wang 等人，2023）等系统引入了不断演化的技能库这一概念，使智能体能够在具身环境中通过反复试错，将可执行代码积累为工具。类似地，SEAgent（Sun 等人，2025年）和 ToolACE-DEV（Huang 等人，2025年）研究了用于操作系统控制的自我演化机制。尽管前景可期，这些演化框架面向游戏化或通用计算机任务，缺乏科学推理所需的严谨性和领域专属逻辑。与此同时，自动化设计方法（Hu 等人，2024年；Shang 等人，2024年）探索在模块化设计空间中搜索最优智能体架构，关注的是组件的排列方式，而非工具本身的演化。

## 3 测试时工具演化

### 3.1 问题定义

我们将测试时工具演化 （TTE）形式化为一种全新的范式，旨在弥补现有静态工具范式中的关键缺口。不同于在问题求解前离线准备工具的现有方法，TTE 允许在问题求解过程中生成并演化工具，这标志着工具生态系统从静态向动态的范式转变。

形式化地说，给定一系列在测试时依次到达的科学问题 $\mathcal{P} = \{P_1, P_2, \ldots, P_t\}$，TTE 的目标是维护一个不断演化的工具库 $L_t$，在能力与效率之间取得平衡。我们将这一任务建模为一个在线优化问题，其中系统旨在最大化累积效用：

$$\max_{\{L_t\}_{t=1}^{T}} \sum_{t=1}^{T} \left(\mathbb{I}(\text{Solved}(P_t, L_t)) - \lambda \cdot |L_t|\right), \quad (1)$$

其中 $\mathbb{I}(\cdot)$ 是用于问题求解的指示函数，例如，准确率，而 $\lambda$ 是用于惩罚工具库扩张的正则化系数。参数训练。我们针对两项主要任务实例化 TTE：用于从头工具合成的 TTE-Zero（$L_0 = \emptyset$），以及用于将预定义工具库跨域适配到新目标领域的 TTE-Adapt。由于工具组合具有组合性质，求解式 1 的全局最优解在计算上难以实现，因此 TTE 框架采用贪心演化策略。在每个步骤 $t$，系统通过工具生成和剪枝机制将 $L_t$ 更新 $L_{t+1}$，以近似最优演化轨迹，而无需显式进行参数训练。

### 3.2 架构概述

如图2所示，我们的框架实现了一个闭环演化工作流，由五个集成模块组成。结构化任务分解将复杂查询分解为可执行的子目标。动态工具检索从工具库中检索现有工具。当检索失败时，生成式工具合成会按需创建新工具。原子工具优化对新工具进行解耦、验证和注册，从而使工具库不断演化。运行时执行引擎执行工具序列，以推导出最终答案。所提出的架构能够在解决现实世界问题的同时，从空工具库状态开始实现工具库的持续增长。

### 3.3 结构化任务分解

问题分析器 充当规划引擎，将科学问题分解为一系列可执行的子目标操作。给定问题 $P$，它将识别所需操作的集合 $\mathcal{O}$：

$$\begin{aligned} \mathcal{O} &= \text{Analyze}(P) \\ &= \{\, O_i : O_i \text{ is required to solve } P \,\}. \end{aligned} \quad (2)$$

该分解过程具备工具感知能力，能够分离出需要计算干预的特定子目标，为工具检索过程做好准备。

### 3.4 动态工具检索

对于每个已识别的操作 $O_i$，系统都会查询动态工具注册表。我们会验证是否存在

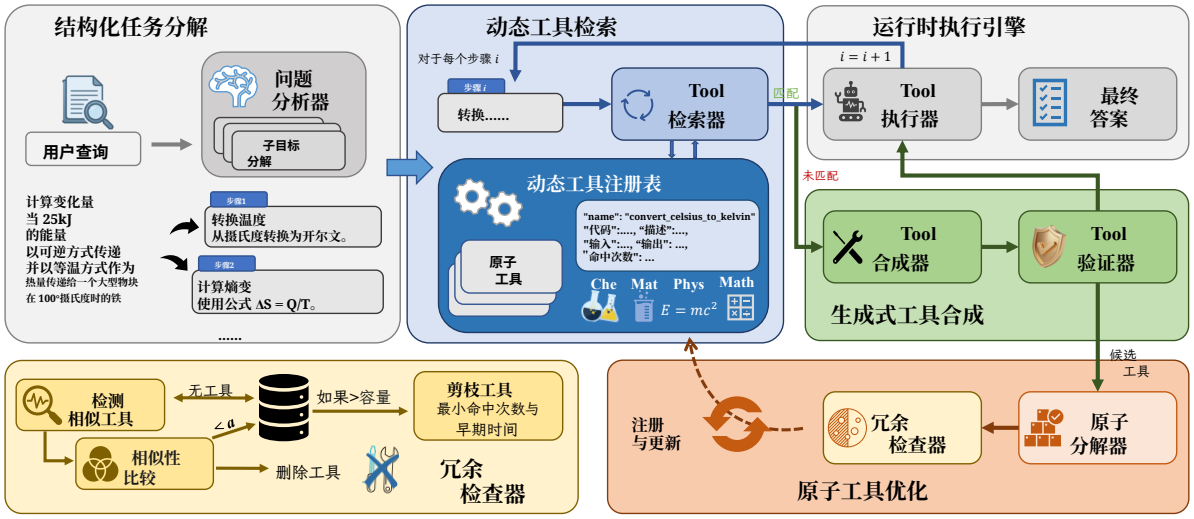



图2：测试时工具演化（TTE）框架的架构。该系统通过一个包含五个集成阶段的闭环工作流运行。(1) 结构化任务分解：问题分析器将复杂的科学查询分解为一系列可执行的子目标。(2) 动态工具检索：系统查询动态工具注册表，以查找现有的原子工具。如果检索失败，则触发 (3) 生成式工具合成：工具合成器即时创建候选工具，并由工具验证器进行严格验证。(4) 原子工具优化：原子分解器将已验证的工具拆解为可复用的原子单元，冗余检查器对其进行筛选，并将其注册以更新工具库。(5) 运行时执行引擎：为所有步骤成功检索或生成所需工具后，工具执行器将执行该序列以合成最终答案。

合适的工具，方法是比较其文本描述的语义相似度：

$$\text{sim}(O_i, T_j) = \cos(\text{embed}(d_{O_i}), \text{embed}(d_{T_j})), \tag{3}$$

其中 $d$ 表示函数描述。系统根据当前工具库 $L$中找到的最高相似度分数作出分支决策：

$$T^* = \begin{cases} \arg\max_{T_j \in L} \text{sim}(O_i, T_j), & \text{if } s_{\max} \geq \tau, \\ \text{Generate}(O_i, P), & \text{otherwise}, \end{cases} \tag{4}$$

其中 $s_{\max} = \max^{T_j \in L} \text{sim}(O_i, T_j)$，$\tau$是使 F1 分数达到最大的阈值。工具检索器在利用与探索之间取得平衡，确保高效复用现有工具（“匹配”路径），同时针对新需求自动触发合成流程（“未匹配”路径）。

### 3.5 生成式工具合成

当检索失败时，生成式工具合成模块会通过严格的生成-验证流程创建新工具。给定 $P$和 $O_i$，工具合成器 通过 $T_{\text{proposed}}$ 提出一个工具思维链推理：

$$P(T_{\text{proposed}} \mid P, O_i) = \prod_{k=1}^{K} P(f_k \mid P, O_i, f_{1:k-1}), \tag{5}$$

其中，$f_k$ 表示函数签名和实现等组成部分，$K$ 表示生成步骤的总数。工具验证器通过语法检查、执行测试和领域验证来确保正确性：

$$P(\text{valid} \mid T_{\text{proposed}}) = P_{\text{syntax}} \cdot P_{\text{exec}} \cdot P_{\text{domain}}. \tag{6}$$

只有通过所有验证检查的工具才能进入精炼阶段。

### 3.6 原子工具优化

为确保工具库不断演进并积累高质量、可复用的资产，有效工具在注册前需经过原子优化。原子分解器首先将复杂的生成工具拆解为基本的“单元工具”。分解过程形式化表示为：

$$\{A_1, \ldots, A_k\} = \text{Decompose}(T), \tag{7}$$

以最大化预期复用改进$\mathbb{E}[R_{\text{atomic}}]$:

$$\mathbb{E}[R_{\text{atomic}}] \geq k \cdot \mathbb{E}[R(T)] \cdot p_{\text{partial}}, \tag{8}$$

其中， $R(\cdot)$表示复用效用函数， $k$表示由分解得出的原子组件数量，$p_{\text{partial}}$ 表示未来问题只需要部分函数的概率。直观而言，单体工具缺乏灵活性。将 $T$ 分解为 $k$ 个原子单位，可实现部分复用，使未来查询能够独立调用特定子函数（$p_{\text{partial}}$）；这种灵活性使分解后的集合比单一的僵化工具产生更高的累积效用。

该冗余检查器 充当把关者。新的原子函数 $A_{new}$ 会与工具库进行比较：

$$A_{new} \in L_{t+1} \Leftrightarrow \max_{A_i \in L_t} \text{sim}(A_{new}, A_i) < \tau. \quad (9)$$

与此同时，动态工具注册表的维护者通过清除使用率较低的工具来维持工具库的效率；当容量 $C$ 超出限制时，这能确保工具库保持精简且切合需求：

$$L_{t+1} = L_t \setminus \{A_i : u(A_i) < \theta_{min} \wedge |L_t| > C\}, \quad (10)$$

其中， $u(A_i)$表示工具 $A_i$的历史使用次数，而 $\theta_{min}$ 是最低使用阈值。

### 3.7 运行时执行引擎

成功检索或生成所需工具后，工具执行器 会将它们整合到最终推理过程中。解答合成形式化表示为 $S = \text{Solve}(P, L_t)$。整个框架形成闭环，运用工具库不断演进的能力，为用户查询生成最终答案 $S$。

## 4 TheSciEvo 基准

### 4.1 基准构建

SciEvo 的一个显著特征是其演化式构建范式。与从静态代码库中整理而成的工具库不同，SciEvo 的工具通过 TTE 框架从零开始引导生成，确保每个工具都能切实满足真实的科学推理需求。

**种子数据来源**。为构建稳健且多样化的种子环境，我们整合了来自三个不同来源的高质量科学问题，包括 SciEval（Sun 等人，2024年）、SciBench（Wang 等人，2024年）以及一个专注于专业领域计算的专有材料科学数据集。我们明确筛选需要

