ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

Large Language Models as Tuning Agents of Metaheuristics

Alicja Martinek[1,2] and Szymon Łukasik[1,2] and Amir H. Gandomi[3,4] *

1 - NASK - National Research Institute, Warsaw, Poland
2 - AGH University of Technology in Kraków, Kraków, Poland
3 - University of Technology Sydney, Sydney, Australia
4 - Obuda University, Budapest, Hungary

Abstract. This study examines whether LLMs can be utilized in metaheuristic tuning through selection of appropriate parameters. Instances of two optimization problems, Travelling Salesman and Graph Coloring, were solved with GA, ACO, PSO, and SA. Experiment involved running these heuristic optimizers with parameter values advised by LLMs. A round of feedback was performed through feeding LLMs with prompts that included initial parameters, average performance, and population variance, where applicable. The results show LLMs exhibit the ability to comprehend the non-trivial task of tuning metaheuristics' parameters. Additionally, feedback runs often outperform results achieved by initial setups, yielding a new application of LLMs.

1 Introduction

Large Language Models (LLMs), since their introduction, made a significant impact on multiple industries. They have been successfully applied in numerous branches of computer science. Most notable use cases include but are not limited to Natural Language Processing, chatbots and assistance systems, content creation, and guidance in code development [1]. The first widely acknowledged LLM, BERT by Google, was introduced in 2018 [2]. However, the true revolution broke out when OpenAI released publicly its ChatGPT in 2022. Since then, the amount of papers and work devoted to LLMs is unprecedented.

Nonetheless, LLMs are not the only area of constant development in the computer science domain. Metaheuristic Algorithms (MHAs) are still a hot topic for many scientists, as their ability to optimize complex tasks is of great value. With the rapidly growing MHAs, it is hard to keep up with all new ideas and implementations. Recent attempts to classify and organize these algorithms result in collections of over 500 distinct methods [3]. Each heuristic optimizer has its own distinct set of parameters. Therefore, it is not trivial to remember and understand all of these setups. To harness such detailed knowledge, it would be beneficial to have guidance and advice on good practices and desired parameter values for a wide range of MHAs.

This paper's main contributions include examining various LLMs and their capabilities as tuning agents of metaheuristics, giving some insightful, prompt

*This work was partially supported by the program "Excellence initiative - research university" for the AGH University of Krakow and by Grant for Statutory Activity from the Faculty of Physics and Applied Computer Science of the AGH University of Krakow.

advice in this context, and comparing tested models. The possibility of having a head start as opposed to the default setting of the algorithm is highly valuable and desired. Especially when the given set is already suited to the problem being solved. This happens with using LLMs, where one can build a prompt containing general information about the task. Such an approach often exceeds the results achieved by running an algorithm with a default setup. Consequently, using LLM-advised and problem-dedicated parameters constitutes a great way of shifting the starting point towards already better and more promising solutions.

2 Related Work

The intersection of Large Language Models and Metaheuristic Algorithms yields an abundance of possible applications. LLMs and MHAs can interact with each other in two directions with regard to the subject of optimization. The first approach, more popular in literature, addresses the exploding popularity of LLMs themselves. This direction employs Evolutionary Algorithms (EAs) to optimize prompts [4], inputs of LLMs. That is a big research field, aiming to find the most appropriate cues that would trigger a desired response of the Large Language Model. Another way of utilizing bio-inspired computation is present in [5]. In this work, the introduced framework mimics the Genetic Algorithm to generate a set of candidate prompts. The alternative body of research is devoted to using LLMs as systems that can design novel metaheuristics [6].

The second direction of LLMs and metaheuristics engagement employs the first to optimize or tune the latter. The aforementioned works outline that, so far, most interest is placed in the opposite direction of the interaction and using LLMs to influence MHAs is not fully explored yet. However, the other perspective of marrying together LLMs and EAs is present in [7]. This paper uses LLM to perform selection and crossover, making it a sole part of the heuristic itself. Such an approach is called an LLM-driven EA. A comprehensive overview of both directions of interactions between LLMs and Evolutionary Algorithms is presented in [8]. A limited amount of existing research in the field makes this subject an exciting and open problem worth further exploration.

