AlphaEvolve: A coding agent for scientific and algorithmic discovery

Google DeepMind AlphaEvolve Team

AlphaEvolve official release: May 14, 2025

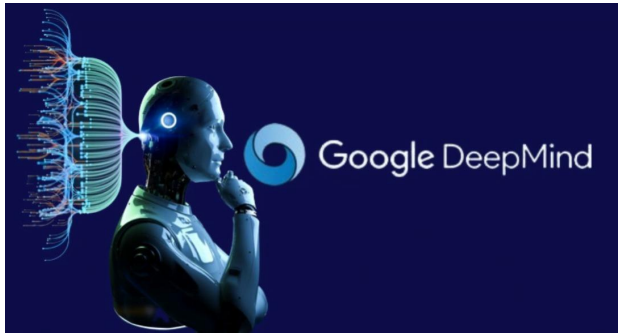
Presented by Lili Pan

作者团队

Google DeepMind

DeepMind：Google旗下人工智能公司

- 2016年3月，DeepMind开发的**AlphaGo**程序以4：1击败韩国**围棋冠军**李世石（Lee Se-dol），成为近年来人工智能领域的里程碑事件。
- 2020年，杰出代表**AlphaFold，**解决了困扰生物学界数十年的**蛋白质结构预测**问题，将过去耗时数年耗资巨额的过程缩短到几秒几分。
- 2025年5月，DeepMind研究团队发布了**Gemini 2.5 Pro** “I/O”。
- 2025年5月，Deepmind发布了设计高级算法的编程体**AlphaEvolve**。



- AlphaZero 专注于游戏
- AlphaEvolve 专注于算法发现和优化

Outline

1 Background

2 Related work

3 Design

4 Experiments

5 Conclusion

背景知识：编码智能体

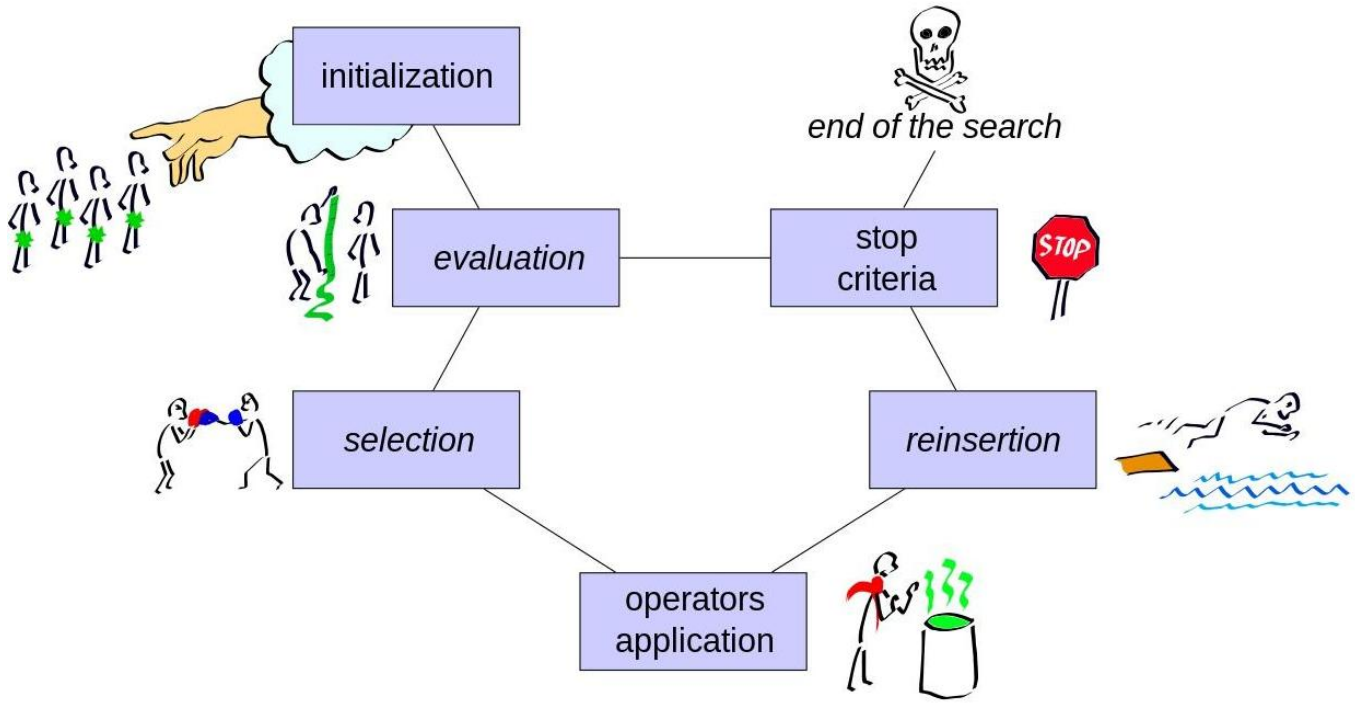
编码智能体

- 能够辅助或自动化编程任务的智能工具或系统
- **AI + 软件工程** 深度融合
  - 理解自然语言需求，自主执行编码任务
  - 涵盖编写代码、调试代码、生成测试用例，创建文档
  - 具备更高自主性，拥有规划、推理、使用工具的能力
- 常用的编码智能体
  - GitHub Copilot（VSCode 集成）
  - Cursor、Devin (Cognition Labs)
  - Aider--允许开发者在本地环境与AI配对编程

相关工作：进化算法

进化算法（Evolutionary Algorithms, EA）

- 基于生物进化原理的元启发式优化算法
- 模拟**自然选择、变异和交叉**等机制，用于解决复杂优化问题



核心：优胜劣汰

AlphaEvolve

Alpha + Evolve 进化编码智能体

- Alpha：通常与谷歌 DeepMind 的一系列突破性 AI相关，如 AlphaGo、AlphaFold
- Evolve：将进化算法应用于 AI 领域

愿景:

进化算法强大的**搜索和优化**能力，自动发现或优化更强大的人工智能算法

- 传统的 AI 研究，eg.深度学习
  - 依赖专家知识和直觉来设计模型结构、损失函数或优化算法
  - 耗时耗力且充满不确定性的过程。

AlphaEvolve 的核心思想：**我们能不能让算法自己“进化”出更好的算法?**

AlphaEvolve

前身FunSearch

- Google DeepMind 于 2023 年 12 月 开发
  - FunSearch 是 "Searching for Functions" 的缩写
  - 核心目标：在庞大的函数空间中，搜索出能够解决特定问题的、最高效或最优秀的函数
- AlphaEvolve 看作是 FunSearch 的重大增强

FunSearch [83]	*AlphaEvolve*
evolves single function	evolves entire code file
evolves up to 10-20 lines of code	evolves up to hundreds of lines of code
evolves code in Python	evolves any language
needs fast evaluation (≤ 20min on 1 CPU)	can evaluate for hours, in parallel, on accelerators
millions of LLM samples used	thousands of LLM samples suffice
small LLMs used; no benefit from larger	benefits from SOTA LLMs
minimal context (only previous solutions)	rich context and feedback in prompts
optimizes single metric	can simultaneously optimize multiple metrics

AlphaEvolve和FunSearch的能力行为对比

AI在科学领域的应用

AI for Scientific and Mathematical Discovery

- AI辅助科学发现已成为热门趋势，在材料科学、生物、物理等各个领域都有应用

如何让AI变得极其可靠，避免LLM的“幻觉”问题?

- **和“代码”打交道：**很多科学AI处理的是自然语言写的“科学猜想”，而 AlphaEvolve 进化的是实实在在的计算机程序 。
- **用“执行”来检验：**判断一个新想法好不好，不是靠LLM自己打分，而是直接运行新生成的代码，用程序的实际输出来评估 。

Outline

1 Background

2 Related work

3 Design

4 Experiments

5 Conclusion

AlphaEvolve overview

1. Human defines “What?”
2. AlphaEvolve figures out “How?”

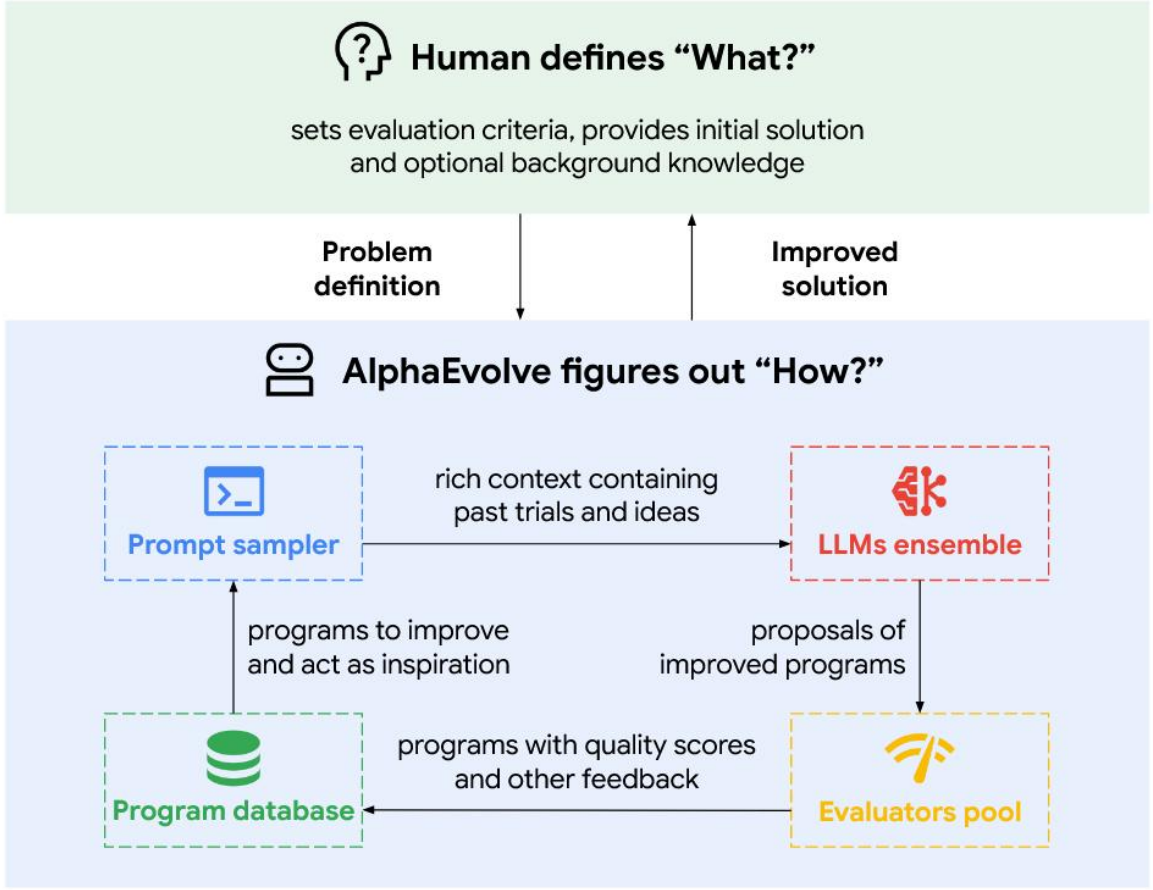
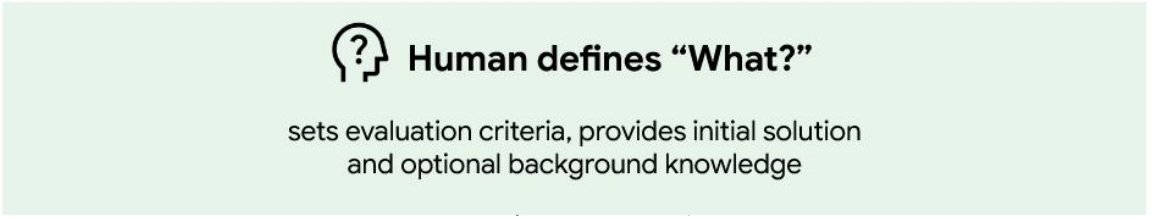


