

OpenEvolve

Asankhaya Sharma

<https://github.com/codelion/openevolve>

AlphaEvolve: A coding agent for scientific and algorithmic discovery

Alexander Novikov^{*}, Ngàn Vŭ^{*}, Marvin Eisenberger^{*}, Emilien Dupont^{*}, Po-Sen Huang^{*}, Adam Zsolt Wagner^{*}, Sergey Shirobokov^{*}, Borislav Kozlovskii^{*}, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli and Matej Balog^{*}
Google DeepMind¹

In this white paper, we present *AlphaEvolve*, an evolutionary coding agent that substantially enhances capabilities of state-of-the-art LLMs on highly challenging tasks such as tackling open scientific problems or optimizing critical pieces of computational infrastructure. *AlphaEvolve* orchestrates an autonomous pipeline of LLMs, whose task is to improve an algorithm by making direct changes to the code. Using an evolutionary approach, continuously receiving feedback from one or more evaluators, *AlphaEvolve* iteratively improves the algorithm, potentially leading to new scientific and practical discoveries. We demonstrate the broad applicability of this approach by applying it to a number of important computational problems. When applied to optimizing critical components of large-scale computational stacks at Google, *AlphaEvolve* developed a more efficient scheduling algorithm for data centers, found a functionally equivalent simplification in the circuit design of hardware accelerators, and accelerated the training of the LLM underpinning *AlphaEvolve* itself. Furthermore, *AlphaEvolve* discovered novel, provably correct algorithms that surpass state-of-the-art solutions on a spectrum of problems in mathematics and computer science, significantly expanding the scope of prior automated discovery methods (Romera-Paredes et al., 2023). Notably, *AlphaEvolve* developed a search algorithm that found a procedure to multiply two 4×4 complex-valued matrices using 48 scalar multiplications; offering the first improvement, after 56 years, over Strassen's algorithm in this setting. We believe *AlphaEvolve* and coding agents like it can have a significant impact in improving solutions of problems across many areas of science and computation.

Scientist / Engineer

Prompt template and configuration

Choice of existing or custom LLMs

Evaluation code

Initial program with components to evolve



Prompt sampler



LLMs ensemble



Evaluators pool



Program database



Best program

Distributed Controller Loop

```
parent_program, inspirations = database.sample()
prompt = prompt_sampler.build(parent_program, inspirations)
diff = llm.generate(prompt)
child_program = apply_diff(parent_program, diff)
results = evaluator.execute(child_program)
database.add(child_program, results)
```



