



Leveraging Large Language Models for the Generation of Novel Metaheuristic Optimization Algorithms

Michal Pluhacek
Tomas Bata University in Zlin
Zlin, Czech Republic
pluhacek@utb.cz

Anezka Kazikova
Tomas Bata University in Zlin
Zlin, Czech Republic
kazikova@utb.cz

Tomas Kadavy
Tomas Bata University in Zlin
Zlin, Czech Republic
kadavy@utb.cz

Adam Viktorin
Tomas Bata University in Zlin
Zlin, Czech Republic
aviktorin@utb.cz

Roman Senkerik
Tomas Bata University in Zlin
Zlin, Czech Republic
senkerik@utb.cz

ABSTRACT

In this paper, we investigate the potential of using Large Language Models (LLMs) such as GPT-4 to generate novel hybrid swarm intelligence optimization algorithms. We use the LLM to identify and decompose six well-performing swarm algorithms for continuous optimization: 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). We leverage GPT-4 to propose a hybrid algorithm that combines the strengths of these techniques for two distinct use-case scenarios. Our focus is on the process itself and various challenges that emerge during the use of GPT-4 to fulfill a series of set tasks. Furthermore, we discuss the potential impact of LLM-generated algorithms in the metaheuristics domain and explore future research directions.

CCS CONCEPTS

• **Mathematics of computing** → **Evolutionary algorithms.**

KEYWORDS

Large Language Models, Metaheuristic Optimization, Swarm Algorithms, Algorithm Generation, Decomposition and Construction, GPT-4

ACM Reference Format:

Michal Pluhacek, Anezka Kazikova, Tomas Kadavy, Adam Viktorin, and Roman Senkerik. 2023. Leveraging Large Language Models for the Generation of Novel Metaheuristic Optimization Algorithms. In *Genetic and Evolutionary Computation Conference Companion (GECCO '23 Companion)*, July 15–19, 2023, Lisbon, Portugal. ACM, New York, NY, USA, 9 pages. <https://doi.org/10.1145/3583133.3596401>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

GECCO '23 Companion, July 15–19, 2023, Lisbon, Portugal

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-0120-7/23/07...\$15.00

<https://doi.org/10.1145/3583133.3596401>

1 INTRODUCTION

In recent years, the field of optimization has experienced significant growth, with numerous algorithms being developed to solve complex problems. Swarm intelligence algorithms have emerged as popular and effective techniques for tackling a variety of optimization tasks [1, 6]. Despite their strengths, these algorithms often face challenges in maintaining population diversity and balancing exploration and exploitation in the search space [4]. To overcome these limitations, researchers have started to investigate hybrid approaches that combine the advantages of multiple algorithms [15]. Swarm intelligence is a research field inspired by the collective behavior of decentralized, self-organized systems, such as natural swarms. Swarm-based algorithms use interactions among simple agents to solve complex optimization problems. The family of swarm-based metaheuristics has grown significantly over the past decades. Nowadays, there are dozens, if not hundreds, of such algorithms [1, 6]. The number and variety of the algorithms make it hard for practitioners to select the proper method for a particular real-world problem. Moreover, the originality of different methods is coming under increased scrutiny from the research community [3]. In such a situation, it is difficult for researchers to orient in the field and identify novel directions for future improvements.

In this work, we explore the possibility of using Large Language Models (LLMs), like GPT-4 [14], to simplify and speed up the process of novel metaheuristic design. Particularly, we focus on the task of hybridization of several methods.

The proposed approach leverages the knowledge contained in GPT-4 to generate an optimization technique that fuses concepts from the selected swarm intelligence algorithms. This study aims to contribute to the research in LLM-generated algorithms and describe the process in detail.

Note: Several sections of this manuscript feature direct outputs from the Large Language Model GPT-4. To maintain the integrity of the reporting, these outputs were left unaltered and are given in *italic*. As noted in the discussion section, the authors recognize that some information provided by GPT-4 is not accurate and, therefore, should not be treated as facts or cited as such. The inclusion of these outputs serves to demonstrate the capabilities and limitations of the LLM and not to assert the correctness of the information provided.

2 METAHEURISTICS

Metaheuristic optimization algorithms efficiently explore large search spaces to find near-optimal solutions for complex problems. Key principles include population-based search for thorough exploration and balancing exploration and exploitation using various techniques, such as mutation, crossover, and neighborhood search. These strategies enable effective navigation of the search space while guiding the algorithms toward optimal solutions.

Furthermore, these algorithms incorporate stochastic processes, such as random initialization, probabilistic selection, and random perturbations, to introduce diversity and randomness into the search process. This prevents premature convergence and encourages exploration of different regions within the search space.

Lastly, adaptability and flexibility are essential aspects of metaheuristic algorithms. They are designed to be easily tailored to various optimization problems and domains through the use of problem-specific operators, objective functions, constraints, parameter tuning, and algorithmic modifications. This adaptability allows metaheuristic algorithms to address a wide range of optimization challenges effectively.

3 LARGE LANGUAGE MODELS

The development of Large Language Models (LLMs) can be traced back to the early days of natural language processing (NLP) [7] and machine learning. Initial efforts in NLP relied on rule-based systems, where explicit rules and grammar structures were manually defined to process and generate human language. However, these systems lacked scalability and adaptability to different languages and domains.