图 1： AlphaEvolve 综合概述

任务设定

2.1 任务设定

- 评估函数
- 初始程序
- 配置和背景知识



a. 评估函数
   - ➢ 可以自动评估任何解决方案好坏的程序（函数）

b. 初始程序
   - ➢ 人类需要提供一个基础版本的代码

c. 配置和背景知识（可选）
   - ➢ 比如选择LLM模型， 或相关论文、代码片段等

```python
def evaluate(eval_inputs) -> dict[str, float]:
  ...
  return metrics
```

```python
# EVOLVE-BLOCK START
"""Image classification experiment in jaxline."""

import jax
...
# EVOLVE-BLOCK-END
```

```
The current model uses a simple ResNet architecture with only
three ResNet blocks. We can improve its performance by
increasing the model capacity and adding regularization. This
will allow the model to learn more complex features and
generalize better to unseen data. We also add weight decay to
the optimizer to further regularize the model and prevent
overfitting. AdamW is generally a better choice than Adam,
especially with weight decay.
```

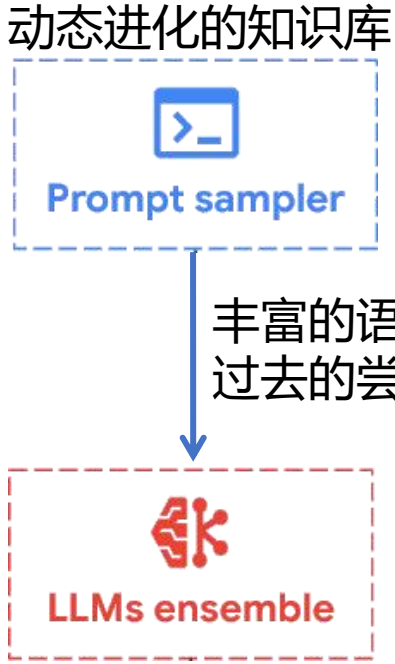
开始进化循环

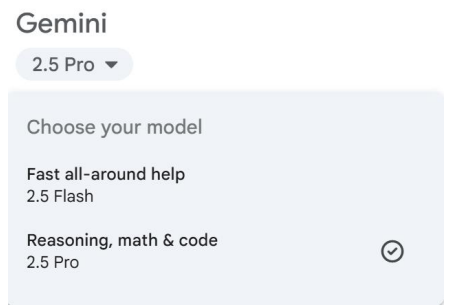
a. 采样与构建提示

- 做什么： 从“**程序数据库**”历史中挑选出一些过去表现最好的程序 。
- 为什么： 这是为了给LLM提供“**灵感**” 。

b. 创意生成

- 做什么： 将上一步构建好的提示发送给一个由多个LLM组成的“**模型集群” (LLMs ensemble)** 。
- 为什么： 这是产生新想法的地方。AlphaEvolve创新地使用了一个模型组合：
  - **Gemini 2.0 Flash**：速度快、成本低（“广度”探索）
  - **Gemini 2.0 Pro**：能力更强，更高质量（“深度”探索）
- 输出格式： LLM被要求以一种特定的SEARCH/REPLACE diff格式返回代码修改建议，这样系统就可以精确地将新代码应用到旧代码上 。





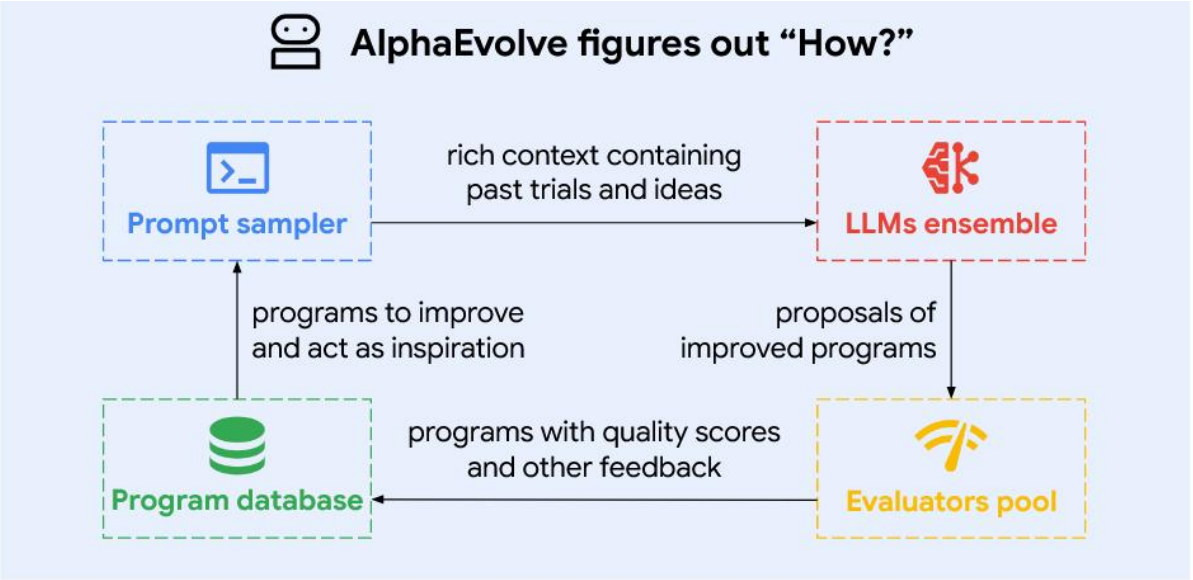

```
<<<<<<< SEARCH
  def optimizer(self, learning_rate):
    return optax.adam(learning_rate)
=======
  def optimizer(self, learning_rate):
    return optax.adamw(learning_rate, weight_decay=1e-4)
>>>>>>> REPLACE
```

```
<<<<<<< SEARCH
  # Original code block to be found and replaced
=======
  # New code block to replace the original
>>>>>>> REPLACE
```