MHAs provide frameworks for search processes. They can be inspired by biology, human behavior, the sociology of mammals, physical phenomena, or even sports strategies [9]. The most widespread examples of heuristic optimizers are Genetic Algorithm (GA), Ant Colony Optimization (ACO), Simulated Annealing (SA), or Particle Swarm Optimization (PSO). Despite the major drawback of increasing computational complexity, these algorithms exhibit extraordinary abilities in a variety of optimization tasks.

Combinatorial optimization problems create a rich playground for scientists to find new approaches to solution seeking. Several challenges and also less research-oriented events are devoted to solving such tasks. However, the oldest and most well-studied problems make the best benchmarks as the primary attention is directed not to the problem itself but to evaluating the novel idea of solving it.

3 Methodology

The aim of this study is to examine if Large Language Models are capable of optimizing MHAs' parameters. As the models of interest, it was decided to evaluate both versions of ChatGPT by OpenAI, Gemini by Google, and Le Chat by Mistral AI models and thoroughly test their ability to tune metaheuristics. To conduct such an experiment, two well-known discrete optimization problems listed below, namely TSP and GCP, were chosen. An instance of TSP *P01* is defined on 15 cities, and it is provided along with TSPLIB [10]. The experimental problem used for the second task, For the Graph Coloring Problem, the *myciel3* graph was taken from [11], and it is defined by the graph of 11 vertices and 20 edges.

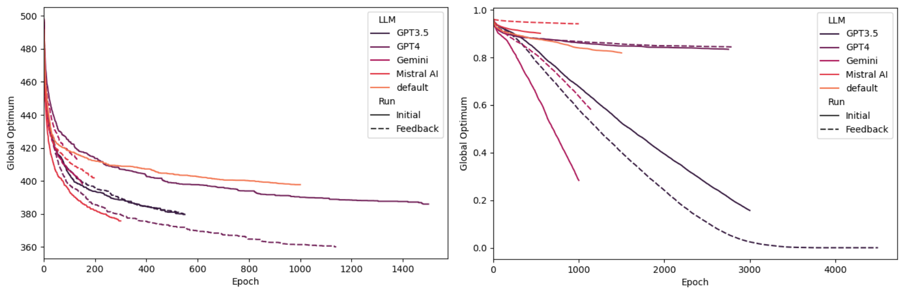
The LLMs themselves selected metaheuristic techniques by prompting which algorithms best suit the given task. Once the problem instances and heuristic optimization methods were chosen, the LLMs were asked to suggest initial parameter values for heuristics. Moreover, the set of default values for each metaheuristic was taken as a source for baseline experiments. Mealpy [12] was determined as a go-to library for handling the optimization part of the experiment. It is a well-established project that includes a huge collection of heuristic optimizers. Another reason for selecting an existing, open-source library as opposed to personal implementations is that all 4 LLMs are aware of Mealpy's code, as documentation was part of their training corpora. Table 1 displays 3 types of exemplary prompts.

Which metaheuristic algorithms would you use to solve a *TSP*?
I want to solve *TSP* (defined on *15 cities*) with *GA*. Its mealpy implementation takes following parameters: *parameters.* Give advice on param values.
For *TSP* task with *15 cities* you previously suggested parameter values for *Genetic Algorithm* metaheuristic. *Parameters and values.* I run the *GA* algorithm 100 times and got the avg global optimum of *0.375* with std of *0.03.* I also measured variance of the population at the beginning and last epoch: *9.598* and *5.675* correspondingly, with std of *0.09* and *0.58.* The solution at last epoch had avg fitness of *0.45* with std of *0.37.* What changes to the parameters would you suggest to improve performance? Keep the epoch*pop_size constant.

Table 1: Exemplary prompts. Text in italics is adjusted for each setup.

When LLM could not give an exact value for a given parameter, the default one was used. In other scenarios, when the answer contained a range of values, the middle point was calculated and passed to the heuristic optimizer. Each MHA was run 100 times to achieve statistical significance, and after all of the runs, the average values and standard deviations were derived.

