НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ноябрь–декабрь 2025 Том 25 № 6 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
November–December 2025 Vol. 25 No 6 http://ntv.ifmo.ru/en/
ISSN 2226-1494 (print) ISSN 2500-0373 (online)

НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

doi: 10.17586/2226-1494-2025-25-6-1134-1141

ReflectivePrompt: Reflective evolution in autoprompting algorithms

Viktor N. Zhuravlev[1], Artur R. Khairullin[2], Ernest A. Dyagin[4], Alena N. Sitkina[4], Nikita I. Kulin[5]✉

[1,2,3,4,5] ITMO University, Saint Petersburg, 197101, Russian Federation

[1] viktor.zhuravlev2003@mail.ru, https://orcid.org/0009-0009-0788-9790
[2] arkhairullin@itmo.ru, https://orcid.org/0009-0007-0442-7764
[3] tsenred@yandex.com, https://orcid.org/0009-0000-3865-4446
[4] sitkina.alena2017@yandex.ru, https://orcid.org/0009-0002-6046-8943
[5] kylin98@list.ru✉, https://orcid.org/0000-0002-3952-6080

Abstract
Autoprompting is the process of automatically selecting optimized prompts for language models, which has been gaining popularity with the rapid advancement of prompt engineering driven by extensive research in the field of Large Language Models. This paper presents ReflectivePrompt — a novel autoprompting method based on evolutionary algorithms that employs a reflective evolution approach for more precise and comprehensive search of optimal prompts. ReflectivePrompt utilizes short-term and long-term reflection operations before crossover and elitist mutation to enhance the quality of the modifications they introduce. This method allows for the accumulation of knowledge obtained throughout the evolution process and updates it at each epoch based on the current population. ReflectivePrompt was tested on 33 datasets for classification and text generation tasks using open-access large language models: T-lite-instruct-0.1 and Gemma3-27b-it. The method demonstrates, on average, a significant improvement (e.g., 28 % on BBH compared to EvoPrompt) in metrics relative to current state-of-the-art approaches, thereby establishing itself as one of the most effective solutions in evolutionary algorithm-based autoprompting.

Keywords
LLM, autoprompting, evolutionary algorithms, reflective evolution, prompt engineering

For citation: Zhuravlev V.N., Khairullin A.R., Dyagin E.A., Sitkina A.N., Kulin N.I. ReflectivePrompt: Reflective evolution in autoprompting algorithms. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2025, vol. 25, no. 6, pp. 1134–1141. doi: 10.17586/2226-1494-2025-25-6-1134-1141

УДК 004.89

ReflectivePrompt: использование рефлексивной эволюции в алгоритмах автопромптинга

Виктор Николаевич Журавлев[1], Артур Рустамович Хайруллин[2], Эрнест Александрович Дягин[3], Алена Николаевна Ситкина[4], Никита Игоревич Кулин[5]✉

[1,2,3,4,5] Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация

[1] viktor.zhuravlev2003@mail.ru, https://orcid.org/0009-0009-0788-9790
[2] arkhairullin@itmo.ru, https://orcid.org/0009-0007-0442-7764
[3] tsenred@yandex.com, https://orcid.org/0009-0000-3865-4446
[4] sitkina.alena2017@yandex.ru, https://orcid.org/0009-0002-6046-8943
[5] kylin98@list.ru✉, https://orcid.org/0000-0002-3952-6080

Аннотация
Автопромптинг — процесс автоматического подбора оптимизированных промптов к языковым моделям, который набирает свою популярность с быстрым развитием промпт-инжиниринга, обусловленного многочисленными исследованиями в области больших языковых моделей. В работе представлен ReflectivePrompt — новый метод автопромптинга на основе эволюционных алгоритмов, использующий подход рефлексивной эволюции для более точного и расширенного поиска оптимальных промптов. ReflectivePrompt использует операции

краткосрочной и долгосрочной рефлексии перед операциями скрещивания и элитарной мутации для повышения качества проводимых ими изменений. Предложенный метод позволяет накапливать знания, полученные на протяжении всей эволюции, и обновлять их на каждой эпохе исходя из текущей популяции. ReflectivePrompt был протестирован на 33 наборах данных по задачам классификации и генерации текста с использованием больших языковых моделей с открытым доступом: T-lite-instruct-0.1, Gemma3-27b-it. Представленный метод продемонстрировал значительное увеличение (например, 28 % в среднем на бенчмарке Big-Bench-Hard относительно EvoPrompt) по метрикам в сравнении с известными методами в данной области, тем самым показал себя одним из самых эффективных вариантов в рамках автопромптинга на основе эволюционных алгоритмов.

Ключевые слова
большие языковые модели, автоматический промптинг, эволюционные алгоритмы, рефлексивная эволюция, промпт-инжиниринг

Ссылка для цитирования: Журавлев В.Н., Хайруллин А.Р., Дягин Э.А., Ситкина А.Н., Кулин Н.И. ReflectivePrompt: использование рефлексивной эволюции в алгоритмах автопромптинга // Научно-технический вестник информационных технологий, механики и оптики. 2025. Т. 25, № 6. С. 1134–1141 (на англ. яз.). doi: 10.17586/2226-1494-2025-25-6-1134-1141

Introduction

Large Language Models (LLMs) have demonstrated significant results in solving Natural Language Processing (NLP) tasks [1, 2]. Prompting and prompt engineering are universal methods for improving the performance of LLMs that do not require access to model weights and gradients during training. Instead, they enhance the efficiency of LLM inference by providing carefully crafted and well-structured instructions (prompts) as input to the model [3]. Currently, there are many different prompting techniques, such as Few-Shot [4], Role-Based [5], Chain-of-Thought [6], Plan-and-Solve [7], and others. What all these techniques have in common is that they can be time-consuming to manually create, iterate, and optimize, often requiring expert knowledge and experience. The reason for this is that models are highly sensitive to input data, necessitating careful and precise application of these techniques [8].