With the advent of machine learning, researchers began to explore data-driven approaches for NLP tasks, such as language modeling. Early language models, such as n-gram models [2], captured the probabilities of sequences of words in a given dataset. While these models showed promise, they were limited in their ability to capture long-range dependencies and complex semantic relationships between words.

The introduction of deep learning techniques, particularly recurrent neural networks (RNNs) [11] and long short-term memory (LSTM) networks [16] allowed for more expressive language models that could capture longer context windows and learn richer representations of language. However, these models still faced challenges in terms of training efficiency and scalability.

In recent years, the emergence of the Transformer architecture [17] has revolutionized the field of NLP, enabling the creation of increasingly large and powerful language models. The Transformer architecture leverages self-attention mechanisms to efficiently capture long-range dependencies and parallelize the training process, allowing for the development of LLMs such as GPT-2, GPT-3, and GPT-4 [14].

4 EXPERIMENT SETUP

The main focus of this work is the process of using LLMs to generate novel metaheuristics. To this end, we design a series of steps (tasks) for the LLM to undertake:

TASK A: Select feasible candidates from the pool of swarm-based metaheuristics for global optimization and provide a basic description of the methods.

TASK B: Identify and describe distinct components of each algorithm suitable for hybridization.

TASK C: Propose a novel hybrid method(s) based on specified goal(s).

TASK D: Provide a pseudo-code and an implementation of the proposed algorithm(s).

TASK E: Provide a reasoning for the design of proposed algorithm(s).

The workflow of the experiments is detailed in Figure 1. The corresponding prompt list is given below:

Prompt 1: "Provide a list of six swarm-based metaheuristics that have shown promising performance in global optimization tasks, and their unique characteristics make them ideal candidates for generating hybrid optimization algorithms using large language models."

Prompt 2: "Provide reasoning for the selection of these algorithms."

Prompt 3: "Identify unique components of each algorithm. Provide a description of each component and reason how it affects the algorithm performance and population dynamic."

Prompt 4: "Suggest a metaheuristic algorithm that would incorporate some of these components with the aim to promote population diversity and balance exploration and exploitation effectively."

Prompt 5: "Provide a description of the proposed ESEEO (Enhanced Swarm Exploration and Exploitation Optimizer) algorithm."

Prompt 6: "Suggest a metaheuristic algorithm that would incorporate some of these components and was effective for expensive optimization tasks, where the the number of objective function evaluations is limited."

Prompt 7: "Provide a description of the proposed LESO (Limited Evaluation Swarm Optimizer) algorithm."

Prompt 8: "Provide the pseudo-code of ESEEO algorithm."

Prompt 9: "Provide working python implementation of ESEEO algorithm."

Prompt 10: "Provide the pseudo-code of LESO algorithm."

Prompt 11: "Provide working python implementation of LESO algorithm."

Prompt 12: "Provide a reasoning for the design of ESEEO."

Prompt 13: "Provide a detailed step-by-step reasoning for the design of ESEEO."

Prompt 14: "Provide a reasoning for the design of LESO."

Prompt 15: "Provide a detailed step-by-step reasoning for the design of LESO."

Please note that the GPT-4 is a stochastic system. Given the same prompt, it may generate different responses depending on various factors, such as the model's parameters, the temperature setting, and the tokens generated during sampling. While the system might generate different responses, they should still be relevant and coherent based on the input prompt. One way to improve the chances of obtaining similar results in hybrid algorithm design would be to include additional information (such as the list of algorithms or components).

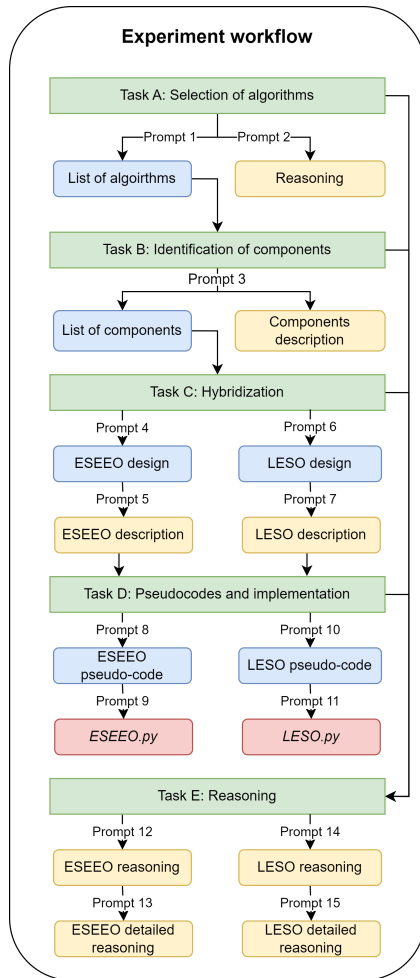


Figure 1: Experimental workflow

The experiment results are presented in the following sections. Subsequently, we evaluate and discuss the effectiveness of the GPT-4 LLM on each of these tasks. However, it is important to note that the evaluation of the performance and other characteristics of the resulting algorithms is not within the scope of this study.

5 TASK A: SELECTION

In the first task, the GPT-4 was prompted to propose a set of six swarm-based algorithms that have shown promising performance in global optimization tasks, and their unique characteristics make them ideal candidates for generating hybrid optimization algorithms using LLMs.