To minimize the effect of the obvious advice of increasing population size or the number of epochs, a constraint of the constant product of these two values was introduced. This approach enables fair comparison between various runs of experiments and fixes the computational cost.



(a) (b)

Fig. 1: (a) Genetic Algorithm solutions to Travelling Salesman Problem - *P01* [10]. (b) Simulated Annealing solutions to Graph Coloring Scheduling Problem - *myciel3* [11].

As a consequence of selecting combinatorial optimization problems, the representation of solutions is a vector adjusted accordingly to the task. To measure the variance of generated populations, the Hamming distance was calculated.

To collect a new (feedback) set of parameter values, prompts containing information about the problem, initial setup, population variance (if applicable), global optimum, and value at last epoch with their corresponding standard deviations were fed to LLMs. MHAs were rerun 100 times with new parameter values and later compared with initial settings as well as with the default one.

4 Experimental Results

Detailed analysis of results is presented with regard to a certain task. Table 2 contains complete findings based on all experiments. Figure 1 presents exemplary convergence curves for 2 problems. It has to be pointed out that due to the constant number of fitness function evaluations between both runs and the fact that each LLM had the freedom to suggest its own values for them, curves end in various epochs.

The first problem was solved with the use of ACO, GA, and SA. Figure 1(a) shows the convergence curve of the Genetic Algorithm. It can be observed that every LLM setup outperformed default settings. Interestingly, GPT4 in feedback run yielded improvement from 386.01 to 359.97. The shape of the curves suggests that if one increased the number of epochs for feedback runs, the results would be even better.

The biggest disproportion can be observed in the case of Simulated Annealing (see Table 2), where the default run settled at 478.315, whereas Mistral AI ended calculations at 397.762 in the initial run. The feedback round of computation resulted in a slight increase in solution quality, down to 391.805.

Solving GCP with Simulated Annealing resulted in the most differentiated outcomes depicted in Figure 1(b). GPT3.5 not only found the global minimum,

			TSP	
LLM	Run	ACO	GA	SA
GPT 3.5	Initial	425.212 ± 14.26 (550)	379.71 ± 16.85 (550)	398.28 ± 28.91 (2000)
	Feedback	422.718 ± 14.65 (550)	379.948 ± 16.81 (550)	400.415 ± 25.1 (3000)
GPT 4	Initial	422.226 ± 14.12 (750)	**386.01 ± 14.47 (1500)**	407.203 ± 31.61 (2000)
	Feedback	423.346 ± 14.61 (500)	**359.97 ± 17.16 (1140)**	407.951 ± 32.69 (2000)
Gemini	Initial	440.241 ± 15.58 (150)	398.193 ± 17.44 (150)	411.658 ± 28.38 (1500)
	Feedback	451.979 ± 16.08 (75)	412.138 ± 16.59 (134)	406.618 ± 27.73 (1725)
Mistral AI	Initial	427.52 ± 16.07 (300)	375.811 ± 17.53 (300)	397.762 ± 26.1 (3000)
	Feedback	422.708 ± 15.92 (600)	401.038 ± 12.8 (200)	391.805 ± 23.6 (5000)
default	Initial	415.719 ± 13.01 (1000)	397.779 ± 10.3 (1000)	478.315 ± 29.95 (1500)
			GCP	
		PSO	GA	SA
GPT 3.5	Initial	0.403 ± 0.04 (400)	0.375 ± 0.05 (500)	**0.157 ± 0.24 (3000)**
	Feedback	0.078 ± 0.08 (300)	0.593 ± 0.03 (500)	**0 ± 0.31 (4500)**
GPT 4	Initial	0.438 ± 0.04 (150)	0.364 ± 0.05 (300)	0.835 ± 0.02 (2750)
	Feedback	0.473 ± 0.04 (120)	0.364 ± 0.03 (450)	0.845 ± 0.02 (2800)
Gemini	Initial	0.44 ± 0.04 (150)	0.387 ± 0.06 (300)	0.283 ± 0.2 (1000)
	Feedback	0.333 ± 0.06 (120)	0.515 ± 0.04 (300)	0.578 ± 0.1 (1150)
Mistral AI	Initial	0.336 ± 0.03 (1000)	0.394 ± 0.09 (300)	0.902 ± 0.01 (550)
	Feedback	0.276 ± 0.03 (714)	0.385 ± 0.09 (150)	0.942 ± 0 (1000)
default	Initial	0.056 ± 0.04 (1000)	0.364 ± 0.04 (1000)	0.819 ± 0.03 (1500)

