# An investigation on the use of Large Language Models for hyperparameter tuning in Evolutionary Algorithms

Leonardo Lucio Custode
leonardo.custode@unitn.it
University of Trento
Trento, Italy

Fabio Caraffini
fabio.caraffini@swansea.ac.uk
Swansea University
Swansea, UK

Anil Yaman
a.yaman@vu.nl
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands

Giovanni Iacca
giovanni.iacca@unitn.it
University of Trento
Trento, Italy

## ABSTRACT

Hyperparameter optimization is a crucial problem in Evolutionary Computation. In fact, the values of the hyperparameters directly impact the trajectory taken by the optimization process, and their choice requires extensive reasoning by human operators. Although a variety of self-adaptive Evolutionary Algorithms have been proposed in the literature, no definitive solution has been found. In this work, we perform a preliminary investigation to automate the reasoning process that leads to the choice of hyperparameter values. We employ two open-source Large Language Models (LLMs), namely Llama2-70b and Mixtral, to analyze the optimization logs online and provide novel real-time hyperparameter recommendations. We study our approach in the context of step-size adaptation for $(1 + 1)$-ES. The results suggest that LLMs can be an effective method for optimizing hyperparameters in Evolution Strategies, encouraging further research in this direction.

## CCS CONCEPTS

• **Information systems** → **Language models**; • **Theory of computation** → **Evolutionary algorithms**.

## KEYWORDS

Evolutionary Algorithms, Large Language Models, Landscape Analysis, Parameter Tuning

**ACM Reference Format:**
Leonardo Lucio Custode, Fabio Caraffini, Anil Yaman, and Giovanni Iacca. 2024. An investigation on the use of Large Language Models for hyperparameter tuning in Evolutionary Algorithms. In *Genetic and Evolutionary Computation Conference (GECCO '24 Companion), July 14–18, 2024, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3638530.3664163


*GECCO '24 Companion, July 14–18, 2024, Melbourne, VIC, Australia*

ACM ISBN 979-8-4007-0495-6/24/07.
https://doi.org/10.1145/3638530.3664163

## 1 INTRODUCTION

It is well established in the field of Evolutionary Computation (EC), and even more broadly in Machine Learning (ML), that the hyperparameters of an algorithm significantly influence its performance. In evolutionary optimization, improper assignment of these parameters can lead to a suboptimal search process and provide inferior solutions [48]. Over the past few decades, various approaches have been proposed to tackle this challenge by adjusting the parameters of the algorithms to positively influence the optimization process. For instance, ontology-based approaches aim to formally represent the knowledge revolving around the EC field [47] to enable inference-based tuning, whereas ML-based approaches [36] learn useful representations for comparing evolutionary runs for landscape analysis. Other approaches pre-define pools of parameter values [19, 20] and automatically choose from them during the optimization process. More recently, Reinforcement Learning (RL) has been considered to automatically train hyperparameter adaptation policies based on data from previously available evolutionary runs [40]. In general, all these approaches aim to find the optimal parameter settings of an Evolutionary Algorithm (EA) *before* or *during* the evolutionary optimization process and are usually referred to as *parameter tuning* or *parameter control*, respectively [9].

Using data acquired from evolutionary runs is certainly a viable approach to leverage the knowledge acquired from past optimization experiences, as shown in [40]. However, the data generated by evolutionary runs can be highly unstructured, and they heavily depend on multiple aspects, such as the problem dimensionalities, the different algorithms and strategies being used, the numerical precision, the verbosity of the logging system in use, etc.

In recent years, Large Language Models (LLMs) have caused a profound revolution in various fields, going beyond the traditional domain of conversational agents. To date, LLMs have been successfully applied to an ever-growing list of domains, such as chemistry [3], protein design [12], robotics [34, 38, 44], swarm motion planning [23], program synthesis [18, 21, 39], game design [24], and urban delivery route optimization [26].

One of the main keys to the success of LLMs is that they are extremely well-suited for analyzing unstructured textual data, which makes it easy to adapt them to conceptually similar tasks without the need for retraining. Some recent literature [5, 32] has investigated the direct usage of LLMs for optimization, although it is important to note that current LLMs are well-suited especially for

high-level discrete optimization tasks (e.g., planning). On the other hand, they are not well-suited for low-level (i.e., numerical) continuous optimization tasks due to (1) the inference cost of querying LLMs, and (2) their limited mathematical reasoning capabilities. Finally, several works have investigated the relationship between LLM and EC [45], both in the cases where LLMs are used to empower EAs, and in the opposite case. However, none of them studied the use case of hyperparameter optimization, which is an essential problem in EC.

In this work, we conduct a preliminary investigation on the possibility of automating "empirical" step-size adaptation in Evolution Strategies (ES) using LLMs. More specifically, we focus on the case of tuning the step-size for $(1 + 1)$-ES using state-of-the-art LLMs, namely Llama2-70b [41] and Mixtral [22]. Our results show that LLMs can compete with well-known traditional mechanisms for step-size adaptation.