The GPT-4 model proposed the following list of algorithms:

- Particle Swarm Optimization (PSO) [9]
- Cuckoo Search (CS) [18]
- Artificial Bee Colony (ABC) [8]
- Grey Wolf Optimizer (GWO) [13]
- Self-Organizing Migrating Algorithm (SOMA)[5]
- Whale Optimization Algorithm (WOA) [12].

Subsequently, we prompted the GPT-4 to provide reasoning for the selection of these particular algorithms with the following result:

The algorithms were selected due to their popularity, effectiveness, and unique features that make them suitable candidates for creating a hybrid algorithm to tackle complex optimization problems. They represent a diverse set of swarm-based optimization algorithms that take inspiration from different natural phenomena.

Particle Swarm Optimization (PSO): PSO is a well-known swarm intelligence optimization technique inspired by the social behavior of bird flocks and fish schools. It has been proven effective in solving a wide range of optimization problems, including continuous, discrete, and combinatorial problems. Its simplicity, ease of implementation, and fast convergence make it an attractive choice.

Cuckoo Search (CS): CS is a metaheuristic algorithm inspired by the brood parasitism behavior of some cuckoo species. It has been applied to various optimization problems, including constrained and unconstrained, single and multi-objective, and continuous and discrete problems. The Lévy flight search strategy, which is employed in CS, introduces randomness that can help escape local optima.

Artificial Bee Colony (ABC): ABC is another popular swarm-based optimization algorithm, inspired by the intelligent foraging behavior of honey bees. The algorithm mimics the search for food sources by employed and onlooker bees, and has been successfully applied to various optimization tasks, including function optimization, machine learning, and engineering design problems.

Grey Wolf Optimizer (GWO): GWO is a recent addition to the swarm-based optimization algorithms, inspired by the social hierarchy and hunting behavior of grey wolves. GWO has shown promising performance in solving several benchmark functions, engineering design problems, and constrained optimization tasks. The algorithm's unique exploration and exploitation strategies have received considerable attention.

Self-Organizing Migrating Algorithm (SOMA): SOMA is a population based optimization algorithm that simulates the migration and adaptation of individuals within a group. It has been applied to various optimization tasks, such as function optimization, engineering design problems, and constrained optimization problems. SOMA's unique approach to global and local search is an attractive feature for creating hybrid algorithms.

Whale Optimization Algorithm (WOA): WOA is inspired by the bubble-net hunting strategy of humpback whales. The algorithm has been shown to be effective in solving various optimization problems, including single and multi-objective optimization tasks, as well as constrained and unconstrained problems. The unique encircling and spiral updating mechanisms in WOA offer different search strategies that can enhance population diversity and convergence.

6 TASK B: COMPONENTS IDENTIFICATION

In this task, the LLM was prompted to identify and describe unique components of each of the six algorithms and reason how these components impact the performance and population dynamics of the algorithm. The summary of the response follows:

Particle Swarm Optimization (PSO):

- **Inertia Weight (w):** In PSO, the inertia weight is an essential component that controls the balance between exploration

and exploitation. It affects the convergence speed and accuracy of the algorithm. A larger inertia weight promotes exploration (global search), while a smaller weight promotes exploitation (local search). Adaptive or time-varying inertia weight strategies can further enhance the performance of PSO.

- **Personal and Global Best:** PSO incorporates the concepts of personal best (p_i) and global best (g) positions. These components promote cooperation among particles, guiding them toward the most promising regions in the search space. The balance between personal and global best updates helps maintain diversity in the swarm and avoids premature convergence.

Cuckoo Search (CS):

- **Lévy Flights:** Cuckoo Search introduces Lévy flights as a method of performing random walks in the search space. Lévy flights exhibit a combination of long and short steps, making them effective for exploring both local and global regions of the search space. This property contributes to maintaining population diversity and improving exploration capabilities.
- **Host Nest Parasitism:** In CS, the cuckoos lay their eggs in other birds' nests, which represents a parasitic behavior. The algorithm simulates this by replacing a randomly chosen solution with a new solution generated by a cuckoo. This replacement strategy introduces randomness into the algorithm and helps in avoiding local optima by exploring new regions of the search space.

Artificial Bee Colony (ABC):

- **Employed and Onlooker Bees:** ABC divides the search agents into two groups: employed bees and onlooker bees. Employed bees are responsible for local search (exploitation) around their current solution, while onlooker bees perform a global search (exploration) based on the fitness values of the employed bees' solutions. This division of labor allows the algorithm to balance exploration and exploitation effectively.
- **Scout Bees:** In ABC, scout bees play a crucial role in maintaining population diversity and avoiding local optima. When a solution cannot be improved after a certain number of iterations, it is considered abandoned, and the employed bee becomes a scout bee. Scout bees perform a random search in the solution space, allowing the algorithm to explore new regions and potentially find better solutions.