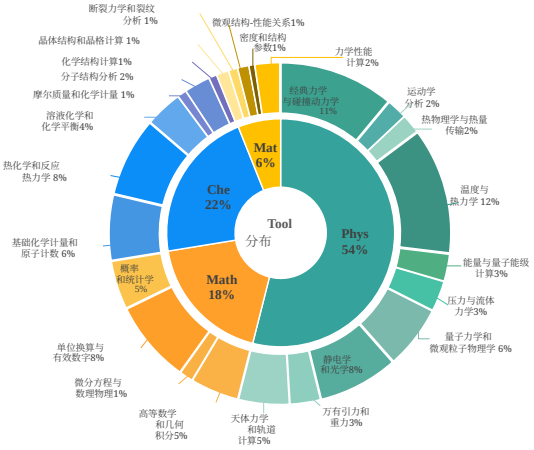



图3：经过整理的 SciEvo 基准测试中的工具分布。SciEvo 涵盖四大科学领域的 25 个子学科：物理学（499 个工具）、化学（192 个）、数学（171 个）和材料（63 个），展现了其对各种科学计算需求的全面覆盖。

多步推理和精确数值解的计算问题，并剔除纯粹的知识检索查询。为确保所选问题涵盖广泛的科学场景，我们采用基于语义聚类的分层抽样策略。具体而言，我们使用嵌入模型（Reimers 和 Gurevych，2019）对所有候选问题进行嵌入，并执行 K 均值聚类，随后从每个簇中均匀抽取实例，以最大限度地提高种子集中的问题多样性。这些问题与解答对提供了可靠工具验证所需的问题上下文（$Q$）和真实值验证信号（$A$）。

**工具库合成**。 我们使用 TTE-Zero 框架引导构建 SciEvo 工具库。通过为智能体初始化一个空工具库，并依次向其提供种子问题，我们模拟了“白板”学习过程。智能体动态生成、执行并验证 Python 函数。只有成功协助推导出正确真实答案的原子函数，才会被永久收录到代码库中。整个过程最终构建出一个经过验证、包含 925 个原子工具的工具库，确保工具集与问题空间 100% 对齐。

### 4.2 分类体系与统计

为便于进行细粒度分析，我们采用混合分类策略，将合成工具组织成层级分类体系。

**领域分类**。我们对生成的工具描述的向量嵌入进行主成分分析（PCA），以识别潜在语义簇。随后，博士级领域专家对这些簇进行审查和完善，建立了一个精确的分类体系，其中包含 25 个子学科，涵盖物理学（10 个）、化学（6 个）、材料科学（5 个）和数学（4 个），如图3所示。该体系确保分类既能体现计算语义，也能涵盖典型的科学区分。

**数据分布**。 完整的 SciEvo 基准包含 1,590 个评估实例，由一个包含 925个演化工具的工具库提供支持。领域专用工具分布涵盖四个主要学科：物理学工具数量最多，共 499 个；其次是化学（192 个）、数学（171 个）和材料（63 个）。如图 3所示，多样化的组成确保了对科学计算基本组件的全面覆盖。

### 4.3 评估指标

为模拟真实的资源限制，所有评估均在工具库容量上限为 $C = 500$的条件下进行。在此设置下，我们遵循标准流程，使用准确率（Acc）评估性能（Wang 等人，2024年；Sun 等人，2024年）。为量化演化工具库 $\mathcal{T}$ 的效用和泛化能力，我们还为 TTE-Zero 定义了工具复用率（TRR@$k$），即至少成功复用 $k$ 次的工具所占比例：

$$\text{TRR@}k = \frac{|\{t \in \mathcal{T} \mid h(t) \geq k\}|}{|\mathcal{T}|}, \quad (11)$$

其中， $h(t)$表示工具 $t$的命中次数。我们报告多个递增阈值下的 TRR@$k$，以衡量不同程度的效用，即，TRR@1 衡量至少使用过一次的非冗余工具所占比例，而 TRR@5 和 TRR@10 则用于识别核心科学原语的涌现。

对于跨领域评估即TTE-Adapt，我们将工具库总量 $\mathcal{T}$分解为预定义集合 $\mathcal{T}_{pre}$ 和新演化集合 $\mathcal{T}_{new}$。我们引入两项分层指标，以区分能力的来源：

$$\text{TRR}_{evol}\text{@}k = \frac{|\{t \in \mathcal{T}_{new} \mid h(t) \geq k\}|}{|\mathcal{T}_{new}|}, \quad (12)$$

$$\text{TRR}_{trans}\text{@}k = \frac{|\{t \in \mathcal{T}_{pre} \mid h(t) \geq k\}|}{|\mathcal{T}_{pre}|}, \quad (13)$$

其中，$\text{TRR}_{evol}$@$k$ 是衡量适应效率的主要基准指标。$\text{TRR}_{evol}$ 越高，表示性能越好，说明系统已成功将新的领域知识整合为高质量、可复用的基本工具，而非生成一次性脚本。相反，$\text{TRR}_{trans}$ 用于监测先验知识的替代情况。在跨领域设置中，通常希望$\text{TRR}_{trans}$ 较低，因为这表明负迁移有所缓解，即丢弃了无关工具即；但该值仍须非零，以确保保留基础的领域通用能力。

## 5 实验

### 5.1 实验设置

**数据集**。 我们在三个不同的基准测试上评估框架，以考察问题求解准确率和工具演化效率，包括 SciBench（Wang 等人，2024年）、SciEval（Sun 等人，2024年）以及经过整理的 SciEvo 数据集。

**基线方法**。 我们将 TTE-Zero 与五种具有代表性的基线方法进行比较，这些方法分为两种范式。为评估不借助外部工具支持时的基础推理能力，我们采用 Basic-COT（思维链）和 Basic-POT（程序思维）。对于使用工具的智能体框架，我们将其与 Creator（Qian 等人，2023）、KTCE（Ma 等人，2025年）和 CheMatAgent（Wu 等人，2025年）进行比较。在 TTE-Adapt 设置下，我们将其与“无工具”基线和“仅源域”基线进行比较，以分离归因于领域专用工具演进的性能提升。

### 5.2 实现细节

**模型架构**。 我们使用三种大语言模型评估该框架，包括 GPT-4o、Qwen2.5-7B-Instruct 和 GPT-3.5-turbo。除非另有说明，主要实验结果均基于 GPT-3.5-turbo 得出，采样温度设为 0。3，以兼顾多样性与确定性。

**检索与排序**。 我们使用 bge-m3（Chen 等人，2024年）构建稠密检索流水线，以生成嵌入，并使用 bge-reranker-v2-m3 进行重排序。对于每个子目标，系统会检索排名前$k$ （$k = 3$）的相关工具，以提供聚焦的上下文。

| 方法 | SciBench | | | | SciEval | | | | SciEvo | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TRR@1 | TRR@2 | TRR@5 | TRR@10 | TRR@1 | TRR@2 | TRR@5 | TRR@10 | TRR@1 | TRR@2 | TRR@5 | TRR@10 |
| KTCE | 0.17 | 0.10 | 0.05 | 0.02 | 0.06 | 0.04 | 0.03 | 0.02 | 0.31 | 0.20 | 0.09 | 0.04 |
| Creator | 0.12 | 0.06 | 0.02 | 0.01 | 0.03 | 0.02 | 0.02 | 0.01 | 0.17 | 0.07 | 0.04 | 0.02 |
| CheMatAgent | 0.43 | 0.27 | 0.18 | 0.13 | 0.20 | 0.16 | 0.09 | 0.05 | 0.62 | 0.42 | 0.28 | 0.17 |
| **TTE-Zero** | **0.89** | **0.71** | **0.40** | **0.21** | **0.48** | **0.35** | **0.15** | **0.05** | **0.99** | **0.94** | **0.66** | **0.41** |

表1：工具复用的层次分析（工具复用率@$k$）。我们报告了阈值为 $k = \{1, 2, 5, 10\}$时的工具复用率。TTE-Zero 在 SciEvo 上实现了近乎完美的利用率（工具复用率@1 $\approx 1.0$），并在更严格的阈值（$k = 5$，10）下始终保持较高的复用率，而基线方法无法生成高频核心基元。

| 方法 | SciBench | SciEval | SciEvo |
|---|---|---|---|
| Basic-COT | 0.27 | 0.18 | 0.33 |
| Basic-POT | 0.31 | 0.21 | 0.36 |
| Creator | 0.27 | 0.22 | 0.49 |
| KTCE | 0.37 | 0.24 | 0.55 |
| CheMatAgent | 0.34 | 0.23 | 0.56 |
| **TTE-Zero** | **0.45** | **0.30** | **0.62** |

表2：各基准测试的准确率比较。TTE-Zero始终优于所有基线方法。

**工具演化与去重**。 为确保工具库紧凑高效，并将容量限制在 $C = 500$以内，我们采用严格的语义去重。我们使用 CodeBERT（Feng 等人，2020年）计算候选工具与现有库条目之间的语义相似度。如果新工具与任何现有工具的最大余弦相似度超过阈值 $\tau = 0.8$，则会被严格拒绝。

**评估协议**。 由 GPT-4.1-nano 裁判验证最终答案的正确性。对于数值结果，我们采用 $10^{-5}$ 的相对容差；对于符号表达式，则要求完全符合规范匹配。评估指标方面，我们报告解答正确性的准确率（Acc），以及用于量化工具库演化质量的工具复用率（TRR@$k$、TRR$_{trans}$@$k$ 和 TRR$_{evol}$@$k$）。

## 6 结果与分析

### 6.1 TTE-Zero的性能

在此设置下，智能体从空工具库（$L_0 = \emptyset$）开始，以评估其合成科学原语并解决实际问题的能力。

**科学基准测试的比较分析**。我们首先评估 TTE-Zero 在三个基准测试中相较于最先进水平基线方法的性能。如表 2所示，TTE-Zero 始终创下新的最先进性能。在 在 SciBench 数据集上，TTE-Zero 的准确率达到 0.45，显著超过表现最强的基线方法 KTCE（0.37）和领域专用智能体 CheMatAgent（0.34）。在我们提出的 SciEvo 基准测试中，性能优势进一步扩大：TTE-Zero 的准确率达到 0.62，而 CheMatAgent 为 0.56，KTCE 为 0.55。这些结果表明，测试时演进工具相较于静态或基于检索的范式具有明显优势，尤其是在需要多步推理的复杂科学问题上。值得注意的是，TTE-Zero 的表现大幅优于标准提示策略（Basic-COT 和 Basic-POT），例如，在 SciEvo 上比 Basic-COT 高出 +0.29，验证了外部工具支持的必要性。

