A Survey on Automatic Generation of Metaheuristic Algorithms Using Large Language Models

Ibrahim Hayatu Hassan
Department of Computer Science
Ahmadu Bello University, Zaria
ihhassan@abu.edu.ng

Fatsuma Jauro
Department of Computer Science
Ahmadu Bello University, Zaria
fjauro@abu.edu.ng

Sada Wasilah
Department of Computer Science
Ahmadu Bello University, Zaria
wmsada@abu.edu.ng

Amina Abubakar Hassan
Department of Computer Science
Ahmadu Bello University, Zaria
ahabubakar@abu.edu.ng

Aliyu Muhammad Kufena
Department of Computer Science
Ahmadu Bello University, Zaria
kufenah@gmail.com

Safinat O. Zakari
Department of Computer Science
Ahmadu Bello University, Zaria
safinatu@abu.edu.ng

Aminu Ominisi Abdulsalami
Department of Computer Science
Ahmadu Bello University, Zaria
lecturer34@gmail.com

Mohammed Abdullahi
Department of Computer Science
Ahmadu Bello University, Zaria
abdullahilwafu@abu.edu.ng

***Abstract*— Designing metaheuristic algorithms such as genetic algorithms, simulated annealing, or particle swarm optimization has traditionally required deep expertise and manual trial-and-error. But with the rise of large language models (LLMs) like GPT-4 and beyond, there is a growing interest in using these models to automatically generate metaheuristic algorithms. This survey explores how LLMs are being applied to assist or even automate the creation of metaheuristics, from generating code and suggesting algorithmic components to fine-tuning strategies based on problem-specific needs. The study highlights state-of-the-art techniques, applications, key challenges, and promising research directions at this new intersection of LLMs and optimization. This survey aims to provide a clear and grounded overview for researchers interested in how LLMs can support the next generation of algorithm design.**

Keywords— Metaheuristics, Large Language Models (LLM), Algorithm Generation, Optimization, Prompt Engineering, GPT-4, AutoML, Generative AI

I. INTRODUCTION

To solve complex optimization problems where traditional exact methods are computationally impractical, metaheuristic algorithms like genetic algorithms (GA), particle swarm optimization (PSO), simulated annealing (SA) etc. are frequently used [1] [2]. These algorithms offer near-optimal solutions to NP-hard problems, which makes them especially useful in fields like engineering, logistics, and artificial intelligence [3]. Effective metaheuristic design, however, necessitates a high level of skill and involves choosing the right operators, parameters, and problem-specific modifications, all of which can be laborious and subject to human error. New developments in large language models (LLMs), like Codex, LLaMA, and GPT-4, have created revolutionary possibilities for algorithm design automation [4]. Because LLMs can comprehend descriptions in natural language and produce executable code, they allow researchers to automate the development of meta-heuristic algorithms that are suited to particular issues. This survey examines how LLMs and metaheuristic algorithm design interact, offering a thorough examination of approaches, uses, difficulties, and potential future developments. The goals are to: (1) examine the ways in which LLMs are used to produce metaheuristics; (2) review various literatures on the use of LLM for new metaheuristic generation, improving existing algorithms or using LLM for metaheuristics hyperparameter optimization; (3) highlight useful applications in a range of fields; and (4) pinpoint important issues and areas in need of additional study.

The structure of the paper is as follows: Background information on metaheuristics and LLMs is given in Section II; automatic algorithm generation techniques are covered in Section III; various literatures on the use of LLM for new metaheuristic generation, improving existing algorithms or using LLM for metaheuristics hyperparameter optimization are presented in Section IV; the real-world applications are examined in Section V; prospect of metaheuristic research community under LLMs are covered in Section VI; and future research directions are presented in Section VII.