Grey Wolf Optimizer (GWO):

- **Leadership Hierarchy:** GWO introduces a social hierarchy among the search agents, with the three best solutions representing the alpha, beta, and delta wolves. This hierarchy is used to guide the other wolves (omegas) in the search space. Each wolf updates its position considering the leaders' positions, ensuring that the search is focused on the most promising areas of the solution space.
- **Encircling and Hunting:** GWO incorporates two main mechanisms for searching the solution space: encircling prey and hunting. The encircling mechanism is based on the wolves converging toward the prey (i.e., the optimal solution), while the hunting mechanism involves the wolves performing a random search around the prey. These mechanisms ensure that the algorithm balances exploration and exploitation effectively.

Self-Organizing Migrating Algorithm (SOMA):

- **Perturbation Vector:** In SOMA, a perturbation vector is used to create variations of the individuals within the population. Each individual generates several trial positions during the search process, and these positions are determined by perturbing the original position with a random weight. This approach helps maintain population diversity and facilitates exploration by allowing individuals to explore different regions of the search space.
- **Migration:** The key concept in SOMA is the migration of individuals toward the best solution found so far. During migration, individuals move in a step-by-step manner, gradually converging toward the target. This mechanism encourages exploitation while still maintaining a degree of exploration by incorporating randomness through the perturbation vector.

Whale Optimization Algorithm (WOA):

- **Shrinking Encircling Mechanism:** WOA utilizes a shrinking encircling mechanism to guide the whales toward the global optimum. Whales move toward the current best solution while their search space shrinks. This mechanism promotes convergence toward the optimal solution while still exploring the surrounding search space.
- **Spiral Hunting:** In WOA, the whales perform a spiral hunting behavior around the current best solution. This behavior is modeled using a logarithmic spiral equation and introduces a stochastic element to the search process. Spiral hunting helps maintain population diversity and enhances the algorithm's exploration capability.

7 TASK C: HYBRIDIZATION

In this task, the GPT-4 model was prompted to propose a novel metaheuristic algorithm hybridizing any number of the previously identified unique components under two scenarios:

- An algorithm that promotes population diversity and balances exploration and exploitation effectively.
- An algorithm for expensive optimization tasks, where the number of objective function evaluations is limited.

To maintain population diversity and balances exploration and exploitation effectively and the LLM proposed combining Particle Swarm Optimization (PSO), Cuckoo Search (CS), and Artificial Bee Colony (ABC) and calling this new algorithm "Enhanced Swarm Exploration and Exploitation Optimizer" (ESEEO).

For the expensive scenario the GPT-4 model proposed "The Limited-Evaluation Swarm Optimizer" (LESO) hybrid metaheuristic combining elements from Particle Swarm Optimization (PSO), Grey Wolf Optimizer (GWO), and Cuckoo Search (CS),

The GPT-4 further provided the following descriptions of the algorithms:

7.1 Enhanced Swarm Exploration and Exploitation Optimizer (ESEEO)

The ESEEO algorithm consists of the following steps:

- *Initialize the population with random solutions. Determine the best solution found so far (global best).*

- For each individual in the population, calculate the fitness and update the personal best. Divide the population into two equal subpopulations (subpopulations A and B).
- For subpopulation A, use PSO with inertia weight and cognitive components to perform exploration and update personal best. This part will ensure a proper balance between exploration and exploitation, considering both the global and local search.
- For subpopulation B, use a combination of CS's Lévy flights and ABC's employed and onlooker bees' phases. This combination facilitates both local and global exploration by incorporating the randomness introduced by Lévy flights and the search for better solutions around the current best solutions.
- After a predefined number of iterations, combine both subpopulations and select the top 50% solutions to form the new population based on their fitness values.
- Redivide the new population into two subpopulations (A and B) again and perform steps 4-5 for a predefined number of iterations or until the stopping criterion is met.

By combining the strengths of PSO, CS, and ABC, the Enhanced Swarm Exploration and Exploitation Optimizer (ESEEO) aims to maintain population diversity and promote both exploration and exploitation more effectively than the individual algorithms alone. This hybrid approach should improve the chances of achieving a better result.

7.2 Limited Evaluation Swarm Optimizer (LESO)

The Limited Evaluation Swarm Optimizer (LESO) is a novel hybrid swarm intelligence algorithm designed to solve expensive optimization problems where the number of objective function evaluations is limited. It combines the strengths of Particle Swarm Optimization (PSO), Grey Wolf Optimizer (GWO), and Artificial Bee Colony's (ABC) local search capabilities to balance exploration and exploitation effectively. The proposed algorithm is particularly suitable for expensive optimization tasks where computational resources or evaluations are constrained.

The LESO algorithm consists of the following steps:

- Initialize the swarm (population) P , algorithm parameters, and the maximum allowed function evaluations N_{max} . Evaluate the fitness of each individual in the swarm. Set the allowed function evaluations N_{eval} to zero initially.
- While N_{eval} is less than N_{max} , perform the following steps:
 - Update the global best solution ($gbest$) found so far.
 - For each individual in the swarm, update their personal best solution ($pbest$), position (x), and velocity (v) using the PSO formulae. If an individual has not improved, consider it as a potential wolf.
 - Select the Alpha, Beta, and Delta wolves from the potential wolves based on their fitness values.
 - Update the position of each individual in the swarm, except the Alpha, Beta, and Delta wolves, using the GWO method. The updated position is based on the positions of the Alpha, Beta, and Delta wolves.
 - Randomly select a subset of individuals from the swarm to act as employed bees.
 - For each employed bee, perform a local search using the ABC's employed and onlooker bees' phases. This local search helps in refining the search around the current best solutions, allowing for better exploitation.