Autoprompting addresses this issue by automating the generation and selection of prompts [9]. It is based on various optimization methods and principles, including reinforcement learning, evolutionary, gradient-based, and gradient-free approaches, among others [9–12]. In particular, prompt optimization can be either discrete or continuous [13]. Continuous optimization involves representing the prompt as a numerical tensor, while discrete optimization treats the prompt as a sequence of tokens. The latter approach offers several advantages: it does not require derivative computations, meaning there is no need to access the model internal parameters and gradients. This allows working with black-box models and avoids additional computational overhead [14]. Additionally, it preserves prompt interpretability, enabling humans to analyze and edit them [14], and allows optimization for any metric (including non-differentiable ones) [11, 15–17].

However, this approach also has challenges: the optimization space for prompts is vast, and prompts generated through search methods may lack diversity [11]. Nevertheless, there are numerous heuristic optimization algorithms that employ stochastic strategies, making the optimization process less sensitive to local optima. Evolutionary algorithms are one such example [18].

In this work, we analyzed the Reflective Evolution algorithm [19] and integrated it to address the problem of automatic prompt generation. The resulting solution, called ReflectivePrompt, was tested on 33 datasets to demonstrate its effectiveness compared to existing methods.

Evolutionary algorithms

Evolutionary algorithms are a family of optimization methods based on the principles of biological evolution: natural selection, mutation, crossover, and inheritance. These algorithms operate with populations of solutions, gradually improving them according to a given fitness function [18]. Among such algorithms, the genetic algorithm [20] can be distinguished which works with gene sequences (in our case, sequences of phrases in prompts). Within this algorithm, starting with an initial population of individuals, selection, crossover of selected individuals (creating offspring based on a combination of parental information), mutation of the offspring (random modification of certain parts), and population update based on offspring evaluations are performed iteratively.

This approach is highly flexible when applied to problems from various domains. The usage of evolutionary operators (crossover, mutation) and population-based search reduces the risk of getting stuck in local optima, maintaining a balance between exploring new solutions and exploiting existing ones, thereby leading to a high diversity of individuals in the final population while ensuring their quality according to the objective function remains high [11, 15–17].

Related works

One solution employing genetic algorithms is EvoPrompt [11]. The improvement of the candidate prompt population occurs iteratively through selection, evolution (generation of new candidates using evolutionary operators), and population updates based on the evaluation of new candidates. The implementation of evolutionary operators (mutation and crossover) is achieved through queries to an LLM, enabling the utilization of its expertise in solving NLP tasks while maintaining prompt readability. During new candidate generation, two parents are first selected from the previous population using roulette-wheel selection (selection phase) [21] followed by the application of crossover and subsequent mutation of the resulting offspring. The study also presents a differential evolution algorithm [22], which involves mutating different segments of two donor prompts from the same population, combining them with a mutating candidate, and performing crossover with the current best prompt. The authors' position EvoPrompt as a general framework for integrating LLMs into evolutionary algorithms, with experimental results demonstrating that differential evolution exhibits superior performance on more complex tasks.

Semantic Prompt Evolution based on a LLM (SPELL) [15] employs a genetic algorithm operating iteratively through repeated reproduction and selection steps where selection is performed via roulette-wheel while reproduction involves generating offspring based on a list of parent prompts and their corresponding scores. Notably, although reproduction is also conducted through LLM queries, this solution lacks explicit separation between crossover and mutation. Instead, it utilizes predefined prompt instructing modifications to the parent prompt set (replacing, adding, or deleting words, altering tone) to generate offspring.

An alternative approach implemented in Plum [16] is based on metaheuristics. Unlike EvoPrompt prompt mutation methodology, Plum explicitly defines a set of prompt modification operations: adding, deleting, rephrasing words/phrases, or swapping their positions, thereby generating multiple neighboring prompts. The solution architecture comprises: a well-defined set of neighboring prompts for each prompt, a metaheuristic algorithm with its inherent hyperparameters, and auxiliary functions (including crossover). The study examines six algorithms: hill climbing [23], simulated annealing [24], genetic algorithm (two variants – with mutation and crossover, and mutation-only) [20], tabu search [25], and harmony search [26]. Each algorithm performs candidate mutation through the application of predefined modification operations, enabling exploration within the discrete prompt space. It should be noted that only the rephrasing operation is executed via LLM queries, while other operations are performed manually. As described by the authors, experimental results demonstrate this approach capability to identify novel structural prompt modifications that enhance performance.

In Promptbreeder [17] paper, the authors propose an extended genetic algorithm mutation approach incorporating predefined mutation prompts and "thinking styles" (concise descriptions of cognitive strategies, e.g., "Let's think step by step"), in addition to utilizing Chain-of-Thought [6] and Plan-and-Solve [7] techniques. At each iteration, candidates are improved through the application of a randomly selected mutation from a uniform distribution. The authors identify five mutation classes: direct mutation, hypermutation, estimation of distribution mutation [27], Lamarckian mutation [28], and prompt crossover/context shuffling. The first two classes further include zero-order and first-order mutations, totaling ten distinct mutations, each implemented through LLM queries. First-order direct mutation modifies candidates using specific mutation prompts, while zero-order mutation utilizes the initial problem statement to address method divergence. When problem specifications lack precision, Lamarckian mutation facilitates prompt reconstruction based on the last output yielding correct results. The algorithm key innovation involves hypermutation which modifies the mutation prompts themselves (via hypermutation prompts), thereby enhancing not only prompt solutions but also the improvement mechanisms. According to the authors, this diversity of operators enables continuous reformulation and representation of problems by LLMs, leading to more effective solutions [17]. This approach demonstrates adaptability across various domains while optimizing prompts and preserving their interpretability.

Proposed method: ReflectivePrompt

Reflective Evolution

Reflective Evolution is an approach described in the article Ye H. et al. [19]. Its essence lies in using a language model to generate prompts aimed at enhancing the efficiency of mutation and crossover operations. The processes of prompt creation are referred to as short-term and long-term reflection. According to the authors, such reflective actions can be interpreted as obtaining a "verbal gradient" within the prompt space. Short-term reflection involves generating crossover prompts based solely on the current parent population, while long-term reflection, as the name suggests, entails accumulating knowledge, dependencies, and methods for improving efficiency throughout the entire evolutionary operation.

The application of reflection helps guide the direction of mutation and crossover operations while also expanding the search space, potentially moving beyond the initial population's predefined prompt space. In the original article, this approach was successfully applied to solving problems, such as Guided Local Search [29], Ant Colony Optimization [30], Electronic Design Automation [31], the Decap Placement Problem [32], the Traveling Salesman Problem [33], and other combinatorial optimization tasks.