The remainder of the paper is organized as follows. Section 2 summarizes the related work on LLMs and their applications to optimization, as well as the early work on EC applied to LLMs. Section 3 describes our methods. Section 4 presents the numerical results. Finally, Section 5 concludes this research.

## 2 RELATED WORK

There is a growing trend to combine the strengths of optimization algorithms with LLMs for various purposes. This synergy is evident also in the EC research community, where the 'LLM' keyword is increasingly prevalent and the increasing number of available *preprints*, see [45] for a recent exhaustive list, is a clear signal that more articles of this kind will soon populate the literature. This highlights a promising direction where LLMs will play an important role in optimization (especially, but not only, evolutionary-based), addressing various challenges associated with optimization algorithms and leading to hybrid systems where optimization algorithms and LLMs work together for mutual benefits.

### 2.1 LLMs as the optimizer

LLMs make it possible to articulate complex tasks using natural language, thus facilitating and automating decision-making in many logistical scenarios, such as planning and scheduling [28], which otherwise would require an expert to formulate the problem rigorously (i.e., as a discrete optimization task). For example, ChatGPT has been used to produce the schedule of a construction project in [30] and to schedule vehicles and drivers to complete a predetermined set of trips in a specified period in [43]. These are combinatorial optimization problems whose operational constraints are difficult to formulate mathematically but easy to communicate to an LLM. Similarly, in [2], users can ask "what if" questions on supply chain demand for a particular scenario in plain text and get answers about the outcomes of an underlying supply chain optimization algorithm.

Hyperparameter optimization is another intriguing application where LLMs can act as the optimizer for other ML models. Compared to traditional hyperparameter optimization methods, the results appear promising [27]. For instance, the studies in [51] and [49] leverage GPT-4 to perform Neural Architecture Search and design autoML frameworks, respectively.

Finally, Optimization by PROmpting (OPRO) is a framework proposed in [5] for using LLMs as an iterative heuristic for gradient-free optimization. In each optimization step, the OPRO LLM generates new solutions from those contained (including their values) in the previous prompt. As with any EC approach, novel candidate solutions are evaluated and added to the next prompt for the next optimization step. This is a general approach that is usable for both discrete and continuous problems. The results presented in [5] for linear regression and Travelling Salesman Problems are promising, highlighting that OPRO can optimize the prompts without human intervention.

### 2.2 EAs for prompt or LLM optimization

The need for a good quality prompt has opened the way for the prompt optimization/engineering field. The existing literature explores different methodologies to optimize prompts. Recent work [31] suggests using gradient descent with beam search and bandit selection assuming that training data and an LLM API are accessible. EC offers various tools for this task, which can also be used when training data and internal LLM information are inaccessible (i.e., the gradient information cannot be exploited). Genetic Algorithms are widely used [11, 50], and ideas borrowed from EAs are explored in the EvoPrompt framework [17]. In [37], an interesting approach based on vectorizing prompts in a continuous subspace employs CMA-ES to perform the optimization process.

LLM architecture search is another intriguing optimization challenge. EAs can greatly help explore these vast search spaces. In the AutoBERT-Zero framework [15], this is done by performing primitive math operations and through an interesting "Operation-Priority" evolution strategy, which exploits prior information from operations during the search. Other studies consider multiple objectives, such as the one in [42], which compares direct search methods and classic heuristics, or the framework in [14], where EAs are used to find the most suitable distribution of hidden units across layers, ensuring that accuracy and latency requirements are met concurrently.

### 2.3 LLMs supporting EAs

LLMs can support EAs in numerous ways and there is growing enthusiasm for employing them to solve various mathematical problems. In this context, the most popular example of incorporating LLM into EAs is probably the FunSearch algorithm [35]. This algorithm has raised a debate among researchers, with some applauding its ability to tackle unresolved mathematical problems and demonstrate genuine "intelligence", while others pointing out its limitations, as being "*remarkable for the shallowness of the mathematical understanding that it seems to exhibit*" [6]. Regardless of its public perception, FunSearch is an interesting, but algorithmically straightforward approach. It involves employing an LLM to generate code for a specific subroutine within a larger program in a Genetic Programming (GP) algorithm, meaning that the LLM is embedded into the EA to perform the mutation step. The idea of using LLMs for code generation in EAs is not new. In [25], they are used with the MAP-Elites algorithm to generate effective mutation operators for GP.

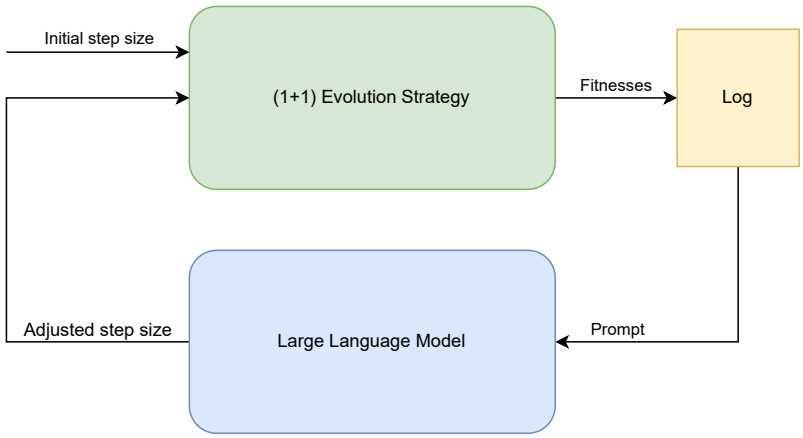


**Figure 1: A block diagram showing the self-adaptive step-size adaptation scheme studied in this work.**

Algorithm design is another application of LLMs to EAs. For example, in [10] LLMs are employed to generate and evolve algorithm components such as initialization, selection, crossover, etc. to create new algorithms.