II. BACKGROUND

A. Metaheuristic Algorithms

Metaheuristics are general techniques designed to produce near-optimal solutions for complex optimization problems[1]. Compared to exact methods, metaheuristics are more suitable for large-scale, non-linear, or multi-modal problems, as they trade optimality for computational efficiency [5]. Some key metaheuristic algorithms are:

- Genetic Algorithms (GA): Drawn from the principles of natural evolution, GAs evolve a population of solutions toward optimality through mechanisms such as crossover, mutation, and selection [6]. They work especially well for combinatorial optimization issues.
- Particle Swarm Optimization (PSO): PSO updates particle positions in a search space guided by local and global best solutions to optimize solutions, drawing inspiration from social behavior in animal groups [7]. PSO is frequently utilized for binary and continuous optimization tasks.
- Simulated Annealing (SA): Based on thermodynamic processes, SA avoids local optima by exploring the solution space by accepting worse solutions with a decreasing probability [8].
- Ant Colony Optimization (ACO): This method, which draws inspiration from ants' foraging habits, employs pheromone-based communication to direct the search for the best routes. [9].

The process of designing these algorithms entails choosing the right operators (such as crossover types in GA) and tuning parameters (such as mutation rates), which frequently calls for a great deal of trial and error and domain expertise.

B. *Large Language Models*

LLMs are deep learning models that have been trained on large text databases, which allows them to do tasks like code synthesis, text generation, and natural language understanding [4]. The ability of models such as GPT-4, LLaMA, and Codex to produce executable code and text that resembles that of a human makes them perfect for automating algorithm design [10]. Their power comes from transformer architectures, which use attention mechanisms to process and produce outputs that are pertinent to the context [11]. Using patterns learned from training data, LLMs can interpret problem descriptions, produce algorithm implementations, and even recommend parameter configurations in the context of metaheuristics.

III. METHODOLOGIES FOR AUTOMATIC GENERATION

The automatic generation of metaheuristic algorithms using LLMs includes several methods. Each uses the models' abilities to simplify algorithm design.

A. *Prompts Engineering*

Prompt engineering is an important method for guiding LLMs to create metaheuristic algorithms. By creating clear input prompts, researchers can define problem areas, limits, and desired features of the algorithm [12]. For instance, a prompt for generating a GA might say, “Design a genetic algorithm for the traveling salesman problem with tournament selection and single-point crossover.” Key approaches include:

- Problem-Specific Prompts: These describe the optimization problem. They include objectives, limitations, and requirement for evaluation. For example, a prompt for a scheduling problem might detail task dependencies and deadlines [13].
- Template-Based Prompts: Standardized templates help preserve consistent algorithm structures. This includes defining initialization, selection, and termination conditions [14].
- Iterative Refinement: To enhance output quality, researchers iteratively improve prompts by incorporating feedback and evaluating the performance of generated algorithms [15].

Prompt engineering is a popular method for initial algorithm generation because it is easy to use and requires little computational power.

B. *Fine-Tuning LLMs*

Fine-tuning adjusts pre-trained LLMs for specific tasks by training them on datasets that include algorithm implementations, problem descriptions, and performance metrics [16]. For instance, fine-tuning a model on a dataset of PSO implementations for benchmark problems (e.g., CEC 2017) can improve its ability to create effective PSO variants. Fine-tuning also helps the model grasp domain-specific details better, but it needs a lot of computational power and carefully selected datasets.

C. *Hybrid Approach*

Hybrid approaches combine LLMs and traditional algorithm design techniques to improve performance. For example, an LLM might create an initial algorithm framework, which is then refined through hyperparameter optimization or evolutionary strategies [17]. Another method combines LLMs with reinforcement learning, where the model learns to enhance algorithm designs based on performance feedback [18]. These approaches take advantage of the strengths of both LLMs and conventional optimization techniques, leading to solid results.

D. *Code Generation and Validation*

For metaheuristic algorithms, LLMs can produce executable code in Python, C++, or Java. To guarantee accuracy and effectiveness, generated algorithms are verified using benchmark problems, such as the CEC or DTLZ test suites [19]. Validation can entail assessing the performance of generated algorithms on real-world datasets or comparing them to well-known implementations. Frameworks for automated testing can make this procedure even more efficient.

IV. LLM-ASSISTED METAHEURISTIC ALGORITHM

The rise of LLM has quickly sparked interest in interdisciplinary research, making "AI for science" a key topic in current discussions [20] [21]. However, the use of LLMs in the metaheuristics community is still quite limited. A few notable examples exist. GPT-4 was used in [22] to create a new hybrid optimization approach based on particle swarm optimization (PSO), cuckoo search (CS), artificial bee colony (ABC), grey wolf optimizer (GWO), self-organizing migrating algorithm (SOMA), and whale optimization algorithm (WOA). This research mainly looked into the interactive process and challenges of using GPT-4 to accomplish a series of tasks through careful prompt engineering.