Proposed solution

In this study we developed a novel approach that combines methods of Reflective Evolution with large language models for the automatic generation of higher-quality prompts — ReflectivePrompt. ReflectivePrompt employs short-term and long-term reflection operations for subsequent use in crossover and elitist mutation. All performed operations and their corresponding queries to the language model were modified and refined to directly optimize prompts.

Specifically, the beginning of each instruction was changed to: "You are an expert in the domain of optimization prompts. Your task is to give hints to design better prompts". This adjustment is motivated by the specifics of the autoprompting task for which reflective evolution was applied. Techniques describing possible modification operations performed during crossover and mutation, previously used in the SPELL algorithm, were incorporated. Thus, the following was added to the model queries defining short-term and long-term reflection: "For example, you can try to recommend word replacements, active/positive voice conversions, adding words, or deleting words". This enables the LLM to generate more precise, well-described hints that affect not only the semantic content of prompts but also their structural aspects.

ReflectivePrompt simplifies user interaction by generating an initial population of prompts based on just a single input prompt. In this approach, the prompt is rephrased using the LLM and structured output techniques [34].

A key feature of ReflectivePrompt is delegating the decision on the specifics of mutation to the model itself. In previously described solutions, the mutation type was either predefined and fixed or randomly selected from a uniform distribution. In this approach, however, the model generates hints autonomously and tends to decide whether to apply structural transformations to the prompt or only modify its semantic meaning and phrasing.

When performing crossover and mutation operations, the LLM is provided with a brief task description which helps generate more problem-targeted prompts while preserving the logical structure of the instruction. Empirical observations have shown that even large models can achieve decent metric values using prompts that are partially or entirely irrelevant to the task. As a result, the final prompts may deviate significantly from the intended meaning. ReflectivePrompt avoids this issue and, in the vast majority of cases, generates semantically correct prompts that are more comprehensible to human perception and logic.

The general scheme of Reflective Evolution within ReflectivePrompt is illustrated in Fig. 1.

Particular attention should be paid to the two selection operations. The parent population selection of prompts chooses pairs of parent prompts from the current population. In this process, each prompt can be included in multiple parent pairs. The main constraint, which is related to the original Reflective Evolution algorithm, is that prompts in a parent pair must have different fitness function values. Parent selection is performed using the roulette-wheel method [21]. The probability vector for being selected for each individual is represented by the normalized vector of their fitness scores. The second selection operation, which mimics the survival of the fittest, also employs the roulette-wheel method, but in this case, the probabilities are obtained by applying a softmax operation with a temperature of 0.1 to the fitness function value vector. This temperature value yields a less uniform distribution in cases where all prompts have approximately similar scores, thereby increasing the probability of selection for individuals with higher fitness values.

Another crucial aspect is the preservation of elite individuals in the population. Before the start of each epoch, the individual that has demonstrated the best performance throughout the entire evolutionary process is reintroduced into the population, even if it was not selected at the end of the previous iteration. This approach enhances the algorithm convergence speed, as the best individuals are not lost over time due to unfavorable selection outcomes.

The examples of ReflectivePrompt optimization are shown below.

Example 1. The optimized prompt for SST-2 dataset:

Initial prompt:

“Please provide a Sentiment Classification task.”

Optimized prompt:

“Could you please conduct a Sentiment Classification exercise on movie reviews?”

Example 2. The optimized prompt for BBH/logical_deduction_three_objects dataset:

Initial prompt:

“Identify the position of each object in a fixed order based on the given statements in each input-output pair.”

Optimized prompt:

“Determine the order of three objects based on the given statements. Each paragraph will contain three objects and a set of statements describing theirproperties, sizes, or positions. You will be provided with a list of statementsabout the objects, and your task is to analyze these statements to deduce thecorrect order of the objects.

For each paragraph, you will find three objects and a set of statements. These statements will describe the objects dimensions, weights, or positions relative to each other. Use this information to deduce the correct order of the objects. For example, consider the paragraph below:

1. The object on the left is smaller than the object in the middle.

2. The object in the middle is larger than the object on the right.

3. The object on the right is transparent.

Based on these statements, determine the correct order of the objects. The possible orderings are:

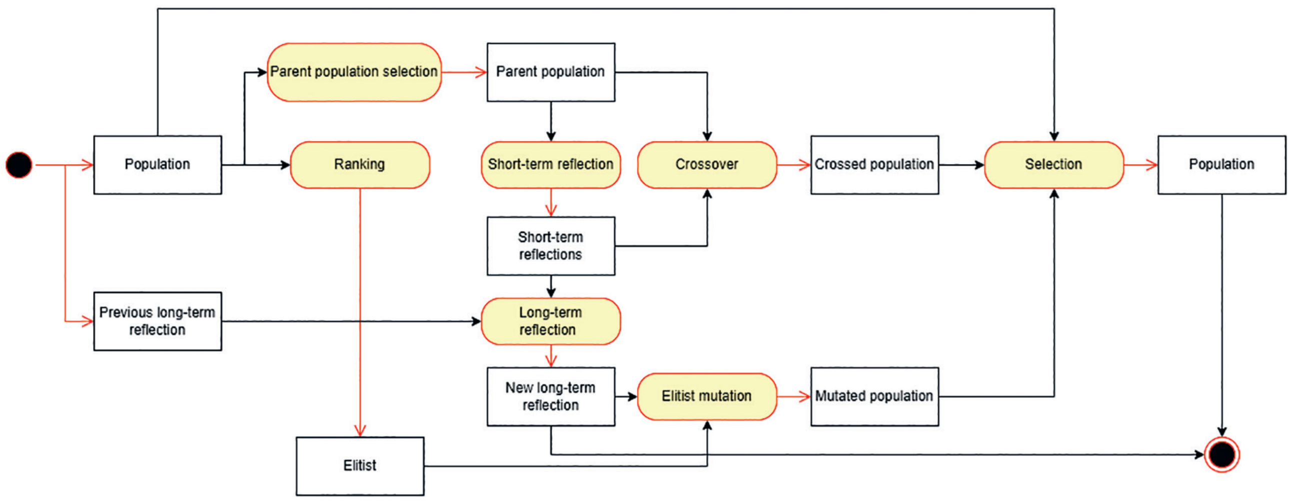

Fig. 1. The Reflective Evolution pipeline in ReflectivePrompt

A. Left, Middle, Right
B. Middle, Left, Right
C. Right, Middle, Left
D. Right, Left, Middle
Please, select the correct order from the options provided."