There are interesting uses for LLMs in other optimization scenarios. The system described in [29] uses these techniques to simplify the formulation of common complex business optimization problems, allowing the optimizer to improve performance in production scheduling. In [46], LLMs check the feasibility of solutions to constraints in discrete spaces to facilitate the application of simulated annealing and other single-solution metaheuristics. LLMs can also function as supervisory logic, e.g., they coordinate agents in the coevolutionary algorithm presented in [7].

## 3 METHODS

In this work, we leverage LLMs to automatically optimize the step-size for $(1+1)$-ES, which generates an offspring by adding, to the current solution, a uniform perturbation in a hyper-sphere of radius $\sigma$, the step-size.

In our method, the LLM is queried every $p$ optimization steps (with $p$ being user-defined, e.g. based on the available computational resources) with the log produced from the beginning of the optimization process to the current generation. The LLM is then asked to produce a recommendation for the next step-size to be used in the subsequent $p$ steps. For the LLM-based step-size adaptation, we use a period of $p = 50$ generations to calculate the new step-size.

We fix the number of fitness evaluations for all the methods to $10^3$. Our code is publicly available on GitHub[1].

### 3.1 Benchmark problems

We run $(1+1)$-ES on the BBOB suite [13], using the IOHProfiler suite [8]. We test all the BBOB problems with dimensionalities 2, 5, and 30, to assess the capabilities of the proposed method to work in both low- and moderate-sized problems. For each function, we perform 10 independent runs with different starting points. Then, we feed the logs to the LLMs, using the prompt described below.

[1]https://github.com/DIOL-UniTN/llm_step_size_adaptation

### 3.2 Prompt format

To provide the LLM with the information needed to tune the step-size, we initially considered three different options: (1) feeding only the fitness values; (2) feeding genotypes and fitnesses; and (3) feeding the genotypes, the relationships between the genotypes (i.e., between the parent solution and its offspring, through the effect of mutation), and the fitnesses. However, after preliminary experiments (not reported here for brevity), we did notice that the last two approaches required a substantially higher amount of tokens, quickly filling the context window of the models. For this reason, we eventually use a prompt that only contained the essential information to make a decision, i.e., the changes in the step-size and the fitnesses of all the last individuals evaluated (depending on the size of the context window).

An example of such a prompt can be seen in Listing 1. Here, the "system" prompt is used to provide a preliminary context to the model to properly address the task. Then, the "user" task represents the actual query. Finally, it is important to note that we request a step-size $s \in [0.001, 0.999]$ to prevent the LLMs from choosing exploding step-sizes.

### 3.3 Algorithms and LLMs

Our comparison investigates the performance of different strategies for adapting the step-size of $(1+1)$-ES. We employ two baseline methods and two LLM-based methods. The two baselines consist of: (1) a constant step-size for the ES (fixed to 0.1); and (2) a well-known rule-based strategy, namely the One-Fifth success rule [33]. The latter changes the step-size by considering the current candidate $x' = x + \mathcal{N}(0, \sigma_s)$, where $x$ is the parent solution, $\mathcal{N}$ is the Gaussian (normal) distribution, and $\sigma_s$ is the step-size, as follows:

$$\sigma_s' = \begin{cases} 1.5\sigma_s & if\ f(x') < f(x) \\ 1.5^{-\frac{1}{4}}\sigma_s & if\ f(x') > f(x) \\ \sigma_s & if\ f(x') = f(x) \end{cases} \quad (1)$$

Regarding the LLM-based step-size adaptation strategies, we employ two widely known open-source state-of-the-art LLMs, namely Llama2-70b and Mixtral. Llama2-70b is the largest model of the Llama2 family [41]. It features 80 transformer layers with a hidden size of 8192 dimensions [4]. It has been trained on $2T$ tokens and has a maximum context size of 4096 tokens.

Mixtral [22], instead, is a mixture-of-experts model made of 8 experts, each with 32 transformer layers with a hidden size of 4096 dimensions. It has a context size of $32k$ tokens.

For both models, we employ Groq's APIs for accelerated inference[2].

## 4 RESULTS

The results obtained by the 4 methods under comparison are shown in Figure 3, aggregating the results for all the problem sizes to facilitate visualization. We observe that, in most cases, the performance of both Llama2-70b and Mixtral is comparable to that of the One-Fifth strategy, suggesting a somehow similar behavior in terms of step-size adaptation. Note that, for the $f5$ function, the methods using constant step-sizes and the one-fifth rule achieve values very close to zero, and thus they are not visible in the boxplot (which uses a logarithmic scale).

