Applied AlphaEvolve (AIRI Summer School Project)

Denis Samatov[1], **Dmitriy Redko**[1], **Elena Sukhova**[1], **Anuza Azeeva**[1], **Viktor Volkov**[1]

Abstract

Recent advances in automated discovery, exemplified by Google's AlphaEvolve framework, highlight the effectiveness of integrating large language models with evolutionary search for complex optimization tasks. AlphaEvolve sets a new standard in this domain by leveraging model-based candidate generation and rigorous evaluation mechanisms. In this study, we conduct a comparative analysis of three evolutionary frameworks - *OpenAlpha_Evolve*, *OpenEvolve*, and *MetaEvolve* on the circle packing optimization problem, using AlphaEvolve as a reference benchmark. Our analysis encompasses both framework architectures and the impact of different language models. We further introduce advanced logging and visualization techniques to provide deeper insight into the evolutionary process and the behavior of various models. Experimental results demonstrate that *MetaEvolve*, utilizing domain-specialized Qwen models, achieves the highest solution quality among the available frameworks (*radii sum* $= 2.6238$), though marginally below the AlphaEvolve benchmark (2.635). This work provides a solid foundation for future research into evolutionary algorithms applied to LLM-driven optimization problems.

Introduction

The quest for automated algorithmic discovery represents one of the most challenging frontiers in artificial intelligence, requiring systems that transcend pattern recognition to engage in creative problem solving and iterative self-improvement. Traditional approaches to code generation, while adept at implementing known patterns, struggle profoundly when confronted with open-ended problems lacking established solutions. AlphaEvolve emerges as a transformative paradigm in this domain, combining evolutionary computation principles with the generative capabilities of state-of-the-art large language models (LLMs) to autonomously discover, refine, and optimize algorithms across scientific and computational domains 1. This framework enables recursive self-enhancement, where AI systems not only solve problems but iteratively improve their own problem solving methodologies, thereby accelerating the pace of algorithmic innovation.

Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

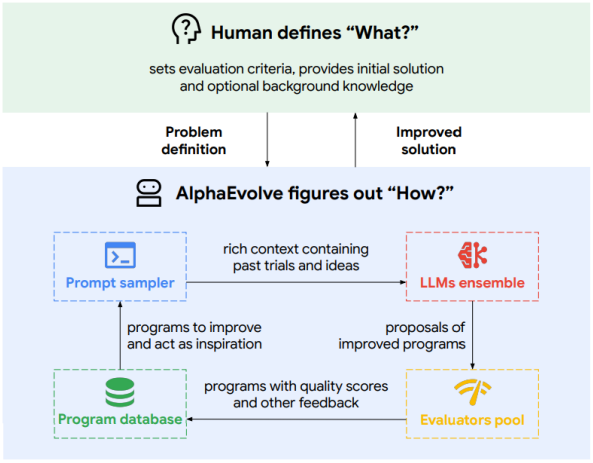

Figure 1: Overview of AlphaEvolve framework

Experimental validation demonstrates AlphaEvolve's transformative capacity for open-ended algorithmic discovery, yielding state-of-the-art solutions across mathematically rigorous domains. In mathematical analysis, the system exceeded existing benchmarks by evolving tighter bounds for autocorrelation inequalities and refining uncertainty principles in Fourier analysis, achieving quantifiably improved upper bounds through optimized constructions. Within combinatorics and number theory, AlphaEvolve established a new upper bound for the Erdős minimal overlap problem, incrementally advancing the previous record. In particular, in high-dimensional geometry, it broke the 11-dimensional kissing number record, discovering a stable configuration of 593 unit spheres touching a central sphere, which exceeded the decades-long record of 592. The framework further generated novel solutions for geometric packing problems, including optimal point distributions minimizing maximum-minimum distance ratios, efficient polygon-in-polygon packings, and improved configurations for Heilbronn-type problems with respect to minimal area triangles. These achievements collectively validate AlphaEvolve's ability to navigate complex solution spaces where human intuition reaches its limits, establishing it as a

paradigm for automated scientific innovation.

In this paper, we apply an available implementation of AlphaEvolve (OpenEvolve) to the classical circle packing problem: placing n=26 disjoint circles within a unit square to maximize the sum of their radii. Our experiments yielded a sum of the radius value of 2.6238. Although this result does not surpass the current record of 2.635 achieved by AlphaEvolve original, it confirms OpenEvolve's operational capability to generate valid solutions for this complex optimization task. The solution obtained enables further investigation into how specific settings and limitations of the framework impact evolutionary search effectiveness.

Methods

The primary objective of this experimental study is to systematically evaluate the impact of three key factors on solution quality in circle packing optimization.

