Some comments on AlphaEvolve

Ernest Davis
Dept. of Computer Science
New York University
New York, NY 10012
davise@cs.nyu.edu

May 22, 2025

The AlphaEvolve system newly announced at DeepMind (AlphaEvolve team, 2025; Novikov, 2025) is an extension of DeepMind's FunSearch system (Romera-Paredes et al., 2023) announced two years ago. The basic idea of AlphaEvolve is the same as FunSearch, but it incorporates a number of significant technical extensions. As a result, whereas FunSearch was successfully applied to only four mathematical problems, AlphaEvolve has been applied successfully to thirteen mathematical problems, to variants of the algorithmic problem of matrix multiplication, and to a handful of problems of practical importance to Google. DeepMind has made FunSearch available to the research community (Ellenberg et al. 2025)[1] but AlphaEvolve has not yet been made available for use or study to anyone outside Google. (I strongly recommend that anyone who plans to read the technical paper on AlphaEvolve (Novikov et al. 2025) first read the paper on FunSearch (Romera-Paredes et al., 2023), which is much more readable.)

I reviewed FunSearch in some depth two years ago (Davis, 2023). At first I planned to write a comparable systematic review of AlphaEvolve here. But after working on it for two days, I found that that would require long, boring repetitions of material in the AlphaEvolve and FunSearch technical papers and in my earlier review. So instead I'll just go through some thoughts, with explanations and justification only to the degree that they seem interesting or non-obvious. Section 1 discusses some aspects of AlphaEvolve that I find particularly impressive. Section 2 raises questions about the significance of some technical claims and of the mathematical and algorithm results that AlphaEvolve found. Section 3 mentions some valuable information that has not been published. Section 4 points out some clear inaccuracies and hype in some of what has been published, both by DeepMind and others.

Disclaimer: I'm not a mathematician. I'm certainly not an expert either on the technology used in AlphaEvolve or on the problems that it has been applied to. If anyone more knowledgeable finds errors here or thinks that I have made serious misjudgments, I will be extremely obliged if you email me about it.

1 What's impressive about AlphaEvolve

Some aspects of AlphaEvolve seem to me quite impressive.

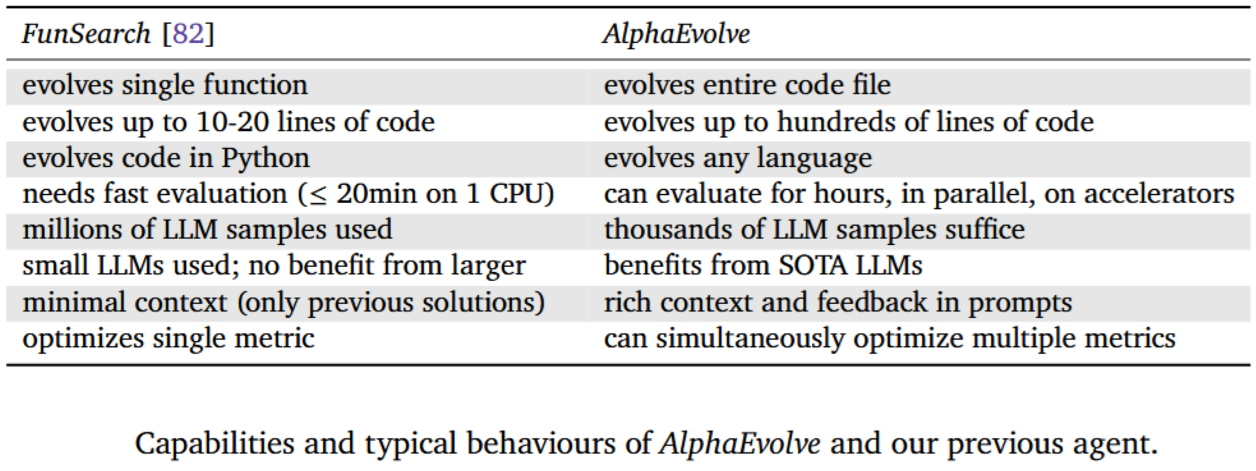
At the time of its announcement, FunSearch had been applied only to four problems, and it seemed reasonable to ask whether it would give useful results on any other problems. (That can happen; for example, DeepMind has conceded that AlphaTensor was a dead end; they have not been able to extend it to problems beyond

[1] Jordan Ellenberg informs me that "it took some serious work, primarily by Kit Fraser-Taliente, to build out an implementation of it that was practically usable by mathematicians." The code is at `https://github.com/kitft/funsearch` .

the original matrix multiplication problems it was applied to.) Since then, Ellenberg et al. (2025) have applied it to two further classes of problems. The number and breadth of problems successfully addressed in AlphaEvolve give a clear positive answer to that; it certainly seems to have the potential to serve a new tool that will sometimes yield improved solutions for certain kinds of optimization problems.