To verify this hypothesis, we studied the step-size used by each of the methods under comparison in Figure 2. Note that the trend shown in Figure 2 contains the average step-size computed on all the functions of the benchmark, divided by all the dimensionalities considered before. We observe that, contrary to our expectations, their impact on the step-size is actually at two opposite extremes. In fact, while the One-Fifth rule pushes the step-size to very high values, the two LLM-based methods tend to progressively reduce the step-size. Yet, their performance appears to be overall similar. Moreover, it is interesting to note that the variance of the LLM-based step-size adaptation techniques is inversely proportional to the problem dimensionality, while the variance of the one-fifth adaptation techniques seems much larger in the highest-dimensionality case.

Finally, to add a further level of comparative evaluation of the four methods, we performed the glicko2-based ranking [16] for fixed-budget scenarios, available in the IOHAnalyzer platform[3]. The results are shown in Tables 1 to 3. In the tables, we observe an interesting trend: Llama2-70b always ranks first, outperforming all the other methods. On the other hand, we see Mixtral as a runner-up when the problem dimensionality is 2, and it gradually ranks worse as the dimensionality of the problem increases. Another interesting remark is that, in the highest-dimensionality case (Table 3), both Mixtral and the One-Fifth rule rank worse than having a constant step-size. On the other hand, in 30 dimensions Llama2-70b is the only method that performs better than simply using a constant step-size.

These preliminary results indicate that it is indeed possible to have well-performing step-size adaptation strategies by using the reasoning capabilities in LLMs. However, our results also show that the outcomes are heavily dependent on the LLM at hand, suggesting that a specialized LLM (e.g., fine-tuned on a large dataset of evolutionary runs) could lead to even better results.

[2]https://groq.com

[3]https://iohanalyzer.liacs.nl/

## 5 CONCLUSIONS

The step-size is a crucial parameter in ES. In fact, it allows the user to directly control the exploration and exploitation capabilities of the algorithm. While several approaches have been proposed for adapting the step-size in ES, there is no one-fits-all strategy that solves this problem. This is due to the fact that step-size control heavily depends on the context at hand, which requires some form of reasoning. In this direction, we performed a preliminary investigation on the use of LLMs for tuning the step-size of $(1+1)$-ES. Our results indicate that it is possible to find adaptation strategies that are competitive with established methods for step-size adaptation, such as the One-Fifth rule. Moreover, given the fixed budget used in our experiments, the approach based on Llama2-70b showed better consistency w.r.t. the other techniques, always ranking as the best algorithm.

The present work suggests several interesting future directions, such as experimenting with different prompts (and the related information therein) provided to the LLMs, fine-tuning existing open-source LLMs to the specific domain of adaptation strategies, conducting a more extensive study on hyperparameter optimization for EAs using LLMs (i.e., using larger budgets, as suggested in [1], and more sophisticated step-size adaptation schemes), and finally developing an LLM-guided EA.

## REFERENCES


[1] Anne Auger. 2009. Benchmarking the (1+1) evolution strategy with one-fifth success rule on the BBOB-2009 function testbed. In *Genetic and Evolutionary Computation Conference Companion*. ACM, New York, NY, USA, 2447–2452.

[2] Beibin Li and Konstantina Mellou and Bo Zhang and Jeevan Pathuri and Ishai Menache. 2023. Large Language Models for Supply Chain Optimization. arXiv:2307.03875.

[3] Boiko, Daniil A and MacKnight, Robert and Kline, Ben and Gomes, Gabe. 2023. Autonomous chemical research with large language models. *Nature* 624, 7992 (2023), 570–578.

[4] Nuo Chen, Ning Wu, Shining Liang, Ming Gong, Linjun Shou, Dongmei Zhang, and Jia Li. 2024. Is Bigger and Deeper Always Better? Probing LLaMA Across Scales and Layers. arXiv:2312.04333.

[5] Chengrun Yang and Xuezhi Wang and Yifeng Lu and Hanxiao Liu and Quoc V. Le and Denny Zhou and Xinyun Chen. 2023. Large Language Models as Optimizers. arXiv:2309.03409.

[6] Ernest Davis. 2024. Using a large language model to generate program mutations for a genetic algorithm to search for solutions to combinatorial problems: Review of (Romera-Paredes et al., 2023). https://cs.nyu.edu/~davise/papers/FunSearch.pdf Accessed on 7/04/2024.

[7] de Zarzà, I. and de Curtò, J. and Roig, Gemma and Manzoni, Pietro and Calafate, Carlos T. 2023. Emergent Cooperation and Strategy Adaptation in Multi-Agent Systems: An Extended Coevolutionary Theory with LLMs. *Electronics* 12, 12 (2023), 19.

[8] Carola Doerr, Hao Wang, Furong Ye, Sander van Rijn, and Thomas Bäck. 2018. IOHprofiler: A Benchmarking and Profiling Tool for Iterative Optimization Heuristics. arXiv:1810.05281.