**工具演化质量分析**。 为了解性能提升究竟源于高效的工具利用，还是单纯的暴力生成，我们分析了工具复用率（TRR）。表1 展示了分层复用统计。一个关键发现是，TTE-Zero 在 SciEvo数据集上的工具利用率近乎完美，TRR@1 达到 0.99，这表明几乎每个生成的工具都成功复用于解决目标问题，从而最大限度地减少计算浪费。相比之下，Creator（TRR@1= 0.17）和 KTCE（TRR@1 = 0.31）等基线方法存在严重冗余，大多数离线生成的工具从未被使用。此外，TTE-Zero 在整合“科学原语”方面表现出更强的能力。在更严格的阈值 $k = 10$ 下，其在 SciEvo 和 SciBench 上的复用率分别维持在 0.41 和 0.21，而 Creator 则降至接近零（分别为 0.02 和 0.01）。这证实了 TTE-Zero 并非只是将工具库塞满，而是会主动演化出高效用、可复用的工具。

**消融研究**。 我们通过比较子目标分解模块的贡献，方

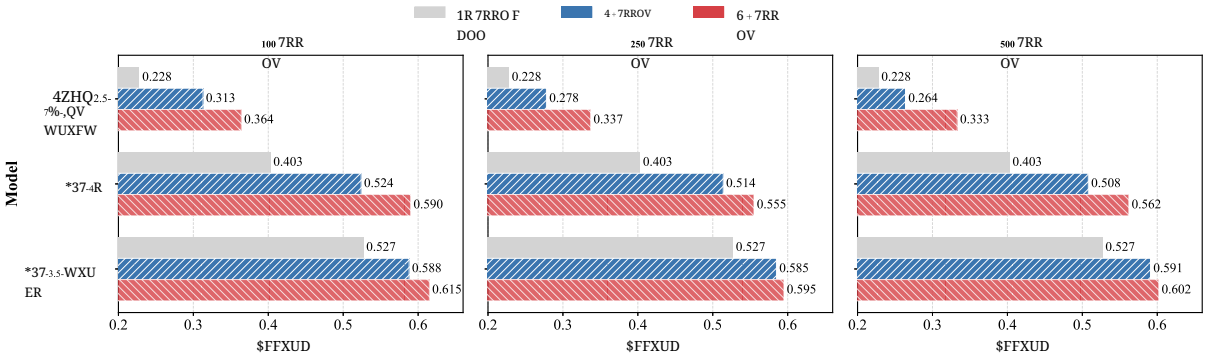



图4：SciEvo 上的准确率比较。我们将“无工具调用”基线与我们的 TTE-Zero方法进行比较，后者采用直接查询（“Q + 工具”）和子目标分解（“S + 工具”）。

| 方法 | 适配：材料 →化学 | | | | | | | | | 适配：材料 →物理学 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 准确率 ↑ | 工具复用率$_{trans}$ (↓) | | | | 工具复用率$_{evol}$ (↑) | | | | 准确率 ↑ | 工具复用率$_{trans}$ (↓) | | | | 工具复用率$_{evol}$ (↑) | | | |
| | | 1 | 2 | 5 | 10 | 1 | 2 | 5 | 10 | | 1 | 2 | 5 | 10 | 1 | 2 | 5 | 10 |
| 无工具 | 0.535 | - | - | - | - | - | - | - | - | 0.535 | - | - | - | - | - | - | - | - |
| 仅源域 | 0.561 | 0.26 | 0.13 | 0.04 | 0.02 | - | - | - | - | 0.585 | 0.38 | 0.20 | 0.03 | 0.01 | - | - | - | - |
| **TTE-Adapt** | **0.595** | **0.23** | **0.10** | **0.01** | **0.00** | **0.24** | **0.11** | **0.02** | **0.01** | **0.618** | **0.25** | **0.11** | **0.01** | **0.01** | **0.32** | **0.16** | **0.03** | **0.01** |

表 3：跨域适应（源领域：材料）上的性能。我们报告了 $k \in \{1, 2, 5, 10\}$ 处的准确率和工具复用率（TRR）。TRR$_{trans}$ 表示保留的源工具（数值越低越好，可缓解负迁移），而 TRR$_{evol}$ 表示新增的目标工具（数值越高越好，有助于知识巩固）。

式是比较两种 TTE 变体：“Q+Tools”（使用原始查询）和“S+Tools”（使用子目标分解），并与“无工具调用”基线进行对比。如图 4所示，在所有评估的模型（包括 Qwen2.5-7B、GPT-4o 和 GPT-3.5-turbo）中，两种工具增强设置的表现均优于“无工具调用”基线。关键的是，“S+Tools”策略始终取得最高准确率。例如，在 Qwen2.5-7B 上，当工具库规模为 100 时，“S+Tools”的表现明显优于“Q+Tools”（0.364 对 0.313），这验证了将复杂的科学查询拆解为细粒度子目标对于精准的工具检索与执行至关重要，进而最大限度地发挥演化工具库的效能。

### 6.2 TTE-Adapt 的性能

我们通过使用预定义的工具库（例如，材料）初始化 TTE-Adapt 框架，并将其适配到新的目标领域（例如，化学和物理学），以评估其可塑性。

**跨域适应。** 表 3 展示了适配性能。TTE-Adapt 始终优于“无工具”和“仅源域”基线方法，在两种设置下都取得了显著的准确率提升。性能提升得益于自适应替换机制，系统通过该机制有效缓解了负向迁移——具体而言，系统会剪除无关的源工具，从而将化学领域的 TRR$_{trans}$@1 从 0.26 降至 0.23；同时，它还会将新知识整合为可复用的基本单元，这一点从演化工具的显著贡献中可见一斑（化学领域的 TRR$_{evol}$@1 $=$ 0.24，以及物理学领域的 0.32）。这一动态调整过程证实，TTE-Adapt 成功重塑了工具分布，使其与目标领域特有的推理模式相适应。

## 7 结论

在本研究中，我们指出并解决了静态工具范式在科学推理中的根本局限。通过引入测试时工具演化（TTE），我们将大语言模型智能体的角色从被动的工具选择者转变为主动的工具创造者。TTE 使智能体能够在推理过程中合成、验证并演化计算原语，确保工具空间在本质上与无界的科学问题空间保持同构。我们的广泛评估证实，TTE 不仅在推理准确率方面达到了新的最先进水平，还能在不同领域实现稳健的工具适应。我们认为，赋予智能体自主演化工具的能力，是实现新一代通用科学人工智能的先决条件。

## 8 局限性

尽管测试时工具演化（TTE）为科学推理引入了一种颇具前景的范式，我们仍承认当前框架存在若干固有局限。

**推理延迟与计算成本**。与静态的基于检索的方法不同，TTE 需要在推理过程中合成并验证工具。与简单的工具选择相比，动态演化过程不可避免地会带来更高的计算开销和更长的延迟。未来的研究可以探索轻量级“元模型”，用于预测是否需要工具，从而跳过对简单查询的演化过程。

**对基础大语言模型代码生成能力的依赖**。TTE 的有效性从根本上受骨干大语言模型代码生成能力的限制。我们的实验表明，使用高容量模型可实现最先进性能。然而，使用较小、能力较弱的开源模型（例如， <7B 参数）时，性能会下降，因为这类模型难以生成语法正确的 Python 基元。

**开放式演化中的安全性与沙箱隔离**。允许智能体在测试时生成并执行任意代码会带来潜在的安全风险，尤其是在开放式科学探索中，生成的脚本可能会无意间消耗过多资源或尝试执行不安全的操作。尽管我们的实验是在严格的沙箱环境中进行，并受到超时约束，但要将 TTE 扩展到自主的现实世界系统，还需要比简单语法检查更稳健的语义级安全验证协议。

## 9 伦理声明

我们认识到，动态生成科学工具可能带来双重用途风险，尤其是在化学或材料科学等领域，代码可能被滥用于有害用途（例如毒素合成）。为降低这些风险，我们在公开发布前对整个演化工具库进行了严格的人工审查。我们严格遵循负责任的披露规范，并确保发布的产物中不包含任何可用于有害用途的工具。

关于本研究发布的产物，即即SciEvo 基准，我们确认所有数据来源均为公开的科学数据库，且符合其预期的研究用途。我们进行了筛查，以确保其中不包含个人身份信息（PII）或冒犯性内容。代码和数据采用 MIT 许可证发布，以促进开放研究，同时限制恶意使用。我们承认，该系统可能会反映底层大语言模型和科学文献中存在的偏见；用户在商业部署前应核实关键计算结果。最后，我们承认仅使用了人工智能助手（例如ChatGPT）进行语言润色。所有科学主张、实验设计和数据分析均为我们的原创工作。

## 参考文献


Tamer Alkhouli、Katerina Margatina、James Gung、Raphael Shu、Claudia Zaghi、Monica Sunkara 和 Yi Zhang。2025年。CONFETTI：通过轮级交互进行对话式函数调用评测。载于计算语言学协会年会论文集，第 7993–8006 页。

Kinjal Basu、Ibrahim Abdelaziz、Kiran Kate、Mayank Agarwal、Maxwell Crouse、Yara Rizk、Kelsey Brad-ford、Asim Munawar、Sadhana Kumaravel、Saurabh Goyal 等。2025年。Nestful：用于评估大语言模型处理嵌套 API 调用序列能力的基准测试。载于自然语言处理实证方法会议论文集，第 33526–33535 页。

M. Bran、A. Cox、O. Schilter、C. Baldassari、A. D. White 和 P. Schwaller。2024年。利用化学工具增强大语言模型。自然机器智能，6：525–535。

Tom Brown、Benjamin Mann、Nick Ryder、Melanie Subbiah、Jared D Kaplan、PrafullaDhariwal、Arvind Neelakantan、Pranav Shyam、Girish Sastry、Amanda Askell 等人。2020年。《语言模型是少样本学习者》。《神经信息处理系统进展》，33：1877–1901。

