CodeEvolve: Evolutionary Agents for Scientific Discovery

AIDDA 2026

Henrique Assumpção[1,2], Diego Ferreira[1,2], Leandro Campos[1,2], Fabricio Murai[3]

[1]Inter Science - Inter&Co., Belo Horizonte, MG, Brazil
[2]Federal University of Minas Gerais, Belo Horizonte, MG, Brazil
[3]Worcester Polytechnic Institute, Worcester, MA, USA

June 2026

DCC
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
UFMG

PROGRAMA DE PÓS-GRADUAÇÃO
CIÊNCIA DA COMPUTAÇÃO
UFMG

inter

Summary of Contents

About

A bit about me:

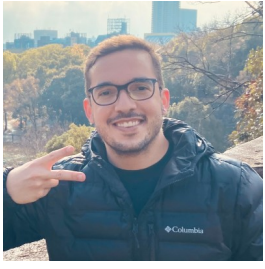
- ML Research Scientist at Inter&Co
- MS Candidate in Computer Science at UFMG
- Creator of CodeEvolve; Contributor to OpenEvolve

Co-Authors:

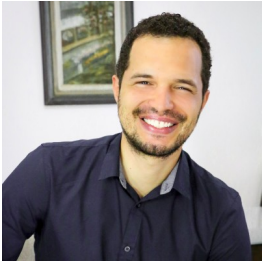
- Collaborative effort across Inter Science, UFMG and WPI.
- Specializing in NLP, Representation Learning and Recommender Systems.

H.A.

Fabricio Murai



Leandro Campos



Diego Ferreira

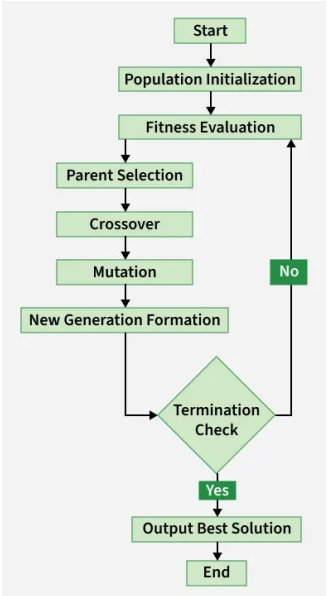
Goals of this talk

Core Objectives:

- **Democratize Algorithmic Discovery:** Evaluate evolutionary coding agents paired with open-source software (OSS) models to match frontier performance.
- **Commercial Independence:** Mitigate business risks and dependency on proprietary APIs (OpenAI/Anthropic) by maximizing open-weight model efficiency.
- **Inference-Time Scaling:** Provide practical, real-world development insights on orchestrating lightweight harnesses for affordable automated discovery.

Evolutionary Algorithms

- **Origins:** Modern GP was popularized by John Koza in 1992 as an extension of Genetic Algorithms.
- **The Process:**
  - *Selection*
  - *Crossover*
  - *Mutation*
- **Goal:** Automated discovery of executable code that optimizes a given fitness function.



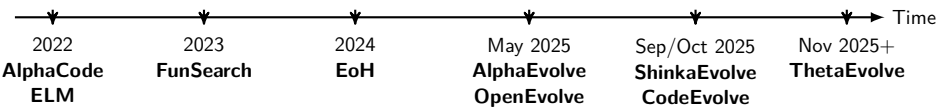
Evolutionary Loop for Agents

Solutions/Prompts $\rightarrow$ Selection $\rightarrow$ Crossover/Mutation (LLM) $\rightarrow$ Evaluation (Sandbox)

Iterative Code Refinement

- **1. Initialization:** A diverse population of initial solution attempts (prompts/code) is generated.
- **2. Selection:** Parent solutions are chosen based on some selection policy (e.g. fitness-based, uniformly random, etc).
- **3. Variation (LLM):** LLMs act as intelligent crossover and mutation operations, generating new solutions.
- **4. Fitness Evaluation:** New solutions are executed in a sandboxed environment, performance metrics and fitness are computed.
- **5. Termination:** The cycle repeats until the desired fitness is achieved or the computational budget is exhausted.