开始进化循环

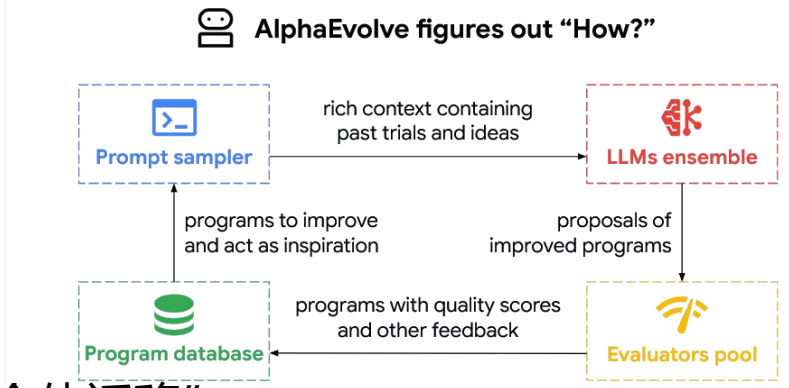
c. **评估**

- 做什么： 将新程序发送到“**评估器池**”， 执行评估代码
- 为什么： 这是质量控制环节，进行打分
  - AlphaEvolve支持“评估级联”
  - 支持并行计算，同时评估多个方案
  - 支持多目标优化

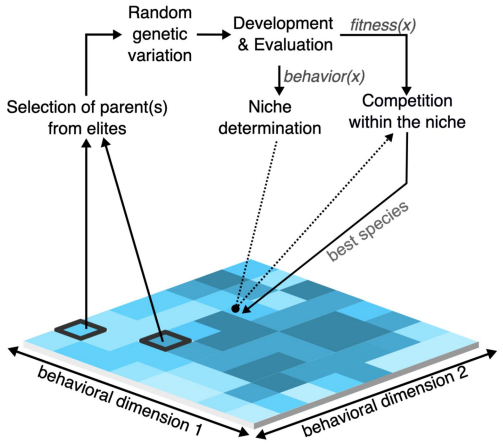


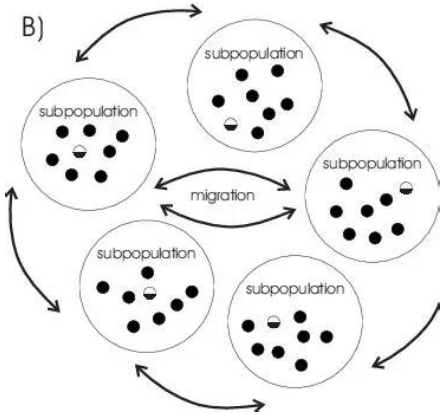
开始进化循环

d. 进化与存储



- 做什么： 将通过评估的信息，添加到“**程序数据库**” 。
  - 简单存储+结合**两种进化算法**
  - **MAP elites**算法：致力于找到不同维度上的精英解
  - **岛屿人口**模型：每个岛屿独立进化，同时不同岛屿之间进行“个体迁移”





- 为什么： 完成进化循环
  - 平衡**Exploitation+Exploration**
  - Exploitation--深度，在当前最好解的基础上继续深挖
  - Exploration--广度，尝试全新的方向，确保**多样性**，避免局部最优

AlphaEvolve应用实例

```python
# EVOLVE-BLOCK START
"""Image classification experiment in jaxline."""

import jax
...
# EVOLVE-BLOCK-END

...

# EVOLVE-BLOCK-START
class ConvNet(hk.Module):
  def __init__(self, num_classes): ...
  def __call__(self, inputs, is_training): ...


def sweep():
  return hyper.zipit([...])
# EVOLVE-BLOCK-END

...

def evaluate(eval_inputs) -> dict[str, float]:
  ...
  return metrics
```

(a)

```
Act as an expert software developer. Your task is to iteratively
improve the provided codebase. [...]

- Prior programs

Previously we found that the following programs performed well
on the task at hand:

top_1_acc: 0.796; neg_eval_log_loss: 0.230; average_score: 0.513

"""Image classification experiment in jaxline."""
[...]
class ConvNet(hk.Module):
  """Network."""

  def __init__(self, num_channels=32, num_output_classess=10):
    super().__init__()
    self._conv1 = hk.Conv2D(num_channels, kernel_shape=3)
    self._conv2 = hk.Conv2D(num_channels * 2, kernel_shape=3)
    self._conv3 = hk.Conv2D(num_channels * 4, kernel_shape=3)
    self._logits_module = hk.Linear(num_output_classes)
[...]
```

(b)