- Update the allowed function evaluations (N_{eval}) by adding the number of fitness evaluations performed in this iteration.
- After the stopping criterion is met, return the global best solution found.

8 TASK D: PSEUDO-CODES AND IMPLEMENTATION

In this task, the LLM was prompted to provide detailed pseudo-codes of the proposed algorithms. The resulting pseudo-codes are presented in Algorithm 1 and Algorithm 2. Further, the GPT-4 was requested to generate an implementation of each algorithm in Python. The generated codes are provided in the supplementary files.

Algorithm 1 Enhanced Swarm Exploration and Exploitation Optimizer (ESEEO)

```

1: Initialize population  $P$ , number of iterations  $T$ , and other algorithm parameters
2: for  $t = 1$  to  $T$  do
3:   Update global best  $gbest$  using fitness values of  $P$ 
4:   Update personal best  $pbest$  for each individual in  $P$ 
5:   Divide the population  $P$  into two equal subpopulations  $A$  and  $B$ 
6:   for each individual  $i$  in subpopulation  $A$  do
7:     Update velocity  $v_i$  of  $i$  using  $v_i = wv_i + c_1r_1(pb_{est_i} - x_i) + c_2r_2(gbest - x_i)$ 
8:     Update position  $x_i$  of  $i$  using  $x_i = x_i + v_i$ 
9:     Apply boundary constraints if necessary
10:    Update  $pbest_i$  for individual  $i$  if the new position has better fitness
11:   end for
12:   for each individual  $j$  in subpopulation  $B$  do
13:     Choose algorithm (CS or ABC) for individual  $j$  with a probability  $p$ 
14:     if algorithm is CS then
15:       Generate a random step size  $s$  from a Lévy distribution with a scale parameter  $\beta > 0$ :  $s \sim L(\beta)$ 
16:       Update position  $x_j$  of  $j$  using  $x_j = x_j + s \odot (x_j - gbest)$ , where  $\odot$  denotes element-wise multiplication
17:       Apply boundary constraints if necessary
18:     else if algorithm is ABC then
19:       Determine if  $j$  is an employed or onlooker bee and update position  $x_j$  accordingly
20:       Apply boundary constraints if necessary
21:     end if
22:     Update  $pbest_j$  for individual  $j$  if the new position has better fitness
23:   end for
24:   Merge subpopulations  $A$  and  $B$  into  $P$ 
25:   Select top 50% of individuals in  $P$  based on their fitness values
26: end for
27: Return the best solution found

```

Algorithm 2 Limited Evaluation Swarm Optimizer (LESO)

```

1: Initialize swarm  $P$ , algorithm parameters, and maximum allowed function evaluations  $N_{max}$ 
2: Evaluate fitness of each individual in  $P$ 
3: Initialize allowed function evaluations  $N_{eval} \leftarrow 0$ 
4: while  $N_{eval} < N_{max}$  do
5:   Update global best  $gbest$ 
6:   for each individual  $i$  in swarm  $P$  do
7:     Update personal best  $pbest_i$ 
8:     Update position  $x_i$  and velocity  $v_i$  of  $i$  using PSO formula:
        $v_i(t+1) = wv_i(t) + c_1r_1(pbest_i - x_i(t)) + c_2r_2(gbest - x_i(t))$ 
        $x_i(t+1) = x_i(t) + v_i(t+1)$ 
9:     Apply boundary constraints if necessary
10:    Update  $gbest$  if the new position has better fitness
11:    if individual  $i$  has not improved then
12:      Consider  $i$  as a potential wolf
13:    end if
14:  end for
15:  Select Alpha, Beta, and Delta wolves from potential wolves based on their fitness
16:  for each individual  $j$  in swarm  $P$  (except Alpha, Beta, and Delta wolves) do
17:    Update position  $x_j$  of  $j$  using GWO method based on positions of Alpha, Beta, and Delta wolves:
        $x_j(t+1) = x_\alpha(t) - A_\alpha D_\alpha + x_\beta(t) - A_\beta D_\beta + x_\delta(t) - A_\delta D_\delta$ 
18:    Apply boundary constraints if necessary
19:    Update  $gbest$  if the new position has better fitness
20:  end for
21:  Randomly select a subset of individuals from  $P$  to act as employed bees
22:  for each employed bee  $k$  do
23:    Perform local search around  $x_k$  using ABC method and update position  $x_k$ 
24:    Update  $gbest$  if the new position has better fitness
25:  end for
26:  Remaining individuals in  $P$  act as onlooker bees
27:  for each onlooker bee  $l$  do
28:    Perform local search around best position found by employed bees using ABC method and update position  $x_l$ 
29:    Update  $gbest$  if the new position has better fitness
30:  end for
31:  Update allowed function evaluations  $N_{eval} \leftarrow N_{eval} + N_{curr}$ 
32: end while
33: Return the best solution found

```

9 TASK E: REASONING

In the final task, the GPT-4 model was prompted to provide detailed reasoning for its choices during designing the hybrid algorithms and as the initial response was deemed too vague, the model was further prompted to provide a step-by-step reasoning. The generated responses are provided below.