[9] Eiben, Ágoston E and Hinterding, Robert and Michalewicz, Zbigniew. 1999. Parameter control in evolutionary algorithms. *IEEE Transactions on evolutionary computation* 3, 2 (1999), 124–141.

[10] Fei Liu and Xialiang Tong and Mingxuan Yuan and Qingfu Zhang. 2023. Algorithm Evolution Using Large Language Model. arXiv:2311.15249.

[11] Chengzhe Feng, Yanan Sun, Ke Li, Pan Zhou, Jiancheng Lv, and Aojun Lu. 2024. Genetic Auto-prompt Learning for Pre-trained Code Intelligence Language Models. arXiv:2403.13588.

[12] Ferruz, Noelia and Höcker, Birte. 2022. Controllable protein design with language models. *Nature Machine Intelligence* 4, 6 (2022), 521–532.

[13] Steffen Finck, Nikolaus Hansen, Raymond Ros, and Anne Auger. 2010. *Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless*

**Listing 1: Prompt template.**

```
{
 "role": "system",
 "content": "You are a powerful and intelligent AI capable of analyzing logs and
 performing reasoning"
},
{
 "role": "user",
 "content": "Q: I am running an optimization process over an unknown function.
 I am using a 1+1-ES to optimize the function.
 f(x) = y indicates an evaluation.
 x1 -> x2 indicates that x1, using the current step size,
 produced a new candidate solution x2.
 It is extremely important that the step size you propose is
 contained between 0.999 and 0.001.
 Here's the log:```txt
 <LOG CONTENT>
 ```
 I am currently using the following step size: <STEP SIZE>.
 Should I change it or not?
 Do you think that the current step size is good enough to make
 the process converge as soon as possible?
 Reply with the following structure:
 `Reasoning: <explanation>
 Recommended step size: <new step size>`"
}
```

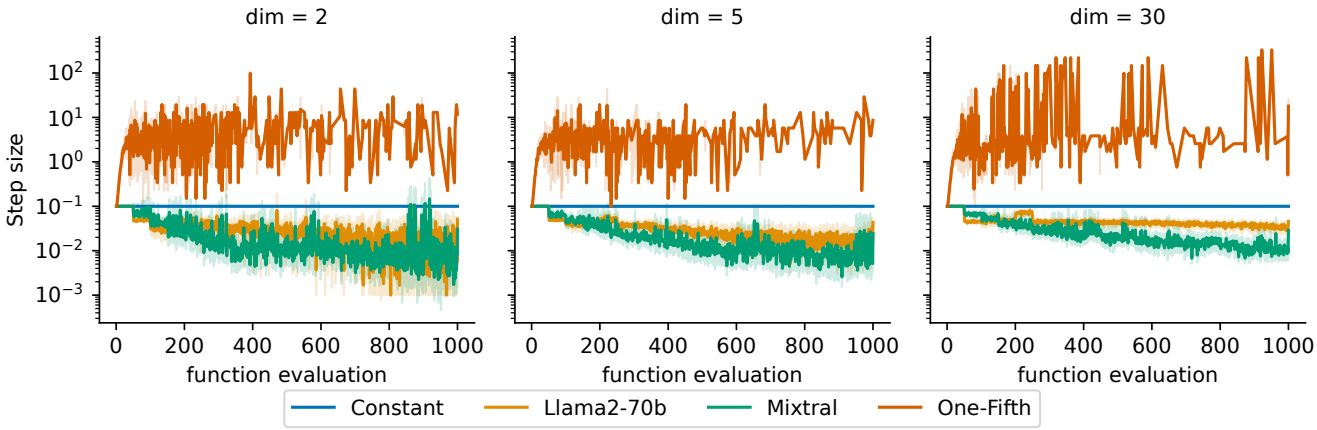


**Figure 2: Average step-size for each function evaluation for all the methods under comparison. The solid lines represent the mean value for the step-size (averaged over all problems and problem dimensionalities), while the shaded area represents the 95% confidence interval.**

*functions*. Technical Report. Citeseer.
[14] Vinod Ganesan, Gowtham Ramesh, and Pratyush Kumar. 2021. SuperShaper: Task-Agnostic Super Pre-training of BERT Models with Variable Hidden Dimensions. arXiv:2110.04711.
[15] Jiahui Gao, Hang Xu, Han Shi, Xiaozhe Ren, LH Philip, Xiaodan Liang, Xin Jiang, and Zhenguo Li. 2022. AutoBERT-Zero: Evolving BERT Backbone from Scratch. In *AAAI Conference on Artificial Intelligence*, Vol. 36, no. 10. AAAI, Washington, DC, US, 10663–10671.
[16] Mark E Glickman. 2012. *Example of the Glicko-2 system*. Technical Report. Boston University.
[17] Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. 2024. Connecting Large Language Models with Evolutionary Algorithms Yields Powerful Prompt Optimizers. arXiv:2309.08532.
[18] Erik Hemberg, Stephen Moskal, and Una-May O'Reilly. 2024. Evolving Code with A Large Language Model. arXiv:2401.07102.
[19] Giovanni Iacca, Fabio Caraffini, and Ferrante Neri. 2015. Continuous Parameter Pools in Ensemble Self-Adaptive Differential Evolution. In *IEEE Symposium Series on Computational Intelligence*. IEEE, New York, NY, USA, 1529–1536.
[20] Giovanni Iacca, Ferrante Neri, Fabio Caraffini, and Ponnuthurai Nagaratnam Suganthan. 2014. A differential evolution framework with ensemble of parameters
```