Table 2: Results of conducted experiments. Global Optimum values are averaged over 100 runs of each setup, reported with a standard deviation. For population-based algorithms (ACO, GA, PSO), the number of fitness function evaluations remains constant. Figures in brackets reflect the amount of epochs each setup was run for.

but its feedback run converged faster than the initial one would. What is more, Gemini outperformed the default run, which in this scenario has a gentle convergence curve. Analysis of convergence curves suggests that increasing the number of epochs for Gemini-based runs would yield the fastest convergence. Notable improvement can also be observed in GPT3.5-based runs of the PSO algorithm, with the average Global Optimum of 0.403 and 0.078 in initial and feedback runs correspondingly.

5 Conclusions

Results obtained throughout experiments gave evidence to support the hypothesis that Large Language Models are capable of tuning metaheuristicsâ parameters. It proves their ability to reason and draw meaningful conclusions based on information delivered by prompts. These models, regardless of the provider, exhibit abilities not only to outperform default algorithm setups but also to adapt to obtained results and adjust parameter values where needed. Moreover, it was observed that there is no single model that outperforms other LLMs.

In terms of prompt engineering - achieved results demonstrate that LLMs do not require detailed information regarding problems being solved. There is no need to feed the exact data via prompts to get practical suggestions for parameter changes.

With the intention of deeper exploration of a presented idea, one could extend this research to include other LLMs, run more feedback rounds and experiment with more benchmark instances, or even train one's own Large Language Model aimed to tune MHAs parameters. It is a new field of research worth further study. The findings of our paper can open up a still-growing field of MHAs to everyone by reducing the knowledge required to consciously use them by involving LLMs in the process.

References

[1] Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. Challenges and applications of large language models, 2023.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2019.

[3] Kanchan Rajwar, Kusum Deep, and Swagatam Das. An exhaustive review of the metaheuristic algorithms for search and optimization: taxonomy, applications, and open challenges. *Artificial Intelligence Review*, 56, 04 2023.

[4] Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *The Twelfth International Conference on Learning Representations*, 2024.

[5] Yongchao Chen, Jacob Arkin, Yilun Hao, Yang Zhang, Nicholas Roy, and Chuchu Fan. Prompt optimization in multi-step tasks (promst): Integrating human feedback and preference alignment, 2024.

[6] Michal Pluhacek, Anezka Kazikova, Tomas Kadavy, Adam Viktorin, and Roman Senkerik. Leveraging large language models for the generation of novel metaheuristic optimization algorithms. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, GECCO '23 Companion, page 1812â1820, New York, NY, USA, 2023. Association for Computing Machinery.

[7] Shengcai Liu, Caishun Chen, Xinghua Qu, Ke Tang, and Yew-Soon Ong. Large language models as evolutionary optimizers, 2023.

[8] Xingyu Wu, Sheng hao Wu, Jibin Wu, Liang Feng, and Kay Chen Tan. Evolutionary computation in the era of large language model: Survey and roadmap, 2024.

[9] Absalom Ezugwu, Amit Shukla, Rahul Nath, Ayo Akinyelu, Ovre Agushaka, Haruna Chiroma, and Pranab Muhuri. Metaheuristics: a comprehensive overview and classiication along with bibliometric analysis. *Artificial Intelligence Review*, 54:1–79, 03 2021.

[10] John Burkardt. TSP Data for the Traveling Salesperson Problem. https://people.sc.fsu.edu/ jburkardt/datasets/tsp/tsp.html. Accessed: 2024-03-01.

[11] Michael Trick. Michael Trick's Operations Research Page. https://mat.gsia.cmu.edu/COLOR/. Accessed: 2024-03-20.

[12] Nguyen Van Thieu and Seyedali Mirjalili. Mealpy: An open-source library for latest meta-heuristic algorithms in python. *Journal of Systems Architecture*, 2023.