AlphaEvolve

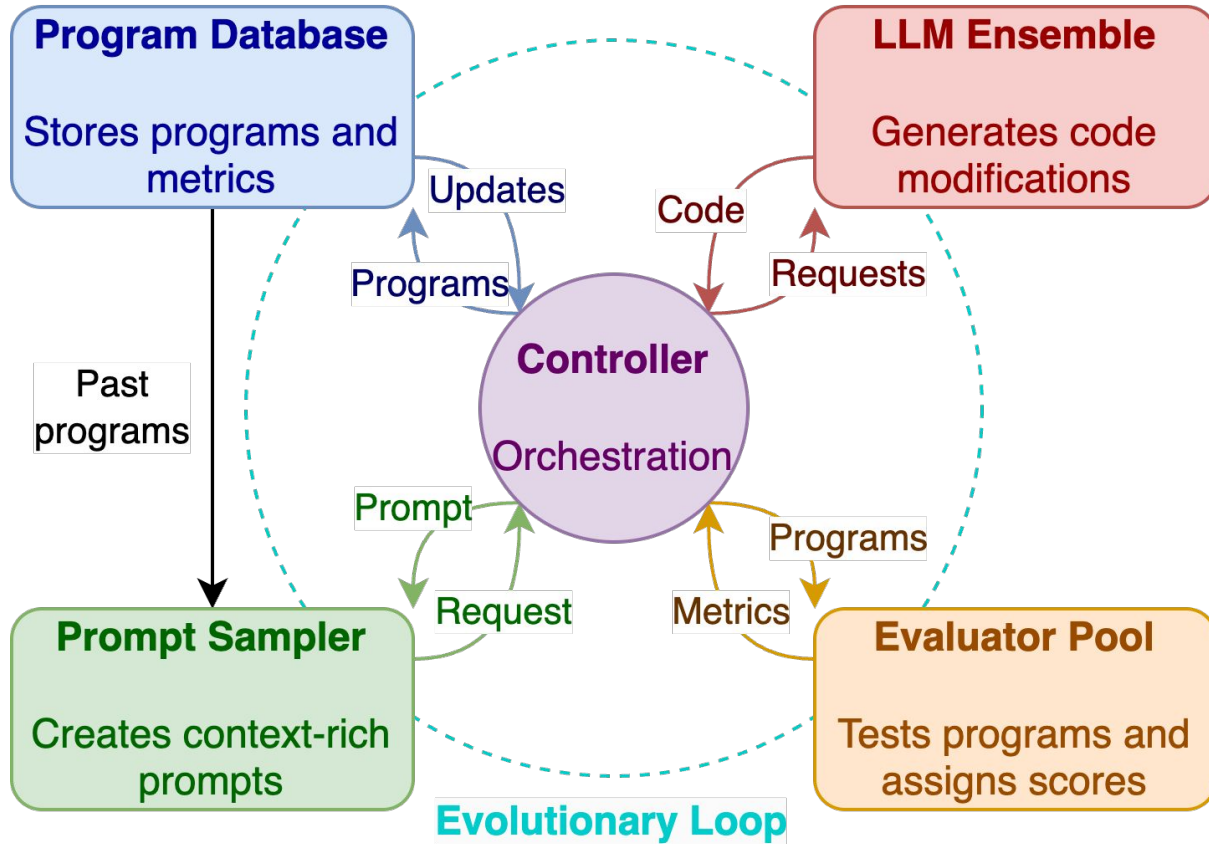
OpenEvolve



Evolutionary Coding Agent

```
def evolve(code):  
    while not optimal:  
        code = mutate(code)  
        evaluate(code)
```

OpenEvolve Architecture



Asynchronous pipeline optimized for maximum throughput

[Introduction](#)[Basic Genetic Algorithm](#)[Island Model GA](#)[MAP-Elites](#)[LLM-Powered Evolution](#)

Welcome to EvoVisual!

Dive into the fascinating world of evolutionary computation. This interactive tool helps you visualize and understand key algorithms and concepts, from basic Genetic Algorithms to cutting-edge ideas like LLM-powered evolution.

</> Basic Genetic Algorithm

Explore the core mechanics of evolution: selection, crossover, and mutation. Watch a population of simple strings evolve towards a target, and see operators in action step-by-step.

[Explore Basic Genetic Algorithm](#)

📁 Island Model GA

See how dividing a population into sub-populations (islands) with occasional migration can improve search diversity and find solutions faster for some problems.

[Explore Island Model GA](#)

🧠 MAP-Elites

Discover how MAP-Elites (Multidimensional Archive of Phenotypic Elites) maintains a collection of diverse, high-performing solutions across different behavioural niches, avoiding premature convergence.

[Explore MAP-Elites](#)

💬 LLM-Powered Evolution

Learn conceptually how Large Language Models (LLMs) can revolutionize evolutionary algorithms by acting as intelligent crossover and mutation operators, especially for complex structures like code.

[Explore LLM-Powered Evolution](#)

Circle Packing Problem (n=26)

The circle packing problem involves placing n non-overlapping circles inside a container (in this case, a unit square) to optimize a specific metric.

For this example:

- We pack exactly 26 circles
- Each circle must lie entirely within the unit square
- No circles may overlap
- We aim to maximize the sum of all circle radii

According to the AlphaEvolve paper, a solution with a sum of radii of approximately 2.635 is achievable for $n=26$.