AlphaEvolve was able to generate improved solutions to three practical optimization problems (or, perhaps, categories of problems) that arise in "Google's computing ecosystem": one problem in data center job scheduling, one in optimizing the tiling strategy for matrix multiplications for training a large language model, and one in hardware circuit design. These have led to measurable improvements in efficiency — a gain of 0.7% in resource utilization, 1% in Gemini training time, and speed-ups of 32% and 15% in various aspects of computation. I can't evaluate these in depth; not enough information is given, and even if it were, I'm too inexpert in these areas to make a reliable judgment. But these do seem impressive.

(Novikov et al. 2025) provides the following table of the ways in which AlphaEvolve goes beyond FunSearch.



FunSearch [82]	*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

Capabilities and typical behaviours of *AlphaEvolve* and our previous agent.

Table 1: From (Novikov et al., 2025)

The fifth item in table 1 — FunSearch generated millions of samples in its search for effective code; AlphaEvolve needs only thousands — seems like a remarkable and unexpected improvement. It would be good to have a deep understanding of how that improvement was achieved. (By contrast, the claim in the third row that AlphaEvolve deals with more programming languages seems to me questionable, as I'll discuss below. The other differences enumerated in Table 1 are the kinds of improvement in capacity, scale, and engineering that one expects to see from one generation of a computer program to the next; certainly very creditable to the development team, but not particularly surprising.)

2 Some doubts

There are some technical statements in the technical paper that seem to me questionable and I am skeptical of the importance of the algorithmic and mathematical results that AlphaEvolve has found, though I am not certain of any of these.

2.1 Technical issues

Line 3 of table 1 claims that FunSearch "evolves code in Python" whereas AlphaEvolve "evolves any language". This is puzzling. On the one hand, FunSearch used the Codey LLM to generate variants, (Romano-Paredes et al. 2023), and the webpage[2] for Codey, the LLM that powered FunSearch, boasts that Codey "supports over 20 programming languages, including Python, Java, JavaScript, Go, ..." so there doesn't seem to have been any inherent reason why FunSearch couldn't have evolved programs in those languages. On the other hand, I don't find any indication of any experiments running AlphaEvolve to generate code in any language other than Python. Moreover, what advantage for solving these optimization problems is actually gained by being able to deal with multiple programming language?

In addition to the differences enumerated in table 1, AlphaEvolve, unlike FunSearch, supplies the user will a variety of knobs that they can turn and settings they can play with to try to improve performance.

> [U]sers can . . . tailor prompts to their specific needs in different ways, such as the following: *Explicit context.* . . . *Stochastic formatting* . . . *Rendered evaluation results,* . . . and *Meta prompt evaluation.*
>
> . . .
>
> AlphaEvolve supports optional mechanisms to make this evaluation more flexible and more efficient: *Evaluation cascade*,. . . *LLM-generated feedback*, . . . and *Parallelized evaluation.*

However, highly user-configurable computer systems are a mixed blessing, as anyone who has seriously engaged with one can attest. On the one hand, they hold out the promise that, if you find the right settings, you can get much better performance than just using the system out of the box with all its default settings. On the other hand, finding the right settings for any particular problem may take an indeterminate amount of experience, insight, and experimentation. How much of each of these was needed to get the results discussed here or to get new results on new problems is anybody's guess. It is not clear how powerful AlphaEvolve is when used "out of the box".

2.2 How significant are the results on matrix multiplication?

Both (Novikov et al, 2025) and many of the write-ups of AlphaEvolve highlight its discovery of new methods to multiply small matrices multiplication using fewer scalar multiplication operations than previously known techniques. In particular, much has been made of its finding of a method for multiplying two 4x4 matrices that uses only 48 scalar multiplications rather than the 49 required by Strassen's method.

Not being at all an expert, I can't say for sure, but I am skeptical that these results are any value in a practical sense or of much interest theoretically. In large measure, my doubt spring from this observation in the caption to table 2 (p. 9) of (Novikov et al., 2025): "For (3, 4, 7), (4, 4, 4), and (4, 4, 8), the algorithms discovered by AlphaEvolve use complex-valued multiplications which can be used for exact multiplication of complex or real-valued matrices." If I am reading this correctly, what this means is that, even if the two matrices you are multiplying are real, the method that AlphaEvolve has generated requires you to carry out some number of multiplications of complex scalars.

That would seem to significantly diminish the practical significance of these results. Multiplying two complex numbers requires three real multiplications. So if the matrices you are multiplying are real, as is the case in many applications, the number of *real* multiplications required by AlphaEvolve's method is not 48; it is somewhere between 50 (if only one complex multiplication is required) and 144 (if all the multiplications required are complex).