AlphaEvolve应用实例

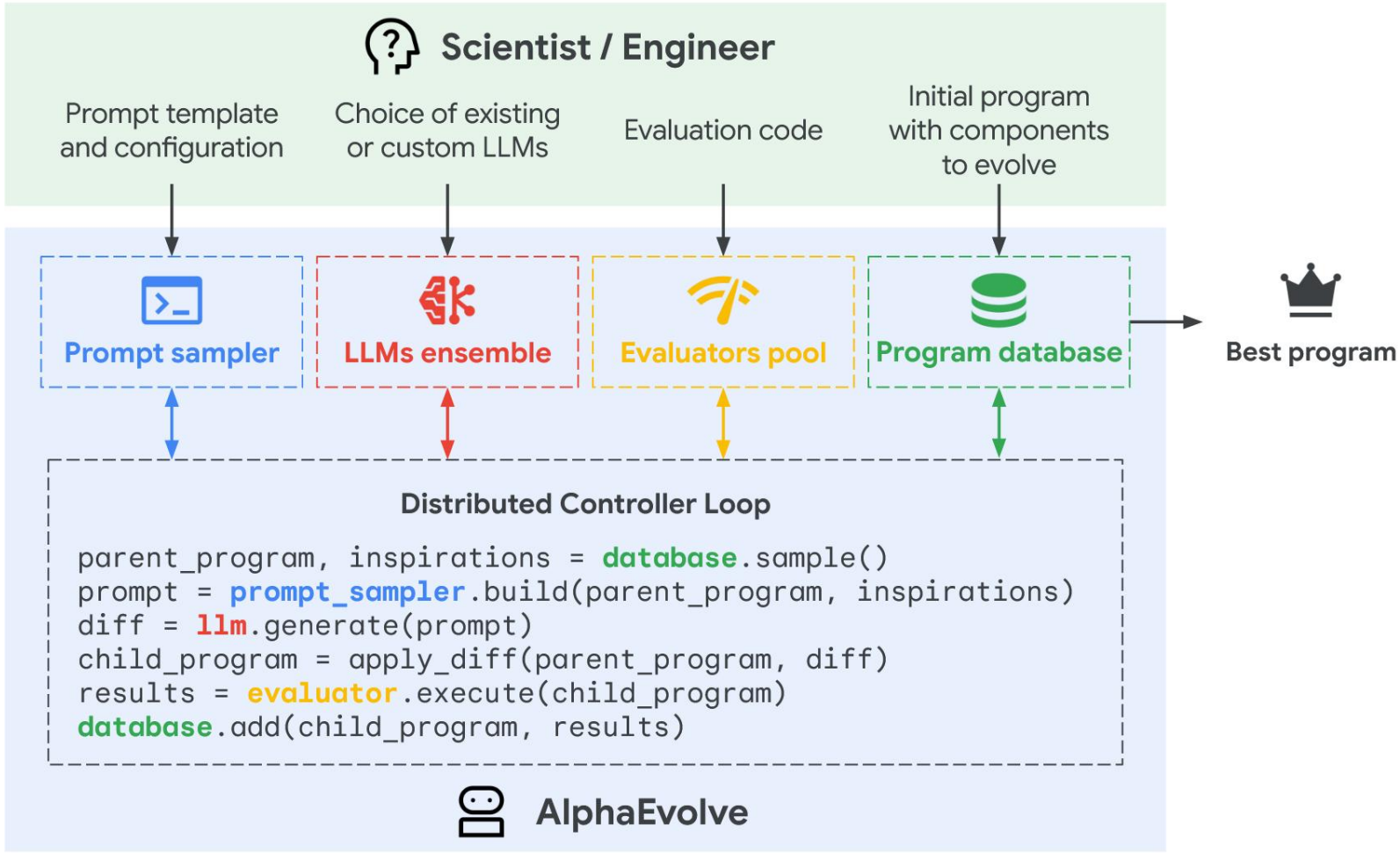
```
- Current program

Here is the current program we are trying to improve (you will
need to propose a modification to it below).

top_1_acc: 0.862; neg_eval_log_loss: 0.387; average_score: 0.624

"""Image classification experiment in jaxline."""
[...]
class ConvNet(hk.Module):
  """Network."""

  def __init__(self, num_channels=32, num_output_classes=10):
    super().__init__()
    self._conv1 = hk.Conv2D(num_channels, kernel_shape=3)
    self._block1 = ResNetBlock(num_channels)
    self._block2 = ResNetBlock(num_channels * 2, stride=2)
    self._block3 = ResNetBlock(num_channels * 4, stride=2)
    self._logits_module = hk.Linear(num_output_classes)
[...]
```

(b) 

```
The current model uses a simple ResNet architecture with only
three ResNet blocks. We can improve its performance by
increasing the model capacity and adding regularization. This
will allow the model to learn more complex features and
generalize better to unseen data. We also add weight decay to
the optimizer to further regularize the model and prevent
overfitting. AdamW is generally a better choice than Adam,
especially with weight decay.


<<<<<<< SEARCH
    self._block1 = ResNetBlock(num_channels)
    self._block2 = ResNetBlock(num_channels * 2, stride=2)
    self._block3 = ResNetBlock(num_channels * 4, stride=2)
=======
    self._block1 = ResNetBlock(num_channels)
    self._block2 = ResNetBlock(num_channels, stride=1)
    self._block3 = ResNetBlock(num_channels * 2, stride=2)
    self._block4 = ResNetBlock(num_channels * 2, stride=1)
    self._block5 = ResNetBlock(num_channels * 4, stride=2)
    self._block6 = ResNetBlock(num_channels * 4, stride=1)
>>>>>>> REPLACE
<<<<<<< SEARCH
  def optimizer(self, learning_rate):
    return optax.adam(learning_rate)
=======
  def optimizer(self, learning_rate):
    return optax.adamw(learning_rate, weight_decay=1e-4)
>>>>>>> REPLACE
```

(c) 