An effective learning-based search operator was found in [23] using the LLM. They introduced a new type of

decomposition-based multi-objective evolutionary algorithm called MOEA/D-LO, which competes well with expert-designed MOEAs. In another study, Liu *et al*. [24] proposed a method for algorithm evolution using LLM, referred to as AEL. Unlike earlier methods, AEL is seen as a model rather than a specific problem approach. It does not need model training and focuses on evolving algorithms. AEL shows promising results in discrete optimization, particularly in the traveling salesman problem (TSP). Liu *et al.* [25] also introduced an LLM-driven evolutionary algorithm, known as LMEA, for solving TSPs. In each round, LMEA directs the LLM to select parent solutions from the population. It then uses search operators to create offspring solutions. These offspring solutions are evaluated and chosen for the next generation. Importantly, LMEA has shown competitive performance on TSP instances with up to 20 cities.

A research conducted by the authors in [26] found that this issue is a combinatorial optimization problem and proposed an LLM-assisted optimizer, or LLMO. They encode the specific neural architecture as a solution and refine it through interaction with Gemini. This work shows the benefits of combining the LLM with EC techniques. In a separate study reported *in* [27] used ChatGPT-3.5 to create a new metaheuristic algorithm called Zoological Search Optimization, or ZSO, based on the standard prompt engineering framework CRISPE, which stands for Capacity and Role, Insight, Statement, Personality, and Experiment. This new animal-inspired algorithm takes cues from the group behaviors of animals to tackle continuous optimization problems. The algorithm includes two search operators: the prey-predator interaction operator and the social flocking operator, which help balance exploration and exploitation. Additionally, there are four variations of the ZSO algorithm with minor adjustments from human interaction. The experimental results and statistical analysis show that ZSO-derived algorithms are efficient and effective compared to over twenty popular algorithms on CEC2014, CEC2022 benchmark functions, and six engineering optimization problems.

A new LLM evolutionary algorithm (LLaMEA) framework was introduced in [28], it uses GPT models for generating and refining algorithms automatically. Given certain criteria and a task definition (the search space), LLaMEA creates, changes, and chooses algorithms based on performance metrics and feedback from runtime evaluations. This framework provides a fresh way to generate optimized algorithms without needing extensive prior knowledge. In Martinek et al. [29], an LLM was used for tuning metaheuristics by selecting the right parameters. The LLM helped pick the best parameters for popular metaheuristic algorithms like GA, ACO, PSO, and SA to solve the Traveling Salesman and Graph Coloring problems. Feedback was gathered by giving LLMs prompts that included initial parameters, average performance, and population variance when relevant. The results indicate that LLMs can understand the complex task of tuning metaheuristic parameters.

Researchers in [30] integrate LLMs into PSO to reduce model evaluations and improve convergence for deep learning hyperparameter tuning. The proposed LLM-enhanced PSO method tackles efficiency and convergence issues by using LLMs, particularly ChatGPT-3.5 and Llama3, to boost PSO performance. This allows for faster achievement of target objectives. The method speeds up search space exploration by replacing underperforming particle placements with the best suggestions from LLMs. Experiments across three scenarios (1) optimizing the Rastrigin function, (2) using Long Short-Term Memory (LSTM) networks for time series regression, and (3) using Convolutional Neural Networks (CNNs) for material classification—show that the method improves convergence rates and lowers computational costs significantly. A new algorithm called the hybrid CLM, inspired by the crayfish optimization algorithm (COA), LSHADE, and the marine predator algorithm (MPA) was presented in [31]. The algorithm was designed using GPT-4 through the evolution of heuristics prompting technique. The hybrid CLM switches adaptively between COA-inspired exploration behavior, LSHADE-driven current-to-pbest mutation and crossover, and MPA-style elite memory to guide population convergence through rank-aware recombination. Hybrid CLM is tested against COA, LSHADE, MPA, the whale optimization algorithm (WOA), and the sailfish optimization algorithm (SFO) across 64 functions (CEC2017, CEC2022, and mathematical benchmarks). The results show that LLM-generated hybrid CLM competes well with top human-designed optimizers on challenging landscapes while reducing execution time by up to 40%. The summary of these findings is shown in TABLE I and Figure 1 shows the taxonomy of large language models in metaheuristic algorithm generation and emerging direction based on the findings.

