The following publication Y. Huang, S. Wu, W. Zhang, J. Wu, L. Feng and K. C. Tan, "Autonomous Multiobjective Optimization Using Large Language Model," in IEEE Transactions on Evolutionary Computation, vol. 30, no. 2, pp. 594-608, April 2026 is available at https://doi.org/10.1109/TEVC.2025.3561001.

Autonomous Multi-Objective Optimization Using Large Language Model

Yuxiao Huang, Shenghao Wu, Wenjie Zhang, Jibin Wu, Liang Feng, and Kay Chen Tan,

***Abstract*—Multi-objective optimization problems (MOPs) are ubiquitous in real-world applications, presenting a complex challenge of balancing multiple conflicting objectives. Traditional multi-objective evolutionary algorithms (MOEAs), though effective, often rely on domain-specific expertise for improved optimization performance, hindering adaptability to unseen MOPs. In recent years, the Large Language Models (LLMs) has revolutionized software engineering by enabling the autonomous generation and refinement of programs. Leveraging this breakthrough, we propose a new LLM-based framework that autonomously designs MOEAs for solving MOPs. The proposed framework includes a robust testing module to refine the generated MOEA through error-driven dialogue with LLMs, a dynamic selection strategy along with informative prompting-based crossover and mutation to fit textual optimization pipeline. Our approach facilitates the design of MOEA without the extensive demands for expert intervention, thereby speeding up the innovation of MOEA. Empirical studies across various MOP categories validate the robustness and superior performance of our proposed framework.**

***Index Terms*—Multi-objective Optimization, Automatic Algorithm Design, Large Language Model**

I. Introduction

MULTI-OBJECTIVE optimization is a crucial field of research that tackles the challenge of simultaneously optimizing two or more conflicting objectives in real-world scenarios [1]. The primary goal of multi-objective optimization is to find a set of solutions, also known as the Pareto Optimal Set [2], where no other solutions outperform all objectives. To solve the multi-objective optimization problems (MOPs), a lot of well-known multi-objective evolutionary algorithms (MOEAs) have been developed in the literature, including non-dominated-sorting-based methods [3]–[5] and decomposition-based techniques [6]–[8]. With the growing demand for multi-objective optimization in real-world applications [9], extensive efforts have been made to improve the search performance of MOEAs [10]–[12], aiming to tackle new multi-objective optimization challenges. While existing MOEAs have shown remarkable success, crafting these methods requires extensive expert resources. Additionally, when new problems arise, applying these algorithms also demands a deep understanding of the problem characteristics and the algorithms themselves to achieve effective solutions, which is not feasible for ordinary users. Therefore, it is desirable to automatically configure the evolutionary search operators within the 'Reproduction' function, as well as the hyper-parameters, to enhance problem-solving in unseen scenarios.

Nowadays, owing to the impressive language processing capabilities, Large Language Models (LLMs) have made significant strides in software engineering tasks. Modern society increasingly leverages these powerful LLMs to address the demands of daily programming tasks [13]–[15]. Particularly, Chen *et al.* enabled LLMs to refine their programs autonomously [16], while Shypula *et al.* proposed to enhance the efficiency of programs via few-shot prompting and chain-of-thought [17]. Furthermore, efforts to create powerful programs using LLMs have explored iterative evolution, leading to new mathematical discoveries [18], autonomous programming [19], enhanced vehicle routing [20], [21], hybrid swarm intelligence [22], and dynamic heuristic adaptation [23]. Moreover, Zhang *et al.* discussed the significant potential of automated heuristic design in automating the development of effective heuristics [24]. As these works demonstrate, LLMs are revolutionizing the field of software engineering through iterative program evolution, marking a new era in algorithmic design for optimization. While existing methods of program evolution have showcased notable successes across various domains, it is worth noting that most of them overlook the handling of generated programs that crash during assessment [21]–[23]. However, crafting innovative programs with LLMs often leads to mistakes [25]. Simply ignoring and removing these programs may result in missing innovative design ideas. Moreover, these methods typically adopt a traditional evolutionary search pipeline, such as selection with a fixed number of parents, which may not fully unleash the power of the LLM-assisted prompting-based textual optimization paradigm.

With the above in mind, we propose to develop a robust producer utilizing powerful commercial LLMs, capable of autonomously creating and refining a diverse array of 'Reproduction' methods within the MOEA for solving various MOPs by slightly altered prompts. In contrast to existing program search pipelines, the primary advantages of this work include the autonomous generation of 'Reproduction' methods for solving MOPs, the autonomous testing and refinement of the generated 'Reproduction' methods to preserve unusual designs, and an effective dynamic selection strategy to enhance the quality of crafting 'Reproduction' methods using LLMs. Particularly, due to the prevalence of execution errors in sophisticated

Corresponding author: Liang Feng

Yuxiao Huang, Shenghao Wu, Jibin Wu and Kay Chen Tan are with the Department of Data Science and Artificial Intelligence, The Hong Kong Polytechnic University, Hong Kong SAR. E-mail: {yuxiao.huang, shenghao.wu, jibin.wu, kctan}@polyu.edu.hk.

Wenjie Zhang is with Department of Electrical and Elctronics Engineering, The Hong Kong Polytechnic University, Hong Kong SAR. E-mail: wenjie_zhang@u.nus.edu

Liang Feng is with College of Computer Science and Chongqing Key Laboratory of Big Data Intelligence and Privacy Computing, Chongqing University, Chongqing 400044, China. E-mail: liangf@cqu.edu.cn.

programs generated by LLMs, our framework incorporates a robust testing module, which is seamlessly integrated into the search progress. This module treats compilation and runtime errors of tested 'Reproduction' methods as indicators, and then guides the autonomous refinement of the 'Reproduction' methods through dialogues with LLMs using the collected errors. To safeguard against the risk of infinite execution loops, we launch this module as an independent process, governed by a predefined execution time limit. Furthermore, to enhance the exploration capabilities of LLMs, we introduce a dynamic selection strategy, which selects effective parent 'Reproduction' methods dynamically, thereby improving the diversity of prompts. More specifically, our proposed framework initiates with a diverse population of 'Reproduction' methods generated through LLMs. As the search process unfolds, our proposed textual crossover and mutation techniques guide the LLM to craft advanced 'Reproduction' methods within the MOEA for solving various MOPs. Notably, these 'Reproduction' methods generated via LLMs can involve various inner search strategies (e.g., selection, crossover, mutation, and other problem-specific search strategies) to effectively address specific MOPs. The code and data involved in this study have been released online[1]. The contributions of this work can be outlined as follows:

- To handle the challenges posed by unseen MOPs, a new LLM-assisted evolution framework is devised, which enables the autonomous ideation and crafting of cutting-edge 'Reproduction' methods.
- To preserve the innovative designs of 'Reproduction' methods, a robust testing module is established, which can autonomously and seamlessly refine the generated methods through interactions with LLM.
- To obtain high-quality 'Reproduction' methods via LLMs, we introduce a dynamic selection strategy, along with informative crossover and mutation for the proposed LLM-based textual optimization paradigm.
- To validate the performance of our proposed framework, we have conducted extensive empirical studies, employing both continuous and combinatorial MOPs, against with human-engineered multi-objective optimization algorithms and the SOTA LLM-assisted method for program design. The obtained results demonstrate the efficacy of our proposed method.

The remainder of this paper is structured as follows. Section II first provides a literature review of existing works on LLM-assisted program evolution for solving diverse problems, followed by the introduction of three distinct MOPs that serves as our case studies. Our proposed framework is presented in Section III. Furthermore, to assess the performance of the proposed framework, comprehensive empirical studies are conducted in Section IV. Lastly, Section V summarizes this work and discusses the directions to be explored in the future.

II. Background

This section begins with a literature review of the LLM-assisted program evolution methods. Subsequently, the mathematical definitions of continuous MOPs and combinatorial MOPs are given, which serve as the key studies. Notably, the rationale behind the selection of these MOPs is to investigate the capability of LLM in devising innovative MOEAs across varied gene encoding modes.

A. Review of LLM-assisted Program Evolution

The emergence of LLMs have marked a new phase in optimization [14], [26], [27]. Within the context of textual optimization, such as program evolution, LLMs empower the generation of code that addresses a wide range of programming requirements [13]. Early study of Lehman *et al.* explored the potential of LLM trained to generate code in the context of genetic programming, which reveals that LLMs can significantly enhance the effectiveness of the programs [28]. Subsequently, Romera-Paredes *et al.* utilized LLMs to make new mathematical discoveries via search novel programs iteratively, indicating the potential software engineering capability by the integration of LLMs and evolutionary computation [18]. Liventsev *et al.* showcased the performance of fully autonomous programming using LLMs, suggesting a future where programs evolve without human guidance [19]. Liu *et al.* proposed a LLM-based algorithm evolution framework to design novel programs for solving traveling salesman problems (TSPs) [20]. Furthermore, they extended their work to develop superior guided local search for combinatorial optimization [21]. Pluhacek *et al.* leveraged LLMs in meta-heuristic optimization to generate hybrid swarm intelligence, potentially outperforming traditional methods [22]. Ye *et al.* introduced LLMs as hyper-heuristics within a reflective evolution framework, allowing for dynamic adaptation [23]. These advancements underscore the transformative role of LLMs in the algorithmic design for optimization tasks.

Despite promising advancements, the exploration of LLM-assisted program evolution in optimization remains in its infancy. To date, no effort has been made to develop innovative MOEAs using LLMs for MOPs. Our proposed methodology aims to bridge this gap, crafting innovative MOEAs that deliver superior search performance across different MOPs.