Tianle Cai、Xuezhi Wang、Tengyu Ma、Xinyun Chen 和 Denny Zhou。2024年。《大语言模型作为工具制造者》。载于国际学习表征会议。

Jianlv Chen、Shitao Xiao、Peitian Zhang、Kun Luo、Defu Lian 和 Zheng Liu。2024年。M3-embedding：通过自知识蒸馏实现多语言、多功能、多粒度文本嵌入。发表于计算语言学协会会议论文集：*ACL 2024*，第 2318–2335 页。

Mark Chen。2021年。评估基于代码训练的大语言模型。*arXiv* 预印本 *arXiv:2107.03374*。

Zhangyin Feng、Daya Guo、Duyu Tang、Nan Duan、Xi- aocheng Feng、Ming Gong、Linjun Shou、Bing Qin、Ting Liu、Daxin Jiang 和 Ming Zhou。2020年。Code- bert：一种用于编程语言和自然语言的预训练模型。载于计算语言学协会：*EMNLP 2020* 论文集。

Shengran Hu、Cong Lu 和 Jeff Clune。2024年。智能体系统的自动化设计。*arXiv*预印本 *arXiv:2408.08435*。

Xu Huang、Weiwen Liu、Xingshan Zeng、Yuefeng Huang、Xinlong Hao、Yuxian Wang、Yirong Zeng、Chuhan Wu、Yasheng Wang、Ruiming Tang 及其他 1 位作者。2025年。Toolace-dev：通过分解和演化实现自我改进的工具学习。*arXiv* 预印本 *arXiv:2505.07512*。

Yankai Jiang、Wenjie Lou、LilongWang、Zhenyu Tang、Shiyang Feng、Jiaxuan Lu、Haoran Sun、Yaning Pan、Shuang Gu、Haoyang Su 等。2025年。SCP：借助自主科学智能体构成的全球网络加速发现。*arXiv*预印本 *arXiv:2512.24189*。

Kiran Kate、Tejaswini Pedapati、Kinjal Basu、Yara Rizk、VijilChenthamarakshan、SubhajitChaudhury、Mayank Agarwal 和 Ibrahim Abdelaziz。2025年。LongFuncEval：衡量长上下文模型的函数调用效果。*arXiv*预印本 *arXiv:2505.10570*。

Minghao Li、Yingxiu Zhao、Bowen Yu、Feifan Song、Hangyu Li、Haiyang Yu、Zhoujun Li、Fei Huang 和 Yongbin Li。2023年。API-bank：一个全面的工具增强型大语言模型基准测试。载于自然语言处理实证方法会议论文集，第 3102–3116 页。Siyu Liu、Jiamin Xu、Beilin Ye、Bo Hu、David J Srolovitz 和 Tongqi Wen。2025年。Mattools：面向材料科学工具的大语言模型基准测试。*arXiv* 预印本 *arXiv:2505.10852*。Zhiyuan Ma、Zhenya Huang、Jiayu Liu、Minmao Wang、Hongke Zhao 和 Xin Li。2025年。自动创建可复用且多样化的工具集，以增强大语言模型推理。*AAAI* 人工智能大会论文集，39(23):24821–24830。Santiago Miret 和 Nandan M Krishnan。2024年。大语言模型是否已准备好用于现实世界材料发现？*arXiv* 预印本 *arXiv:2402.05200*。Erik Nijkamp、Bo Pang、Hiroaki Hayashi、Lifu Tu、Huan Wang、Yingbo Zhou、Silvio Savarese 和 Caiming Xiong。2022 年。一种程序合成的对话式范式。*arXiv* 预印本 *arXiv:2203.13474*，30。Shishir G Patil、Tianjun Zhang、Xin Wang 和 Joseph E Gonzalez。2024年。Gorilla：连接海量 API 的大语言模型。神经信息处理系统进展，37:126544–126565。

Qiwei Peng、Yekun Chai 和 Xuhong Li。2024年。HumanEval-XL：用于跨语言自然语言泛化的多语言代码生成基准。载于计算语言学、语言资源与评价联合国际会议论文集，第 8383–8394 页。

Cheng Qian、Chi Han、Yi Fung、Yujia Qin、Zhiyuan Liu 和 Heng Ji。2023。Creator：为大语言模型创建工具，以拆解抽象推理与具体推理。发表于计算语言学协会：*EMNLP 2023*论文集，第 6922–6939 页。

Yujia Qin、Shihao Liang、Yining Ye、Kunlun Zhu、Lan Yan、Yaxi Lu、Yankai Lin、Xin Cong、Xiangru Tang、Bill Qian、Sijie Zhao、Lauren Hongyu Ruan、Zhiyuan Liu 和 Maosong Sun。2024年。ToolLLM：助力大语言模型掌握 16000+ 真实世界 API。发表于国际学习表征会议。

Nils Reimers 和 Iryna Gurevych。2019。Sentence-BERT：使用孪生 BERT 网络生成句子嵌入。发表于自然语言处理经验方法会议暨国际自然语言处理联合会议论文集，第 3982–3992 页。

Timo Schick、Jane Dwivedi-Yu、Roberto Dessì、Roberta Raileanu、Maria Lomeli、Eric Hambro、Luke Zettlemoyer、Nicola Cancedda 和 Thomas Scialom。2023。Toolformer：语言模型可以自学使用工具。发表于神经信息处理系统进展，36:68539–68551。

Yu Shang、Yu Li、Keyu Zhao、Likai Ma、Jiahe Liu、Fengli Xu 和 Yong Li。2024年。Agentsquare：在模块化设计空间中自动搜索大语言模型智能体。2024年。*arXiv* 预印本 *arXiv:2410.06153*。Liangtai Sun、Yang Han、Zihan Zhao、Da Ma、Zhennan Shen、Baocai Chen、Lu Chen 和 Kai Yu。2024年。Sci-eval：用于科学研究的多层级大语言模型评测基准。载于*AAAI* 人工智能大会论文集，第 38 卷，第 19053–19061 页。Zeyi Sun、Ziyu Liu、Yuhang Zang、Yuhang Cao、Xiaoyi Dong、Tong Wu、Dahua Lin 和 Jiaqi Wang。2025年。SEAgent：通过自主从经验中学习实现自我进化的计算机操作智能体。*arXiv* 预印本 *arXiv:2508.04700*。Haiyuan Wan、Chen Yang、Junchi Yu、Meiqi Tu、Jiaxuan Lu、Di Yu、Jianbao Cao、Ben Gao、Jiaqing Xie、Ao-ran Wang 等。2025年。Deepresearch arena：首个通过以研讨会为基础的任务考察大语言模型研究能力的测试。*arXiv* 预印本 *arXiv:2509.01396*。Guanzhi Wang、Yuqi Xie、Yunfan Jiang、Ajay Mandlekar、Chaowei Xiao、Yuke Zhu、Linxi Fan 和 Anima Anandkumar。2023。Voyager：一种采用大语言模型的开放式具身智能体。载于神经信息处理系统大会。

Xiaoxuan Wang、Ziniu Hu、Pan Lu、Yanqiao Zhu、Jieyu Zhang、Satyen Subramaniam、Arjun R. Loomba、Shichang Zhang、Yizhou Sun 和 Wei Wang。2024年。SCIBENCH：评估大语言模型解决大学水平科学问题的能力。载于国际机器学习大会论文集，*ICML 2024*，第 235 卷，第 2072–2099 页。

Georg Wölflein、Dyke Ferber、Daniel Truhn、Ognjen Arandjelovic 和 Jakob Nikolas Kather。2025年。大语言模型智能体构建智能体工具。载于计算语言学协会年会论文集，第 26092–26130 页。

Mengsong Wu、YaFei Wang、Yidong Ming、Yuqi An、Yuwei Wan、Wenliang Chen、Binbin Lin、Yuqiang Li、Tong Xie 和 Dongzhan Zhou。2025年。Chemagent：通过基于树搜索的工具学习增强大语言模型在化学和材料科学领域的能力。*arXiv* 预印本 *arXiv:2506.07551*。

Cheng Yang、Jiaxuan Lu、Haiyuan Wan、Junchi Yu 和 Feiwei Qin。2025年。从是什么到为什么：用于基于证据推理化学反应条件的多智能体系统。*arXiv* 预印本 *arXiv:2509.23768*。

Shunyu Yao、Jeffrey Zhao、Dian Yu、Nan Du、Izhak Shafran、Karthik R Narasimhan 和 YuanCao。2022年。《ReAct：让语言模型中的推理与行动协同增效》。载于国际学习表征会议。

Botao Yu、Frazier N Baker、Ziru Chen、Garrett Herb、Boyu Gou、Daniel Adu-Ampratwum、Xia Ning 和 Huan Sun。2025年。《使用工具还是不使用工具？工具对用于化学问题求解的语言智能体的影响》。载于计算语言学协会论文集：*NAACL 2025*，第7620–7640页。

Lifan Yuan、Yangyi Chen、Xingyao Wang、Yi Fung、Hao Peng 和 Heng Ji。2024年。《CRAFT：通过创建和检索专用工具集来自定义大语言模型》。载于国际学习表征会议。

Huan Zhang、Yu Song、Ziyu Hou、Santiago Miret 和 Bang Liu。2024年。HoneyComb：一种用于材料科学的灵活大语言模型智能体系统。载于计算语言学协会论文集：*EMNLP 2024*，第 3369–3382 页。

## 完整算法工作流

附录提供了测试时工具演化（TTE）的端到端算法细节，这些内容因篇幅限制而未在正文中呈现，其中包括完整的闭环演化流程，以及确保测试时稳健性的故障处理逻辑。

### A.1 端到端测试时工具演化

算法1 展示了完整的 TTE 流程。相同的流程适用于 TTE-Zero（从空工具库开始）和 TTE-Adapt（从预定义的源工具库开始），因为演化循环完全相同，唯一的区别在于注册表的初始化方式。为确保清晰性和可复现性，我们明确区分：（i） $\tau_{\text{ret}}$ 用于检索接受（是否复用检索到的工具），以及（ii）$\tau_{\text{dup}}$ 用于去重（是否将新的原子工具视为冗余）。这两个阈值不必相同，可以分别调整。

### A.2 失败处理与回退

TTE 的设计能够在工具合成不完善时保持稳健。当验证失败（语法、运行时或领域约束）时，所提出的工具不会被注册。后续执行可以尝试仅通过推理来解决子目标，也可以在中间值仍然可用时，使用部分工具链继续执行。在我们的实现中，主要采用保守策略：仅注册经过验证的工具；工具生成失败时，则轻量回退到直接推理或程序思维，确保系统能够平稳降级，而不是不断累积有缺陷的工具。