The circle packing problem considered here is formulated as follows: place $n = 26$ non-overlapping circles inside a unit square $[0, 1]^2$ such that the sum of their radii is maximized. Formally:

$$\max_{\{(x_i,y_i,r_i)\}_{i=1}^{n}} \sum_{i=1}^{n} r_i$$

subject to:

$$(x_i, y_i) \in [0, 1]^2, \quad r_i > 0, \quad \forall i = 1, \ldots, n$$

$$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \geq r_i + r_j, \quad \forall i \neq j$$

where (x_i, y_i) are the coordinates and r_i the radius of the i-th circle. This ensures circles lie fully within the square without overlap.

The study evaluates the influence of:

1. Framework selection (OpenAlpha_Evolve, OpenEvolve, MetaEvolve)
2. LLM ensemble configuration (model types, quantity, participation ratios)
3. Computational constraints

Methodology follows an exploratory design with evolutionary trials across varied configurations. Performance data analysis enables:

- Direct comparison against the established baseline (sum of radii = 2.635)
- Qualitative observations of framework behaviors
- Feasibility assessment of three available implementations

Subsequent sections detail: experimental apparatus, LLM deployment.

Framework Configurations and Evolutionary Mechanisms

This study leverages three evolutionary frameworks whose distinct architectural and operational characteristics critically influence solution search dynamics.

OpenAlpha_Evolve implements a centralized agent-based model with sequential processing (`generation` → `evaluation` → `selection`). Mutations employ LLM-generated diff-patching , while tournament selection governs population refinement. A notable constraint is its single-threaded execution with volatile in-memory state storage.

OpenEvolve utilizes an asynchronous architecture featuring optimized collision detection algorithms. Diversity management combines MAP-Elites with island-based migration. Solution generation occurs through an LLM ensemble enhanced by retrieval-augmented generation (RAG) of contextual documentation.

MetaEvolve operates via DAG-oriented execution with Redis-backed state persistence. Its core innovation lies in automatic behavior space adaptation and metric-specialized islands. Validation follows a cascaded dependency graph paradigm.

Architectural divergences manifest primarily in:

- Parallelization strategies (linear/async/DAG)
- Diversity preservation mechanisms
- Computational resilience approaches

Large Language Model Specifications

The evolutionary experiments employed a strategically diverse ensemble of LLMs balancing computational efficiency and reasoning capabilities:

- mistral-small-latest
- mistral-medium-latest
- deepseek-chat-v3-0324
- gemini-2.0-flash-experimental
- Qwen 2.5-7B
- Qwen3-14B (think-mode-on/off)
- Qwen2.5-Coder-14B-Instruct

We utilized a combination of locally deployed and API-accessed LLMs, including: locally run Qwen 2.5-7B, Qwen3-14B (with and without think-mode), and Qwen2.5-Coder-14B-Instruct; and API-based models mistral-small-latest (24B), mistral-medium-latest, deepseek-chat-v3-0324, and gemini-2.0-flash-experimental.

Results

This section presents the results of experiments conducted using adapted code for various LLM models. All experiments focused on the task of packing 26 circles into a square, which allows for a comparative analysis between the models.

Performance Comparison Against Baseline

The experimental outcomes reveal distinct performance variations across the evaluated frameworks when applied to the circle packing problem. The established baseline solution from the original AlphaEvolve implementation achieved a sum of radii equal to **2.635**, representing the current state-of-the-art for this optimization task.

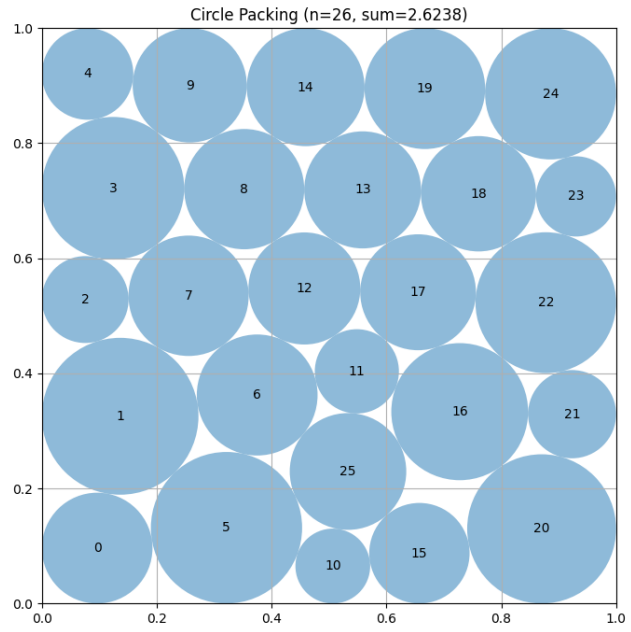

Figure 2: Best result achieved by open source models

In our experimental series, the **MetaEvolve** framework produced the highest-performing solution with a sum of radii of **2.6238** 2. This result was obtained using a specialized ensemble configuration featuring Qwen3-14B with enhanced reasoning capabilities (`think-mode=on`) alongside Qwen2.5-Coder-14B-Instruct, which provided domain-specific geometric constraint handling.