B. Continuous Multi-objective Optimization Problems

Continuous Multi-objective Optimization Problems (CMOPs) constitute a pivotal category of optimization problems. These problems, characterized by continuous real numbers as decision variables, entail the optimization of multiple conflicting objectives. Considering a minimization CMOP, the mathematical definition can be given by:

$$\begin{aligned} &\text{Minimize} \\ &\qquad \mathbf{f}(\mathbf{x}) = [f^1(\mathbf{x}), f^2(\mathbf{x}), \ldots, f^k(\mathbf{x})] \\ &\text{Subject to} \\ &\qquad \mathbf{x} \in \mathcal{X} \end{aligned} \tag{1}$$

where k represents the number of objectives, $\mathbf{x}$ denotes the decision variables and $\mathcal{X}$ specifies the feasible search space. Generally, the goal of CMOP is to find a set of Pareto-optimal solutions, $\mathcal{P} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots\}, \mathbf{x} \in \mathcal{X}$, which cannot

[1]https://github.com/RunningPhoton/LLMCG-MOEA/tree/master

be improved in any objective without degrading another. The research value of CMOPs lies in their applicability to real-world scenarios, such as engineering design [29], finance [30], and environmental management [31].

C. Combinatorial Multi-objective Optimization Problems

In contrast to the continuous MOPs which involve real-valued decision variables, combinatorial MOPs focus on discrete decision variables, which present challenges due to their discontinuity and lack of smoothness. In what follows, we introduce two distinct combinatorial MOPs, each characterized by unique gene encoding.

1) Multi-objective Knapsack Problems: MOKPs are combinatorial optimization problems that involve selecting a subset of items from a given set, subject to a capacity constraint [32]. Each item has multiple associated objectives (e.g., different types of profits). The goal is to find a Pareto-optimal set of items that maximizes the profits, subject to the capacity constraint. The mathematical definition of MOKPs can be expressed as follows. Given n items with associated weights $\mathbf{w}_i$, k types of profits $\mathbf{p}_j^i$ $(i = 1, 2, \cdots, k)$, and a knapsack capacity C, the goal is to find a binary vector $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \ldots]$ to maximize the profits given by:

$$\begin{aligned} &\text{Maximize} \\ &\quad \mathbf{f}(\mathbf{x}) = [f^1(\mathbf{x}), f^2(\mathbf{x}), \ldots, f^k(\mathbf{x})] \\ &\text{Where} \\ &\quad f^i(\mathbf{x}) = \sum_{j=1}^{n} \mathbf{p}_j^i \cdot \mathbf{x}_j \\ &\text{Subject to} \\ &\quad \sum_{j=1}^{n} \mathbf{w}_j \cdot \mathbf{x}_j \leq C, \\ &\quad \mathbf{x}_j \in \{0, 1\}, \quad j = 1, 2, ..., n \end{aligned} \tag{2}$$

MOKPs find applications in many real-world optimization problems, such as food order optimization [33], engineering and operational research [34], cargo loading and project selection [35]. The versatility of MOKPs in addressing these diverse challenges underscores their significance in strategic planning and optimization across various industries.

2) Multi-objective Traveling Salesman Problems: MOTSPs extend the traditional Traveling Salesman Problem (TSP) by incorporating several minimization objectives [36]. In a MOTSP, a salesman is tasked with visiting each city in a set exactly once, while optimizing not just for the total travel distance, but also for additional objectives such as time, cost, or environmental impact. Mathematically, consider a set of n cities and distinct k types of pairwise distances $\mathbf{d}^i(u, v)$, where the u and the v specifies two different vertexes. The goal is to find a permutation π of the n cities that minimizes the k traveling distances simultaneously, which can be given by:

$$\begin{aligned} &\text{Minimize} \\ &\quad \mathbf{f}(\pi) = [f^1(\pi), f^2(\pi), \ldots, f^k(\pi)] \\ &\text{Where} \\ &\quad f^i(\pi) = \sum_{i=1}^{n-1} \mathbf{d}^i(\pi(i), \pi(i+1)) \\ &\text{Subject to} \\ &\quad \pi \in permutations(1, 2, \ldots, n) \end{aligned} \tag{3}$$

Multi-objective Traveling Salesman Problems (MOTSPs) have practical implications in logistics, transportation, and network design. Particularly, MOTSPs streamline complex decision-making processes in areas such as efficient routing for delivery services [37], sustainable urban development [38], and resource management in healthcare systems [39]. The adaptability of MOTSPs to cater to multiple objectives simultaneously is what makes them invaluable in the pursuit of operational excellence and sustainability across diverse sectors.

III. Proposed Method

In this section, the proposed method for crafting MOEAs via LLM is introduced. The goal of our proposed method is to automatically design effective reproduction method within the MOEA, which involves a set of inner search operators such as selection, crossover, mutation, solution repair, etc. The workflow of our proposed paradigm has been depicted in Fig.

Algorithm 1: Pseudocode of the operator evolution.

Input:
G_{ev}: Number of generations to evolve operators.
N_{ev}: Number of operators in each population.
PBs: Multi-objective problems.
Output:
$\mathcal{P}_{ev}$: The best evolutionary operators.

Obtain the initial operator population $\mathcal{P}_{ev}$ and the scores on PBs via 'Operator Initialization' (Alg. 2).
for $gen \leftarrow 1$ *to* G_{ev} **do**
  for $i \leftarrow 1$ *to* N_{ev} **do**
    Obtain selection probabilities of operators in $\mathcal{P}_{ev}$ based on their scores.
    Generate a random integer N_s, $1 < N_s \leq N_{max}$
    Obtain a new operator RC leveraging LLM and N_s selected operators through 'Operator Crossover' (Alg. 3).
    Obtain a new operator $\hat{RC}$ leveraging LLM and the generated operator RC through 'Operator Mutation' (Alg. 4).
    Obtain score of $\hat{RC}$ via parallel evaluation on PBs.
    $\mathcal{P}_{ev} = \mathcal{P}_{ev} \cup \{\hat{RC}\}$.
  end
  Update $\mathcal{P}_{ev}$ based on scores.
end

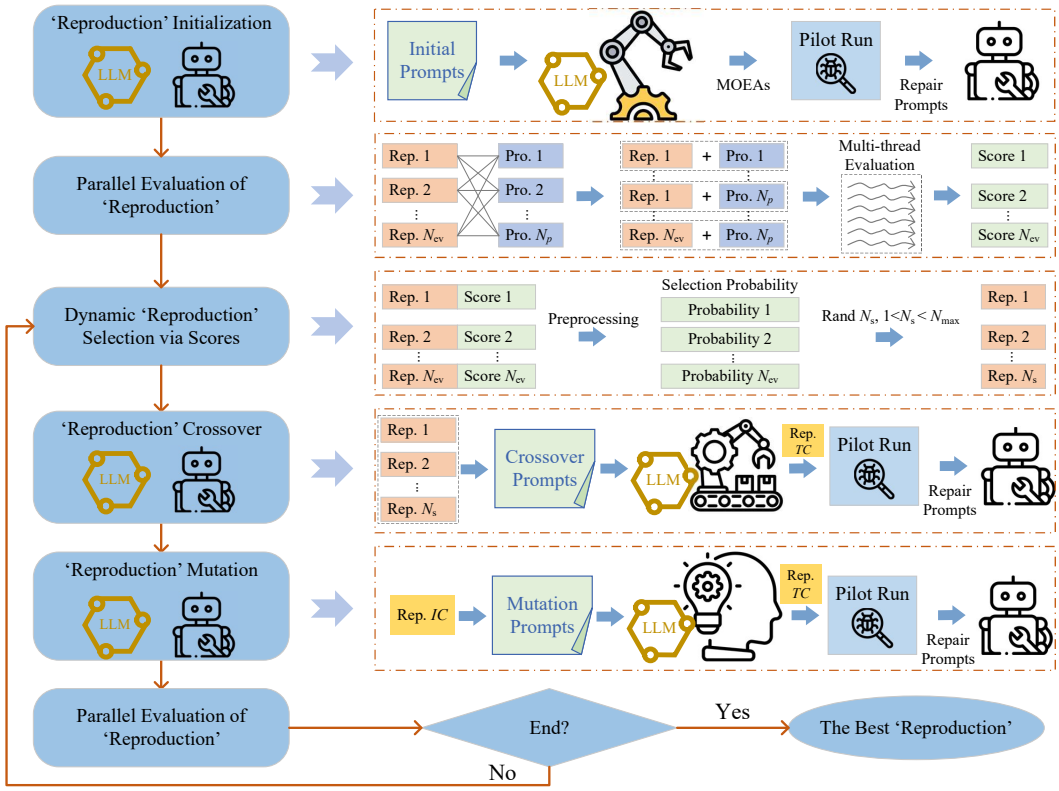


Fig. 1: Illustration of the proposed MOEA evolution pipeline via LLM.

1. In contrast to existing program evolution pipelines [18], [20], [28], our proposed method integrates an LLM-assisted debugging process, encompassing Pilot Run & Repair, into the core search stages: 'Reproduction' Initialization, 'Reproduction' Crossover, and 'Reproduction' Mutation. Additionally, the Dynamic 'Reproduction' Selection is developed to complement the LLM-based crossover mechanism, facilitating the handling of textual data.

As illustrated in Fig. 1 and outlined in Alg. 1, the paradigm begins with the phase of 'Reproduction' Initialization, where an initial population of 'Reproduction' methods is generated based on the 'Initial Prompts' (line 1 of Alg. 1). These 'Reproduction' methods undergo Parallel Evaluation, receiving scores based on their efficacy across MOPs. These scores inform the Dynamic 'Reproduction' Selection (lines 4-5), bringing diversity in prompts within "Crossover" by altering the number of selected parents, thus fostering the creation of innovative 'Reproduction' methods during 'Reproduction' Crossover (line 6) and 'Reproduction' Mutation (line 7). The new 'Reproduction' function obtained through mutation (i.e., $\hat{RC}$) is evaluated in parallel to obtain its score, which is then integrated into the population, followed by an elitist strategy-based update of the population (lines 8-11). The 'Reproduction' evolution persists, driven by enhanced optimization performance, until the termination criteria are met.

The paradigm's iterative design guarantees ongoing refinement, cycling through 'Reproduction' deployment and assessment until an predefined endpoint is reached. With this framework, the most effective MOEAs—demonstrating consistent superiority in MOPs—is remained. This LLM-aided approach enhances program evolution with human-like reasoning, improving the flexibility and strength of MOEAs. In what follows, the details of each component are introduced.