Experimental Results

ReflectivePrompt was evaluated on 33 datasets for text classification and generation tasks. As baselines and reference points for comparison, we used results from EvoPrompt, SPELL, PromptBreeder, and Plum. The autoprompting algorithms were executed using large language models from different families and sizes (T-lite-instruct-0.1[1], Gemma3-27b-it [35]). This choice of LLMs was made due to the use of open source white-box models which are more user-friendly and can be utilized by everyone. Also the significant difference in the number of model parameters leads to better testing coverage and makes our results more unbiased.

[1] T-lite-instruct-0.1 — Hugging Face, 2025. URL: https://huggingface.co/AnatoliiPotapov/T-lite-instruct-0.1 (accessed: 19.06.2025).

Classification tasks

For classification tasks, the following datasets and benchmarks were used: MNLI, MR, SST-2, YAHOO, and BBH (a subset of datasets with strictly formatted answers that can be treated as classification tasks). The metric selected for evaluation and optimization during evolution was the F1-score. The results of each method are presented in Fig. 2.

Generation tasks

ReflectivePrompt and its counterparts were evaluated on the following datasets: BBH (dyck_languages, multistep_arithmetic_two, object_counting, word_sorting), GSM8K, and SamSUM. The metric used for evaluation and optimization was METEOR. The main results are shown in Fig. 3.

Discussion

The conducted experiments demonstrate that ReflectivePrompt effectively handles both classification and text generation tasks. Across all evaluated datasets, ReflectivePrompt either outperformed or matched the performance of existing evolutionary algorithm-based autoprompting methods. The method showed particularly strong results on the BBH benchmark, comprising 23 classification tasks and 4 text generation tasks. For

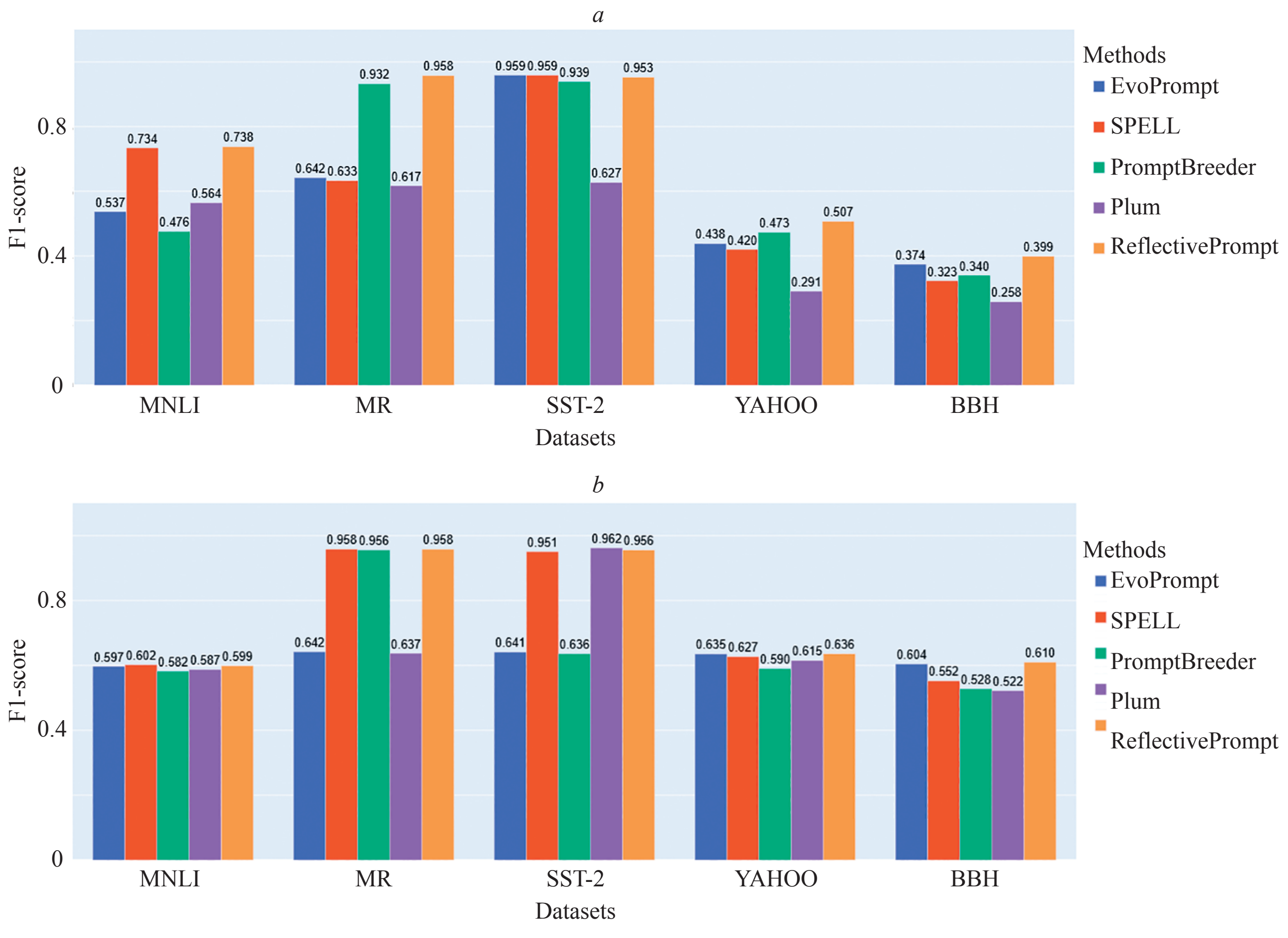

Fig. 2. Histogram of F1-score values.
Modes T-lite-instruct-0.1 (*a*) and Gemma3-27b-it (*b*)