V. APPLICATION

Applications of LLMs for the automatic creation of metaheuristic algorithms in a variety of fields have shown their adaptability and significance:

- Logistics and Transportation: LLMs have produced algorithms for vehicle routing that optimize delivery schedules and reduce fuel expenses. For instance, a logistics company's delivery efficiency increased, thanks to an LLM-generated ACO algorithm.
- Machine Learning: To improve model accuracy and decrease training times, automated metaheuristic design has been applied to hyperparameter optimization in neural networks.
- Engineering Design: LLMs have developed structural optimization algorithms that, for example, minimize material consumption in bridge design while preserving structural integrity.
- Bioinformatics: To speed up drug discovery process, customized metaheuristics for protein folding and sequence alignment have been developed.
- Finance: Under intricate constraints, LLMs have developed algorithms for portfolio optimization that balance risk and return

These uses demonstrate how LLMs can modify metaheuristics to fit particular problem domains, cutting down on development time and improving output.

Thus, using LLM to design novel metaphor-based MAs is hopeful. Moreover, during the LLM era, what benefits can LLM bring to the metaheuristic community, and what advantages can metaheuristics can contribute.

TABLE I: SUMMARY OF EXISTING LITERATURE

Source	LLM	Method	Generated Algorithm	Benchmark Problems	Key Findings
[22]	GPT-4	prompt-driven experimental design process to synthesize and code new optimization algorithms	generated two new hybrid optimization algorithms—**ESEEO** (Enhanced Swarm Exploration and Exploitation Optimizer) and **LESO** (Limited Evaluation Swarm Optimizer) based on: PSO, CS, ABC, GWO, SOMA, and WOA.	None was used. The evaluation was purely qualitative	GPT-4 demonstrates strong potential for algorithm design by accurately identifying existing algorithms, generating coherent hybrid designs, and producing consistent pseudocode and executable code.
[23]	GPT-3.5	prompt engineering.	MOEA/D-LLM: Decomposition-based Multiobjective Evolutionary Algorithm with an LLM as the operator.	The algorithms were Tested on standard multi-objective benchmarks: RE21–RE25, ZDT1–ZDT6, and UF1–UF9.	MOEA/D-LLM and MOEA/D-LO outperformed traditional MOEAs, showing pre-trained LLMs can serve as adaptable, automated components for evolutionary optimization
[24]	GPT-3.5-turbo and GPT-4	Algorithm Evolution using Large Language Model (AEL)	Heuristic algorithms for the Traveling Salesman Problem (TSP).	Evaluation was performed on various TSP sizes against greedy, direct LLM-generated algorithms, and the LKH3 solver baseline.	AEL outperformed human and model-based heuristics, achieving lower optimality gaps, better scalability, and reduced need for domain expertise or training.
[25]	GPT-3.5	Uses natural language prompts to guide the LLM in generating solutions	LLM-driven Evolutionary Algorithm (LMEA)	The study evaluates LMEA on the Traveling Salesman Problem (TSP)	LMEA achieved competitive results against traditional heuristics and OPRO
[26]	Gemini	A novel prompt-driven optimization	LLM-Assisted Optimizer (LLMO)	The study uses ARNAS tasks based on the NAS-Bench-201 search space with CIFAR-10 and CIFAR-100 datasets.	LLMO demonstrates competitive performance compared to six well-known metaheuristic algorithms (GA, PSO, DE, CMA-ES, JADE, SHADE) in ARNAS tasks.
[27]	GPT-3.5	The CRISPE framework (Capacity and Role, Insight, Statement, Personality, and Experiment) was employed for prompt engineering to guide ChatGPT-3.5 in generating the algorithm	Zoological Search Optimization (ZSO) and its variants were generated	Six Engineering Optimization Problems (e.g., Cantilever Beam Design, Gear Train Design), CEC2014, and CEC2022 benchmarks	ZSO and its variants demonstrated competitive performance against 20 state-of-the-art metaheuristic algorithms
[28]	GPT-3.5, GPT-4, GPT-4o	A feedback via IOHexperimenter benchmarking.	LLaMEA (LLM-based Evolutionary Algorithm)	5-D, 10-D, 20-D BBOB (Black-Box Optimization Benchmark) problems using IOHprofiler (24 functions).	Generated algorithms outperformed CMA-ES and DE on 5-D BBOB benchmarks; competitive in higher dimensions; first framework to evolve full metaheuristics autonomously using LLMs.
[29]	GPT-3.5, GPT-4, Gemini, Mistral (Le Chat)	Prompt-driven parameter tuning: LLMs recommend and refine metaheuristic parameters through feedback.	LLM-based Parameter Tuning Framework	Travelling Salesman Problem (TSP) (15-city P01, TSPLIB) and Graph Coloring Problem (GCP) (myciel3 graph).	LLM-tuned parameters outperformed default and initial setups; GPT-4 and Mistral showed best performance; feedback runs yielded improved optimization results. Demonstrated LLMs can understand and enhance parameter tuning of MHAs.
[30]	ChatGPT-3.5, Llama-3	LLM-driven PSO	LLM-Enhanced PSO (LLM-PSO) – Hybrid framework combining PSO	Rastrigin Function, LSTM, CNN.	LLM-PSO reduced computational complexity by 20–60% while maintaining accuracy; ChatGPT-3.5 achieved 60% reduction in model calls; Llama-3 achieved 20–40% improvement in regression tasks; validated efficiency for deep learning hyperparameter tuning.
[31]	hybrid CLM,	LLM EoH	hybrid CLM	CEC2017, CEC2022, Friedman rank and Wilcoxon rank	hybrid CLM algorithm delivers both speed and accuracy