AlphaEvolve小结



AlphaEvolve发展过程的扩展视图

Outline

1 Background

2 Related work

3 Design

4 Experiments

5 Conclusion

实验结果

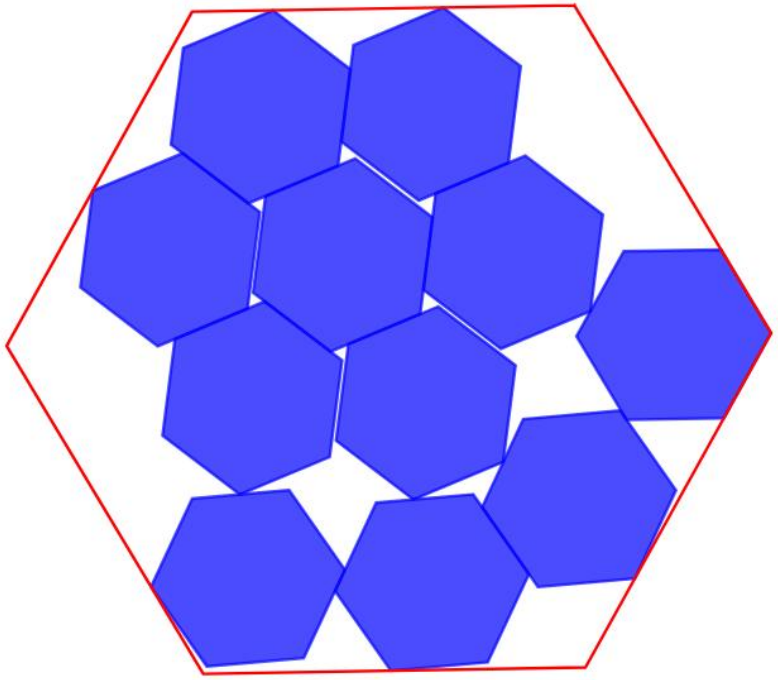
1. 发现更快的矩阵乘法算法

破解56年未解数学难题！

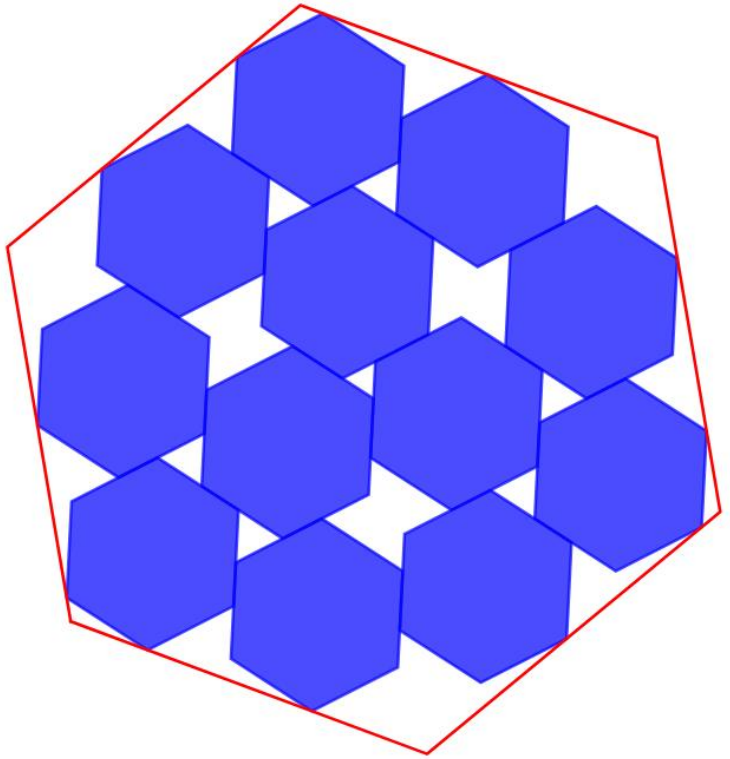
- **矩阵乘法**是计算机科学中的一个基础运算，寻找**更快的算法**一直是学界难题 。
- 重大突破：AlphaEvolve 发现了一种新算法，能用**48次**标量乘法完成两个 4x4 复数矩阵的相乘 。这是**56年来**首次在该领域对 **Strassen 算法（需要49次乘法）**的改进 。
- 广泛成果：在超过**50个**不同的矩阵尺寸上，AlphaEvolve 发现了**14种**优于当前最优**（SOTA）**水平的算法，并在其余多数情况下达到了已知最佳水平 。