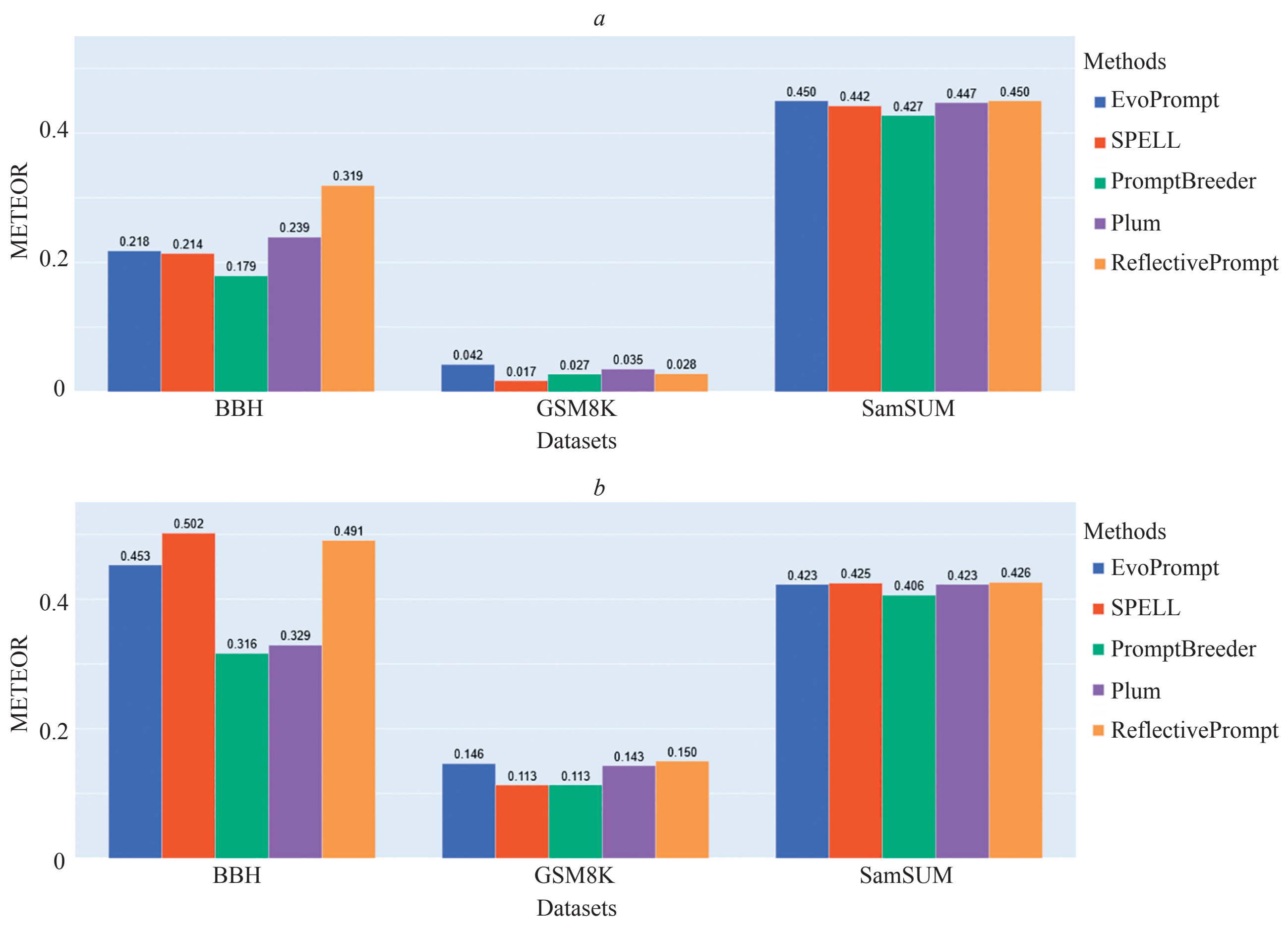

Fig. 3. Histogram of METEOR scores for text generation datasets. Modes T-lite-instruct-0.1 (*a*) and Gemma3-27b-it (*b*)

classification tasks, the average F1-score improved by 6.59 % on the T-lite-instruct-0.1 model and by 0.96 % on the Gemma3-27b-it model. In text generation tasks, the average METEOR score increased by 33.34 % on the T-lite-instruct-0.1 model (comparisons and improvements were calculated relative to the maximum average metrics achieved by existing solutions).

It should be noted that ReflectivePrompt performance significantly depends on the underlying LLM. The effectiveness of Reflective Evolution relies on the quality of generated hints, and weaker language models may produce suggestions that are not fully relevant to the optimization task.

This work creates a scope for future research into Reflective Evolution for autoprompting applications. The current ReflectivePrompt implementation could potentially be further refined for more targeted prompt optimization. Moreover, the concept of Reflective Evolution could be generalized and adapted to other metaheuristic optimization algorithms, representing a promising direction for future studies. For example, there was a recent research where ReflectivePrompt evolution outperforms reinforcement learning on a group of benchmarks [36].

Conclusion

The proposed ReflectivePrompt algorithm, which employs Reflective Evolution for prompt optimization, was evaluated on 33 datasets covering various natural language processing domains. It demonstrated consistent improvements over existing evolutionary algorithm-based autoprompting methods. ReflectivePrompt proves to be a competitive solution, showing that exploring Reflective Evolution for autoprompting can yield significant benefits and advance current methods to new levels of performance.