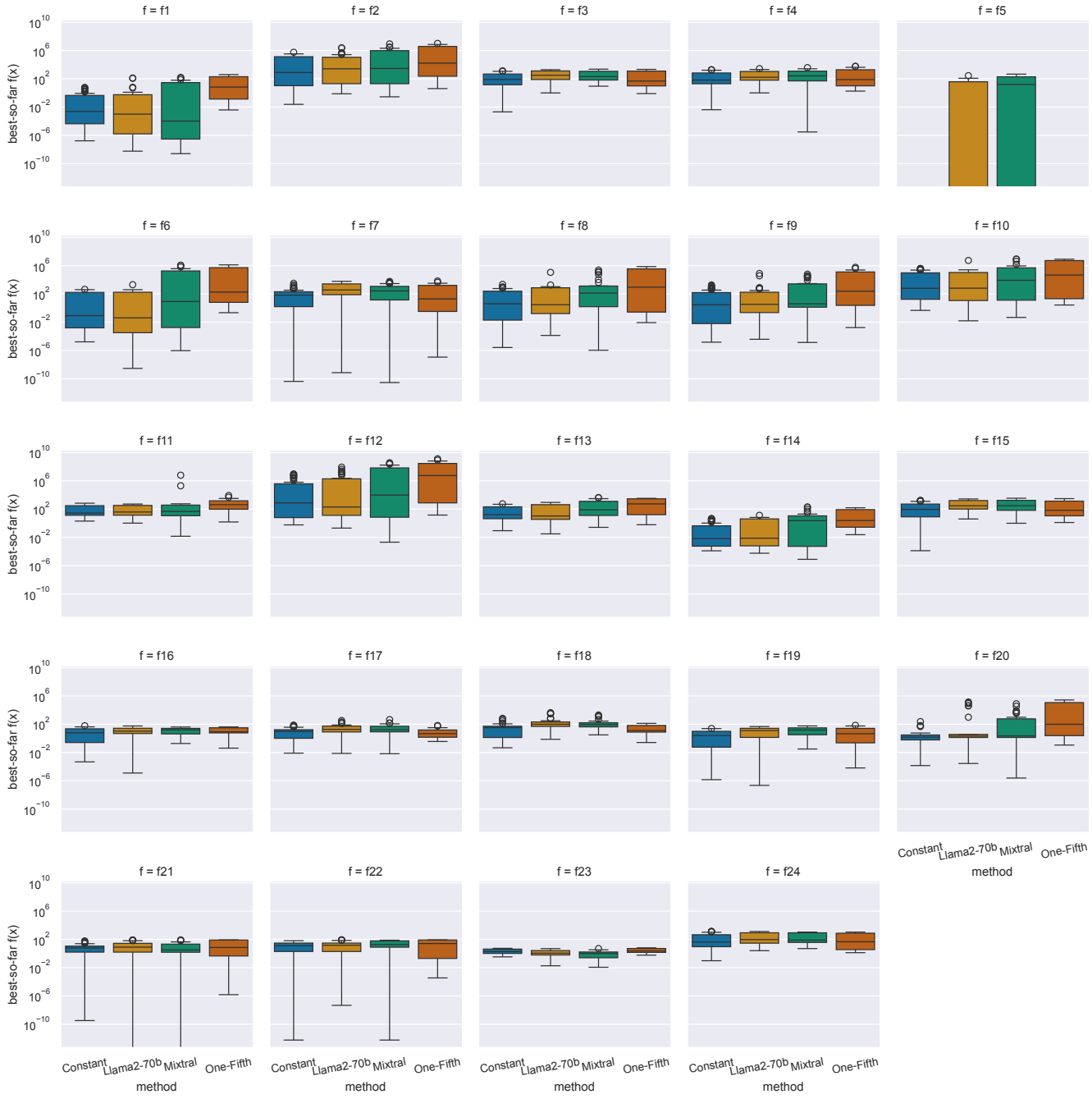


**Figure 3: Boxplot of the best fitness obtained in 10 runs by each of the methods on each function from the BBOB suite. Note that we aggregate the values from 10 runs on all the considered problem sizes.**

and strategies and pool of local search algorithms. In *Applications of Evolutionary Computation: 17th European Conference, EvoApplications 2014, Granada, Spain, April 23-25, 2014, Revised Selected Papers 17*. Springer, Berlin Heidelberg, Germany, 615–626.