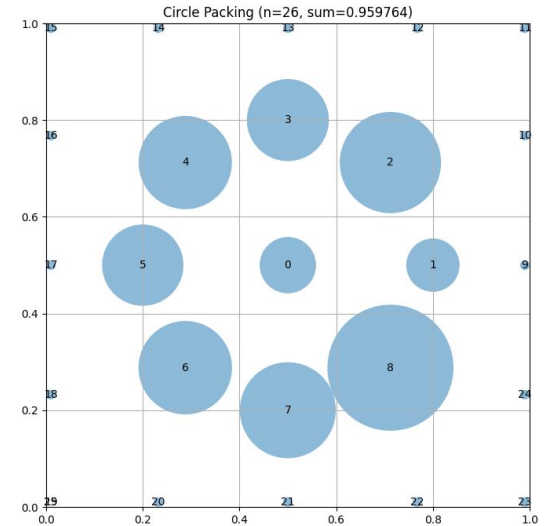
Our goal was to match or exceed this result.

Initial Program

```
# Initial attempt
# Place a large circle in the center
centers[0] = [0.5, 0.5]

# Place 8 circles around it in a ring
for i in range(8):
    angle = 2 * np.pi * i / 8
    centers[i + 1] = [0.5 + 0.3 * np.cos(angle), 0.5 + 0.3 *
np.sin(angle)]

# Place 16 more circles in an outer ring
for i in range(16):
    angle = 2 * np.pi * i / 16
    centers[i + 9] = [0.5 + 0.7 * np.cos(angle), 0.5 + 0.7 *
np.sin(angle)]
```



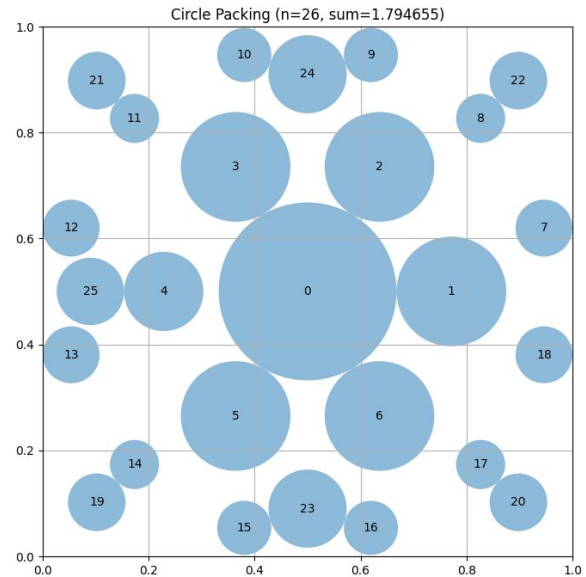

```

# Generation 10
# Parameters for the arrangement (fine-tuned)
r_center = 0.1675 # Central circle radius

# 1. Place central circle
centers[0] = [0.5, 0.5]
radii[0] = r_center

# 2. First ring: 6 circles in hexagonal arrangement
r_ring1 = 0.1035
ring1_distance = r_center + r_ring1 + 0.0005 # Small gap for
stability
for i in range(6):
    angle = 2 * np.pi * i / 6
    centers[i+1] = [
        0.5 + ring1_distance * np.cos(angle),
        0.5 + ring1_distance * np.sin(angle)
    ]
    radii[i+1] = r_ring1

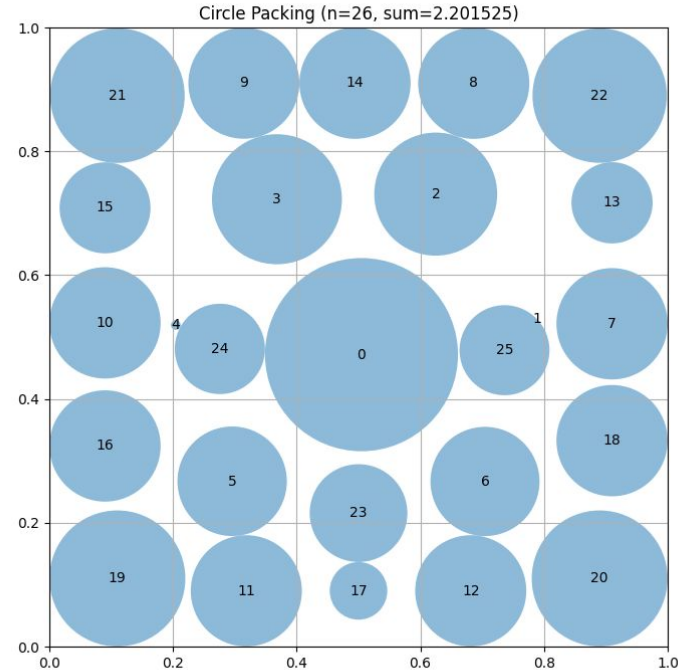
```