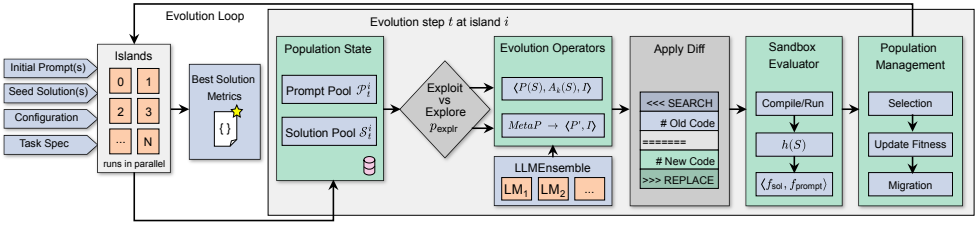
Overview of Frameworks



Overview of CodeEvolve

A Transparent Framework for Meta-Optimization

- **Island-Based GA:** Maintains diversity through isolated populations and parallel search trajectories.
- **Weighted LLM Ensemble:** Dynamically selects the best-performing models for specific mutation tasks.
- **Modular Operators:**
  - *Inspiration-based Crossover:* Recombines logic from top-tier "parent" programs.
  - *Meta-Prompting:* Evolves the instructions themselves to guide the search.
  - *Depth-based Refinement:* Targets specific code segments for local exploitation.



Architecture of the CodeEvolve Evolutionary Pipeline

LLM Backbones

Comparing Proprietary and Open-Weight Models

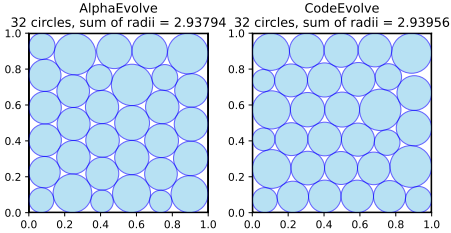
- Our experiments evaluate the scalability and reasoning capabilities of:
- **Gemini 2.5 (Flash/Pro):** Serving as the high-throughput, closed-source baseline for complex reasoning.
- **Qwen3-Coder-30B:** Representing the state-of-the-art in open-weight coding models.
- **Key finding:** Open-weight models often match or exceed closed-source performance at significantly lower compute costs.

Google Gemini 2.5

Alibaba Qwen3-Coder30B

Benchmark Problems

- **Geometric Packing:** Packing circles and hexagons in order to optimize a given metric.
- **Min-Max Distance Ratio:** Placing points in d-dimensional space to minimize the ratio of maximum to minimum distance.
- **Autocorrelation Inequalities:** Constructing optimal step-functions for convolution bounds in additive combinatorics.
- **Evaluation:** Programs are executed in a sandbox and scored against rigid mathematical fitness signals.



Overview of Results

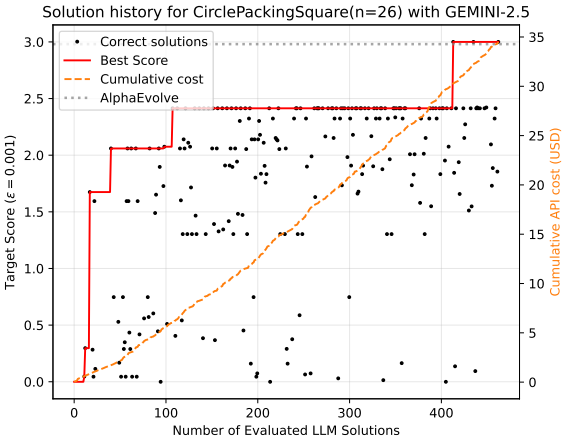
Problem	AlphaEvolve	CodeEvolve	
	Gemini 2.5	Qwen3-30B	Gemini 2.5
`CirclePackingSq` ($n = 26$) ($\uparrow$)	2.63586	**2.63598**	2.63597
`CirclePackingSq` ($n = 32$) ($\uparrow$)	2.93794	**2.93956**	2.93950
`CirclePackingRect` ($n = 21$) ($\uparrow$)	**2.36583**	2.36339	**2.36583**
`HexagonPacking` ($n = 11$) ($\downarrow$)	**3.93009**	4.02786	3.93794
`HexagonPacking` ($n = 12$) ($\downarrow$)	**3.94191**	4.01092	4.00001
`MinMaxDist` ($n = 16, d = 2$) ($\downarrow$)	12.88927	12.88925	**12.88923**
`MinMaxDist` ($n = 14, d = 3$) ($\downarrow$)	4.16585	4.16765	**4.16579**
`FirstAutocorrIneq` ($\downarrow$)	**1.50316**	1.51343	1.55438
`SecondAutocorrIneq` ($\uparrow$)	**0.96102**	0.88110	0.87067