[2]`https://lablab.ai/tech/google/codey`

If you are doing an immense series of multiplications of dense matrices of complex numbers where high degrees of precision are required, so that the cost of scalar multiplications dominates everything else, then AlphaEvolve's algorithm might give measurable gains in efficiency. But that's probably a fairly rare case.

Also, if the method is applied recursively to matrices, it will give an asymptotic running time of $O(n^{\log(48)/4}) = O(n^{2.792})$ whereas the theoretically best (albeit completely impractical) algorithms currently known have a running time of $O(n^{2.37})$.[3]

2.3 How significant are the mathematical results?

AlphaEvolve was applied to "over 50" mathematical problems within its scope. The solutions it found were at least as good as the best-known solutions for 75% of the problems; for 20% (i.e. 13) of the problems, it found a better solution than had been previously known. It is certainly possible that, for some of these problems, AlphaEvolve failed to find a better solution because in fact no better solution exists. Of the 13 problems where it found better solutions, seven are problems in elementary geometry (meaning that one only needs high school geometry to understand the statement of the problem, not by any means that finding the solution is easy);[4] five are in real analysis; and one is in combinatorics.

Is this a significant contribution to mathematical knowledge? I can't judge. One would have to hear from an expert about how exciting these results are, and I have not seen any such reactions. The problems were *suggested* by the distinguished mathematicians Terence Tao, Jordan Ellenberg, and Javier Gomez Serrano, so we can be sure that they are meaningful problems, but that is a different statement.

Not yet having heard from experts, I think there are grounds for doubt. Let us consider the result that has been most highlighted: AlphaEvolve there is a solution for the "kissing problem" in 11 dimensions that has 593 spheres, whereas the best previous known solutions had 592. The kissing problem asks the following Consider a central sphere S of radius 1. How many spheres of radius 1 can you place around it such that they all touch ("kiss") S but no two overlap? The problem has a long and distinguished history. Newton and mathematician David Gregory debated about its value in three dimensions; Newton conjectured that the obvious solution with 12 spheres was optimal while Gregory conjectured that it should be possible to somehow fit 13. (The question was not resolved until 1953, when Newton was proved right.) The exact value of the kissing number in n dimensions is known for $n = 1$, 2, 3, 4, 8, and 24; for all other dimensions, there is a gap between the best known solution and the best proved upper bound.

How excited do mathematicians in the area feel on learning that in 11 dimensions there is a solution with 593 kissing spheres, not merely 592? I can't say. The known upper bound on the kissing number in 11 dimensions is 868, so this is a 0.36% improvement in the known range of possible values. Mathematicians have at times gotten excited about much smaller percentage gains in known results.

On the other hand it is instructive to contrast this with the two papers that found the value of 592 using two different methods. In Ganzinov (2025), improved values of 510 for 11 dimensions, 592 for 10 dimensions, and 1932 for 14 dimensions emerges as a consequence of a study of geometric symmetries. The specific case of 11-dimensional space is one paragraph in a 26 page paper. It would appear that the solution was found through by purely theoretical analysis, though no doubt it was checked in a computer program. De Laat and Leijenhorst (2024) extended traditional interior-point optimization methods to find improved values on the kissing number in dimensions 11 through 23. Both of these seem to the ignorant outsider (me) as more interesting mathematically and more likely to be fruitful than the mathematically more arbitrary approach taken in AlphaEvolve.

[3]Thanks to Greg Kuperberg for correcting an error in an earlier version.

[4]They are also elementary in the technical sense that, for specific values of the measure, they can be expressed in the existential theory of real arithmetic, which is decidable. For instance it is straightforward to translate the statement "The kissing number in 11 dimensions is less than 869" into a statement, "There are no real solutions of the set of equations and inequalities $P_1(\cdot) = 1 \dots P_{869}(\cdot) = 1, Q_1(\cdot) \geq 1, Q_{377,146}(\cdot) \geq 1$" where each of the P's and Q's is a quadratic polynomial and there are a total of 9559 variables.

3 Additional information that I hope the AlphaEvolve team will publish

The AlphaEvolve team has thus far published only the statements of the 13 problems where AlphaEvolve found an improved solution, not for the more than 37 where it didn't. For the purpose of evaluation, it would be very helpful if they gave the statements of all 50. For instance, did they attempt to solve the kissing problem only in dimension 11, where there is an especially large gap between lower and upper bound, or did they try other dimensions as well. (In their online material[5] they offer the lame justification, "To avoid clutter, we exclude those problems where the results matched but did not outperform the best known constructions.")

It would also be very helpful to get much more information about what was involved in obtaining the results: how much human time was expended, how many attempts with different settings of AlphaEvolve were attempted, how much computing resources were spent. This information should be provided, both for the successful and for the unsuccessful attempts.