实验结果

目标：将**n个边长为1**的、互不重叠的**正六边形**（单位六边形）放入一个更大的正六边形中，并使得这个外层大六边形的**边长尽可能小** 。



n=11，从3.943改进到了3.931

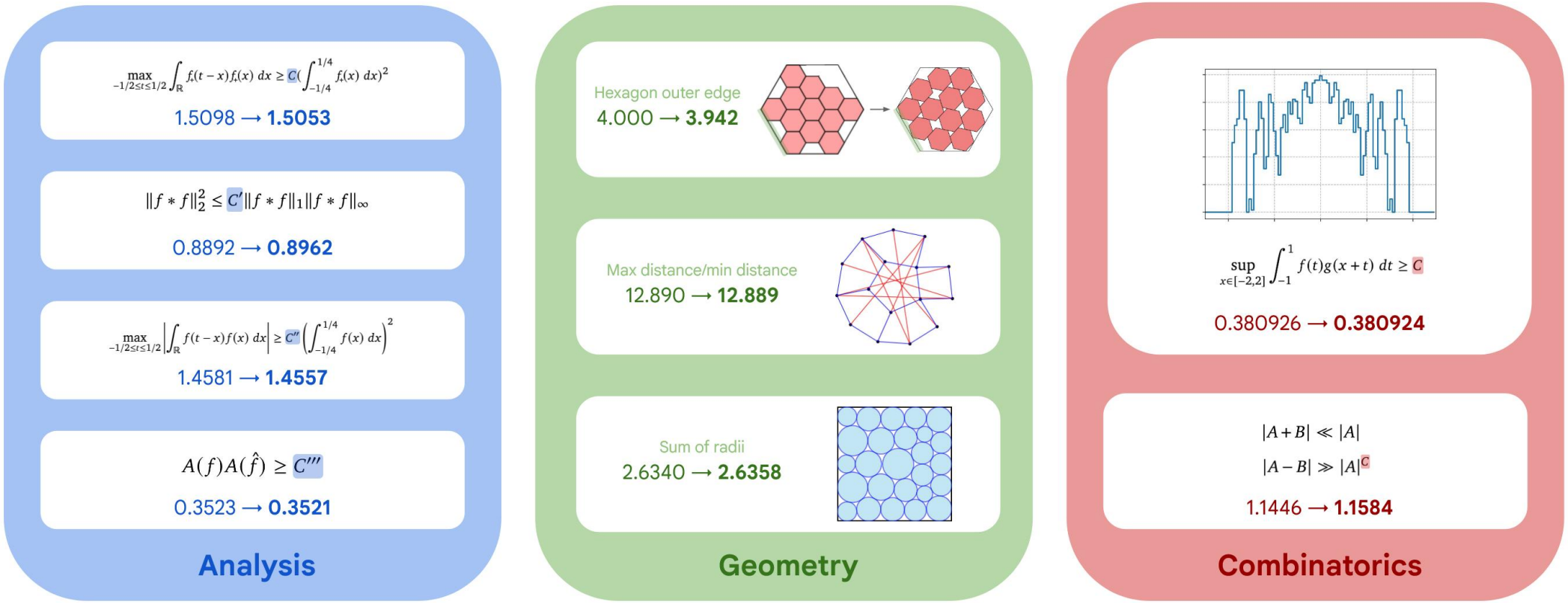


n=12，从4.0减小到了3.942

实验结果

2. 解决多个开放数学难题

AlphaEvolve被应用于超过**50个**来自不同数学分支的开放问题 。它大约在 **75%**的案例中重新发现了已知的最佳构造，并在约 **20%** 的问题上超越了现有SOTA水平，发现了新的、可被证明更优的构造。