After 200+ iterations

```
# Generation 100
# Row 1: 5 circles
centers[0] = [0.166, 0.166]
centers[1] = [0.333, 0.166]
centers[2] = [0.500, 0.166]
centers[3] = [0.667, 0.166]
centers[4] = [0.834, 0.166]

# Row 2: 6 circles (staggered)
centers[5] = [0.100, 0.333]
centers[6] = [0.266, 0.333]
# ... additional circles
```



```

# Final solution with scipy.optimize
def construct_packing():
    # ... initialization code ...

    # Objective function: Negative sum of radii (to maximize)
    def objective(x):
        centers = x[:2*n].reshape(n, 2)
        radii = x[2*n:]
        return -np.sum(radii)

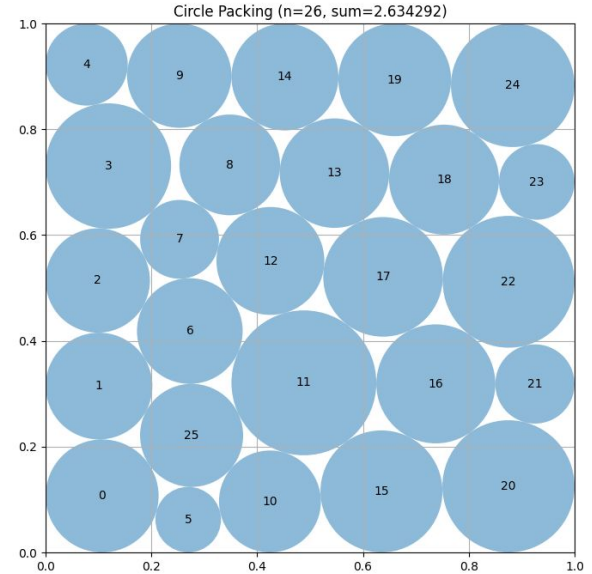
    # Constraint: No overlaps and circles stay within the unit square
    def constraint(x):
        centers = x[:2*n].reshape(n, 2)
        radii = x[2*n:]

        # Overlap constraint
        overlap_constraints = []
        for i in range(n):
            for j in range(i + 1, n):
                dist = np.sqrt(np.sum((centers[i] - centers[j])**2))
                overlap_constraints.append(dist - (radii[i] + radii[j]))

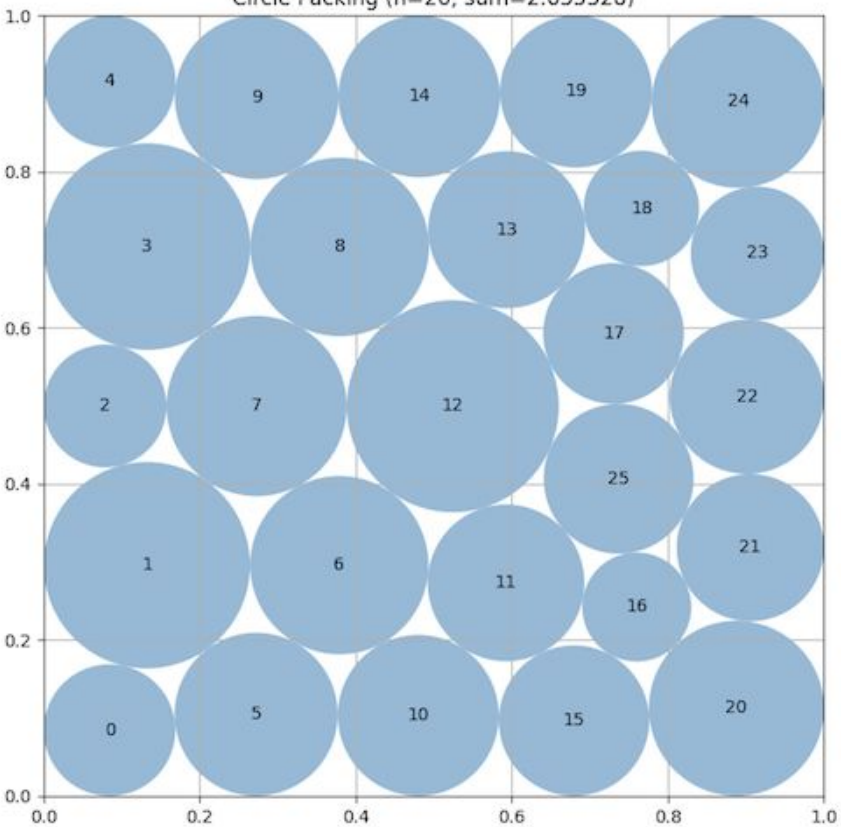
        # ... boundary constraints ...

    # Optimization using SLSQP
    result = minimize(objective, x0, method='SLSQP', bounds=bounds,
constraints=constraints)