[21] Jain, Naman and Vaidyanath, Skanda and Iyer, Arun and Natarajan, Nagarajan and Parthasarathy, Suresh and Rajamani, Sriram and Sharma, Rahul. 2022. Jigsaw: Large Language Models Meet Program Synthesis. In *International Conference on Software Engineering*. Association for Computing Machinery, New York, NY, USA, 1219–1231.

[22] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of Experts. arXiv:2401.04088.

**Table 1: Glicko2-based ranking of the methods on functions from the BBOB suite with dimensionality 2.**

| Step-size adaptation method | Rating | Deviation | Volatility | Games | Win | Draw | Loss |
|---|---|---|---|---|---|---|---|
| Llama2-70b | 1532 | 18.9 | 0.0480 | 5700 | 3189 | 9 | 2502 |
| Mixtral | 1510 | 18.0 | 0.0431 | 5700 | 2853 | 9 | 2838 |
| Constant step-size | 1479 | 17.5 | 0.0409 | 5700 | 2705 | 6 | 2989 |
| One-Fifth rule | 1479 | 17.3 | 0.0392 | 5700 | 2640 | 2 | 3058 |

**Table 2: Glicko2-based ranking of the methods on functions from the BBOB suite with dimensionality 5.**

| Step-size adaptation method | Rating | Deviation | Volatility | Games | Win | Draw | Loss |
|---|---|---|---|---|---|---|---|
| Llama2-70b | 1562 | 17.8 | 0.0415 | 5700 | 3537 | 0 | 2163 |
| One-Fifth rule | 1495 | 16.6 | 0.0365 | 5700 | 2778 | 0 | 2922 |
| Mixtral | 1481 | 18.1 | 0.0436 | 5700 | 2695 | 0 | 3005 |
| Constant step-size | 1459 | 16.7 | 0.0359 | 5700 | 2390 | 0 | 3310 |

**Table 3: Glicko2-based ranking of the methods on functions from the BBOB suite with dimensionality 30.**

| Step-size adaptation method | Rating | Deviation | Volatility | Games | Win | Draw | Loss |
|---|---|---|---|---|---|---|---|
| Llama2-70b | 1570 | 17.3 | 0.0378 | 5700 | 3591 | 0 | 2109 |
| Constant step-size | 1540 | 16.6 | 0.0354 | 5700 | 3202 | 0 | 2498 |
| One-Fifth rule | 1457 | 16.7 | 0.0361 | 5700 | 2363 | 0 | 3337 |
| Mixtral | 1430 | 17.5 | 0.0390 | 5700 | 2244 | 0 | 3456 |


[23] Jiao, Aoran and Patel, Tanmay P and Khurana, Sanjmi and Korol, Anna-Mariya and Brunke, Lukas and Adajania, Vivek K and Culha, Utku and Zhou, Siqi and Schoellig, Angela P. 2023. Swarm-GPT: Combining Large Language Models with Safe Motion Planning for Robot Choreography Design. arXiv:2312.01059.

[24] Lanzi, Pier Luca and Loiacono, Daniele. 2023. ChatGPT and other large language models as evolutionary engines for online interactive collaborative game design. arXiv:2303.02155.

[25] Lehman, Joel and Gordon, Jonathan and Jain, Shawn and Ndousse, Kamal and Yeh, Cathy and Stanley, Kenneth O. 2023. Evolution through large models. In *Handbook of Evolutionary Machine Learning*. Springer, Singapore, 331–366.

[26] Liu, Yang and Wu, Fanyou and Liu, Zhiyuan and Wang, Kai and Wang, Feiyue and Qu, Xiaobo. 2023. Can language models be used for real-world urban-delivery route optimization? *The Innovation* 4, 6 (2023), 8 pages.

[27] Michael R. Zhang and Nishkrit Desai and Juhan Bae and Jonathan Lorraine and Jimmy Ba. 2023. Using Large Language Models for Hyperparameter Optimization. arXiv:2312.04528.

[28] Vishal Pallagani, Kaushik Roy, Bharath Muppasani, Francesco Fabiano, Andrea Loreggia, Keerthiram Murugesan, Biplav Srivastava, Francesca Rossi, Lior Horesh, and Amit Sheth. 2024. On the Prospects of Incorporating Large Language Models (LLMs) in Automated Planning and Scheduling (APS). arXiv:2401.02500.

[29] Pivithuru Thejan Amarasinghe and Su Nguyen and Yuan Sun and Damminda Alahakoon. 2023. AI-Copilot for Business Optimisation: A Framework and A Case Study in Production Scheduling. arXiv:2309.13218.

[30] Samuel A. Prieto, Eyob T. Mengiste, and Borja García de Soto. 2023. Investigating the Use of ChatGPT for the Scheduling of Construction Projects. *Buildings* 13, 4 (2023), 16 pages.

[31] Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic Prompt Optimization with "Gradient Descent" and Beam Search. arXiv:2305.03495.

[32] Qingyan Guo and Rui Wang and Junliang Guo and Bei Li and Kaitao Song and Xu Tan and Guoqing Liu and Jiang Bian and Yujiu Yang. 2023. Connecting Large Language Models with Evolutionary Algorithms Yields Powerful Prompt Optimizers. arXiv:2309.08532.