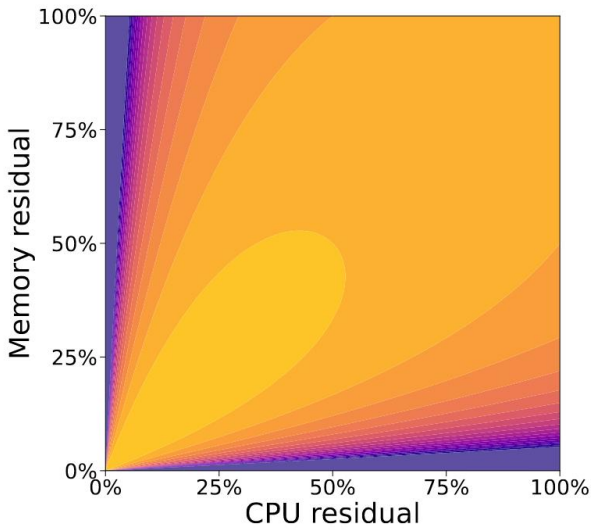


实验结果

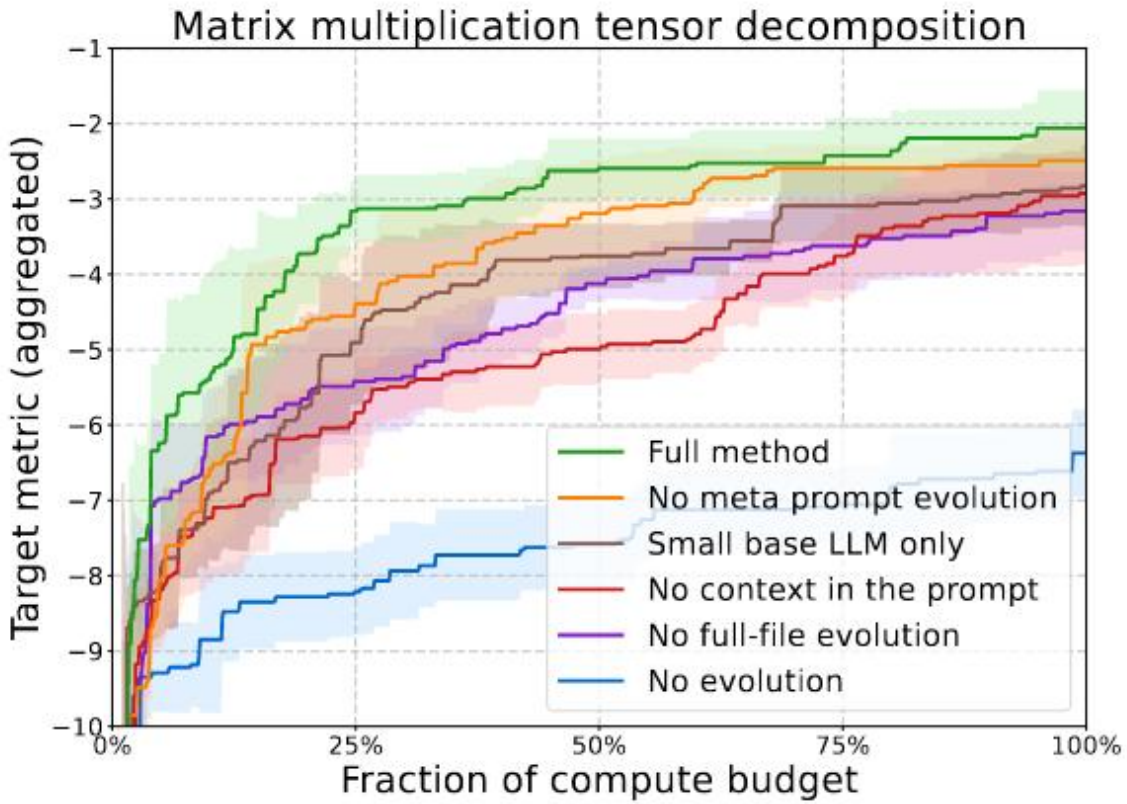
3. 优化谷歌的计算生态系统

- **数据中心调度：**该函数平均为谷歌全球计算资源挽回了**0.7%**的被“搁浅”的资源，相当于节省了大量本会被浪费的算力 。
- **加速Gemini模型训练：**AlphaEvolve优化了用于训练Gemini模型的TPU上的矩阵乘法内核的“切片”策略 。使内核的平均速度提升了**23%**，并使得Gemini模型的总训练时间减少了**1%** 。
- **辅助硬件电路设计**
- **优化编译器生成代码**

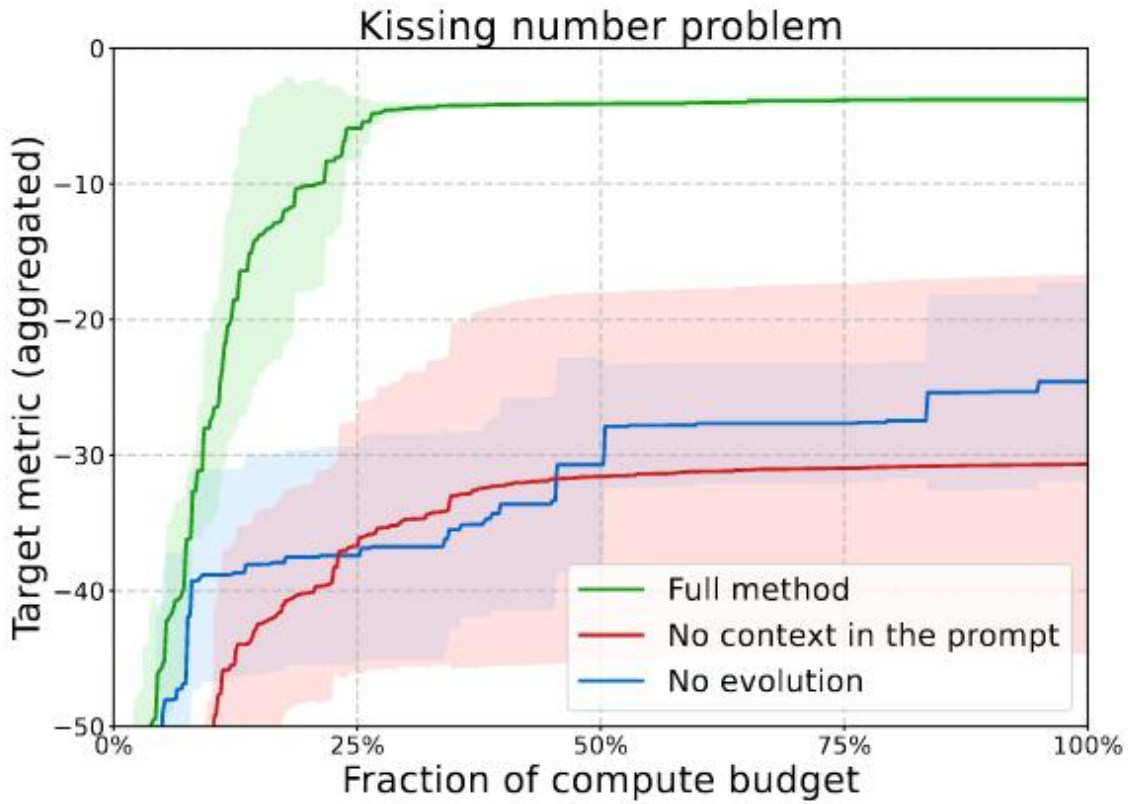
```python
def alpha_evolve_score(required, free):
  cpu_residual = required.cpu / free.cpu
  mem_residual = required.mem / free.mem

  return -1.0 * (cpu_residual + mem_residual +
                 mem_residual / cpu_residual +
                 cpu_residual / mem_residual)
```



消融实验



寻找低秩张量分解加速矩阵乘法



亲吻数问题

Outline

1 Background

2 Related work

3 Design

4 Experiments

5 Conclusion

Conclusion

- **Conclusion**
  - AlphaEvolve通过一个自动化的流程，利用**进化循环**逐步改进算法。像生物进化一样“优胜劣汰”，最终迭代出更优秀的算法，从而实现新的科学发现或技术突破。

- **Inspiration**

➢ 能不能用到我们的场景？
  - 代码生成：基于静态分析进行**评估**，通过分析**编译后的二进制文件或源代码**来评估其性能，根据日志、编译器反馈进行优化。
  - 记忆更新：模仿动态数据库，打造**动态记忆库**，结合进化算法

➢ 能不能进一步提高？
  - AlphaEvolve进化过程中产生的**优化算法“突变”**，作为高质量的训练数据，用来微调下一代LLM
  - AlphaEvolve很依赖**评估**函数，没有标准、需要实验？**人机回环、混合评估**

➢ 问题可以泛化吗？
  - AlphaEvolve可以看作**优化框架**，需要优化的问题可以尝试和AlphaEvolve结合

Q&A

2025.09.20