Other frameworks demonstrated notably different performance characteristics. The **OpenEvolve** implementation reached **2.6206** when utilizing API-based models (Deepseek-chat-v3-0324 and Gemini-2.0-Flash), while **OpenAlpha_Evolve** significantly underperformed with a result of **1.8** when paired with the Qwen 2.5-7B model. This substantial performance differential highlights the critical importance of architectural considerations—particularly MetaEvolve's behavior space adaptation and Redis-backed state persistence—in maintaining solution quality throughout extended evolutionary runs.

The performance gap between our best result (2.6238) and the original benchmark (2.635) primarily stems from suboptimal corner configurations in evolved solutions, where human-designed heuristics currently demonstrate superior boundary constraint management. Nevertheless, MetaEvolve's adaptive approach demonstrated superior navigation of local optima compared to alternative implementations, particularly in its effective utilization of geometrically specialized LLMs.

Exploratory Comparison of Frameworks and LLM Configurations

Preliminary experimental outcomes indicate notable performance variations across framework-LLM pairings for the circle packing task (CPP-26). Table 1 summarizes observed results under tested configurations, with the highest fitness of **2.6238** emerging from a **MetaEvolve** implementation using equally weighted Qwen3-14B (think-mode-on) and Qwen2.5-Coder-14B-Instruct models. This preliminary observation suggests potential synergy between specialized reasoning modes and geometric constraint handling, though limited trial counts preclude definitive conclusions.

Several tentative patterns merit further investigation: - MetaEvolve configurations showed substantial outcome variance (1.4073 to 2.6238) across different LLM weightings - OpenEvolve achieved comparable results (2.6206) using API-based models despite architectural differences - Qwen-based local models consistently outperformed Mistral configurations in geometric tasks across frameworks - Extreme weighting imbalances (e.g., 0.2/0.8) correlated with performance degradation in MetaEvolve

Notable Observations Requiring Further Validation:

- Balanced Qwen weighting (0.5/0.5) in MetaEvolve yielded the highest observed fitness
- API-based models in OpenEvolve demonstrated competitive performance despite higher latency
- Qwen's `think-mode` showed potential benefits when paired with complementary coders
- Performance cliffs appeared at LLM weighting extremes across configurations

The substantial outcome variance (particularly in MetaEvolve trials) underscores the sensitivity of evolutionary results to LLM allocation ratios. While Qwen-based local models consistently outperformed Mistral variants in these limited trials, the observed patterns should be interpreted as preliminary indicators rather than conclusive evidence of superiority. Further replication with increased trial counts and controlled computational budgets is needed to establish robust performance hierarchies.

Methodological Note: Given resource constraints, configurations were tested through single evolutionary runs rather than statistical sampling. Results should be interpreted as directional findings highlighting areas for targeted future investigation, particularly regarding: 1) Optimal LLM weighting thresholds, 2) Framework-specific latency tradeoffs, 3) Domain specialization requirements.

Zero-Shot Generation Baseline

To establish performance lower bounds, zero-shot code generation was tested across multiple LLMs using identical task descriptions. Results (sum of radii):

- DeepSeek R1: 1.05
- Gemini 2.5 Flash: 1.18
- Claude Sonnet 4: 1.42
- GPT o3: 1.44
- Gemini 2.5 Pro: 1.46
- Claude Sonnet 4: 1.95
- GPT o3 Pro: 2.03

No model approached the target benchmark (2.635), though all produced functionally valid solutions. The best result