A. 'Reproduction' Initialization

This section introduces the initialization of 'Reproduction' methods by prompting LLM. It is commonly acknowledged that well-crafted prompts are pivotal for the effective initialization of algorithms, which can significantly speed up the evolution of 'Reproduction' methods. In light of this, we intend to employ detailed task descriptions and requirements to prompt the LLM, with the aim of engendering superior 'Reproduction' methods. The prompts for initialization are presented in the Fig. 2, titled by "INITIALIZATION". As depicted, the prompts can be briefly divided into three parts, i.e., 'System', 'Description of Task' and 'Requirements', respectively. The 'System' gives the initial prompts for LLM with a brief description for the task. The 'Description of Task' outlines the details of the task for the LLM, while the 'Requirements' provides the specific requirements on the task. Within these prompts, #PROBLEM#, #PROBLEM_DESC# and #FORMAT# are provided by users, which present the name of the problem, the informative introduction to the problem with optimization objectives, and the format to design the 'Reproduction' method, respectively. Examples of these placeholders have been given in the **Supplementary Materials**. Employing these comprehensive prompts enables the LLM to generate MOEAs that may achieve higher score

Algorithm 2: Pseudocode of the 'Reproduction' Initialization.

Input:
N_{ev}: Number of 'Reproduction' methods in each population.
PBs: Multi-objective problems.
Output:
$\mathcal{P}_{ev}$: The initialization population of 'Reproduction' methods.

```
P_ev ← {}
for i ← 1 to N_ev do
    RC ← None
    while RC is None do
        Obtain an 'Reproduction' TC via prompting LLM.
        Obtain new RC via Alg. 5.
    end
    Obtain score of RC via parallel evaluation on PBs.
    P_ev = P_ev ∪ {RC}
end
```

across MOPs, as delineated in Alg. 2. To further validate the MOEAs produced by the LLM engine, it will undergo Pilot Run & Repair as discussed in Section III-E, specifically lines 3-7 of Alg. 2. The generation process continues until an 'Reproduction' successfully passes the validation tests or a rectified 'Reproduction' is produced through the repair process outlined in Alg. 5.

INITIALIZATION

[System]:You are an expert in designing intelligent evolutionary search strategies that can solve **#PROBLEM#** efficiently and effectively. **#PROBLEM_DESC#**
[Description of Task]:Your task is to evolve a superior evolutionary operator with Python for tackling **#PROBLEM#**, with the goal of achieving top search performance across **#PROBLEM#**. You have to provide me the Python code with a single function namely 'next_generation' following the format and the requirements given below, which are matched with their functionalities. Specific format is given by: **#FORMAT#**
[Requirements]:You have to return me a single function namely 'next_generation', keep the format of input and the format of output unchanged, and provide concise descriptions in the annotation. Please return me an XML text using the following format: <next_generation>...</next_generation> where '...' gives only the entire code without any additional information. To enable direct compilation for the code given in '...', please don't provide any other text except the single Python function namely 'next_generation' with its annotation. No Explanation Needed!!

Fig. 2: Prompt design for 'Reproduction' Initialization

B. Dynamic 'Reproduction' Selection via Scores

This section introduces the proposed selection strategy developed for LLM-assisted evolution of MOEAs. In traditional evolutionary algorithms [40]–[42], this component is typically responsible for selecting a predetermined number of parents (commonly two) for subsequent operations such as crossover. In contrast, generative pre-trained transformer (GPT) models [43], [44] may yield similar responses (i.e., codes) when provided with fixed prompt—effectively replicating the same 'Reproduction' methods. This may lead to a stagnation in the 'Reproduction' evolution progress, as the newly generated 'Reproduction' methods, when underperforming compared to the existing methods within the population, may cause the process to become ensnared in local optima. Considering the aforementioned issue, this study introduces a dynamic selection mechanism to facilitate the generation of novel 'Reproduction' methods during crossover. Specifically, as illustrated in lines 4-5 of Alg. 1, the selection probabilities for the individual 'Reproduction' methods within $\mathcal{P}_{ev}$ are determined based on their performance scores on MOPs. A higher score signifies enhanced search capability and, consequently, an increased likelihood of selection. The selection probability for each 'Reproduction' is computed as follows:

$$P_s(i) = \frac{\exp(score_i)}{\sum_{k=1}^{N_{ev}} \exp(score_k)} \tag{4}$$

where N_{ev} denotes the number of 'Reproduction' methods within the population. Additionally, a random integer N_s, where $1 < N_s \leq N_{max}$, is generated to designate the count of parent 'Reproduction' methods chosen for crossover.

C. 'Reproduction' Crossover

Utilizing the dynamically selected parent 'Reproduction' methods, this study performs informative crossovers by engaging a LLM with these 'Reproduction' methods. The LLM, acting as an intelligent engine, is expected to generate innovative 'Reproduction' methods that outperform the parent 'Reproduction' methods in terms of scores. The prompts for

CROSSOVER

[Description of Task]:I will showcase several evaluated 'next_generation' functions in XML format, with their scores obtained on the **#PROBLEM#**. Your task is to conceive an advanced function with the same input/output formats, termed 'next_generation', that should draw inspiration from the high-qualified cases while differentiating itself from them. The format is given by: **#FORMAT#**. Below, you will find the *[N_s]* evaluated 'next_generation' functions in XML texts, each accompanied by its corresponding score. *[CODES]*
[Requirements]:Kindly devise an innovative 'next_generation' method with XML that retains the identical input/output structure. No Explanation Needed!

Fig. 3: Prompt design for 'Reproduction' Crossover

Algorithm 3: Pseudocode of the 'Reproduction' Crossover.

Input:
$\hat{\mathcal{P}}_{ev}$: Selected 'Reproduction' methods via Dynamic 'Reproduction' Selection.
Output:
RC: Validated or Repaired 'Reproduction'.

```
1  RC ← None
2  while RC is None do
3  |  Obtain a new 'Reproduction' TC via prompting
   |   LLM with P̂_ev.
4  |  Obtain new RC via Alg. 5.
5  end
```

the LLM are structured as Fig. 3, named by "CROSSOVER". Besides the user designed #PROBLEM#, #FORMAT#, and #CODES#, the "CROSSOVER" contains dynamically changed texts, i.e., *[N_s]* and *[CODES]*, which indicate the number of selected 'Reproduction' methods and the codes of these methods, respectively. It should be noted that, the *[N_s]* and *[CODES]* are automatically generated during our proposed method. The dynamic nature of these prompts is likely to facilitate the generation of varied 'Reproduction' methods by the LLM, as outlined in line 3 of Alg. 3. However, as outlined in line 4 of Alg. 1, the resultant 'Reproduction' TC requires additional validation or refinement to obtain RC, akin to the proposed 'Reproduction' Initialization.

D. 'Reproduction' Mutation

This section introduces the process of 'Reproduction' Mutation. The fundamental goal of this mutation component is to implement strategic modifications to the 'Reproduction' methods derived from the crossover phase. As illustrated in lines 1-7 of Alg. 4, these alterations occur with a predefined probability (i.e., $\frac{1}{2}$). Given an input 'Reproduction' method,

Algorithm 4: Pseudocode of the 'Reproduction' Mutation.

Input:
IC: Input 'Reproduction' gained via crossover.
Output:
$\mathcal{P}_{ev}$: The best 'Reproduction' methods.

```
// 'Reproduction' Mutation
1  if rand < 1/2 then
2  |  RC ← None
3  |  while RC is None do
4  |  |  Obtain a new 'Reproduction' TC via
   |  |   prompting LLM with IC.
5  |  |  Obtain new RC via Alg. 5.
6  |  end
7  end
8  else
9  |  RC ← IC
10 end
```

i.e., IC, we intend to alter the code slightly with the prompts given by Fig. 4, where the *[CODE]* represents the textual data of IC. Once gained a modified 'Reproduction' function by leveraging the LLM, the 'Reproduction' function (i.e., TC) will be validated or repaired through Alg. 5 akin to 'Reproduction' Crossover (line 5 of Alg. 4).

The mutation mechanism is designed to be adaptive, allowing for the fine-tuning of 'Reproduction' methods based on the evolving demands of the MOPs at hand. Such component reduces the risk of premature convergence and maintains healthy diversity within the population.

MUTATION

[Description of Task]:I will introduce an evolutionary search function namely 'next_generation' in XML format. Your task is to meticulously refine this function and propose a novel one that may obtain superior search performance on **#PROBLEM#**, ensuring the input/output formats, function name, and core functionality remain unaltered. The original function is given by: *[CODE]*
[Requirements]:Please return me an innovative 'next_generation' function with the same format. No Explanation Needed!

Fig. 4: Prompt design for 'Reproduction' Mutation

E. Pilot Run & Repair

Since the 'Reproduction' methods generated via LLMs may encounter various errors during compilation and execution, which can lead to disastrous outcomes in MOEA evolution, it is of great significance to design a component that ensures the quality of 'Reproduction' methods produced by LLMs. With this goal in mind, we introduce a Pilot Run & Repair component in this section, seamlessly integrated into the program's evolution process. This component treats compilation and runtime errors as indicators and engages in dialogues with LLMs to guide the search of innovative 'Reproduction' methods. Alg. 5 outlines the workflow of the proposed component. It assesses the quality of the tested 'Reproduction' method (referred to as TC) on the toy problems (given by PBs) within a specified budgeted running time ($MaxT$). If necessary, it proceeds to repair the method by identifying errors and engaging in dialogues with LLMs. The procedure initiates a while loop with a maximum repair time (N_{trail}). Within the loop, as observed in lines 3-4, an independent process is launched to execute RC on PBs, adhering to the time budget of $MaxT$. The process provides its running state and any encountered errors during execution (line 5). Notably, the error may be empty due to unknown issues (such as an endless loop). Subsequently, if $state = True$, it signifies that the tested 'Reproduction' successfully passed the pilot run and does not require further repair through LLM dialogues (lines 6-8). Conversely, when $state = False$ and $error$ is not empty, it indicates that errors occurred during testing and can be leveraged for code repair using LLMs (lines 9-11). The code repair prompts are illustrated in Fig. 5, termed as 'REPAIR'. *[ERROR]* represents the descriptions collected

Algorithm 5: Pseudocode of Pilot Run & Repair.