4 Hype

As is almost inevitable with AI products these days, the announcement and associated publicity have been accompanied by hype, misleading statements, and outright untruths.

The problems start with the title of the technical paper, "*AlphaEvolve*: A coding agent for scientific and algorithmic discovery". There are two problems with this. First, "scientific" should be "mathematical". There is zero evidence and little reason to expect that AlphaEvolve will be useful for science outside of mathematics. Likewise the first sentence of the article, which starts "Discovering new high-value knowledge, such as making a novel scientific discovery ..." suggests strongly that AlphaEvolve has something to do with making novel scientific discoveries, which it doesn't.

Second, AlphaEvolve is in no sense an agent. Here I have to rant. An agent is something that acts. A robot or a self-driving car is an agent; they do physical things. A commercial web pages that sells shoes is an agent; it places a charge on your credit card and arranges for a pair of shoes to be mailed to you. A thermostat is an agent; it turns the furnace on and off. AlphaEvolve is no more an agent than a desk calculator or the C compiler. The word "agent" had a fairly specific and useful meaning in AI research from the early 1980s until a few months ago; then it was taken up as a hot buzzword by Google, OpenAI, and everyone else; now it has become completely vacuous. End rant.

The title of the *Nature* news article (Gibney, 2025) "DeepMind unveils 'spectacular' general-purpose science AI" is even worse. 'General-purpose' is the exact opposite of the truth; even within math-oriented AI, the range of problems that AlphaEvolve can address is remarkably narrow. A quote from Simon Frieder in the body of the article makes the same point: "[AlphaEvolve] will probably be applied only to the 'narrow slice' of tasks that can be presented as problems to be solved through code."

But the hype prize goes to the self-styled "visionary leader" and "futurist" Antoine Tardif (2025) who wrote an article entitled, "AlphaEvolve: Google DeepMind's Groundbreaking Step toward AGI". 'Nuff said.

Acknowledgements

Thanks to Greg Kuperberg for correcting an error in an earlier draft of this and to Jordan Ellenberg for information about his further experiments with FunSearch.

[5]`https://github.com/google-deepmind/alphaevolve_results/blob/main/README.md`

References

Davis, Ernest (2024). "Comment on (Romera-Paredes et al., 2023), 'Mathematical discoveries from program search with large language models".
`https://cs.nyu.edu/~davise/papers/FunSearchComment.pdf`

Ellenberg, Jordan et al. (2025). "Generative Modeling for Mathematical Discovery". ArXiv preprint 2503.11061.
`https://www.arxiv.org/abs/2503.11061`

Fawzi, Alhussein et al.(2022). "Discovering faster matrix multiplication algorithms with reinforcement learning." *Nature,* **610** (7930):47-53. `https://doi.org/10.1038/s41586-022-05172-4`

Fawzi, Alhussein and B. Romera-Paredes (2023). "FunSearch: Making new discoveries in mathematical sciences using Large Language Models". Google DeepMind Blog.
`https://deepmind.google/discover/blog/funsearch-making-new-discoveries-in-mathematical-sciences-using-l`

Ganzhinov, Mikhail. "Highly symmetric lines." Linear Algebra and its Applications (2025). Volume 722, 1 October 2025, Pages 12-37 `https://www.sciencedirect.com/science/article/pii/S0024379525001946`

Gibney, Elizabeth (2025). "DeepMind unveils 'spectacular' general-purpose i science AI". *Nature,* News, May 14, 2025. `https://www.nature.com/articles/d41586-025-01523-z`

Google (2023). "Palm 2.0 Technical Report" ArXiv preprint 2305.10403. `https://arxiv.org/pdf/2305.10403`

Leijenhorst, Nando, and David de Laat. "Solving clustered low-rank semidefinite programs arising from polynomial optimization." Mathematical Programming Computation 16, no. 3 (2024): 503-534. `https://www.arxiv.org/abs/2202.12077`

Novikov, Alexander et al. (2025). "AlphaEvolve: A coding agent for scientific and algorithmic discovery."
`https://storage.googleapis.com/deepmind-media/DeepMind.com/Blog/alphaevolve-a-gemini-powered-coding-agen`
`AlphaEvolve.pdf`

Romera-Paredes, Bernardino et al. (2023). "Mathematical discoveries from program search with large language models". *Nature* accelerated article preview.
`https://doi.org/10.1038/s41586-023-06924-6`

Tardif, Antoine (2025). "AlphaEvolve: Google DeepMind's Groundbreaking Step toward AGI", *Unite.AI* May 17, 2025. https://www.unite.ai/alphaevolve-google-deepminds-groundbreaking-step-toward-agi/