9.1 ESEEO - reasoning

The design of the Enhanced Swarm Exploration and Exploitation Optimizer (ESEEO) was guided by the aim of creating a hybrid optimization algorithm that synergistically combines key components from multiple swarm-based methods to achieve superior performance. The algorithm incorporates aspects of Particle Swarm Optimization (PSO), Cuckoo Search (CS), and Artificial Bee Colony (ABC).

The primary motivation behind the selection of these components is to achieve a balance between exploration and exploitation in the optimization process. The algorithm starts by dividing the population into two subpopulations (A and B). For subpopulation A, PSO's exploration and exploitation capabilities are employed, utilizing inertia weight and cognitive components. This approach ensures a proper balance between global and local search while considering both global and personal best solutions.

For subpopulation B, the algorithm leverages the randomness introduced by CS's Lévy flights and the search for better solutions around the current best solutions provided by the ABC algorithm. This combination facilitates both local and global exploration, allowing search agents to explore the search space more effectively.

The Enhanced Swarm Exploration and Exploitation Optimizer (ESEEO) is designed to leverage the strengths of Particle Swarm Optimization (PSO), Cuckoo Search (CS), and Artificial Bee Colony (ABC) algorithms. Detailed step-by-step reasoning behind the proposal follows:

- **Initialization:** The algorithm starts with a randomly initialized population. This provides an initial set of diverse solutions in the search space, allowing the algorithm to explore different regions during the optimization process.
- **Global best update:** By identifying the best solution found so far in the entire population, the algorithm maintains a record of the highest fitness value achieved, which helps guide the search process towards optimal regions in the search space.
- **Personal best update:** For each individual in the population, the algorithm calculates its fitness and updates its personal best. This ensures that the particles remember the best solution they have found so far, which further contributes to the exploration-exploitation balance.
- **Subpopulation division:** Dividing the population into two equal subpopulations (A and B) is an essential step to diversify the search strategies and exploit the unique strengths of the underlying algorithms.
- **Subpopulation A (PSO):** Using the PSO strategy for exploration and personal best update helps achieve a balance between exploration (searching new regions in the search space) and exploitation (improving solutions within the current regions). The cognitive components of PSO contribute to finding a balance between the individual's knowledge and the swarm's knowledge.
- **Subpopulation B (CS and ABC):** Combining CS's Lévy flights and ABC's employed and onlooker bees' phases in subpopulation B enhances both local and global exploration. Lévy flights introduce randomness and long-range steps, improving the search for diverse solutions. In contrast, the employed and onlooker bees in ABC search for better solutions in the

neighborhood of the current best solutions, thus intensifying the search in promising regions.

- **Subpopulations merging:** After a predefined number of iterations, merging the two subpopulations and selecting the top 50% of solutions based on their fitness values combines the best solutions found by the different strategies, preserving population diversity.
- **Iterative process:** Redividing the merged population into two subpopulations (A and B) and repeating steps 4-5 allows the algorithm to iteratively explore and exploit the search space while preserving population diversity. This iterative process continues until a predefined number of iterations is reached or the stopping criterion is met.

By combining the strengths of PSO, CS, and ABC in a single hybrid algorithm, the Enhanced Swarm Exploration and Exploitation Optimizer (ESEEO) aims to achieve better results by maintaining population diversity and effectively promoting both exploration and exploitation.

9.2 LESO - reasoning

The design of the Limited Evaluations Swarm Optimizer (LESO) was driven by the need to create a hybrid optimization algorithm tailored to expensive optimization tasks, where the number of objective function evaluations is limited. The algorithm combines key components from three swarm-based methods: Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), and Grey Wolf Optimizer (GWO).

The primary motivation behind the selection of these components is to achieve a balance between exploration and exploitation while minimizing the number of objective function evaluations. PSO's exploration capability is enhanced by incorporating an adaptive inertia weight to prevent premature convergence. GWO's encircling and hunting behaviors are introduced to ensure efficient exploitation of promising areas within the search space.

Additionally, the LESO algorithm adopts the employed and onlooker bee phases from the ABC algorithm, which allows for a more effective exploration of the search space. This mechanism enables search agents to focus their efforts on areas with higher potential for improvement and helps in escaping local optima.

To further reduce the number of evaluations, LESO employs an adaptive learning mechanism for the search agents. This mechanism enables the search agents to adjust their exploration and exploitation tendencies dynamically during the optimization process, focusing their efforts on areas with higher potential for improvement.

The rationale behind the proposal of the Limited Evaluation Swarm Optimizer (LESO) is rooted in the need to develop a hybrid swarm intelligence algorithm that can efficiently solve expensive optimization problems with limited function evaluations. To achieve this, it combines the strengths of the Particle Swarm Optimization (PSO), Grey Wolf Optimizer (GWO), and the Artificial Bee Colony's (ABC) local search capabilities.

A step-by-step reasoning behind the proposal follows:

Selection of algorithms: the algorithm combines PSO, GWO, and ABC because they each offer unique advantages for exploration and exploitation. PSO is known for its simplicity and efficiency, while GWO's hierarchical structure allows for effective exploration, and ABC's local search capabilities make it ideal for exploitation. By combining these algorithms, the aim is to create a robust optimizer

that can effectively tackle expensive optimization tasks with limited evaluations.