## B 各智能体模块的提示词

本节详细介绍为 TTE 框架的三个核心模块设计的系统提示词。我们严格要求采用基于 JSON 或 XML 的输出格式，以确保解析稳健，并与 Python 执行环境无缝集成。

### B.1 结构化任务分解

该问题分析器使用图5中的提示词，将高层科学查询转换为线性执行计划。

### B.2 动态工具检索

该工具检索器根据图 6 中的提示，从工具库中选择现有的原语，

**算法 1**完整的测试时工具演化。

**要求：** 用户问题 $P$、初始工具库 $L$、工具库容量 $C$
**确保：** 解决方案 $S$（或失败）

$\mathcal{O} \leftarrow$ 分解$(P)$
链 $\leftarrow [\,]$
**对于** 每个操作 $O_i \in \mathcal{O}$ **执行**
  $\mathcal{T}_i \leftarrow$ 检索前K项$(L,\ O_i,\ k)$
  $(T^\star,\ s^\star) \leftarrow$ 最大值参数$_{T \in \mathcal{T}_i}$ 相似度$(T,\ O_i)$
  **如果** $s^\star \geq \tau_{\text{ret}}$ **那么**
    链。追加$(T^\star)$；  $u(T^\star)\ +\!= 1$
  **else**
    $T_{\text{new}} \leftarrow$ 工具合成$(P,\ O_i)$
    好的 $\leftarrow$ 验证工具$(T_{\text{new}})$
    **如果**满足 = **否则**
      **继续**
    **结束如果**
    $\mathcal{A} \leftarrow$ 原子分解$(T_{\text{new}})$
    **对**每个原子工具 $A \in \mathcal{A}$**执行**
      **如果** 最大$_{T \in L}$ 相似度去重$(A,\ T) < \tau_{\text{dup}}$ **则**
        $L \leftarrow L \cup \{A\}$；  $u\,(A)\ \leftarrow 1$
      **else**
        $T_{\text{match}} \leftarrow$ 最大值参数$_{T \in L}$ 相似度重复$\,(A,\ T)$
        $u\,(T_{\text{match}})\ +\!= 1$
      **结束条件判断**
    **结束循环**
    $L \leftarrow$ 仅在需要时剪枝$\,(L,\ C)$
    链.追加$(T_{\text{new}})$
  **结束条件判断**
**结束循环**
$S \leftarrow$ 执行链$(P$,链$)$
**如果** $S =$ 失败 **则**
  $S \leftarrow$ 回退$(P)$
**结束 if**
**返回** $S$

强制执行“禁止幻觉”策略。

### B.3 工具合成与推理

图 7 展示了生成式工具合成模块所使用的混合提示。它同时处理工具合成和最终答案生成。

## C SciEvo 各学科结果

表4报告了 Sci- Evo 在三种设置下各学科的表现：(i)“无工具”（直接推理），(ii)“Q+工具”（使用原始问题作为查询来检索工具），以及(iii)“S+ 工具 ”（使用分解后的子问题作为查询来检索工具）。

在所有模型–学科组合中，工具增强都比直接推理带来明显优势。更重要的是，子问题驱动检索（S+工具）通常更加稳健，峰值性能也高于使用原始问题检索工具（Q+工具）；这一点在化学和物理学中尤为明显，因为工具选择容易受到单位、常数和领域特定公式的影响。

一个重要规律是，化学从结构化分解中获益最大：化学

问题分析器提示词

```
[系统]
你是一位经验丰富的计算科学家，拥有广泛的跨学科专业知识。
你擅长将复杂的科学问题分解为一系列可编程计算步骤，这些步骤可以由
预定义的原子工具执行。

[任务]
将用户问题分解为一系列具体的计算子任务。
每个子任务都必须是一个具体的计算步骤（而非分析、                    讨论或描述）。
这些子任务必须构成连贯的执行顺序：前面步骤的输出可以作为后续步骤的输入。
持续分解，直至涵盖完整的计算流程                              ，但不要给出最终数值答案。

[严格输出格式]
仅返回严格的 JSON（不得添加额外文本、Markdown 或注释）：
{
  "原始_问题":          "...",
  "子任务": [
    {"步骤": 1 ，"描述":          "..."} ，
    {"步骤": 2 ，"描述":          "..."} ，
    ...
  ]
}

[用户]
{query}
```

图 5：问题分析器用于将用户查询分解为结构化执行计划的提示。

工具执行器提示

```
[系统]
你是一个严格的工具调用代理。你必须仅通过调用一个预定义的原子工具来作答。
你的目标是从提供的工具目录中选出最相关的单个工具并调用它。

[核心规则]
1) 不要直接回答问题。不要用自然语言进行计算。
2) 只能调用一个工具。如果没有匹配的工具，则返回空的工具调用列表。
3) 不要捏造工具。只能使用系统提供的工具。
4) 输出必须采用标准的 OpenAI 工具调用 JSON 格式。不要添加其他文本。

[用户]
问题：{sub_question_or_operation}
工具目录：{tool_catalog_with_signatures}
```

图6：工具执行器用于调用工具库中原语的提示。

类查询往往涉及多种操作（单位换算、理想气体关系、化学计量），此时分解后的子问题能为检索提供更明确的语义信号，并降低选中无关工具的概率。相比之下，材料科学的基线表现通常较高，而且每个问题可能只需较少种类的原子操作，因此分解带来的增量收益相对较小。

我们还观察到，最佳配置取决于模型和学科。这促使我们在附录F中进行工具库规模分析，该分析解释了为什么在问题级检索下，增加工具清单并不总能单调提升性能。

## D 工具可复用性分析

我们通过分析生成工具在各基准测试中的调用频率（命中次数），研究其可复用性。如直方图（图8）和核密度估计曲线（图9）所示，基线方法呈现出严重的左偏分布，绝大多数工具都集中在最低频率区间（1 ∼ 2 次调用）。对“一次性”工具的高度依赖表明，基线方法倾向于使用单体脚本过拟合特定查询，从而导致较高的冗余和较差的迁移能力。

相比之下，TTE 展示出概率密度显著向右偏移的趋势，有效地将质量重新分布到中高频范围（例如， 10 ∼ 50+）。这一现象表明——

提示词 面向工具合成器和工具执行器

```
[系统]
你是一支由多个科学学科领域的专家组成的专业团队。
你可以将逐步拆解的子问题与可执行的 Python
代码。

[输入]
主问题:
{main_question}

子问题（按顺序处理）。每项包含一个子问题及其对应的代码。
如果代码为空 / null,                则将其视为   "缺失"。
{(步骤_i、子问题_i、代码_i)}                # 对 i=1..n 重复

[执行规则]
1) 如果子问题提供了非空且有效的代码，则模拟其执行并写下结构化结果。
2) 如果子问题缺少代码，或所提供的代码无法解决该子问题，则生成一段可运行的 Python
函数:
    - 使用 snake_case 函数名
    - 添加清晰的文档字符串，说明输入/输出规范和单位
    - 添加一个最简测试示例
    然后模拟在测试示例上执行你的函数。

[数字和单位]
- 浮点数请使用科学计数法，并保留 6 位有效数字。
- 对于 [0,1] 中的概率/比率              ，四舍五入到小数点后 4 位。
- 所有物理量都必须包含国际单位制单位。如果缺少单位，请明确说明默认假设。

[严格输出格式]
第 1 部分：仅针对缺失代码的子问题，输出一个用 <code> ... </code> 包裹的 JSON 列表:
<code>
[
  {
    "子_问题":          "..."  ,
    "名称":  "函数      _名称",
    "代码":  "完整的 Python 函数（字符串形式）",
    "text_描述":              "函数简要摘要",
    "io_描述": {"input":              "..."  , "输出":      "..."}  ,
    "test_example": {"输入": {...}       , "结果":      "··· 或 null"} ,
    "错误":   "可选"
  }
]
</代码>

第 2 部分：将最终答案包裹在 <答案> ... </答案> 中:
<答案>
纯自然语言结论（不要提及                                    "代码" 或“函数）。
</答案>

[简洁性]
除 <代码>和 <答案>外，不允许出现任何额外文本。
```

Fig图 7：用于合成新工具并推导最终科学结论的混合提示 .

这标志着一种质的转变：从生成临时解决方案，转向发现原子级计算原语。通过演进能够捕捉基本操作（例如标准公式或单位换算）的工具，TTE 降低了冗余，并确保所学习的工具库由通用模块构成，这些模块能够通过组合来解决各种科学问题。

## E 评估指标说明

### E.1 TTE-Zero 的评估指标

在 TTE-Zero 设置中，系统完全从头开始演化工具库 $\mathcal{T}$，我们采用工具复用率（TRR@$k$）来量化合成函数的效用和泛化能力——性。

$$\mathrm{TRR@}k = \frac{|\{t \in \mathcal{T} \mid h(t) \geq k\}|}{|\mathcal{T}|}. \quad (14)$$

我们通过不断提高阈值来解读 TRR@$k$ ，以捕捉演化质量的不同维度。TRR@1 衡量至少成功执行过一次的非冗余工具所占的比例。接近 1.0 的值表示计算浪费极少，说明生成过程精准，避免创建“死代码”或臆造的函数。
TRR@2 反映即时可迁移性，用于识别那些足够稳健、能够处理多个不同查询的工具，而不是仅对单一实例过拟合。尤其是 TRR@5 和 TRR@10 等高阶指标