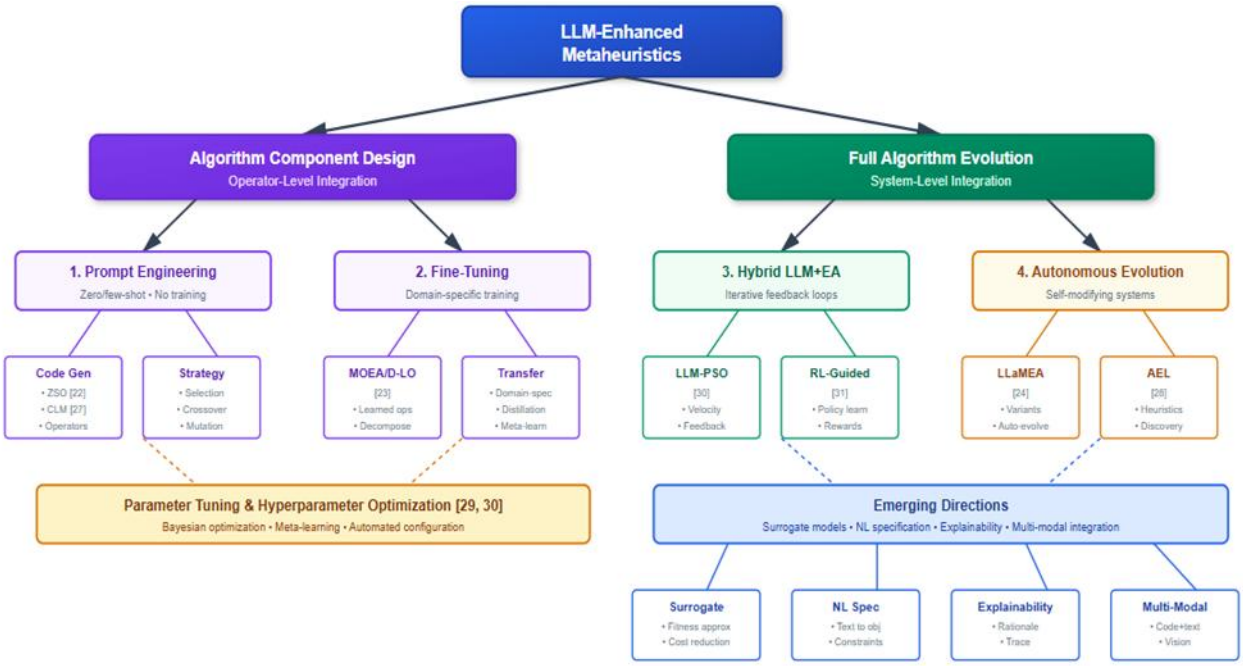

Fig. 1 Taxonomy of LLM Applications in Metaheuristic Algorithm Design and Emerging Direction

VI. Prospects of the Metaheuristics Community under LLM Era

Integrating LLMs into MAs offers great promise and can result in several beneficial outcomes for both sides. Let's look at the potential benefits of bringing LLMs into the MA community and the other way around.

A. Benefits of Introducing LLMs to Metaheuristics

Integrating LLMs into the realm of MA holds significant promise and can lead to several mutually beneficial outcomes. Let's explore the potential benefits and advantages of introducing LLMs to the MA community and vice versa:

- Improved Problem Representation: LLMs can improve how we represent problem instances. They provide a better and more context-aware understanding of the optimization landscape. The natural language processing (NLP) features in LLMs allow for a more expressive and human-like description of problems. This helps in defining and interpreting problems effectively.
- Knowledge Integration: Pre-trained LLMs can incorporate large amounts of domain-specific knowledge, giving metaheuristics a clear understanding of the problem space. Fine-tuning LLMs on specific problem instances can help in finding relevant features and patterns, which supports metaheuristics in making decisions.
- Dynamic Adaptation: LLMs can help with adaptive metaheuristics by changing their parameters based on real-time information and changing problem conditions. The continuous learning abilities of LLMs can improve adaptability, allowing metaheuristics to respond well to shifts in the optimization landscape.
- Multi-Objective Optimization: LLMs can help manage multi-objective optimization problems by understanding and showing the trade-offs between conflicting objectives. The natural language understanding in LLMs can assist in creating and explaining complex objective functions in a way that is easier for people to read.

B. Advantage of Metaheuristics for LLMs

- Training Data Augmentation: Metaheuristics (MAs) can be used to create varied and difficult training examples for LLMs, which improves their strength and ability to generalize. Exploration-exploitation strategies in metaheuristics can help with better data sampling during training.
- Hyperparameter Tuning: MAs can be used to improve the hyperparameters of large language models. This boosts their performance on specific tasks. Evolutionary algorithms and swarm intelligence can effectively search the hyperparameter space for the best configurations.