Input:
N_{trail}: Number of trails for code repair.
TC: The tested 'Reproduction' method.
PBs: The testing multi-objective problems.
$MaxT$: The maximum running time.
Output:
RC: Repaired 'Reproduction'.

```
1  iter ← 0
2  while iter < N_trail do
3      RC ← TC
4      Start an independent process to run RC on PBs
         under a limited running time MaxT.
5      Obtain the running state and error occurred
         during the testing run.
6      if state is True then
7          Return RC
8      end
9      else if state is False and error is not empty then
10         Obtain new TC based on communication with
             LLM using the informative error collected.
11     end
12     else
13         Return None
14     end
15     iter ← iter + 1
16 end
17 Return None
```

by the error handler automatically. Otherwise, if no specific issues are identified, $None$ is returned to denote default testing failure (lines 12-14). The repair process continues until the specified maximum repair time (N_{trail}) is met, ultimately returning $None$.

REPAIR

[Description of Task]:The code you provided for me cannot pass my demo test on **#PROBLEM#**. The error is given by: *[ERROR]*. Can you correct the code according to the errors?
[Requirements]:Please return me a refined 'next_generation' with the same XML format, i.e., <next_generation>...</next_generation>, where the '...' represents the code snippet. No Explanation Needed!

Fig. 5: Prompt design for repairing 'Reproduction' methods

F. Parallel Evaluation of 'Reproduction'

After the 'Reproduction' methods successfully complete the 'Pilot Run', a parallel evaluation is conducted to measure their effectiveness on the MOPs using a normalized score. As depicted in Fig. 1, each 'Reproduction' method, denoted as 'Rep. $*$', is systematically paired with a validation problem, referred to 'Pro. $*$'). Following this, the pairs are independently processed in multiple threads, each yielding an 'Reproduction' method's performance score for a specific MOP. It is worth noting that, the normalized score for the i-th 'Reproduction' method, aggregated across the MOPs, is calculated as follows:

$$score_i = MEAN(PS) - STD(PS) \tag{5}$$

Here, '$MEAN$' and 'STD' denote the mean and standard deviation of the collected scores, respective. PS specifies the scores obtained on the MOPs. By maximizing the normalized $score$ through LLM-assisted 'Reproduction' evolution, we aim to derive an 'Reproduction' method that not only excels consistently across MOPs but also demonstrates robust competitiveness across diverse MOPs.

IV. Experimental Study

In this section, we conduct comprehensive empirical studies to rigorously assess the efficacy of our proposed framework within the domain of multi-objective optimization. This section begins by delineating the experimental settings of validation and testing MOPs, the performance metrics across different MOP categories, and the setup of compared handcrafted MOEAs and the existing state-of-the-art method to search novel heuristics. Subsequently, the results obtained on COMPs, MOKPs and MOTSPs are presented and discussed to validate the performance of the proposed method, which is denoted by LLMMOP.

A. Experimental Settings

Due to the limited hardware resources, the commercial LLMs are employed in this study. Moreover, we select one continuous MOP and two combinatorial MOPs with very different gene encoding modes to validate the performance of our proposed framework: the CMOP with continuous variables, the MOKP with binary variables, and the MOTSP with permutation-based variables. Each MOP category is further subdivided into two distinct sets: a validation group and a testing group. The validation group consists of smaller-scale problem instances that are utilized during the evolutionary progression of 'Reproduction' methods in LLMMOP. On the other hand, the testing group comprises larger-scale problem instances that serve to assess the performance of the best 'Reproduction' method optimized through our proposed LLMMOP.

For the CMOPs, we select 'ZDT1' to 'ZDT6' (two objectives) and 'DTLZ1' to 'DTLZ7' (three objectives) [45] with default configurations (with decision variables ranging from 10 to 30) in [46] as the validation group, while their counterparts with more decision variables serve as the testing group. The performance score for each CMOP instance, denoted by PS_i, is calculated via $PS_i = 1 - IGD_i$, where IGD_i denotes the Inverted Generational Distance (IGD) value [47] obtained for the i-th problem instance. A lower IGD value signifies a higher score and indicates a more effective search performance by the 'Reproduction' method. In the case of the MOKP, we have randomly generated 10 problem instances with item counts ranging from 50 to 200 for the validation group, and 20 problem instances with different knapsack capacity

TABLE I: Configurations of hand-crafted MOEAs for solving different MOP categories.

		Decomposition-based	Non-dominated Sorting-based		
		CTAEA	NSGA2	AGEMOEA	SMSEMOA
Population Size		100	100	100	100
Generation		200	200	200	200
Crossover	CMOP	SBX	SBX	SBX	SBX
	MOKP	TwoPointCrossover	TwoPointCrossover	TwoPointCrossover	TwoPointCrossover
	MOTSP	OrderCrossover	OrderCrossover	OrderCrossover	OrderCrossover
Mutation	CMOP	PM	PM	PM	PM
	MOKP	BitflipMutation	BitflipMutation	BitflipMutation	BitflipMutation
	MOTSP	InversionMutation	InversionMutation	InversionMutation	InversionMutation
Repair	MOKP	RandWeightRepair	RandWeightRepair	RandWeightRepair	RandWeightRepair

and number of items (ranging from 100 to 200) for the testing group. The score for each MOKP instance is given by: $PS_i = 1 - HV_i$, where HV_i denotes the Hypervolume (normalized between 0 and 1) [48] achieved on the i-th problem instance. Since the MOKP is a maximization problem, a lower Hypervolume implies a higher score and more favorable results. Additionally, for the MOTSP, we have generated 20 problem instances with 30 vertices for the validation group. For the testing group, we have created 20 problem instances with varying numbers of vertices, ranging from 100 to 200, and diverse vertex distributions. The score for each MOTSP instance is determined by: $PS_i = HV_i$. As the MOTSP is a minimization problem, a higher normalized Hypervolume indicates a higher score and superior search capability.

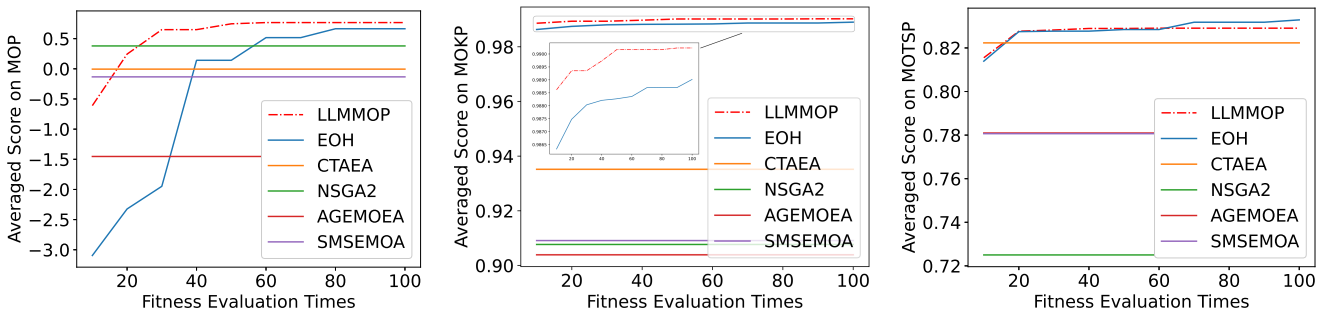
To demonstrate the search performance of the 'Reproduction' method optimized via the proposed LLMMOP, we adopt four hand-crafted multi-objective optimization algorithms with well-configured evolutionary operators for handling the three MOP categories based on [46], CTAEA [49], NSGA2 [3], AGEMOEA [50], SMSEMOA [51]. The configurations of these methods are detailed in Table I, ensuring a fair comparison by maintaining the same population size and the maximum number of generations across all algorithms (also configured in the 'Reproduction' method obtained through LLMMOP). As can be observed, for CMOPs, we employ the 'Simulated Binary Crossover (SBX)' [52] and 'Polynomial Mutation (PM)' [53] operators; MOKPs are addressed using 'Two-Point Crossover' [54] and 'Bit-Flip Mutation' [55]; while MOTSPs are optimized with 'Order Crossover' [56] and 'Inversion Mutation' [57]. Additionally, for the efficient resolution of MOKPs, the commonly adopted 'RandWeightRepair' strategy is employed, wherein items are randomly removed to meet the capacity constraints. Since the search space of 'Reproduction' is not predefined and the textual data can only be effectively processed by LLMs, we employ the state-of-the-art program search method (called by EOH) [21] for comparisons in solving these MOPs. The primary differences between EOH and our proposed LLMMOP lie in their application and methodology. EOH focuses on discovering new heuristics for combinatorial optimization problems, while LLMMOP targets novel reproduction methods for MOPs, addressing both black-box continuous and white-box combinatorial problems. To address the inherent randomness in LLM responses, we have performed three independent runs to compare the performance of our proposed LLMMOP with EOH across three case problems. Furthermore, each hand-crafted MOEA and the best reproduction methods identified by EOH and our proposed LLMMOP are subjected to 10 independent runs for each MOP instance to ensure robust statistical analysis. The hyper-parameters specific to our proposed method and EOH are as follows:

- Population size configured in EOH and LLMMOP to search programs: $N_{ev} = 10$.
- Number of fitness evaluation times for EOH and LLMMOP: 100.
- Maximum number of selected parents in LLMMOP: $N_{max} = N_{ev}/2$.
- Employed commercial LLMs: gpt-4-1106-preview, gpt-4o, Gemini-1.5-pro, Claude-3-opus.
- Temperature setting for the language model: 0.5.
- Maximum running time for Pilot Run: $MaxT = 2000s$.
- Number of trials for code repair in LLMMOP: $N_{trial} = 2$.

The temperature for each LLM employed in this study is set at 0.5 to balance creativity and robustness within the 'Reproduction' evolution framework. Other temperature configurations can also be applied accordingly. The maximum running time $MaxT$ and the number of trials $N_{trial} = 2$ for code repair are configured empirically.