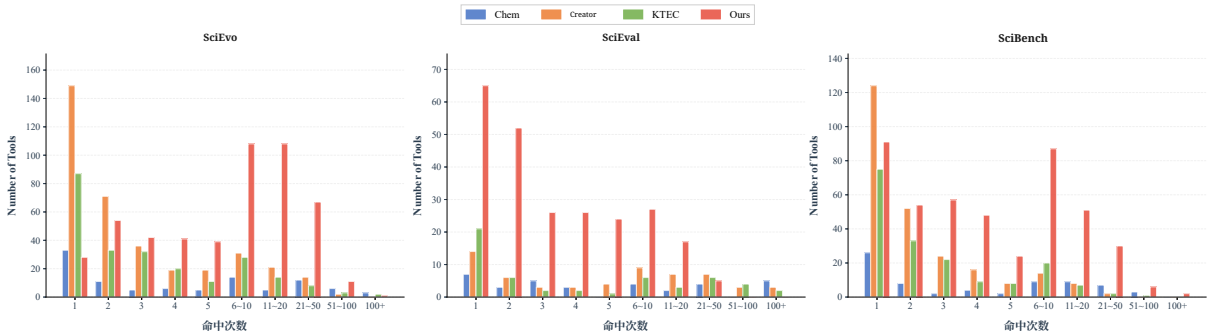



图8：三个基准测试中工具使用频率（命中次数）的直方图。横轴表示工具的复用频率，纵轴表示工具数量。

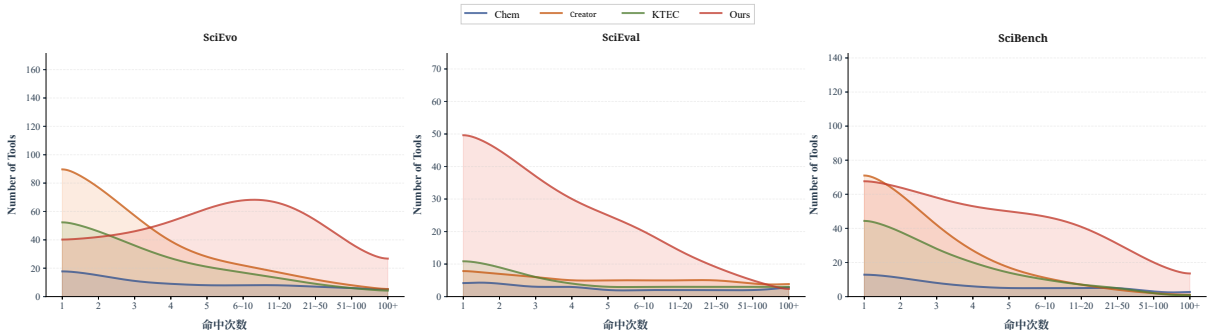



图9：工具利用率的核密度估计（KDE）。分布曲线展示了工具可复用性分布的变化。

可作为核心科学原语出现的指示变量。在这些阈值下，较高的值表明系统已经自主发现并整合了基础的领域算子（例如特定的单位转换器或热力学方程求解器），这些算子对于解决广泛类型的问题至关重要。

### E.2 TTE-Adapt 的指标

在跨域适应设置下，即 TTE-Adapt 中，系统必须在稳定性（保留有用的先验知识）与可塑性（获取新的领域特定能力）之间取得平衡。为严格评估这一动态过程，我们将最终工具库 $\mathcal{T}$ 分解为两个互不相交的集合：（i）预定义的源域子集 $\mathcal{T}_{\text{pre}}$，由源域迁移而来；（ii）新演化的子集 $\mathcal{T}_{\text{new}}$，在目标域推理期间自主合成。我们引入两个分层指标 $\text{TRR}_{\text{trans}}@k$ 和 $\text{TRR}_{\text{evol}}@k$，以区分能力的来源。

**工具复用率$_{\text{evol}}@k$。** 知识巩固指标（越高越好）。这是评估适配质量的主要指标。在标准的代码生成方法中，模型往往会生成“用完即弃”的脚本，也就是只能解决单个查询、缺乏泛化能力的一次性解决方案。较高的工具复用率$_{\text{evol}}@k$（尤其是对于 $k \geq 5$）表明，系统已成功将新领域的“物理定律”或“核心原语”提炼为可复用的原子函数。这证实工具库的扩展是高效的：系统用一组精简且高质量的新工具解决了许多问题，而不是通过臃肿的冗余脚本库过拟合。

**工具复用率$_{\text{trans}}@k$。** 负迁移缓解指标。该指标用于监测先验知识的效用。在跨领域设置中（例如，材料 $\rightarrow$ 化学），我们预期工具复用率$_{\text{trans}}$ 会低于领域内设置。较低的数值表明系统能够正确识别并剔除与目标领域无关或有害的源工具（例如，舍弃不适用于分子的特定材料属性计算器）。不过，该数值应保持非零。非零的保留率意味着系统保留了领域无关能力（例如，基础代数、统计函数、单位换算），这些能力具有普遍适用性。

| 模型 | 学科 | 无工具 | Q + 工具 | | | S + 工具 | | |
|---|---|---|---|---|---|---|---|---|
| | | | 100 | 250 | 500 | 100 | 250 | 500 |
| GPT-3.5-turbo | Che | 0.548 | 0.580 | 0.540 | 0.551 | **0.647** | 0.589 | 0.619 |
| | Math | 0.503 | 0.548 | 0.585 | **0.598** | 0.564 | 0.552 | 0.546 |
| | Phy | 0.524 | 0.600 | 0.629 | **0.642** | 0.634 | 0.609 | 0.639 |
| | Mat | 0.535 | 0.625 | 0.589 | 0.607 | 0.617 | **0.631** | 0.607 |
| GPT-4o | Che | 0.412 | 0.536 | 0.539 | 0.539 | **0.638** | 0.594 | 0.578 |
| | Math | 0.341 | 0.376 | 0.366 | 0.366 | **0.502** | 0.482 | 0.487 |
| | Phy | 0.346 | 0.533 | 0.534 | 0.487 | **0.615** | 0.557 | 0.576 |
| | Mat | 0.515 | **0.652** | 0.618 | 0.642 | 0.605 | 0.588 | 0.608 |
| Qwen2.5-7B | Che | 0.211 | 0.317 | 0.291 | 0.312 | **0.400** | 0.358 | 0.352 |
| | Math | 0.175 | 0.246 | 0.246 | 0.201 | 0.256 | 0.241 | **0.286** |
| | Phy | 0.227 | 0.332 | 0.302 | 0.304 | **0.368** | 0.365 | 0.360 |
| | Mat | 0.301 | 0.358 | 0.274 | 0.240 | **0.433** | 0.384 | 0.334 |

表4：基于 TTE 框架的 SciEvo 基准测试各学科表现。Che：化学，Math：数学，Phy：物理学，Mat：材料。子问题分解（“S+工具”）在所有学科中始终优于输入主问题（“Q+工具”）。

**替代效应**。 通过分析 (工具复用率$_{\text{trans}}$，工具复用率$_{\text{evol}}$)的联合变化轨迹，我们可以诊断适配策略。表 3 所示的实证结果证明了替代效应：随着领域差距增大，TTE 会自主降低对 $\mathcal{T}_{\text{pre}}$ (较低的 工具复用率$_{\text{trans}}$) 的依赖，并通过提高 $\mathcal{T}_{\text{new}}$( 较高的 工具复用率 $_{\text{evol}}$) 的合成来弥补。这与静态基线形成鲜明对比，后者会遭遇“强行套用”，试图用不匹配的旧工具解决新问题。

## F 工具过载现象

表4 的实证分析揭示了一种反直觉的非单调趋势：将工具清单从 100 个原子基元扩充到 500 个，并不能始终带来性能提升。在某些配置中，尤其是依赖直接查询到工具匹配的配置，工具库规模扩大反而会降低问题求解准确率。我们将这一现象称为“工具过载现象”。

**理论分析**。我们将这种性能下降归因于工具库丰富度与检索鲁棒性之间固有的张力。随着工具库扩充，向量空间的语义密度随之增加，最优工具与高相似度干扰项之间的距离不可避免地缩小。这会导致检索冲突：语义相近但功能不同的工具——例如，两个输入假设略有不同的热力学计算器变体——会在最近邻搜索过程中挤掉正确的候选项。

此外，即使成功检索到正确的工具，这些高相似性干扰项的存在也会在上下文窗口中引入显著的上下文干扰。语言模型被迫在细微不同的函数签名之间进行精细区分，从而增加了选择过程的认知负荷。“选择瘫痪”会消耗模型的推理能力，增加其选择次优工具或产生参数幻觉的可能性，进而抵消更大能力集合所带来的理论优势。

**对可扩展智能体系统的启示**。这些发现表明，扩展工具库所需的不只是简单地积累函数，还需要对检索机制进行架构创新。为减轻工具库扩展带来的噪声，未来的系统必须超越扁平相似性搜索。潜在的解决方案包括分层索引策略：先确定相关的工具领域，再选择具体的原子函数；或者采用不确定性感知检索机制，根据查询的语义歧义程度动态调整检索候选项的数量。

## G 案例研究

我们详细考察了两个科学推理场景，以说明静态工具库的具体失效模式，以及测试时工具演化（TTE）框架如何解决这些问题。

### G.1 案例1：摩尔质量估算

**问题定义**。 任务要求根据气态化合物的密度（1.23 kg m$^{-3}$）、温度（330 K）

| 设置 | 结果与诊断 |
|---|---|
| 无工具 | 预测为 76.9 g $mol^{-1}$ （不正确）。模型可能进行了错误的变换，或丢失了单位一致性，且未经可执行验证。 |
| Q+工具 | 预测值为 $1.73 \times 10^2$ g $mol^{-1}$ （错误）。不过，工具上下文中可能包含许多无关的函数，增加选择噪声，使成功在不同实例之间的可靠性降低。 |
| S+工具 | 预测值为 $1.69 \times 10^2$ g $mol^{-1}$ （正确）。分解过程分离出缺失的操作“计算通过 $V_m = RT/P$ 求摩尔体积”，并触发定向工具合成，生成一个显式的可执-行函数，并进行单位处理。 |

表5：案例1的执行结果。子问题分解能够实现定向工具合成并减少检索歧义。

| Step | 子目标描述 | 工具操作 | 工具状态 | 中间结果 |
|---|---|---|---|---|
| 1 | 将密度从 $kg/m^3$ 转换为 $g/L$ | 转换_密度(1.23) | 已检索 | 1.23 $g/L$ |
| 2 | 将压力从 $kPa$ 转换为 $Pa$ | 转换_压力(20) | 已检索 | 20，000 $Pa$ |
| 3 | 计算摩尔体积 $V_m$ ($PV = nRT$) | 计算_摩尔_体积(20000, 330) | 已演化 | 13.738 $L/mol$ |
| 4 | 计算摩尔质量 $M = \rho \times V_m$ | 计算_摩尔_质量(1.23, 13.738) | 已检索 | **169.0** $g/mol$ |