- Transfer Learning and Adaptation: Metaheuristics can help with the transfer learning process for LLMs by assisting in adjusting pre-trained models to new tasks or areas. Adaptive algorithms can optimize the finetuning process. This leads to faster convergence and better performance for specific tasks.
- Resource Distribution and Scaling: Metaheuristics can improve how computational resources are allocated for training large language models. This

helps make the process more efficient and cuts down on training time. Scalability algorithms ensure that LLMs can effectively manage larger model sizes and datasets.

Introducing LLMs to the metaheuristic community is exciting; however, it may also raise some concerns.

C. Drawback of using LLM to Produce New Metaheuristics

While LLMs have notable abilities, there are numerous downsides to using them for producing novel metaheuristics algorithms:

- Lack of Domain Expertise: LLMs might not have the deep knowledge needed to create effective and innovative metaheuristics. They produce content based on patterns in the training data, and this data may not include the latest or highly specialized information.
- Quality and Novelty of Concepts: The designs produced by LLMs may not be unique or innovative. They often blend existing concepts instead of forming entirely novel ones. This can restrict the uniqueness and effectiveness of the generated algorithms.
- Intricacy and Implementation Problems: The algorithms suggested by LLMs might be too complicated or hard to implement successfully. They may recommend steps that make sense on paper but are not practical in real-world situations.
- Scalability and Efficacy Problems: The suggested algorithms might not be scalable or efficient for largescale problems. LLMs might overlook practical constraints such as computational resources and runtime efficiency.
- Lack of Innovation in Operators and Mechanisms: The evolutionary operators or mechanisms (e.g., selection, crossover, mutation) proposed by LLMs might not introduce significant improvements over existing methods. They might reuse well-known operators without substantial modifications.
- Reliance on Training Data: LLMs are limited by the data they were trained on. If the training data does not include recent advances or specific areas of metaheuristic research, the generated algorithms might be outdated or incomplete.
- Ethical and Bias Concerns: The generated algorithms could inadvertently incorporate biases present in the training data. This can lead to biased or unfair optimization outcomes, which is particularly problematic in sensitive applications.