B. Results and Discussions

In this section, we first evaluate the efficacy of the proposed LLMMOP on the previously discussed validation MOPs. Fig. 6 illustrates the convergence trajectories of EOH and the proposed LLMMOP across these MOPs over three independent runs. The X-axis denotes the fitness evaluation times, while the Y-axis shows the average scores achieved by the best-performing 'Reproduction' methods at the current search stage over three independent runs. As shown in Fig. 6 (a), the averaged performance of 'Reproduction' generated by EOH and LLMMOP at the early stage is relatively low compared to hand-crafted MOEAs due to the black-box nature of CMOPs, which provide limited problem information. Despite initial



(a) Validation CMOPs (b) Validation MOKPs (c) Validation MOTSPs

Fig. 6: The convergence curves of the proposed 'Reproduction' evolution framework on different types of MOPs.

TABLE II: Averaged IGD values obtained by CTAEA, NSGA2, AGEMOEA, SMSEMOA, the best 'Reproduction' method achieved via EOH, and the best 'Reproduction' method optimized via the proposed LLMMOP on testing CMOP instances over 10 independent runs. Superior averaged results on each instance are highlighted in bold font. The symbols $-$, $+$, and $\approx$ indicate that the corresponding algorithm achieved significantly worse, significantly better, and approximately equivalent results compared to LLMMOP, respectively.

Problem	N.VAR	CTAEA	NSGA2	AGEMOEA	SMSEMOA	EOH	LLMMOP
zdt1	50	2.313e-02$-$	9.998e-03$-$	1.098e-02$-$	6.838e-03$+$	**6.308e-03**$+$	1.133e-02
zdt2	50	6.199e-02$-$	**1.320e-02**$\approx$	2.779e-02$\approx$	5.123e-02$\approx$	2.471e-02$+$	3.996e-02
zdt3	50	3.211e-02$-$	8.365e-03$+$	7.526e-03$+$	1.234e-02$+$	**6.167e-03**$+$	1.947e-02
zdt4	50	3.199e+01$\approx$	3.673e+01$-$	**3.140e+01**$\approx$	3.194e+01$\approx$	3.510e+01$\approx$	3.180e+01
zdt5	80	1.836e-01$-$	**1.266e-01**$+$	1.424e-01$+$	1.493e-01$\approx$	1.468e-01$\approx$	1.587e-01
zdt6	50	2.445e+00$-$	1.321e+00$-$	1.052e+00$-$	1.476e+00$-$	7.635e-01$-$	**1.984e-02**
dtlz1	50	1.179e+02$-$	2.503e+02$-$	1.861e+02$-$	1.679e+02$-$	1.555e+02$-$	**6.519e+01**
dtlz2	50	**6.032e-02**$+$	8.110e-02$\approx$	6.666e-02$+$	6.678e-02$+$	8.582e-02$-$	7.758e-02
dtlz3	50	4.434e+02$-$	5.438e+02$-$	4.360e+02$-$	5.435e+02$-$	3.513e+02$-$	**2.054e+02**
dtlz4	50	**5.819e-02**$+$	8.066e-02$+$	4.409e-01$+$	3.421e-01$+$	2.649e-01$\approx$	3.982e-01
dtlz5	50	1.065e-01$-$	1.111e-02$+$	1.443e-02$\approx$	**6.240e-03**$+$	1.565e-02$\approx$	1.564e-02
dtlz6	50	2.506e+01$-$	2.658e+01$-$	2.801e+01$-$	1.847e+01$-$	2.328e+01$-$	**2.318e-01**
dtlz7	50	1.440e-01$-$	1.238e-01$\approx$	1.272e-01$\approx$	**9.460e-02**$\approx$	1.197e-01$-$	1.169e-01
Averaged IGD	-	4.780e+01	6.609e+01	5.257e+01	5.877e+01	4.359e+01	**2.335e+01**

trials, the generated 'Reproduction' methods for CMOPs were unsatisfactory. However, the EOH and the proposed LLMMOP allow the LLM to iteratively refine 'Reproduction' methods through feedback, showcasing the advantages of iterative program search pipeline over one-off code generation techniques. Finally, both 'Reproduction' methods obtained via EOH and LLMMOP achieved highest performance scores on the validation CMOPs. Fig. 6 (b) reveals that the 'Reproduction' methods generated through EOH and LLMMOP outperforms hand-crafted algorithms from the start in the MOKP domain, which leverages on the specific problem properties within the prompts. Notably, the optimization capabilities of these 'Reproduction' methods continued to improve as the evolution progresses. Similarly, Fig. 6 (c) shows that both EOH and LLMMOP started with a high initial score against NSGA2, AGEMOEA and SMSEMOA. By iteratively improving the performance, the 'Reproduction' methods gained via EOH and LLMMOP ultimately achieved competitive performance against CTAEA. Moreover, these figures indicate that the 'Reproduction' methods initialized by our proposed LLMMOP outperformed those initialized via EOH. This superior performance is primarily attributed to the high-quality initialization provided by LLMMOP. However, it is important to note that the subsequent search process also plays a crucial role in refining these initial solutions to achieve optimal results. The search process allows for further exploration and exploitation of the solution space, leading to more refined and higher-quality results. To delve deeper into the search capabilities of the best 'Reproduction' optimized via the EOH and the proposed LLMMOP over three independent runs, we present a detailed analysis of the results obtained on the testing CMOPs, MOKPs and MOTSPs.

1) Evaluation on CMOPs: Table II presents the averaged IGD values achieved by CTAEA, NSGA-II, AGEMOEA, SMSEMOA, the best 'Reproduction' obtained by EOH (namely 'EOH' in the table), and the best 'Reproduction' searched by LLMMOP (namely 'LLMMOP' in the table) across testing

TABLE III: Averaged HV values obtained by CTAEA, NSGA2, AGEMOEA, SMSEMOA, the best 'Reproduction' obtained via EOH, and the best 'Reproduction' generated by the proposed LLMMOP on testing MOKPs over 10 independent runs. Superior averaged results (Lower HV values) on each instance are highlighted in bold font. The symbols $-$, $+$, and $\approx$ indicate that the corresponding algorithm achieved significantly worse, significantly better, and approximately equivalent results compared to LLMMOP, respectively.

Problem	N.VAR	Capacity	CTAEA	NSGA2	AGEMOEA	SMSEMOA	EOH	LLMMOP
MOKP1	183	889	1.842e-02$-$	1.948e-02$-$	1.582e-02$-$	1.717e-02$-$	6.671e-03$-$	**4.374e-03**
MOKP2	132	610	1.012e-02$-$	9.229e-03$-$	8.909e-03$-$	9.192e-03$-$	3.374e-03$-$	**2.882e-03**
MOKP3	184	993	1.116e-02$-$	1.087e-02$-$	9.307e-03$-$	9.516e-03$-$	5.502e-03$-$	**1.889e-03**
MOKP4	141	664	1.809e-02$-$	1.644e-02$-$	1.472e-02$-$	1.568e-02$-$	5.667e-03$\approx$	**4.775e-03**
MOKP5	100	517	2.099e-02$-$	1.978e-02$-$	1.876e-02$-$	1.860e-02$-$	9.634e-03$-$	**7.393e-03**
MOKP6	146	691	1.530e-02$-$	1.393e-02$-$	1.247e-02$-$	1.288e-02$-$	4.522e-03$\approx$	**4.071e-03**
MOKP7	131	672	2.299e-02$-$	2.047e-02$-$	1.939e-02$-$	1.924e-02$-$	1.132e-02$\approx$	**9.190e-03**
MOKP8	160	803	9.441e-03$-$	8.774e-03$-$	7.068e-03$-$	8.296e-03$-$	3.333e-03$\approx$	**3.187e-03**
MOKP9	141	721	1.515e-02$-$	1.401e-02$-$	1.364e-02$-$	1.389e-02$-$	7.815e-03$-$	**5.307e-03**
MOKP10	111	582	1.290e-02$-$	1.144e-02$-$	1.075e-02$-$	1.093e-02$-$	6.372e-03$-$	**3.932e-03**
MOKP11	112	365	2.035e-02$-$	1.855e-02$-$	1.890e-02$-$	1.890e-02$-$	7.851e-03$-$	**3.027e-03**
MOKP12	169	514	1.613e-02$-$	1.518e-02$-$	1.289e-02$-$	1.224e-02$-$	5.284e-03$-$	**3.080e-03**
MOKP13	123	414	1.236e-02$-$	8.718e-03$-$	6.551e-03$-$	6.911e-03$-$	**2.217e-03**$+$	2.558e-03
MOKP14	112	574	8.270e-03$-$	8.169e-03$-$	7.463e-03$-$	7.341e-03$-$	4.521e-03$-$	**1.436e-03**
MOKP15	179	1099	4.525e-03$-$	4.192e-03$-$	2.762e-03$-$	3.057e-03$-$	1.139e-03$-$	**5.356e-04**
MOKP16	114	387	1.545e-02$-$	1.427e-02$-$	1.356e-02$-$	1.395e-02$-$	**7.069e-03**$+$	8.257e-03
MOKP17	122	877	1.837e-02$-$	1.906e-02$-$	1.726e-02$-$	1.723e-02$-$	**7.202e-03**$\approx$	8.244e-03
MOKP18	123	371	3.905e-02$-$	3.658e-02$-$	3.492e-02$-$	3.477e-02$-$	1.481e-02$-$	**6.352e-03**
MOKP19	124	712	2.637e-03$-$	2.214e-03$-$	1.543e-03$-$	1.747e-03$-$	**2.748e-04**$+$	6.059e-04
MOKP20	151	1345	1.158e-02$-$	1.160e-02$-$	7.527e-03$-$	7.959e-03$-$	1.853e-03$-$	**1.606e-03**
Averaged HV	-	-	1.516e-02	1.415e-02	1.271e-02	1.297e-02	5.822e-03	**4.135e-03**

CMOP instances based on 10 independent runs. 'N.VAR' gives the number of decision variables for each CMOP instance and the 'Average IGD' row reflects the mean IGD value across all evaluated problems. To facilitate a comprehensive performance comparison among these MOEAs, Wilcoxon rank-sum tests at a 95% confidence level have been conducted. The symbols '$-$', '$\approx$', and '$+$' indicate that the method in the corresponding column performs statistically significantly worse, equivalent, or better compared to LLMMOP on the given test problem. As can be observed from the table, the best 'Reproduction' generated by our proposed LLMMOP obtained the lowest 'Averaged IGD' against EOH and other hand-crafted MOEAs, demonstrates the superior generalization performance of LLMMOP. This success is primarily attributed to the significantly superior problem-solving capabilities of the 'Reproduction' obtained by the proposed LLMMOP on more challenging optimization tasks, such as ZDT6, DTLZ1 and DTLZ6. Although the 'Reproduction' generated by EOH performed well on simpler CMOPs, it struggled with more difficult tasks.