[33] Ingo Rechenberg. 1973. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien derbiologischen Evolution.* Frommann-Holzboog Verlag, Stuttgart, Germany.

[34] Ren, Allen Z and Dixit, Anushri and Bodrova, Alexandra and Singh, Sumeet and Tu, Stephen and Brown, Noah and Xu, Peng and Takayama, Leila and Xia, Fei and Varley, Jake and others. 2023. Robots that ask for help: Uncertainty alignment for large language model planners. arXiv:2307.01928.

[35] Romera-Paredes, Bernardino and Barekatain, Mohammadamin and Novikov, Alexander and Balog, Matej and Kumar, M Pawan and Dupont, Emilien and Ruiz, Francisco JR and Ellenberg, Jordan S and Wang, Pengming and Fawzi, Omar and others. 2023. Mathematical discoveries from program search with large language models. *Nature* (early access) (2023), 1–3.

[36] Moritz Vinzent Seiler, Pascal Kerschke, and Heike Trautmann. 2024. Deep-ELA: Deep Exploratory Landscape Analysis with Self-Supervised Pretrained Transformers for Single- and Multi-Objective Continuous Optimization Problems. arXiv:2401.01192.

[37] Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. 2022. Black-Box Tuning for Language-Model-as-a-Service. In *International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 162)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (Eds.). PMLR, Honolulu, HI, US, 20841–20855.

[38] Sun, Lingfeng and Jha, Devesh K and Hori, Chiori and Jain, Siddarth and Corcodel, Radu and Zhu, Xinghao and Tomizuka, Masayoshi and Romeres, Diego. 2023. Interactive Planning Using Large Language Models for Partially Observable Robotics Tasks. arXiv:2312.06876.

[39] Tao, Ning and Ventresque, Anthony and Saber, Takfarinas. 2023. Program Synthesis with Generative Pre-trained Transformers and Grammar-Guided Genetic Programming Grammar. In *IEEE Latin American Conference on Computational Intelligence*. IEEE, New York, NY, USA, 6 pages.

[40] Michele Tessari and Giovanni Iacca. 2022. Reinforcement learning based adaptive metaheuristics. In *Genetic and Evolutionary Computation Conference Companion*. ACM, New York, NY, USA, 1854–1861.

[41] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288.

[42] Christophe Tribes, Sacha Benarroch-Lelong, Peng Lu, and Ivan Kobyzev. 2024. Hyperparameter Optimization for Large Language Model Instruction-Tuning. arXiv:2312.00949.
[43] Voß, Stefan. 2023. Successfully Using ChatGPT in Logistics: Are We There Yet?. In *Computational Logistics*. Springer Nature, Cham, Switzerland, 3–17.
[44] Wang, Lirui and Ling, Yiyang and Yuan, Zhecheng and Shridhar, Mohit and Bao, Chen and Qin, Yuzhe and Wang, Bailin and Xu, Huazhe and Wang, Xiaolong. 2023. GenSim: Generating Robotic Simulation Tasks via Large Language Models. arXiv:2310.01361.
[45] Xingyu Wu, Sheng hao Wu, Jibin Wu, Liang Feng, and Kay Chen Tan. 2024. Evolutionary Computation in the Era of Large Language Model: Survey and Roadmap. arXiv:2401.10034.
[46] Xianggen Liu and Pengyong Li and Fandong Meng and Hao Zhou and Huasong Zhong and Jie Zhou and Lili Mou and Sen Song. 2021. Simulated annealing for optimization of graphs and sequences. *Neurocomputing* 465 (2021), 310–324.
[47] Yaman, Anil and Hallawa, Ahmed and Coler, Matt and Iacca, Giovanni. 2017. Presenting the ECO: evolutionary computation ontology. In *Applications of Evolutionary Computation*. Springer, Cham, Switzerland, 603–619.
[48] Yaman, Anil and Iacca, Giovanni and Caraffini, Fabio. 2019. A comparison of three differential evolution strategies in terms of early convergence with different population sizes. In *AIP Conference Proceedings*, Vol. 2070. AIP Publishing, Melville, NY, USA, 4 pages.
[49] Shujian Zhang, Chengyue Gong, Lemeng Wu, Xingchao Liu, and Mingyuan Zhou. 2023. AutoML-GPT: Automatic Machine Learning with GPT. arXiv:2305.02499.
[50] Jiangjiang Zhao, Zhuoran Wang, and Fangchun Yang. 2023. Genetic Prompt Search via Exploiting Language Model Probabilities. In *International Joint Conference on Artificial Intelligence*, Edith Elkind (Ed.). International Joint Conferences on Artificial Intelligence Organization, Macao, 5296–5305.
[51] Mingkai Zheng, Xiu Su, Shan You, Fei Wang, Chen Qian, Chang Xu, and Samuel Albanie. 2023. Can GPT-4 Perform Neural Architecture Search? arXiv:2304.10970.