References

1. Kadavath S., Conerly T., Askell A., Henighan T., Drain D., Perez E., et al. Language models (mostly) know what they know. *arXiv*, 2022, arXiv:2207.05221. https://doi.org/10.48550/arXiv.2207.05221
2. Wei J., Bosma M., Zhao V.Y., Guu K., Yu A.W., Lester B., et al. Finetuned language models are zero-shot learners. *arXiv*, 2021, arXiv:2109.01652. https://doi.org/10.48550/arXiv.2109.01652

Литература

1. Kadavath S., Conerly T., Askell A., Henighan T., Drain D., Perez E., et al. Language models (mostly) know what they know // arXiv. 2022. arXiv:2207.05221. https://doi.org/10.48550/arXiv.2207.05221
2. Wei J., Bosma M., Zhao V.Y., Guu K., Yu A.W., Lester B., et al. Finetuned language models are zero-shot learners // arXiv. 2021. arXiv:2109.01652. https://doi.org/10.48550/arXiv.2109.01652

3. Liu P., Yuan W., Fu J., Jiang Z., Hayashi H., Neubig G. Pre-train, prompt, and predict: a systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 2023, vol. 55, no. 9, pp. 1–35. https://doi.org/10.1145/3560815
4. Brown T.B., Mann B., Ryder N., Subbiah M., Kaplan J., Dhariwal P., et al. Language models are few-shot learners. *arXiv*, 2020, arXiv:2005.14165. https://doi.org/10.48550/arXiv.2005.14165
5. Wang N., Peng Z., Que H., Liu J., Zhou W., Wu Y., et al. RoleLLM: benchmarking, eliciting, and enhancing role-playing abilities of large language models. *Proc. of the Annual Meeting of the Association for Computational Linguistics*, 2024, pp. 14743–14777. https://doi.org/10.18653/v1/2024.findings-acl.878
6. Wei J., Wang X., Schuurmans D., Bosma M., Ichter B., Xia F., et al. Chain-of-thought prompting elicits reasoning in large language models. *arXiv*, 2022, arXiv:2201.11903. https://doi.org/10.48550/arXiv.2201.11903
7. Wang L., Xu W., Lan Y., Hu Z., Lan Y., Lee R.K.-W., Lim E.-P. Plan-and-solve prompting: improving zero-shot chain-of-thought reasoning by large language models. *Proc. of the 61st Annual Meeting of the Association for Computational Linguistics*, 2023, vol. 1, pp. 2609–2634. https://doi.org/10.18653/v1/2023.acl-long.147
8. Leidinger A., van Rooij R., Shutova E. The language of prompting: What linguistic properties make a prompt successful?. *Proc. of the Findings of the Association for Computational Linguistics: EMNLP*, 2023, pp. 9210–9232. https://doi.org/10.18653/v1/2023.findings-emnlp.618
9. Shin T., Razeghi Y., Logan R.L., Wallace E., Singh S. AutoPrompt: Eliciting knowledge from language models with automatically generated prompts. *Proc. of the Conference on Empirical Methods in Natural Language*, 2020, pp. 4222–4235. https://doi.org/10.18653/v1/2020.emnlp-main.346
10. Kwon M., Kim G., Kim J., Lee H., Kim J. StablePrompt: automatic prompt tuning using reinforcement learning for large language models. *arXiv*, 2024, arXiv:2410.07652. https://doi.org/10.48550/arXiv.2410.07652
11. Guo Q., Wang R., Guo J., Li B., Song K., Tan X., et al. EvoPrompt: Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. *arXiv*, 2023, arXiv:2309.08532. https://doi.org/10.48550/arXiv.2309.08532
12. Prasad A., Hase P., Zhou X., Bansal M. GrIPS: gradient-free, edit-based instruction search for prompting large language models. *Proc. of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, 2023, pp. 3845–3864. https://doi.org/10.18653/v1/2023.eacl-main.277
13. Schulhoff S., Ilie M., Balepur N., Kahadze K., Liu A., Si C., et al. The prompt report: a systematic survey of prompt engineering techniques. *arXiv*, 2024, arXiv:2406.06608. https://doi.org/10.48550/arXiv.2406.06608
14. Liu P., Yuan W., Fu J., Jiang Z., Hayashi H., Neubig G. Pre-train, prompt, and predict: a systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 2023, vol. 55, no. 9, pp. 1–35. https://doi.org/10.1145/3560815
15. Li Y.B., Wu K. Spell: semantic prompt evolution based on a LLM. *arXiv*, 2023, arXiv:2310.01260. https://doi.org/10.48550/arXiv.2310.01260
16. Pan R., Xing S., Diao S., Sun W., Liu X., Shum K., et al. Plum: prompt learning using metaheuristic. *arXiv*, 2023, arXiv:2311.08364. https://doi.org/10.48550/arXiv.2311.08364
17. Fernando C., Banarse D., Michalewski H., Osindero S., Rocktäschel T. Promptbreeder: self-referential self-improvement via prompt evolution. *arXiv*, 2023, arXiv:2309.16797. https://doi.org/10.48550/arXiv.2309.16797
18. Eiben A.E., Smith J.E. *Introduction to Evolutionary Computing*. Springer, 2015, 287 p.
19. Ye H., Wang J., Cao Z., Berto F., Hua C., Kim H., et al. ReEvo: large language models as hyper-heuristics with reflective evolution. *arXiv*, 2024, arXiv:2402.01145. https://doi.org/10.48550/arXiv.2402.01145
20. Holland J.H. Genetic algorithms. *Scientific American*, 1992, vol. 267, no. 1, pp. 66–72. https://doi.org/10.1038/scientificamerican0792-66
21. Lipowski A., Lipowska D. Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and its Applications*, 2012, vol. 391, no. 6, pp. 2193–2196. https://doi.org/10.1016/j.physa.2011.12.004
22. Storn R., Price K. Differential evolution — a simple and efficient heuristic for global optimization over continuous spaces. *Journal of*