PSO component: PSO's update rules for individuals' personal best solutions and velocities were incorporated to maintain a balance between exploration and exploitation. This allows the swarm to be attracted towards better solutions while still exploring the search space.

GWO component: By selecting the Alpha, Beta, and Delta wolves from potential wolves (those that haven't improved), the algorithm can encourage exploration by mimicking GWO's natural hierarchy. The positions of other individuals are updated based on the positions of the Alpha, Beta, and Delta wolves, which helps guide the search toward promising regions of the search space.

ABC local search: LESO introduces the local search capability of ABC's employed and onlooker bees to refine the search around the current best solutions. This addition allows for better exploitation of the search space, which is crucial for expensive optimization tasks where function evaluations are limited.

10 DISCUSSION

In this section, we take a look at the way the LLM managed to fulfill the set tasks.

10.1 TASK A

Assignment: In the first task, the GPT-4 model was prompted to provide a list of six swarm-based metaheuristics that "have shown promising performance in global optimization tasks, and their unique characteristics make them ideal candidates for generating hybrid optimization algorithms using LLMs."

Result assessment: A correct number of algorithms was proposed; all proposed algorithms do exist and are usable for global optimization. As requested, all algorithms belong to the swarm family. The algorithm descriptions are factually correct, albeit general.

10.2 TASK B

Assignment: The second task requested the GPT-4 to identify the unique (distinct) components of each algorithm.

Result assessment: The model did provide two components for each algorithm (the number was not defined by the user). It could be argued that some of the algorithms encompass a different number of unique components. Further, the distinctiveness of these components could also be questioned. The description of the components does not contain factual errors.

10.3 TASK C

Assignment: Propose a novel hybrid method(s) based on user-specified goal

Result assessment: GPT-4 successfully proposed two hybrid algorithms (ESEEO and LESO) by combining selected components from the six swarm-based algorithms proposed earlier and further provided step-by-step descriptions of the hybrid algorithms. However, a thorough literature review is necessary to validate the novelty of the proposed algorithms. Google Scholar search for the names of proposed algorithms found no match. The criteria for selecting particular components are, however, unclear. Interestingly,

the GPT-4 selected in both cases only three components. And in both scenarios utilizes components from PSO and ABC.

10.4 TASK D

Assignment: The task was to generate pseudo-codes and Python implementation of the proposed algorithms.

Result assessment: While the LLM generated pseudo-codes and Python implementation that at first glance resemble the described algorithms, closer inspection reveals numerous errors and inconsistencies.

ESEEO: Both the algorithm description and the pseudo-code fail to specify the subsequent action after selecting the fitter half of the population (line 25). This crucial selection step is also absent from the Python implementation. Additionally, neither the pseudocode nor the Python implementation includes a counter or check for the predefined number of iterations mentioned in the algorithm description.

LESO: Although both the algorithm description and pseudo-code detail the roles of employed and onlooker bees, the Python code only utilizes the employed bee behavior. Additionally, the counting of function evaluations in the pseudocode is defined in an impractical manner, as all evaluations are added to the counter at the end of each iteration. This approach is not well-suited for expensive optimization tasks. Furthermore, the function evaluation counting is inaccurately implemented in the Python code. In the ABC segment of the algorithm, the function evaluation counter is only updated if the new solution outperforms the old one. This method is clearly incorrect, as the objective function has already been evaluated irrespective of the solution's quality.

Moreover, noticeable inconsistencies exist between the pseudo-codes, such as discrepancies in terminology (e.g., "number of iterations" versus "maximum allowed function evaluations") and the notation used in the PSO update formula(s). Despite these shortcomings, it is worth mentioning that both Python implementations ran flawlessly, including the execution of the artificial test function and results report, without requiring any modifications. This indicates that the LLM successfully generated working code, even though it did not perfectly align with the algorithm description.

10.5 TASK E

Assignment: Provide reasoning for the specific design of the proposed hybrid algorithm(s)

Result assessment: The provided reasoning has some critical flaws, including inaccuracies such as the mention of adaptive inertia weight in the LESO algorithm, which is absent from its pseudo-code and Python implementation. The LESO's reasoning also refers to an "adaptive learning mechanism," which is not present in the algorithm.

In contrast, the ESEEO algorithm's reasoning is more nuanced and accurate. Though some explanations are general, the reasoning for ESEEO correctly describes the rationale for combining key components from multiple swarm-based methods to balance exploration and exploitation in the optimization process.

11 CONCLUSION

In this paper, we have investigated the process of employing the Large Language Model GPT-4 to generate novel hybrid metaheuristics. Overall, the results are promising, as the LLM was able to adhere to the instructions without significant errors and produce working code that aligns with the algorithm descriptions and pseudo-codes. However, when prompted to elaborate on the reasoning or provide specific details about the process, inaccuracies appeared more frequently. It is evident that there remain limitations and challenges to be addressed, making it crucial to approach GPT-4 outputs with caution. Thorough validation and fact-checking by human users are essential to ensure the reliability and accuracy of the generated information.

Given the limited scope of this initial study, several directions for future research are apparent. Firstly to undertake performance studies of LLM-generated algorithms and compare their actual behavior with the desired (and claimed) one. Secondly, investigate the novelty of LLM-generated algorithms and better clarify the exact mechanism of algorithm/component selection. There is an argument to be made that the model might be biased to prefer popular (frequently published) algorithms [10], regardless of their actual performance or suitability for hybridization.