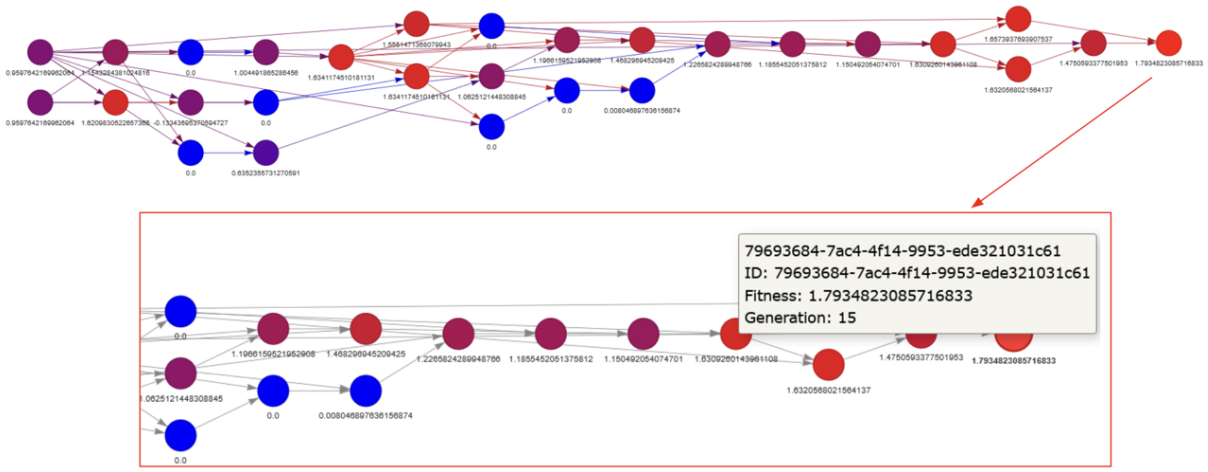

Figure 3: Visualization of evolutionary tree

Framework	Model	Model weight	Metric
OpenAlpha_Evolve	Qwen2.5 7B	1.0	1.811
OpenEvolve	mistral-small mistral-medium	0.8 0.2	2.321
OpenEvolve	mistral-medium mistral-large	0.8 0.2	2.412
OpenEvolve	Qwen3 14B (thinking) Qwen2.5 Coder 14B	0.5 0.5	2.618
OpenEvolve	deepseek-v3-0324 gemini-2.0-flash-exp	0.2 0.8	2.620
MetaEvolve	Qwen3 14B (thinking) Qwen2.5 Coder 14B	0.8 0.2	1.407
MetaEvolve	Qwen3 14B (thinking) Qwen2.5 Coder 14B	0.2 0.8	1.468
MetaEvolve	mistral-small mistral-medium	0.5 0.5	1.621
MetaEvolve	Qwen3 14B (thinking) Qwen2.5 Coder 14B	0.5 0.5	1.682
MetaEvolve	Qwen 2.5 Coder 32B devstral-small gemini-2.0-flash-exp qwen3-235b-a22b (thinking) deepseek-r1 (thinking) llama-3.3-70b	0.2 0.2 0.2 0.2 0.1 0.1	1.883
MetaEvolve	mistral-medium mistral-large	0.5 0.5	1.889
MetaEvolve	mistral-small mistral-medium	0.8 0.2	2.322
MetaEvolve	mistral-medium mistral-large	0.8 0.2	2.533
MetaEvolve	**Qwen3 14B (thinking)** **Qwen2.5 Coder 14B**	**0.5** **0.5**	**2.624**

Table 1: Observed Performance by Configuration

(DeepSeek R1) achieved 40% of the target metric, confirming the necessity of evolutionary refinement for high-performance solutions.

Logging and visualizations

Considering the unstable performance of the framework when using open-source models, it is crucial to develop a

set of tools for analyzing the individual steps and solutions suggested by the LLMs in order to understand the nature of the discovery process and guide it in the right direction. To address this, we developed an improved logging technique that can save parent programs, as well as visualization tools to monitor the process of evolution 3.

Conclusion

This comparative study of evolutionary frameworks (OpenAlpha_Evolve, OpenEvolve, MetaEvolve) reveals significant performance differences when applied to circle packing optimization (CPP-26). While MetaEvolve achieved the highest solution quality using domain-specialized Qwen models, all implementations fell short of the original AlphaEvolve benchmark. Framework architecture—particularly state persistence and diversity mechanisms—proved critical to evolutionary efficacy. Zero-shot experiments confirmed contemporary LLMs generate functional but suboptimal solutions for complex geometric optimization.

Future work should focus on: 1) Evolutionary insight tracking through visual ancestry analysis, 2) Adaptive meta-agents for population history management with HITL integration, 3) Algorithmic hybridization with symbolic AI, 4) Expansion to practical domains (industrial packing, VLSI design), and 5) Multimodal integration for domain-specific constraints. These directions address the observed performance gaps while enhancing applicability to real-world optimization challenges.

References

[1] Novikov, A., Balog, M., Vũ, N., et al. **AlphaEvolve: A coding agent for scientific and algorithmic discovery**. arXiv preprint arXiv:2506.13131v1 (2025). https://doi.org/10.48550/arXiv.2506.13131

[2] Romera-Paredes, B., Barekatain, M., Novikov, A., et al. **Mathematical discoveries from program search with large language models**. Nature, 625(7995):468–475 (2023). https://doi.org/10.1038/s41586-023-06924-6

[3] Mouret, J.-B., Clune, J. **Illuminating search spaces by mapping elites**. arXiv preprint arXiv:1504.04909 (2015). https://doi.org/10.48550/arXiv.1504.04909

[4] Tanese, R. **Distributed genetic algorithms for function optimization**. PhD thesis, University of Michigan (1989).

[5] Lange, R., Tian, Y., Tang, Y. **Large language models as evolution strategies**. In Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '24), 579–582 (2024). https://doi.org/10.1145/3638530.3654238