AlphaEvolve: where language models meet mathematical discovery

Abdelwahid Benslimane

wahid.benslimane@gmail.com

Introduction

On May 14, 2025, Google DeepMind officially introduced AlphaEvolve, an AI system that combines Gemini-based language models with evolutionary algorithms to autonomously discover and optimize algorithms across a range of scientific and technical domains (https://deepmind.google/discover/blog/alphaevolve-a-gemini-powered-coding-agent-for-designing-advanced-algorithms/). This announcement comes with major breakthroughs in the field of mathematics.

The first breakthrough involves the resolution of the Kissing Number problem, which I won't go into detail here, but it's a centuries-old mathematical question. The problem seeks to determine the maximum number of identical spheres that can touch a central sphere without overlapping. In an 11-dimensional space, the exact number of spheres is still unknown, but the range within which this number lies is established. The upper bound is 868, while AlphaEvolve has contributed to the problem by raising the lower bound from 592 to 593.

AlphaEvolve has also enhanced Strassen's matrix multiplication algorithm by further reducing the number of operations required. To illustrate this with a concrete example, which I will explain in more detail later in this article, Strassen reduced the complexity from $O(n^3)$ to $O(n^{2.81})$, enabling the multiplication of two 2 $\times$ 2 matrices, which traditionally required 8 multiplications using the standard method, to be achieved with just 7 multiplications, at the cost of a few additional additions. Of course, the larger the matrix, the more significant the gain. AlphaEvolve has built upon this by developing new techniques that optimize this process even further. These advancements lead to significant performance gains, particularly in fields such as artificial intelligence and big data processing, where matrix multiplication is crucial. As a result, AlphaEvolve's algorithms allow for faster and more efficient calculations, optimizing modern computing systems.

Strassen’s Algorithm

Strassen's algorithm applies specifically to the multiplication of square matrices, that is, matrices of dimension $n \times n$. It works optimally when the matrices have sizes of $2^k \times 2^k$ (for example, 2×2, 4×4, 8×8, etc.), as it recursively divides the matrices into four equal submatrices.

It only becomes more efficient than the naive method for large matrices (typically $n \geq 128$ or more), since for small sizes, the overhead from the additional additions can outweigh the benefit of reducing the number of multiplications. In the following, I will show how Strassen’s algorithm allows you to multiply two 2×2 matrices using only 7 multiplications, compared to the 8 multiplications required by the standard method.

Let's do the math...

Let’s define matrices A and B:

A = $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$

B = $\begin{bmatrix} e & f \\ g & h \end{bmatrix}$

We aim to calculate the following matrix product:

M = $A \cdot B = \begin{bmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{bmatrix}$

This is the standard matrix multiplication, which requires 8 multiplications and 4 additions. Now, let’s see how Strassen’s method works. It begins by computing 7 intermediate products, defined as follows:

$$\begin{aligned}
&M_1 = (a + d)(e + h) \\
&M_2 = (c + d)e \\
&M_3 = a(f - h) \\
&M_4 = d(g - e) \\
&M_5 = (a + b)h \\
&M_6 = (c - a)(e + f) \\
&M_7 = (b - d)(g + h)
\end{aligned}$$

These intermediate products are then combined using the following formulas to calculate the elements of matrix M:

$$\begin{aligned}
&M_{11} = M_1 + M_4 - M_5 + M_7 \\
&M_{12} = M_3 + M_5 \\
&M_{21} = M_2 + M_4 \\
&M_{22} = M_1 - M_2 + M_3 + M_6
\end{aligned}$$

The resulting matrix M is:

$$M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}$$

How AlphaEvolve discovered more efficient matrix multiplication algorithms

AlphaEvolve was able to discover more efficient methods that reduce the number of multiplications even further (though the exact number depends on the matrix size) by combining reinforcement learning and evolutionary search, two core approaches in artificial intelligence. These algorithms, generated by LLMs, are represented as mathematical structures (such as computation graphs), which the system modifies over time through mechanisms analogous to genetics: mutation, crossover, and natural selection. The most efficient variants, those that perform matrix multiplication with the fewest operations, are preserved and refined over successive generations.

In parallel, AlphaEvolve learns to intelligently navigate the search space. With each attempt, it receives a performance signal, for example, an evaluation of the computational efficiency of the algorithm (number of multiplications, additions, numerical stability, etc.). This feedback guides its learning process, allowing it to adapt and produce increasingly effective algorithms.

For more details, you can refer to the paper released by Google, available here: https://storage.googleapis.com/deepmind-media/DeepMind.com/Blog/alphaevolve-a-gemini-powered-coding-agent-for-designing-advanced-algorithms/AlphaEvolve.pdf.