2) Evaluation on MOKPs: To assess the performance of our proposed method on MOKPs, Table III enumerates the average HV values achieved by CTAEA, NSGA-II, AGEMOEA, SMSEMOA, the optimal 'Reproduction' method gained by EOH, and the optimal 'Reproduction' method obtained through the proposed LLMMOP. Columns 'N.VAR' and 'Capacity' denote the number of items and the capacity of the knapsack, respectively. These MOKP instances vary in the number of decision variables and knapsack capacities. Additionally, the 'Average HV' row aggregates the HV metrics, providing a summarized view of the algorithms' overall performance across all testing instances. Similarly, the symbols '$-$', '$\approx$', and '$+$' indicate that the method in the corresponding column performs statistically significantly worse, equivalent, or better compared to LLMMOP on the given test problems. From Table III we can observe that the 'Reproduction' gained by LLMMOP significantly outperforms the competing methods across most testing cases with varied problem properties. The 'Reproduction' method gained via the proposed LLMMOP obtained the best performance in terms of the 'Averaged HV', followed by the 'Reproduction' optimized via EOH. This demonstrates the LLMMOP's capacity to construct powerful search operators within the 'Reproduction' method when provided with clearly defined problem attributes in the prompts. A closer look at the best 'Reproduction' generated by LLMMOP in the **Supplementary Materials** reveals an effective repair function crafted by the proposed LLMMOP, which strategically removes the heaviest item from the knapsack, thereby obtaining peak performance on the MOKPs.

3) Evaluation on MOTSPs: This section further validates the performance of the proposed LLMMOP against EOH

and hand-crafted MOEAs on the testing MOTSPs. Table IV tabulates the average Hypervolume (HV) values achieved by CTAEA, NSGA-II, AGEMOEA, SMSEMOA, the best 'Reproduction' obtained via LLMMOP, and the best 'Reproduction' gained via the proposed LLMMOP on testing MOTSP instances. The columns 'N.VAR' and 'Distribution' indicate the number of vertices and their distribution for each MOTSP instance, respectively. '$-$', '$\approx$', and '$+$' indicate that the method in the corresponding column performs statistically significantly worse, equivalent, or better compared to LLMMOP on the given test problems. As can be observed, these testing MOTSPs vary in both distribution and vertex scale, enabling the assessment of the generalization performance of the 'Reproduction' methods obtained via EOH and LLMMOP. From the table, we can observe that the 'Reproduction' method searched by LLMMOP obtained the best generalization performance on the testing MOTSPs with different vertex distributions, followed by the 'Reproduction' method obtained via EOH. This further verifies the advantages of the LLM-assisted 'Reproduction' evolution pipeline. Interestingly, while the best 'Reproduction' method identified by EOH achieved top performance on the validation MOTSPs, the method discovered by the proposed LLMMOP yielded superior or competitive results. This demonstrates that 'Reproduction' methods excelling on validation MOTSPs may not necessarily perform best on testing MOTSPs. This discrepancy between training and testing highlights the importance of considering this factor when applying these methods to real-world applications.

4) Efficiency of Algorithms: To evaluate the search efficiency of the 'Reproduction' methods generated via our proposed LLMMOP, we focus on the execution time (wall clock time) as a primary metric, which is measured in seconds for a fair comparison across the algorithms CTAEA, NSGA2, AGEMOEA, SMSEMOA, and the 'Reproduction' method obtained via EOH. The investigation was conducted using testing instances of CMOP, MOKP, and MOTSP.

The empirical evaluation of the search efficiency, as shown in Table V, reveals a significant variance in running time across the competing algorithms. NSGA2 demonstrated the best efficiency across all test problems. The best reproduction method generated by LLMMOP showed competitive performance on CMOP and MOKP but had the longest running time of 841 seconds on MOTSP, indicating the worst efficiency among the compared MOEAs. Despite this, its solution quality for MOTSPs was superior, demonstrating that its strategic, albeit time-intensive, approach results in high-quality solutions. This highlights the robustness of the proposed LLMMOP in handling different MOP categories.

C. Investigation on Different LLMs

In this section, we validate the performance of our proposed LLMMOP framework using various commercial LLMs, i.e., "gpt-4-1106-preview", "gpt-4o", "Gemini-1.5-pro" and "Claude-3-opus". For brevity, we denote them as GPT4, GPT4O, GEMINI, and CLAUDE, respectively. Moreover, the same temperature (0.5) is adopted for each LLM. Fig. 7 has illustrated the convergence curves obtained on CMOP validation dataset using these LLMs. As can be observed, different LLMs exhibit distinct behaviors within the proposed LLMMOP paradigm. Particularly, the proposed LLMMOP demonstrates consistent effectiveness across mainstream LLMs, including those with varying code-generation capabilities (e.g., GPT4 and GPT4O). Although these models may initially produce suboptimal solutions, the iterative search mechanism progressively improves the quality of generated code through systematic feedback. The proposed method effectively balances and leverages the diverse capabilities of LLMs, ultimately achieving competitive results across different models.

Moreover, CLAUDE demonstrates superior performance in generating innovative 'Reproduction' methods compared to other LLMs. For the optimal 'Reproduction' method generated by CLAUDE (with the complete code provided in the **Supplementary Materials**), hybrid strategies are employed in selection, crossover, and mutation, distinguishing it from traditional hand-crafted MOEAs. Specifically, intelligent selection is conducted according to the selection pressure, dynamically adjusted based on the search progress. Additionally, the adaptive crossover and mutation processes are tailored to the diversity and convergence stages of the current population. Different stages of diversity and convergence trigger distinct crossover operators and mutation schemes, enhancing the dynamic nature of the search strategy. These insights suggest an intuitive design principle: employing a hybrid search strategy and dynamically adapting search operators can lead to better optimization performance. While these strategies are innovative, they align with the intuitive approaches a human expert might consider, such as balancing exploration and exploitation and adapting strategies based on population characteristics. However, the numerous parameters configured in such a hybrid reproduction method highlight the significant impact of LLM-assisted design on MOEA performance.

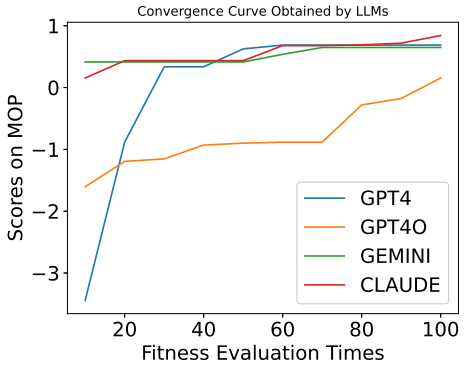


Fig. 7: Convergence curves gained by the proposed LLMMOP using different LLMs.

D. Ablation Study

In this section, the ablation studies are conducted to demonstrate the efficacy of the informative prompts, the proposed Pilot Run & Repair component, and the dynamic selection strategy for enhanced LLM-based crossover.

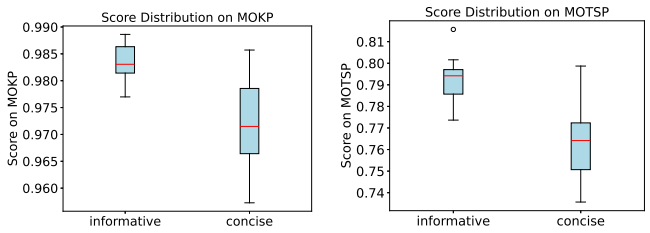
TABLE IV: Averaged HV values obtained by CTAEA, NSGA2, AGEMOEA, SMSEMOA, the best 'Reproduction' generated by EOH, and the best 'Reproduction' generated by the proposed LLMMOP on testing MOTSPs over 10 independent runs. Superior averaged results (larger HV values) are highlighted in bold font. The symbols $-$, $+$, and $\approx$ indicate that the corresponding algorithm achieved significantly worse, significantly better, and approximately equivalent results compared to LLMMOP, respectively.