Another possible direction for future research could involve using LLM-generated algorithms as components of hybrid optimization approaches or leveraging LLM-generated algorithms to improve the performance of existing metaheuristics through adaptive parameter tuning or operator selection.

One additional aspect to consider for future research involves the ongoing enhancement of large language models (LLMs). It would be beneficial to replicate similar experiments using more advanced models and observe any differences in the outcome.

Despite their potential, it is important to acknowledge the limitations inherent to Large Language Models. By their very design, LLMs are incapable of independent reasoning or understanding, and their output is inherently probabilistic. Thus, while they can generate innovative and seemingly insightful content, they do not guarantee the semantic correctness or reliability of the generated output. The 'black-box' nature of these models further challenges the predictability and trustworthiness of their results.

As with any emerging technology, the use of LLM-generated algorithms in optimization raises important ethical and societal concerns. Future research should explore these issues and develop guidelines and best practices for the responsible application of LLM-generated algorithms in various domains. Ultimately, we envision a future where LLM-generated metaheuristic algorithms become an integral part of the optimization research landscape, enabling researchers and practitioners to rapidly develop and deploy customized optimization solutions for a wide range of applications,

ACKNOWLEDGMENTS

The research presented in this paper was partially supported by the Internal Grant Agency of the Tomas Bata University in Zlin, under project number IGA/CebiaTech/2023/004, and resources of A.I.Lab at the Faculty of Applied Informatics, Tomas Bata University in Zlin (ailab.fai.utb.cz).

REFERENCES

- [1] Christian Blum and Andrea Roli. 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM computing surveys (CSUR)* 35, 3 (2003), 268–308. ISBN: 0360-0300 Publisher: Acm New York, NY, USA.
- [2] Peter F. Brown, Vincent J. Della Pietra, Peter V. Desouza, Jennifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational linguistics* 18, 4 (1992), 467–480.
- [3] Christian L. Camacho-Villalón, Marco Dorigo, and Thomas Stützle. 2022. Exposing the grey wolf, moth-flame, whale, firefly, bat, and antlion algorithms: six misleading optimization techniques inspired by bestial metaphors. *International Transactions in Operational Research* (2022). Publisher: Wiley Online Library.
- [4] Maurice Clerc and James Kennedy. 2002. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE transactions on Evolutionary Computation* 6, 1 (2002), 58–73. ISBN: 1089-778X Publisher: IEEE.
- [5] Donald Davendra and Ivan Zelinka. 2016. *Self-Organizing Migrating Algorithm: Methodology and Implementation*. Vol. 626. Springer.
- [6] Absalom E. Ezugwu, Amit K. Shukla, Rahul Nath, Andronicus A. Akinyelu, Jeffrey O. Agushaka, Haruna Chiroma, and Pranab K. Muhuri. 2021. Metaheuristics: a comprehensive overview and classification along with bibliometric analysis. *Artificial Intelligence Review* 54 (2021), 4237–4316. ISBN: 0269-2821 Publisher: Springer.
- [7] Julia Hirschberg and Christopher D. Manning. 2015. Advances in natural language processing. *Science* 349, 6245 (2015), 261–266. ISBN: 0036-8075 Publisher: American Association for the Advancement of Science.
- [8] Dervis Karaboga. 2005. *An idea based on honey bee swarm for numerical optimization*. Technical Report. Technical report-tr06, Erciyes university, engineering faculty, computer
- [9] James Kennedy and Russell Eberhart. 1995. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, Vol. 4. IEEE, 1942–1948.
- [10] Li Lucy and David Bamman. 2021. Gender and Representation Bias in GPT-3 Generated Stories. In *Proceedings of the Third Workshop on Narrative Understanding*. Association for Computational Linguistics, Virtual, 48–55. <https://doi.org/10.18653/v1/2021.nuse-1.5>
- [11] Larry Medsker and Lakhmi C. Jain. 1999. *Recurrent neural networks: design and applications*. CRC press.
- [12] Seyedali Mirjalili and Andrew Lewis. 2016. The whale optimization algorithm. *Advances in engineering software* 95 (2016), 51–67. ISBN: 0965-9978 Publisher: Elsevier.
- [13] Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis. 2014. Grey wolf optimizer. *Advances in engineering software* 69 (2014), 46–61. ISBN: 0965-9978 Publisher: Elsevier.
- [14] OpenAI. 2023. OpenAI GPT-4. <https://openai.com/research/gpt-4>
- [15] Günther R. Raidl, Jakob Puchinger, and Christian Blum. 2019. Metaheuristic hybrids. *Handbook of metaheuristics* (2019), 385–417. ISBN: 331991085X Publisher: Springer.
- [16] Greg Van Houdt, Carlos Mosquera, and Gonzalo Nápoles. 2020. A review on the long short-term memory model. *Artificial Intelligence Review* 53 (2020), 5929–5955. ISBN: 0269-2821 Publisher: Springer.
- [17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
- [18] Xin-She Yang and Suash Deb. 2009. Cuckoo search via Lévy flights. In *2009 World congress on nature & biologically inspired computing (NaBIC)*. Ieee, 210–214.