On the one hand, the collaboration between LLMs and metaheuristics holds immense potential for advancing the capabilities of optimization techniques and creating more intelligent, adaptive, and user-friendly solutions across a wide range of applications. On the other hand, the limitations of introducing LLMs to the MA community must be carefully considered when developing new metaheuristic algorithms. Combining their generative capabilities with domain expertise and rigorous evaluation is essential for producing effective and practical optimization solutions. In summary, this interdisciplinary synergy can pave the way for groundbreaking developments in artificial intelligence and optimization.

VII. Conclusion and Future Direction

The use of large language models in the automatic creation of metaheuristic algorithms marks a significant change in optimization research. By taking advantage of their ability to understand language and generate code, LLMs allow for quick, tailored algorithm design. This reduces the need for manual expertise. Their applications in logistics, machine learning, engineering, and bioinformatics show their potential to transform these fields. However, challenges like computational costs, generalization, and ethical issues need to be tackled to fully achieve their benefits.

Future research directions include:

- Lightweight LLMs: Creating smaller, efficient LLMs for environments with limited resources to make access more widespread.
- Improving generalization by using transfer learning and multi-domain training.
- Establishing standard measures for evaluating generated algorithms.
- Exploring ethical guidelines to ensure responsible use of automated algorithm design.

This survey highlights the transformative ability of LLMs in metaheuristic algorithm design and calls for ongoing exploration in this interdisciplinary field.

References

[1] K. Sörensen, " Metaheuristics—the metaphor exposed," *International Transactions in Operational Research,* vol. 22, no. 1, pp. 3-18, 2015.

[2] I. H. Hassan, A. Mohammed and M. A. Masama, "Metaheuristic algorithms in network intrusion detection," in *Comprehensive Metaheuristics: Algorithms and Applications*, Academic Press, 2023, pp. 95-129.

[3] E. K. B. E. K. K. G. &. K. G. Burke, earch methodologies: introductory tutorials in optimization and decision support techniques., New York: Springer, 2014.

[4] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal and D. ... Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems (NeurIPS 2020)*, 2020.

[5] I. H. Hassan, A. Mohammed, Y. S. Ali, I. Jeremiah and S. A. Abdulraheem, "Metaheuristic algorithms in text clustering," in *Comprehensive Metaheuristics,: Algoriths and Applications*, Academic Press, 2023, pp. 131-152.

[6] J. H. Holland, "Adaptation in Natural and Artificial Systems," *MIT Press,* vol. 1992, 1992.

[7] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN*, 1995.

[8] P. M. Pardalos and T. D. Mavridou, "Simulated annealing.," in *In Encyclopedia of Optimization* , Cham: Springer International Publishing., 2024, pp. 1-3.

[9] C. Blum, "Ant colony optimization: A bibliometric review," *Physics of life reviews,,* vol. 51, pp. 87-95, 2024.

[10] J. Alammar and M. Grootendorst, Hands-on large language models: language understanding and generation, O'Reilly Media, Inc., 2024.

[11] Y. Li, X. Li, H. Wu, Y. Zhang, X. Cheng, S. Zhong and F. Xu, "Attention is all you need for llm-based code vulnerability localization," *arXiv e-prints,* pp. arXiv-2410, 2024.

[12] X. Wang and D. Zhou, "Chain-of-thought reasoning without prompting," in *Advances in Neural Information Processing Systems*, 2024.

[13] N. G. H. Vu, K. Wang and G. G. Wang, "Effective prompting with ChatGPT for problem formulation in engineering optimization," *Engineering Optimization,* pp. 1-18, 2025.