Overview of Results

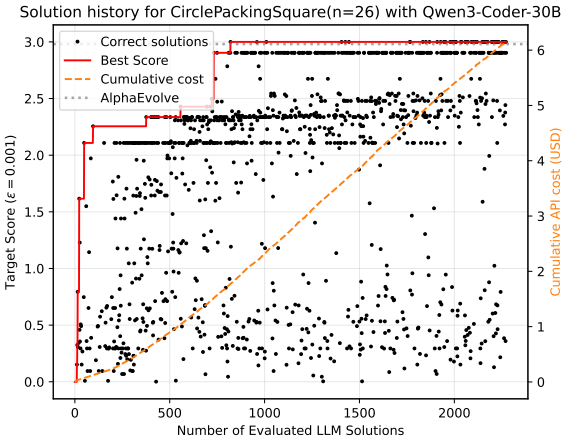
Some Observations:

- Ok, we can beat AlphaEvolve in **some** of the benchmark problems using a lightweight harness, **but do we actually need this harness**?
- Do we have enough evidence to justify trying to build harnesses that are better suited for OSS models?
- What can the average user expect in terms of costs?

Sample Efficiency

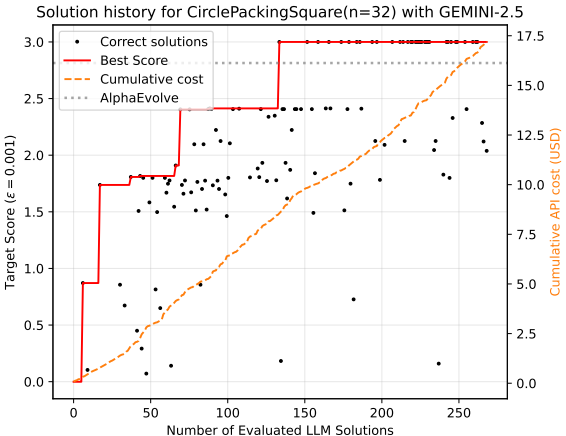


Gemini 2.5 Convergence Profile

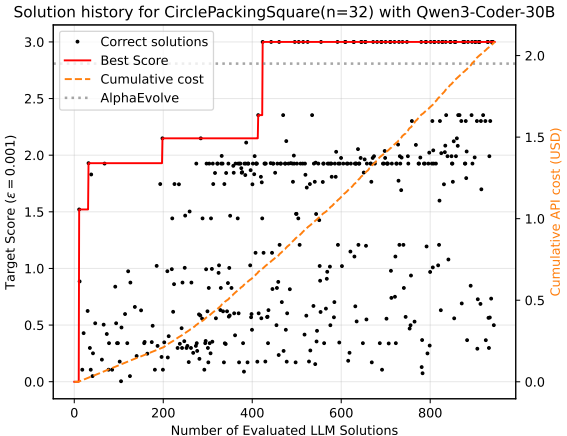


Qwen3-Coder-30B Convergence Profile

Sample Efficiency

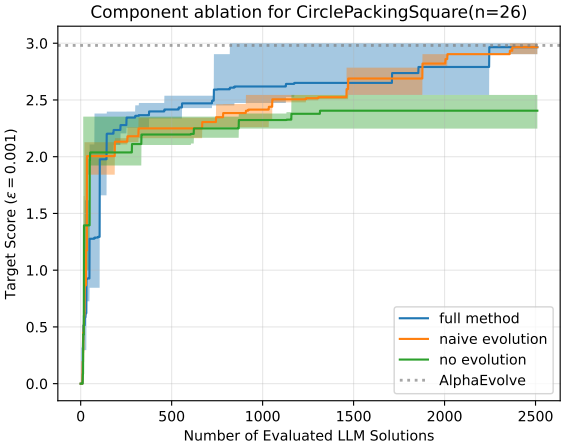


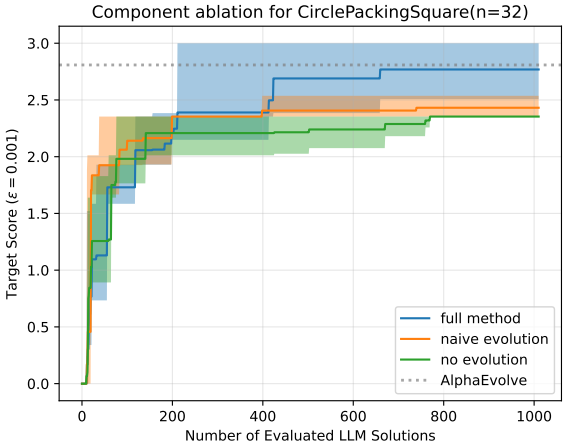
Gemini 2.5 Convergence Profile



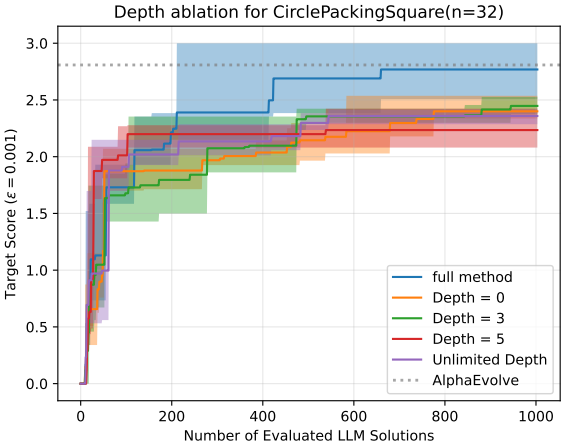
Qwen3-Coder-30B Convergence Profile

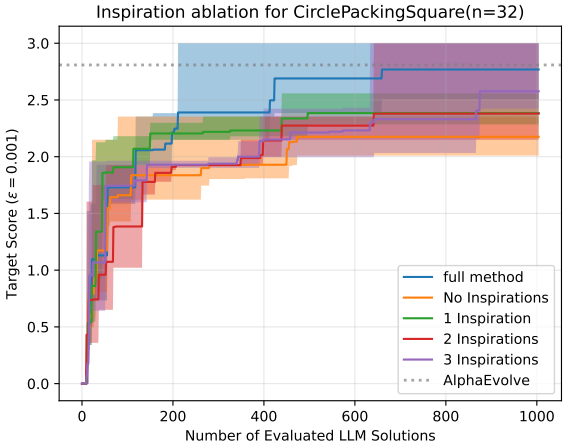
Does Evolution Matter?





Does Evolution Matter?





Current Limitations

Open Challenges in Automated Discovery:

- **Hyperparameter Sensitivity:** Calibration of island topologies, migration intervals, and operator depths introduces complex tuning layers.
- **Maintainability vs. Fitness:** Optimization solely rewards the raw performance metric, occasionally yielding dense, unrefactored codebases.
- **Compute Footprint:** Despite significant savings over cloud APIs, large-scale iterative runs demand intensive parallel hardware budgets.

Conclusion

The Power of Evolution-Augmented LLMs

- **Synergy is Key:** Combining evolutionary search with LLMs creates a discovery engine far more powerful than zero-shot prompting alone.
- **Evolution as an Equalizer:** Experiments show that the evolutionary loop allows OSS models like Qwen to match or surpass proprietary ones.
- **Strategic Independence:** This paradigm reduces dependency on closed-source APIs, offering a path toward high-performance, private, and reproducible automated discovery.
- **The Loop $>$ The Model:** The architecture of the search process is often more critical for discovery than the raw size of the underlying LLM.

Thank you!

Questions?

henriquesoares@dcc.ufmg.br
henriqueassumpcao.github.io

CodeEvolve GitHub