表6：案例1的逐步执行轨迹。工具状态 列突显了系统的自适应行为：对于标准操作（步骤1、2、4），系统会检索现有工具；而对于步骤3中缺失的基本操作，则会自主开发新工具。

和压力（20 kPa），估算其摩尔质量。真实值为 169 g $mol^{-1}$。此问题是对系统处理多步推理、严格单位一致性以及缺失计算原语能力的一项关键测试。

**性能比较**。 如表 5所示，基线方法要么容易产生幻觉，要么精度不足。 无工具基线依赖参数化知识，因此得出的数值虽然在物理上看似合理，却并不正确（76。9 g $mol^{-1}$），这可能是由于错误地应用了理想气体定律的变形式。*Q*+ 工具设置会检索通用密度计算器，但受到上下文窗口噪声的影响，得出近似结果（173 g $mol^{-1}$）。相比之下，我们的*S*+工具 框架通过强制采用结构化执行路径，得出了精确的解析解（169 g $mol^{-1}$）。

**演化执行轨迹**。表格展示了 TTE 的核心优势6，详细说明了逐步解决问题的过程。 问题分析器 首先将复杂查询分解为四个原子子问题。对于步骤 1、2 和 4，系统成功在现有工具库中找到高相似性匹配，并检索到标准单位换算工具和算术工具。然而，在步骤 3，系统遇到了缺口：工具库中包含通用气体定律函数，却缺少一种能直接根据压力和温度计算摩尔体积的专用原语。检测到

这一检索失败后，工具合成器 便会触发。它会生成一个专用的原子函数计算_摩尔_体积（如图10所示），该函数能正确处理气体常数 $R$ 以及从 $m^3$ 到 $L$ 的单位换算。随后立即执行这个“演化”工具，弥补了导致基线方法失败的计算缺口。

### G.2 案例2：电镀化学计量

**问题定义**。 该问题涉及计算通过电解沉积在托盘上的银的质量（电流：8。46 A，时间：8。0 h），随后根据镀层厚度为0。00254 厘米、密度为10。5 g/$cm^3$，确定托盘的表面积。真实值为质量33。98 g、面积1275。6 $cm^2$ 。此任务要求结合应用法拉第电解定律与几何体积—面积关系。

**性能比较**。表7 对结果进行了对比。 无工具基线模型通常无法正确整合物理常数（法拉第常数），导致数量级误差。 *Q*+工具模型生成了一个单体函数，该函数虽然正确识别了物理公式，但很可能误解了时间参数或化学计量的上下文，因此得出285。6 克的结果，与真实值相差甚远。相比之下，*S*+工具通过将问题分解为电荷计算、化学计量转换和几何推导，实现了高精度（31。6 克 和 1283 厘米$^2$），并对结果进行了验证

```
def 计算_摩尔_体积(压力_帕,                    温度_开尔文):
    """
    根据理想气体定律计算摩尔体积 Vm：Vm = RT/P。

    参数:
        压力_帕（浮点数）：压力，单位为 Pa
        温度_开尔文（浮点数）：温度，单位为 K

    返回:
        浮点数：以升/摩尔为单位的摩尔体积
    """
    R = 8.314462618       # 焦耳/(摩尔*开尔文)
    vm_m3_per_mol = (R * 温度_k) / 压力_pa
    vm_L_per_mol =vm_m3_per_mol * 1000.0
    返回 vm_升_每_摩尔
```

图10：案例1中合成原子函数的摘录，该函数能够正确进行摩尔质量计算。

独立执行每个步骤。

**演化执行轨迹**。表 8展示了自适应工作流。系统将物理问题分解为一系列逻辑步骤。步骤 1 和步骤 4 使用现有工具库中的工具进行基本单位换算和质量与物质的量关系计算。然而，对于步骤 2（根据电荷计算电子的物质的量）、步骤 5（计算体积）和步骤 6（根据体积和厚度推导面积），检索系统返回的匹配结果相似度较低。因此， 工具合成器演化出了专用基元：计算_电子的物质的量__电子（纳入法拉第常数）以及计算_面积。图11 展示了演化出的面积计算工具，它明确处理了几何关系 $A = V/t$，弥合了化学输出与几何输入之间的差距。

## H 数据集比较与独特性

SciEvo 通过建立一个能够同时评估科学推理准确性和工具演化过程有效性的基准测试，填补了当前评估方案中的关键空白。正如表 9所总结的，现有基准测试通常将这些能力分开评估，而 SciEvo 则将二者结合起来，以模拟现实世界科学研究的开放性。

### H.1 与现有基准测试的比较

现有基准测试可分为三类，但都未能充分体现测试时演化范式。首先，SciBench 等科学推理基准测试（Wang 等人，2024年）以及 SciEval（Sunetal.，2024年）侧重问题求解，但假设设置固定。SciBench 提供问题集，却不提供可执行工具，迫使模型依赖内部参数化知识，或使用缺乏统一接口的外部计算器。SciEval 提供多层级评估，但缺少评估工具生成的机制。其次，ToolBench（Qin 等人，2024年）、API-Bank（Li 等人，2023）和 BFCL（Patil 等人，2024年）等函数调用基准测试侧重检索和调用静态库。尽管 CONFETTI（Alkhouli等人，2025年）和 NEST-FUL（Basu等人，2025年）等近期研究探索了复杂的嵌套调用，LongFuncEval（Kate等人，2025年）评估了长上下文检索，但它们都基于“封闭世界”假设，即工具集不可变。第三，代码生成基准测试，例如例如，HumanEval（Peng 等人，2024年）侧重生成独立的代码片段。尽管涉及合成，但它们并不评估生成的代码能否作为可复用的库组件（工具），以供后续任务维护和检索。SciEvo 则独特地整合了这些维度，要求智能体不仅解决科学问题，还要维护并演化由原子基元构成的持久工具库。

### H.2 领域覆盖与工具模态

SciEvo的一大独特优势在于其全面的学科覆盖。以往的领域专用智能体依赖人工整理的静态工具包。例如，ChemCrow（Bran 等人，2024年）提供了一个包含约 19 种工具的专用工具库，分为通用推理（4 种）、分子操作（8 种）、安全检查（3 种）和反应

| 设置 | 结果与诊断 |
|---|---|
| 无工具 | 由于缺少领域常数（法拉第常数）和几何公式，预测值错误。中间值出现幻觉的情况很常见。 |
| Q+工具 | 预测为 285.6 克和 10,800 厘米$^2$ （错误）。模型尝试进行单步计算。虽然代码逻辑在语法上正确，但单体执行路径未能将时间单位与具体的问题约束相匹配（可能过度放大了持续时间），从而导致了较大的偏差。 |
| S+工具（TTE） | 预测为 31.6 克和 $1.28 \times 10^3$ 厘米$^2$ （正确）。分解强制进行了逐步验证。系统演化出了用于电子摩尔计算和面积推导的专用工具，确保各接口处的量纲一致性。 |

表7：案例2的执行结果。逐步分解可防止多阶段物理问题中的误差传播。

| Step | 子目标描述 | 工具操作 | 工具状态 | 中间结果 |
|---|---|---|---|---|
| 1 | 计算总电荷 $Q = I \times t$ | 计算_电荷(8.46, ...) | 已检索 | 30, 500 $C$ |
| 2 | 计算电子的物质的量 $n = Q/F$ | 计算_电子的物质的量_, 共_（30500） | 演化 | 0.316 $mol$ |
| 3 | 化学计量（银氧化 +1） | [逻辑推理] | - | 0.316 $mol$ $Ag$ |
| 4 | 将物质的量转换为质量（摩尔质量） | 计算_物质的量（质量, ...） | 已检索 | 31.63 $g$ |
| 5 | 计算体积 $Volume = mass/$密度 | 计算_体积(31.63,10.5) | 已演化 | 3.26 $cm^3$ |
| 6 | 计算托盘面积 $A = V/$厚度 | 计算_面积(3.26, 0.00254) | 演化而成 | **1283.5** $cm^2$ |

表8：逐 步执行案例2的过程追踪。当找不到完全匹配的工具时，系统会演化出特定的物理学和几何学工具（步骤2和6），而在适当情况下则会复用标准化学工具（步骤4）。

处理（4 种）。类似地，CheMatAgent（Wu 等人，2025年）将其扩展至材料领域，提供约 34 种化学工具（例如，摩尔质量、溶液浓度）和 95 种材料科学工具（例如，晶体结构分析、相图计算）。然而，这些工具库是静态的，且局限于特定领域。它们不支持基础数学和物理学，而这两者在跨学科研究中无处不在。正如我们的分析所示，SciEvo 不仅涵盖化学和材料领域，还填补了数学和物理学方面的空白。至关重要的是，与 ChemCrow 或 CheMatAgent 中的静态定义不同，SciEvo 中的工具是动态合成的，这意味着其工具库的覆盖范围在理论上没有上限，能够根据推理轨迹，从基础算术发展到复杂的热力学模拟。

# I 理论分析

本节对测试时工具演化框架所依据的机制进行了严格的形式化。我们推导了工具可复用性的理论界限，分析了工具库扩展对检索保真度的影响，量化了序列推理中的误差传播，并证明了在我们的剪枝策略下库规模的收敛性。

## I.1 原子分解带来的效用提升

我们分析将单体工具分解为原子函数的预期效用。设 $\mathcal{Q}$ 为所有可能未来查询的空间。单体工具 $T$定义为由 $k$ 个相互独立的原子操作 $\{a_1, a_2, \ldots, a_k\}$组合而成。对于任意查询 $q \in \mathcal{Q}$，设 $S(q)$表示解决 $q$所需的原子操作集合。

**定义**。 我们使用指示变量来定义工具的适用性。单体工具 $T$适用于查询 $q$，当且仅当该查询需要 $T$ 所实现的完整操作集，或需要一个足够相似的操作集，以便检索并以单体方式执行 $T$。为进行严格分析，我们假设当 $S(q) \supseteq \{a_1, \ldots, a_k\}$ 时， $T$ 会被复用。反之，原子工具 $A_i$ （对应操作 $a_i$）在 $a_i \in S(q)$ 时会被复用。令 $R(T)$ 和$R(A_i)$ 分别为随机变量，表示单体工具和第 $i$ 个原子工具在包含 $M$ 个查询的序列中的复用次数。