3. Liu P., Yuan W., Fu J., Jiang Z., Hayashi H., Neubig G. Pre-train, prompt, and predict: a systematic survey of prompting methods in natural language processing // ACM Computing Surveys. 2023. V. 55. N 9. P. 1–35. https://doi.org/10.1145/3560815
4. Brown T.B., Mann B., Ryder N., Subbiah M., Kaplan J., Dhariwal P., et al. Language models are few-shot learners // arXiv. 2020. arXiv:2005.14165. https://doi.org/10.48550/arXiv.2005.14165
5. Wang N., Peng Z., Que H., Liu J., Zhou W., Wu Y., et al. RoleLLM: benchmarking, eliciting, and enhancing role-playing abilities of large language models // Proc. of the Annual Meeting of the Association for Computational Linguistics. 2024. P. 14743–14777. https://doi.org/10.18653/v1/2024.findings-acl.878
6. Wei J., Wang X., Schuurmans D., Bosma M., Ichter B., Xia F., et al. Chain-of-thought prompting elicits reasoning in large language models // arXiv. 2022. arXiv:2201.11903. https://doi.org/10.48550/arXiv.2201.11903
7. Wang L., Xu W., Lan Y., Hu Z., Lan Y., Lee R.K.-W., Lim E.-P. Plan-and-solve prompting: improving zero-shot chain-of-thought reasoning by large language models // Proc. of the 61st Annual Meeting of the Association for Computational Linguistics. 2023. V. 1. P. 2609–2634. https://doi.org/10.18653/v1/2023.acl-long.147
8. Leidinger A., van Rooij R., Shutova E. The language of prompting: What linguistic properties make a prompt successful? // Proc. of the Findings of the Association for Computational Linguistics: EMNLP. 2023. P. 9210–9232. https://doi.org/10.18653/v1/2023.findings-emnlp.618
9. Shin T., Razeghi Y., Logan R.L., Wallace E., Singh S. AutoPrompt: Eliciting knowledge from language models with automatically generated prompts // Proc. of the Conference on Empirical Methods in Natural Language. 2020. P. 4222–4235. https://doi.org/10.18653/v1/2020.emnlp-main.346
10. Kwon M., Kim G., Kim J., Lee H., Kim J. StablePrompt: automatic prompt tuning using reinforcement learning for large language models // arXiv. 2024. arXiv:2410.07652. https://doi.org/10.48550/arXiv.2410.07652
11. Guo Q., Wang R., Guo J., Li B., Song K., Tan X., et al. EvoPrompt: Connecting large language models with evolutionary algorithms yields powerful prompt optimizers // arXiv. 2023. arXiv:2309.08532. https://doi.org/10.48550/arXiv.2309.08532
12. Prasad A., Hase P., Zhou X., Bansal M. GrIPS: gradient-free, edit-based instruction search for prompting large language models // Proc. of the 17th Conference of the European Chapter of the Association for Computational Linguistics. 2023. P. 3845–3864. https://doi.org/10.18653/v1/2023.eacl-main.277
13. Schulhoff S., Ilie M., Balepur N., Kahadze K., Liu A., Si C., et al. The prompt report: a systematic survey of prompt engineering techniques // arXiv. 2024. arXiv:2406.06608. https://doi.org/10.48550/arXiv.2406.06608
14. Liu P., Yuan W., Fu J., Jiang Z., Hayashi H., Neubig G. Pre-train, prompt, and predict: a systematic survey of prompting methods in natural language processing // ACM Computing Surveys. 2023. V. 55. N 9. P. 1–35. https://doi.org/10.1145/3560815
15. Li Y.B., Wu K. Spell: semantic prompt evolution based on a LLM // arXiv. 2023. arXiv:2310.01260. https://doi.org/10.48550/arXiv.2310.01260
16. Pan R., Xing S., Diao S., Sun W., Liu X., Shum K., et al. Plum: prompt learning using metaheuristic // arXiv. 2023. arXiv:2311.08364. https://doi.org/10.48550/arXiv.2311.08364
17. Fernando C., Banarse D., Michalewski H., Osindero S., Rocktäschel T. Promptbreeder: self-referential self-improvement via prompt evolution // arXiv. 2023. arXiv:2309.16797. https://doi.org/10.48550/arXiv.2309.16797
18. Eiben A.E., Smith J.E. Introduction to Evolutionary Computing. Springer, 2015. 287 p.
19. Ye H., Wang J., Cao Z., Berto F., Hua C., Kim H., et al. ReEvo: large language models as hyper-heuristics with reflective evolution // arXiv. 2024. arXiv:2402.01145. https://doi.org/10.48550/arXiv.2402.01145
20. Holland J.H. Genetic algorithms // Scientific American. 1992. V. 267. N 1. P. 66–72. https://doi.org/10.1038/scientificamerican0792-66
21. Lipowski A., Lipowska D. Roulette-wheel selection via stochastic acceptance // Physica A: Statistical Mechanics and its Applications. 2012. V. 391. N 6. P. 2193–2196. https://doi.org/10.1016/j.physa.2011.12.004
22. Storn R., Price K. Differential evolution — a simple and efficient heuristic for global optimization over continuous spaces // Journal of

Global Optimization, 1997, vol. 11, no. 4, pp. 341–359. https://doi.org/10.1023/a:1008202821328

23. Russell S., Norvig P. *Artificial Intelligence: a Modern Approach*. Pearson, 2009, 1152 p.
24. Kirkpatrick S., Gelatt C.D., Vecchi M.P. Optimization by simulated annealing. *Science*, 1983, vol. 220, no. 4598, pp. 671–680. https://doi.org/10.1126/science.220.4598.671
25. Glover F. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 1986, vol. 13, no. 5, pp. 533–549. https://doi.org/10.1016/0305-0548(86)90048-1
26. Geem Z.W., Kim J.H., Loganathan G.V. A new heuristic optimization algorithm: harmony search. *Simulation*, 2001, vol. 76, no. 2, pp. 60–68. https://doi.org/10.1177/003754970107600201
27. Larranaga P. A review on estimation of distribution algorithms. *Genetic Algorithms and Evolutionary Computation*, 2002, vol. 2, pp. 57–100. https://doi.org/10.1007/978-1-4615-1539-5_3
28. Ross B.J. A Lamarckian evolution strategy for genetic algorithms. *Practical Handbook of Genetic Algorithms*, 2019, pp. 1–16. https://doi.org/10.1201/9780429128356-1
29. Voudouris C., Tsang E.P., Alsheddy A. Guided local search. *International Series in Operations Research & Management Science*, 2010, vol. 146, pp. 321–361. https://doi.org/10.1007/978-1-4419-1665-5_11
30. Dorigo M., Maniezzo V., Colorni A. Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 1996, vol. 26, no. 1, pp. 29–41. https://doi.org/10.1109/3477.484436
31. Shibasaka K., Kanazawa K., Yasunaga M. Decoupling-capacitor allocation problem solved by genetic algorithm. *Proc. of the IEEE Electrical Design of Advanced Packaging Systems Symposium (EDAPS)*, 2013, pp. 225–228. https://doi.org/10.1109/edaps.2013.6724430
32. Kim H., Kim M., Berto F., Kim J., Park J. DevFormer: a symmetric transformer for context-aware device placement. *arXiv*, 2022, arXiv:2205.13225. https://doi.org/10.48550/arXiv.2205.13225
33. Gohil A., Tayal M., Sahu T., Sawalpurkar V. Travelling salesman problem: parallel implementations & analysis. *arXiv*, 2022, arXiv:2205.14352. https://doi.org/10.48550/arXiv.2205.14352
34. Liu M.X., Liu F., Fiannaca A.J., Koo T., Dixon L., Terry M., Cai C.J. “We Need Structured Output”: towards user-centered constraints on large language model output. *Proc. of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, 2024, pp. 1–9. https://doi.org/10.1145/3613905.3650756
35. Kamath A., Ferret J., Pathak S., Vieillard N., Merhej R., Perrin S., et al. Gemma 3 technical report. *arXiv*, 2025, arXiv:2503.19786. https://doi.org/10.48550/arXiv.2503.19786
36. Agrawal L.A., Tan S., Soylu D., Ziems N., Khare R., Opsahl-Ong K., et al. GEPA: reflective prompt evolution can outperform reinforcement learning. *arXiv*, 2025, arXiv:2507.19457. https://doi.org/10.48550/arXiv.2507.19457