Problem	N.VAR	Distribution	CTAEA	NSGA2	AGEMOEA	SMSEMOA	EOH	LLMMOP
MOTSP1	190	uniform	3.537e-01$-$	3.878e-01$-$	3.874e-01$-$	3.757e-01$-$	4.739e-01$\approx$	**4.814e-01**
MOTSP2	172	uniform	3.681e-01$-$	3.999e-01$-$	4.152e-01$-$	3.995e-01$-$	5.012e-01$\approx$	**5.105e-01**
MOTSP3	152	uniform	3.786e-01$-$	4.265e-01$-$	4.279e-01$-$	4.080e-01$-$	**5.364e-01**$\approx$	5.297e-01
MOTSP4	102	uniform	4.475e-01$-$	4.949e-01$-$	5.041e-01$-$	4.787e-01$-$	6.135e-01$\approx$	**6.177e-01**
MOTSP5	136	uniform	3.864e-01$-$	4.364e-01$-$	4.480e-01$-$	4.250e-01$-$	5.400e-01$-$	**5.600e-01**
MOTSP6	165	uniform	3.735e-01$-$	4.102e-01$-$	4.201e-01$-$	4.077e-01$-$	5.102e-01$\approx$	**5.134e-01**
MOTSP7	149	uniform	3.751e-01$-$	4.221e-01$-$	4.210e-01$-$	3.993e-01$-$	5.222e-01$-$	**5.334e-01**
MOTSP8	148	uniform	3.730e-01$-$	4.262e-01$-$	4.315e-01$-$	4.093e-01$-$	5.209e-01$\approx$	**5.283e-01**
MOTSP9	100	clustered	6.388e-01$-$	6.977e-01$-$	7.139e-01$-$	7.076e-01$-$	8.415e-01$\approx$	**8.452e-01**
MOTSP10	150	clustered	5.024e-01$-$	5.434e-01$-$	5.547e-01$-$	5.553e-01$-$	6.822e-01$-$	**7.133e-01**
MOTSP11	200	clustered	5.820e-01$-$	6.283e-01$-$	6.427e-01$-$	6.374e-01$-$	7.583e-01$\approx$	**7.672e-01**
MOTSP12	100	exponential	4.662e-01$-$	5.208e-01$-$	5.277e-01$-$	4.986e-01$-$	**6.408e-01**$\approx$	6.366e-01
MOTSP13	150	exponential	3.967e-01$-$	4.406e-01$-$	4.425e-01$-$	4.194e-01$-$	5.412e-01$\approx$	**5.453e-01**
MOTSP14	200	exponential	3.453e-01$-$	3.882e-01$-$	3.930e-01$-$	3.780e-01$-$	4.708e-01$\approx$	**4.717e-01**
MOTSP15	100	normal	4.200e-01$-$	4.802e-01$-$	4.818e-01$-$	4.642e-01$-$	**6.079e-01**$\approx$	6.057e-01
MOTSP16	150	normal	3.652e-01$-$	4.093e-01$-$	4.233e-01$-$	3.958e-01$-$	5.051e-01$\approx$	**5.185e-01**
MOTSP17	200	normal	3.195e-01$-$	3.529e-01$-$	3.601e-01$-$	3.473e-01$-$	4.457e-01$\approx$	**4.515e-01**
MOTSP18	100	poisson	4.420e-01$-$	4.981e-01$-$	4.944e-01$-$	4.735e-01$-$	6.008e-01$\approx$	**6.023e-01**
MOTSP19	150	poisson	3.660e-01$-$	4.065e-01$-$	4.157e-01$-$	3.916e-01$-$	5.038e-01$\approx$	**5.111e-01**
MOTSP20	200	poisson	3.155e-01$-$	3.581e-01$-$	3.703e-01$-$	3.456e-01$-$	4.402e-01$\approx$	**4.427e-01**
Averaged HV	-	-	4.108e-01	4.564e-01	4.638e-01	4.459e-01	5.628e-01	**5.693e-01**

TABLE V: Running time (wall clock time measured in seconds) obtained by CTAEA, NSGA2, AGEMOEA, SMSEMOA, the best 'Reproduction' obtained through EOH, and the best 'Reproduction' optimized through LLMMOP on testing MOPs. Smaller running time on each category is highlighted in bold font.

	CMOP	MOKP	MOTSP
CTAEA	114	215	247
NSGA2	**23**	**128**	**153**
AGEMOEA	50	144	168
SMSEMOA	1490	170	157
EOH	152	398	357
LLMMOP	32	131	841

1) In-depth Analysis of prompting schemes: To evaluate the effectiveness of our informative prompting scheme against the concise prompting scheme (refer to the **Supplementary Materials** that is designed based on [21]) in LLM-assisted program generation, we measured the quality of the generated 'Reproduction' populations using MOKP and MOTSP validation datasets, as illustrated in Fig. 8. For the MOKP, the informative prompting scheme yields a higher median score, indicating a potential capability in generating optimal 'Reproduction' methods. For the MOTSP, the differences are more pronounced. The informative prompting consistently outperforms the concise approach, with higher median scores and broader ranges, which reveal the insights of crafting effective prompts to enhance the quality of the generated 'Reproduction' methods. By providing informative prompts that include problem data and common steps to design the 'Reproduction', we can guide the LLM to develop high-quality 'Reproduction' methods more efficiently.



(a) Quality on MOKP (b) Quality on MOTSP

Fig. 8: Quality of initialized population of 'Reproduction' methods with different prompting schemes.

2) In-depth Analysis of Code Quality Among LLMs: As we discussed in the introduction, a common challenge of

TABLE VI: Quality of 'Reproduction' generated with different LLMs

LLM	Tests	Success	Failed	Repaired (%)
GPT4	20	9 + 9	2	81.8%
GPT4O	20	16 + 2	2	50.0%
GEMINI	20	16 + 3	1	75.0%
CLAUDE	20	10 + 7	3	70.0%

using LLMs in program design is the occurrence of execution errors in the generated code, which may prevent the generation of ideal designs. To validate the efficacy of the proposed Pilot Run & Repair component, we tested the code quality generated by GPT4, GPT4O, GEMINI and CLAUDE, using the initialization prompts. Particularly, each LLM was tasked to generate 20 'Reproduction' methods, which were then assessed for their operational viability. During the testing phase, any errors encountered triggered a repair process, which is designed to systematically address and resolve coding errors. The results have been summarized in Table VI, which presents a comparative analysis of the initial 'Reproduction' generation and subsequent repair success rates. As can be observed from, the 'Tests' column records the number of attempts made to generate 'Reproduction' methods. The "A + B" within the 'Success' column represents the total number of successfully generated 'Reproduction' methods, where "A" quantifies the instances without further interaction while "B" counts the successfully repaired instances via the proposed method. Moreover, 'Failed' tallies the number of unrepaired 'Reproduction' methods after one trial, while 'Repaired' reflects the proportion of successfully corrected 'Reproduction' methods through the repair process. The data presented in Table VI clearly demonstrates the superior initial success rates of GPT4O and GEMINI, with an impressive 80% of their generated methods being immediately deployable. Conversely, GPT4 shows a lower initial success but an outstanding repair success rate of 81.8%, underscoring the 'Pilot Run & Repair' component's capability to effectively rectify errors. CLAUDE presents a balanced profile, showcasing a consistent performance in both initialization and subsequent repairability.

3) Investigation on Dynamic Selection Strategy: The selectiopn strategy plays an important role in evolutionary progresses. To validate the performance of the developed dynamic selection strategy, tailored for LLM-based textual optimization tasks, we present the convergence curves of the proposed LLMMOP (denoted by 'GPT4') and its counterpart lacking this strategy (denoted by 'GPT4-no-dc'). As can be observed in Fig. 9, the 'GPT4' model, equipped with the dynamic selection strategy, shows a significant improvement in terms of scores gained on CMOPs, indicating an effective adaptation and optimization capability brought by the developed dynamic selection strategy. In contrast, despite a good initialization for the 'Reproduction' methods, the 'GPT4-no-dc' model's performance remains stagnant, again underscoring the necessity and impact of a dynamic selection strategy in the proposed LLMMOP.

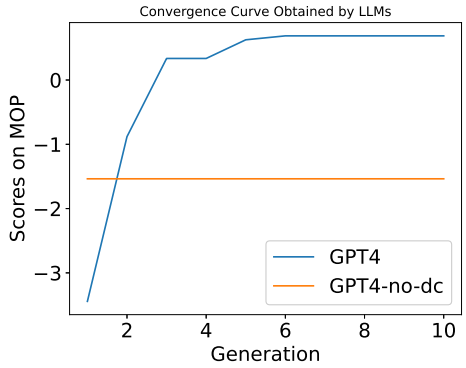


Fig. 9: Convergence curves gained by the proposed LLMMOP (denoted by 'GPT4') and its counterpart without dynamic selection strategy (denoted by GPT4-no-dc).

V. Conclusion

In this work, we have embarked on a journey to leverage the capabilities of LLMs to search novel MOEAs, specifically targeting multi-objective optimization. Our proposed framework, namely LLMMOP, introduces significant advancements in 'Reproduction' design, moving towards an autonomous methodology that reduces the need for expert intervention. The 'Pilot Run & Repair' mechanism within LLMMOP has been crucial in refining the evolution process of 'Reproduction' methods, enhancing their robustness and paving the way for fully autonomous programming. Empirical studies highlight the superiority of the proposed LLMMOP, demonstrating its superior performance over the SOTA LLM-assisted program search method and traditional human-crafted MOEAs.

The avenue of this field is promising, with the potential for LLMs to further revolutionize the optimization domain. In the future, we would like to explore how integrating expert knowledge with the proposed method can enhance search efficiency for black-box optimization problems. Furthermore, we would also like to investigate the capabilities of various LLMs to maximize their potential in enhancing the quality and consistency of generated algorithms.

VI. Acknowledgements

This work was supported in part by the National Key R&D Program of China under Grant 2022YFC3801700; and in part by the National Natural Science Foundation of China under Grant U21A20512; and in part by the Research Grants Council of the Hong Kong SAR under Grant C5052-23G, Grant PolyU 15229824, Grant PolyU 15218622, and Grant PolyU 15215623; and in part by the PolyU Start-up Fund for Research Assistant Professor (RAPs) through the Strategic Hiring Scheme under Grant P0045620.

References

[1] N. Gunantara, "A review of multi-objective optimization: Methods and its applications," *Cogent Engineering*, vol. 5, no. 1, p. 1502242, 2018.

[2] D. A. Van Veldhuizen, G. B. Lamont *et al.*, "Evolutionary computation and convergence to a pareto front," in *Late breaking papers at the genetic programming 1998 conference*. Citeseer, 1998, pp. 221–228.

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[4] Y. Zhou, W. Zhang, J. Kang, X. Zhang, and X. Wang, "A problem-specific non-dominated sorting genetic algorithm for supervised feature selection," *Information Sciences*, vol. 547, pp. 841–859, 2021.

[5] W. Deng, X. Zhang, Y. Zhou, Y. Liu, X. Zhou, H. Chen, and H. Zhao, "An enhanced fast non-dominated solution sorting genetic algorithm for multi-objective problems," *Information Sciences*, vol. 585, pp. 441–453, 2022.

[6] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on evolutionary computation*, vol. 11, no. 6, pp. 712–731, 2007.

[7] M. Asafuddoula, T. Ray, and R. Sarker, "A decomposition-based evolutionary algorithm for many objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 3, pp. 445–460, 2014.