[14] X. Kang, Z. Wang, X. Jin, W. Wang, K. Huang and Q. Wang, "emplate-Driven LLM-Paraphrased Framework for Tabular Math Word Problem Generation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

[15] S. Ye, Z. Sun and G. G. L. L. Q. L. Z. &. L. Y. Wang, "Prompt alchemy: Automatic prompt refinement for enhancing code generation," *arXiv preprint ,* p. arXiv:2503.11085., 2025.

[16] B. Bordelon, A. Atanasov and C. Pehlevan, "A dynamical model of neural scaling laws," *arXiv preprint ,* p. arXiv:2402.01092.

[17] F. Ghaddar, I. Ahmad, A. Salman and M. Alfailakawi, "Large Language Model-Aided Design of Hybrid Coa, Lshade and Mpa Inspired Algorithm.," *SSRN,* pp. 1-32, 2025.

[18] D. Chen and Y. Huang, "Integrating reinforcement learning and large language models for crop production process management optimization and control through a new knowledge-based deep learning paradigm," *Computers and Electronics in Agriculture,* vol. 232 , p. 110028., 2025.

[19] N. Tang, "Towards effective validation and integration of llm-generated code," in *2024 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2014.

[20] W. Liang, G. A. Tadesse, D. Ho, F.-F. L., M. Zaharia, C. Zhang and J. Zou, "Advances, challenges and opportunities in creating data for trustworthy AI.," *Nature Machine Intelligence,* vol. 4, no. 8, pp. 669-677, 2022.

[21] I. Grossmann, M. Feinberg, D. C. Parker, N. A. Christakis, P. E. Tetlock and W. A. Cunningham, "AI and the transformation of social science research," *Science,* vol. 380, no. 6650, pp. 1108-1109, 2023.

[22] M. Pluhacek, A. Kazikova, T. Kadavy, A. Viktorin and R. Senkerik, "Leveraging large language models for the generation of novel metaheuristic optimization algorithms," in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, 2023.

[23] F. Liu, X. Lin, S. Yao, Z. Wang, X. Tong, M. Yuan and Q. Zhang, "Large language model for multiobjective evolutionary optimization," in *International Conference on Evolutionary Multi-Criterion Optimization*, Singapore, 2025.

[24] F. Liu, X. Tong, M. Yuan and Q. Zhang, "Algorithm evolution using large language model," *arXiv preprint,* p. rXiv:2311.15249, 2023.

[25] S. Liu, C. Chen, X. Qu, K. Tang and Y. S. Ong, "Large language models as evolutionary optimizers," in *2024 IEEE Congress on Evolutionary Computation (CEC)*, Yokohama, Japan, 2024.

[26] R. Zhong, Y. Cao, J. Yu and M. Munetomo, "Large language model assisted adversarial robustness neural architecture search," in *2024 6th International Conference on Data-driven Optimization of Complex Systems (DOCS)*, 2024.

[27] R. Zhong, Y. Xu, C. Zhang and J. Yu, " Leveraging large language model to generate a novel metaheuristic," *Cluster Computing,* vol. 27, no. 10, pp. 13835-13869., 2024.

[28] N. van Stein and T. Bäck, "Llamea: A large language model evolutionary algorithm for automatically generating metaheuristics," *IEEE Transactions on Evolutionary Computation.,* vol. 29, no. 2, pp. 331-345, 2025.

[29] A. Martinek, S. Lukasik and A. H. Gandomi, "Large Language Models as Tuning Agents of Metaheuristics," in *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning.*, Bruges, Belgium, 2024.

[30] S. Hameed, B. Qolomany, S. B. Belhaouari, M. Abdallah, J. Qadir and A. Al-Fuqaha, "Large Language Model Enhanced Particle Swarm Optimization for Hyperparameter Tuning for Deep Learning Models," *IEEE Open Journal of the Computer Society,* vol. 6, no. 2025, pp. 574-585, 2025.

[31] F. Ghaddar, I. Ahmad, A. Salman and M. Alfailakawi, "Large Language Model-Aided Design of Hybrid Coa, Lshade and Mpa Inspired Algorithm. Lshade and Mpa Inspired Algorithm," *SSRN,* 2025.