Global Optimization. 1997. V. 11. N 4. P. 341–359. https://doi.org/10.1023/a:1008202821328

23. Russell S., Norvig P. Artificial Intelligence: a Modern Approach. Pearson, 2009. 1152 p.
24. Kirkpatrick S., Gelatt C.D., Vecchi M.P. Optimization by simulated annealing // Science. 1983. V. 220. N 4598. P. 671–680. https://doi.org/10.1126/science.220.4598.671
25. Glover F. Future paths for integer programming and links to artificial intelligence // Computers and Operations Research. 1986. V. 13. N 5. P. 533–549. https://doi.org/10.1016/0305-0548(86)90048-1
26. Geem Z.W., Kim J.H., Loganathan G.V. A new heuristic optimization algorithm: harmony search // Simulation. 2001. V. 76. N 2. P. 60–68. https://doi.org/10.1177/003754970107600201
27. Larranaga P. A review on estimation of distribution algorithms // Genetic Algorithms and Evolutionary Computation. 2002. V. 2. P. 57–100. https://doi.org/10.1007/978-1-4615-1539-5_3
28. Ross B.J. A Lamarckian evolution strategy for genetic algorithms // Practical Handbook of Genetic Algorithms. 2019. P. 1–16. https://doi.org/10.1201/9780429128356-1
29. Voudouris C., Tsang E.P., Alsheddy A. Guided local search // International Series in Operations Research & Management Science. 2010. V. 146. P. 321–361. https://doi.org/10.1007/978-1-4419-1665-5_11
30. Dorigo M., Maniezzo V., Colorni A. Ant system: Optimization by a colony of cooperating agents // IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics). 1996. V. 26. N 1. P. 29–41. https://doi.org/10.1109/3477.484436
31. Shibasaka K., Kanazawa K., Yasunaga M. Decoupling-capacitor allocation problem solved by genetic algorithm // Proc. of the IEEE Electrical Design of Advanced Packaging Systems Symposium (EDAPS). 2013. P. 225–228. https://doi.org/10.1109/edaps.2013.6724430
32. Kim H., Kim M., Berto F., Kim J., Park J. DevFormer: a symmetric transformer for context-aware device placement // arXiv. 2022. arXiv:2205.13225. https://doi.org/10.48550/arXiv.2205.13225
33. Gohil A., Tayal M., Sahu T., Sawalpurkar V. Travelling salesman problem: parallel implementations & analysis // arXiv. 2022. arXiv:2205.14352. https://doi.org/10.48550/arXiv.2205.14352
34. Liu M.X., Liu F., Fiannaca A.J., Koo T., Dixon L., Terry M., Cai C.J. “We Need Structured Output”: towards user-centered constraints on large language model output // Proc. of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems. 2024. P. 1–9. https://doi.org/10.1145/3613905.3650756
35. Kamath A., Ferret J., Pathak S., Vieillard N., Merhej R., Perrin S., et al. Gemma 3 technical report // arXiv. 2025. arXiv:2503.19786. https://doi.org/10.48550/arXiv.2503.19786
36. Agrawal L.A., Tan S., Soylu D., Ziems N., Khare R., Opsahl-Ong K., et al. GEPA: reflective prompt evolution can outperform reinforcement learning // arXiv. 2025. arXiv:2507.19457. https://doi.org/10.48550/arXiv.2507.19457

Authors

Viktor N. Zhuravlev — Student, ITMO University, Saint Petersburg, 197101, Russian Federation, https://orcid.org/0009-0009-0788-9790, viktor.zhuravlev2003@mail.ru

Artur R. Khairullin — Student, ITMO University, Saint Petersburg, 197101, Russian Federation, https://orcid.org/0009-0007-0442-7764, arkhairullin@itmo.ru

Ernest A. Dyagin — Student, ITMO University, Saint Petersburg, 197101, Russian Federation, https://orcid.org/0009-0000-3865-4446, tsenred@yandex.com

Alena N. Sitkina – Student, ITMO University, Saint Petersburg, 197101, Russian Federation, https://orcid.org/0009-0002-6046-8943, sitkina.alena2017@yandex.ru

Nikita I. Kulin — PhD Student, ITMO University, Saint Petersburg, 197101, Russian Federation, sc 57222386134, https://orcid.org/0000-0002-3952-6080, kylin98@list.ru

Авторы

Журавлев Виктор Николаевич — студент, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, https://orcid.org/0009-0009-0788-9790, viktor.zhuravlev2003@mail.ru

Хайруллин Артур Рустамович — студент, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, https://orcid.org/0009-0007-0442-7764, arkhairullin@itmo.ru

Дягин Эрнест Александрович — студент, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, https://orcid.org/0009-0000-3865-4446, tsenred@yandex.com

Ситкина Алена Николаевна — студент, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, https://orcid.org/0009-0002-6046-8943, sitkina.alena2017@yandex.ru

Кулин Никита Игоревич — аспирант, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, sc 57222386134, https://orcid.org/0000-0002-3952-6080, kylin98@list.ru

Received 11.06.2025
Approved after reviewing 01.09.2025
Accepted 21.11.2025

Статья поступила в редакцию 11.06.2025
Одобрена после рецензирования 01.09.2025
Принята к печати 21.11.2025