```

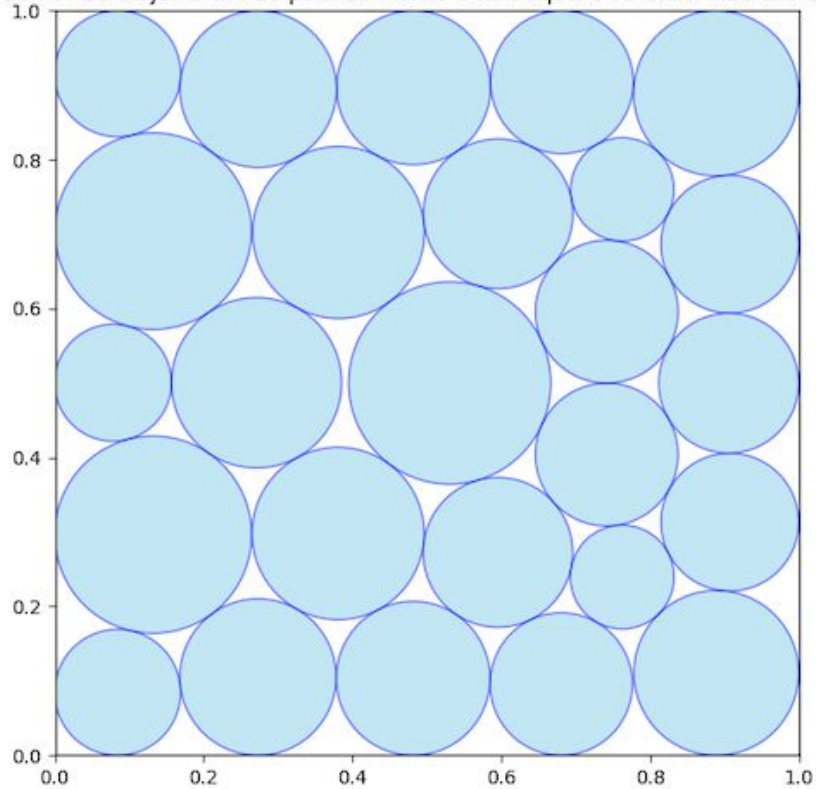


Circle Packing (n=26, sum=2.635328)



OpenEvolve

A collection of 26 disjoint circles packed inside a unit square to maximize the sum of radii



AlphaEvolve

Things to watch out for!

- Choosing the right abstraction at which to do the evolutionary search
- Preventing and allowing the use of existing libraries and APIs
- Guiding a population of candidate programs via prompting v/s a single program
- Robust cascading evaluations
- Requires human ingenuity in formulating the problem

New paradigm

- For inference time scaling of LLMs
- Distinct from existing sequential or parallel test time computing approaches
- Combines genetic algorithms driven search with LLMs for evolutionary coding agents
- Distill evolutionary agents to next version of base LLMs

Thank You!

- Questions?
- Links
 - OpenEvolve - <https://github.com/codelion/openevolve>
 - EvoVisual - <https://evovisual-advanced-evolutionary-concepts-577160257370.us-west1.run.app/>