**定理 1（分解下界）**。令$p_{\text{partial}}$ 表示查询需要一部分操作（而非全部操作）的概率，具体而言，对任意 $i$， $P(a_i \in S(q) \mid \{a_1 \ldots a_k\} \not\subseteq S(q)) \geq \delta$，其中某个 $\delta > 0$。分解后的原子工具库的预期总复用率严格高于

```
def 计算_面积(体积_厘米3,          厚度_厘米):
    """
    计算托盘的面积。

    参数:
        体积_厘米3 (浮点数): 银的体积（厘米^3）
        厚度_厘米（浮点数）：银涂层的厚度（厘米）

    返回:
        浮点数: 面积（厘米^2）
    """
    # 面积 =体积 / 厚度
    面积 =体积    _厘米3 / 厚度       _cm
    返回 面积
```

图 11：案例2步骤 6中合成原子函数的摘录。该工具经过即时演化，将电化学结果（体积）与几何要求（面积）关联起来。

| 数据集 | 领域 | 工具模态 | 演化 | 推理类型 | 工具覆盖范围 |
|---|---|---|---|---|---|
| SciBench（Wang 等人，2024年） | 科学（通用） | None | No | 思维链 | - |
| SciEval（Sun 等人，2024年） | 科学（通用） | None | No | 多层级 | - |
| ToolBench（Qin 等人，2024年） | 通用 | 静态 API | No | API 调用 | 通用 API |
| BFCL(Patil等人，2024年) | 通用 | 静态函数 | No | AST 解析 | Python/Java/JS |
| ChemCrow(Bran等人，2024年) | 化学 | 静态工具库 | No | ReAct | 19（化学） |
| CheMatAgent (Wu 等人，2025年) | 化学与材料 | 静态库 | No | ReAct | 130（Chem+数学） |
| HumanEval（Peng 等人，2024年） | 计算机科学/编程 | 一次性代码 | No | 代码生成 | - |
| **SciEvo（我们的方法）** | **科学（通用）** | **动态/演化** | **Yes** | **TTE-推理** | **925** |

表9：SciEvo 与现有基准测试的比较。SciEvo 是唯一支持在多个科学领域进行测试时演化的框架，而其他框架依赖静态库，或仅专注于代码生成而不涉及库管理。

单体工具：

$$\mathbb{E}\left[\sum_{i=1}^{k} R(A_i)\right] \geq k \cdot \mathbb{E}[R(T)] + \Delta_{\text{flexibility}}, \quad (15)$$

其中 $\Delta_{\text{flexibility}}$ 是表示部分复用效用的正值项。

**证明。** 考虑单个查询 $q$。令 $X_T^{(q)}$ 为指示变量，表示是否复用了 $T$；令 $X_i^{(q)}$ 为指示变量，表示是否复用了 $A_i$。根据定义，如果复用了 $T$，则所有底层操作均处于激活状态，这意味着 $X_T^{(q)}=1 \implies \forall i,\ X_i^{(q)} = 1$。因此，$X_i^{(q)} \geq X_T^{(q)}$ 几乎必然成立。对查询分布取期望，可得 $\mathbb{E}[X_i^{(q)}] = P(a_i \in S(q))$ 和 $\mathbb{E}[X_T^{(q)}] = P(\{a_1 \dots a_k\} \subseteq S(q))$。根据全概率公式，我们展开原子复用概率：

$$P(a_i \in S(q)) = P(a_i \in S(q) \mid \text{Mono})P(\text{Mono}) + P(a_i \in S(q) \mid \text{Partial})P(\text{Partial}), \quad (16)$$

由于 $P(a_i \in S(q) \mid \text{Mono}) = 1$，第一项恰好等于 $\mathbb{E}[X_T^{(q)}]$。第二项表示“部分匹配”的情形，即单体工具失败而原子工具成功。对所有 $k$ 工具和所有 $M$ 查询求和，并运用期望的线性性，可得：

$$\sum_{i=1}^{k} \mathbb{E}[R(A_i)] = M \sum_{i=1}^{k} \Big( \mathbb{E}[X_T^{(q)}] + P(a_i \in S(q), \text{Partial}) \Big). \quad (17)$$

项 $M \sum \mathbb{E}[X_T^{(q)}]$ 等于 $k \cdot \mathbb{E}[R(T)]$。由于 $p_{\text{partial}} > 0$，且这些操作的边际效用非零，其余总和严格为正。因此，分解通过捕获单体工具所遗漏的子问题的边际效用，保证了更高的预期可复用性。

□

### I.2 不断增长的工具库中的检索精度

我们对“工具过载”现象进行形式化研究。设工具库的大小为 $N$。对于给定的查询，令 $s_r$ 表示唯一相关工具的相似度分数，该分数取自一个

概率密度函数为 PDF$f_r(s)$、累积分布函数为 $F_r(s)$ 的分布。令 $\{s_i\}_{i=1}^{N-1}$表示 $N-1$ 个无关工具（干扰项）的分数，它们独立取自一个噪声分布，其概率密度函数为 PDF$f_n(s)$、累积分布函数为 $F_n(s)$。检索系统会选择分数最高的工具。

**定理2（单调退化）**。假设相关分布与噪声分布的支集存在重叠，使得对于某个 $s$，$F_n(s) < 1$，其中 $f_r(s) > 0$，则正确检索相关工具的概率（记为 $P_N(\text{success})$）是库规模$N$的严格递减函数。

**证明**。当相关工具的得分$s_r$ 大于所有 $N-1$ 个干扰项得分中的最大值时，就会检索到该工具。令 $M_{N-1} = \max\{s_1, \ldots, s_{N-1}\}$。独立随机变量的最大值的累积分布函数等于各自累积分布函数的乘积，因此 $P(M_{N-1} \le x) = [F_n(x)]^{N-1}$。成功概率即 $s_r > M_{N-1}$ 成立的概率。我们对 $s_r$ 的所有可能取值积分：

$$\begin{aligned} P_N(\text{success}) &= \int P(M_{N-1} < s \mid s_r = s) f_r(s)\, ds \\ &= \int [F_n(s)]^{N-1} f_r(s)\, ds. \end{aligned} \tag{18}$$

为了确定关于 $N$ 的变化趋势，我们将$N$ 视为连续变量，并使用莱布尼茨法则在积分号下求导：

$$\frac{\partial}{\partial N} P_N(\text{success}) = \int_{-\infty}^{\infty} f_r(s) \frac{\partial}{\partial N} [F_n(s)]^{N-1}\, ds. \tag{19}$$

计算积分内部的导数：

$$\frac{\partial}{\partial N} [F_n(s)]^{N-1} = [F_n(s)]^{N-1} \ln(F_n(s)). \tag{20}$$

由于 $F_n(s)$ 是累积分布函数，$0 \le F_n(s) \le 1$，这意味着 $\ln(F_n(s)) \le 0$。对于分布有重叠且检索并非无效的任何区域，$F_n(s) < 1$，因此对数严格为负。因此，被积函数处处非正，并且在重叠区域上严格为负。由此可得，$\frac{\partial P_N}{\partial N} < 0$，这证明了在不改进检索机制（例如通过子问题分解）的情况下扩展工具库必然会提高错误率。

### I.3 工具库增长的稳定性

我们对工具库规模 $L(t)$ 的时间动态进行建模，以证明所提出的演化机制会使系统趋于稳定，而非无限增长。

**动态模型**。工具库规模的变化率由两种相反的力量决定：检索失败时生成新工具，以及剪除效用低的工具。设 $\lambda_g$ 为最大潜在生成率。随着工具库增长，找到匹配项的概率会提高，从而抑制新工具的生成。我们用逻辑斯蒂项（$1 - L(t)/K$）对这种饱和现象进行建模，其中 $K$ 表示语义空间的有效容量。设 $\lambda_p$ 为剪枝率，它与当前工具库规模成正比（假设低于使用阈值的工具比例恒定）。描述该系统的微分方程为：

$$\frac{dL}{dt} = \lambda_g \left(1 - \frac{L(t)}{K}\right) - \lambda_p L(t). \tag{21}$$

**定理 3（收敛）**。对于任意非负初始条件 $L(0) \ge 0$，工具库规模 $L(t)$ 将渐近收敛到稳定的平衡点 $L^*$。

**证明**。我们将微分方程重新整理为具有常系数的标准线性形式：

$$\frac{dL}{dt} = \lambda_g - \left(\frac{\lambda_g}{K} + \lambda_p\right) L(t). \tag{22}$$

令 $A = \lambda_g$ 和 $B = \frac{\lambda_g}{K} + \lambda_p$。方程变为 $\frac{dL}{dt} = A - BL(t)$。令 $\frac{dL}{dt} = 0$，即可求得平衡点 $L^* = \frac{A}{B} = \frac{\lambda_g K}{\lambda_g + \lambda_p K}$。该一阶线性常微分方程的通解为：

$$L(t) = L^* + (L(0) - L^*)\, e^{-Bt}. \tag{23}$$

由于 $B > 0$（生成率、容量和剪枝率均为正），项 $e^{-Bt}$ 会随着 $t \to \infty$ 增大而衰减至零。因此，无论工具库最初为空还是已满，系统都会自主调节，使其趋向稳态规模 $L^*$。这证明了 TTE 框架能够抵御工具库规模的爆炸式增长，从而确保长期计算效率。

## J 未来方向与更广泛的影响

向演化工具生态系统的转变，为非平稳环境中的智能提供了一种可泛化的范式。通过将工具视为自适应能力，而非静态资源，测试时工具演化框架使

智能体能够应对开放式挑战。我们概述了扩展并增强这一范式鲁棒性的关键方向。

**生命周期管理**。 工具库的无界增长要求进行严格维护，以保持检索效率。未来研究必须应对可塑性（获取新工具）与稳定性（保留核心能力）之间的权衡。智能剪枝和分层索引等机制对于遗忘过时的原语、整合高效用函数以及防止知识饱和至关重要。

**鲁棒性与校准**。 提升合成工具的可靠性是一项优先任务。未来系统应纳入形式化验证或不确定性感知生成，以确保代码安全。此外，我们设想采用元认知校准机制，使智能体根据置信度动态权衡检索与演化的成本，并通过自我纠正循环，在执行失败后迭代完善工具逻辑。

**多模态前沿**。 现实世界问题要求解读示意图或仪器读数。将 TTE 扩展到多模态情境，需要演化用于基于视觉的分析或图操作的工具。协同演化感知与推理能力，是迈向能够开展端到端科学研究的完全自主智能体的关键步骤。