[8] C. Dai, Y. Wang, and M. Ye, "A new multi-objective particle swarm optimization algorithm based on decomposition," *Information Sciences*, vol. 325, pp. 541–557, 2015.

[9] R. Tanabe and H. Ishibuchi, "An easy-to-use real-world multi-objective optimization problem suite," *Applied Soft Computing*, vol. 89, p. 106078, 2020.

[10] Y. Huang, W. Zhou, Y. Wang, M. Li, L. Feng, and K. C. Tan, "Evolutionary multitasking with centralized learning for large-scale combinatorial multi-objective optimization," *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2023.

[11] X. Xue, C. Yang, L. Feng, K. Zhang, L. Song, and K. C. Tan, "Solution transfer in evolutionary optimization: An empirical study on sequential transfer," *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2023.

[12] Y. Feng, L. Feng, S. Kwong, and K. C. Tan, "A multi-form evolutionary search paradigm for bi-level multi-objective optimization," *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2023.

[13] L. Belzner, T. Gabor, and M. Wirsing, "Large language model assisted software engineering: prospects, challenges, and a case study," in *International Conference on Bridging the Gap between AI and Reality*. Springer, 2023, pp. 355–374.

[14] X. Wu, S.-h. Wu, J. Wu, L. Feng, and K. C. Tan, "Evolutionary computation in the era of large language model: Survey and roadmap," *arXiv preprint arXiv:2401.10034*, 2024.

[15] V. Alto, *Modern Generative AI with ChatGPT and OpenAI Models: Leverage the capabilities of OpenAI's LLM for productivity and innovation with GPT3 and GPT4*. Packt Publishing Ltd, 2023.

[16] X. Chen, M. Lin, N. Schärli, and D. Zhou, "Teaching large language models to self-debug," *arXiv preprint arXiv:2304.05128*, 2023.

[17] A. Shypula, A. Madaan, Y. Zeng, U. Alon, J. Gardner, M. Hashemi, G. Neubig, P. Ranganathan, O. Bastani, and A. Yazdanbakhsh, "Learning performance-improving code edits," *arXiv preprint arXiv:2302.07867*, 2023.

[18] B. Romera-Paredes, M. Barekatain, A. Novikov, M. Balog, M. P. Kumar, E. Dupont, F. J. Ruiz, J. S. Ellenberg, P. Wang, O. Fawzi *et al.*, "Mathematical discoveries from program search with large language models," *Nature*, vol. 625, no. 7995, pp. 468–475, 2024.

[19] V. Liventsev, A. Grishina, A. Härmä, and L. Moonen, "Fully autonomous programming with large language models," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2023, pp. 1146–1155.

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

[21] F. Liu, T. Xialiang, M. Yuan, X. Lin, F. Luo, Z. Wang, Z. Lu, and Q. Zhang, "Evolution of heuristics: Towards efficient automatic algorithm design using large language model," in *Forty-first International Conference on Machine Learning*, 2024.

[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, pp. 1812–1820.

[23] H. Ye, J. Wang, Z. Cao, and G. Song, "Reevo: Large language models as hyper-heuristics with reflective evolution," *arXiv preprint arXiv:2402.01145*, 2024.

[24] R. Zhang, F. Liu, X. Lin, Z. Wang, Z. Lu, and Q. Zhang, "Understanding the importance of evolutionary search in automated heuristic design with large language models," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2024, pp. 185–202.

[25] M. Dunne, K. Schram, and S. Fischmeister, "Weaknesses in llm-generated code for embedded systems networking," in *2024 IEEE 24th International Conference on Software Quality, Reliability and Security (QRS)*, 2024, pp. 250–261.

[26] B. Huang, X. Wu, Y. Zhou, J. Wu, L. Feng, R. Cheng, and K. C. Tan, "Exploring the true potential: Evaluating the black-box optimization capability of large language models," *arXiv preprint arXiv:2404.06290*, 2024.

[27] Y. Huang, W. Zhang, L. Feng, X. Wu, and K. C. Tan, "How multimodal integration boost the performance of llm for optimization: Case study on capacitated vehicle routing problems," *arXiv preprint arXiv:2403.01757*, 2024.

[28] J. Lehman, J. Gordon, S. Jain, K. Ndousse, C. Yeh, and K. O. Stanley, "Evolution through large models," in *Handbook of Evolutionary Machine Learning*. Springer, 2023, pp. 331–366.

[29] M. Janga Reddy and D. Nagesh Kumar, "An efficient multi-objective optimization algorithm based on swarm intelligence for engineering design," *Engineering Optimization*, vol. 39, no. 1, pp. 49–68, 2007.

[30] S. C. Chiam, K. C. Tan, and A. Al Mamum, "Evolutionary multi-objective portfolio optimization in practical context," *International Journal of Automation and Computing*, vol. 5, pp. 67–80, 2008.

[31] Y. Wang, C. Chen, Y. Tao, Z. Wen, B. Chen, and H. Zhang, "A many-objective optimization of industrial environmental management using nsga-iii: A case of china's iron and steel industry," *Applied energy*, vol. 242, pp. 46–56, 2019.

[32] C. Bazgan, H. Hugot, and D. Vanderpooten, "Solving efficiently the 0–1 multi-objective knapsack problem," *Computers & Operations Research*, vol. 36, no. 1, pp. 260–279, 2009.

[33] A. P. Khandekar and A. Nargundkar, "Dynamic programming approach to solve real-world application of multi-objective unbounded knapsack problem," in *Intelligent Systems and Applications: Select Proceedings of ICISA 2022*. Springer, 2023, pp. 417–422.

[34] Y. Du, Z. Feng, and Y. Shen, "A mixed-factor evolutionary algorithm for multi-objective knapsack problem," in *International Conference on Intelligent Computing*. Springer, 2022, pp. 51–67.

[35] Z. Song, W. Luo, X. Lin, Z. She, and Q. Zhang, "On multiobjective knapsack problems with multiple decision makers," in *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2022, pp. 156–163.

[36] W. Peng, Q. Zhang, and H. Li, "Comparison between moea/d and nsga-ii on the multi-objective travelling salesman problem," in *Multi-objective memetic algorithms*. Springer, 2009, pp. 309–324.

[37] I. Khan, M. K. Maiti, and K. Basuli, "Multi-objective traveling salesman problem: an abc approach," *Applied Intelligence*, vol. 50, pp. 3942–3960, 2020.

[38] W. Li, "Solving multi-objective traveling salesman problem," in *The Traveling Salesman Problem: Optimization with the Attractor-Based Search System*. Springer, 2023, pp. 83–95.

[39] Y. Shuai, S. Yunfeng, and Z. Kai, "An effective method for solving multiple travelling salesman problem based on nsga-ii," *Systems Science & Control Engineering*, vol. 7, no. 2, pp. 108–116, 2019.

[40] T. Bäck and H.-P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary computation*, vol. 1, no. 1, pp. 1–23, 1993.

[41] T. Bartz-Beielstein, J. Branke, J. Mehnen, and O. Mersmann, "Evolutionary algorithms," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 4, no. 3, pp. 178–195, 2014.

[42] Z. He and G. G. Yen, "Many-objective evolutionary algorithms based on coordinated selection strategy," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 2, pp. 220–233, 2016.

[43] Q. Zhu and J. Luo, "Generative pre-trained transformer for design concept generation: an exploration," *Proceedings of the design society*, vol. 2, pp. 1825–1834, 2022.

[44] R. Luo, L. Sun, Y. Xia, T. Qin, S. Zhang, H. Poon, and T.-Y. Liu, "Biogpt: generative pre-trained transformer for biomedical text generation and mining," *Briefings in bioinformatics*, vol. 23, no. 6, p. bbac409, 2022.

[45] Y. Tian, X. Xiang, X. Zhang, R. Cheng, and Y. Jin, "Sampling reference points on the pareto fronts of benchmark multi-objective optimization problems," in *2018 IEEE congress on evolutionary computation (CEC)*. IEEE, 2018, pp. 1–6.

[46] J. Blank and K. Deb, "Pymoo: Multi-objective optimization in python," *Ieee access*, vol. 8, pp. 89 497–89 509, 2020.

[47] Y. Sun, G. G. Yen, and Z. Yi, "Igd indicator-based evolutionary algorithm for many-objective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 2, pp. 173–187, 2018.

[48] L. Bradstreet, *The hypervolume indicator for multi-objective optimisation: calculation and use*. University of Western Australia Perth, 2011.

[49] K. Li, R. Chen, G. Fu, and X. Yao, "Two-archive evolutionary algorithm for constrained multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 2, pp. 303–315, 2018.
[50] A. Panichella, "An adaptive evolutionary algorithm based on non-euclidean geometry for many-objective optimization," in *Proceedings of the genetic and evolutionary computation conference*, 2019, pp. 595–603.
[51] N. Beume, B. Naujoks, and M. Emmerich, "Sms-emoa: Multiobjective selection based on dominated hypervolume," *European Journal of Operational Research*, vol. 181, no. 3, pp. 1653–1669, 2007.
[52] K. Deb, R. B. Agrawal *et al.*, "Simulated binary crossover for continuous search space," *Complex systems*, vol. 9, no. 2, pp. 115–148, 1995.
[53] K. Deb, K. Sindhya, and T. Okabe, "Self-adaptive simulated binary crossover for real-parameter optimization," in *Proceedings of the 9th annual conference on genetic and evolutionary computation*, 2007, pp. 1187–1194.
[54] H. Ouerfelli and A. Dammak, "The genetic algorithm with two point crossover to solve the resource-constrained project scheduling problems," in *2013 5th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO)*. IEEE, 2013, pp. 1–4.
[55] J. Garnier, L. Kallel, and M. Schoenauer, "Rigorous hitting times for binary mutations," *Evolutionary Computation*, vol. 7, no. 2, pp. 173–203, 1999.
[56] I. Ono, M. Yamamura, and S. Kobayashi, "A genetic algorithm for job-shop scheduling problems using job-based order crossover," in *Proceedings of IEEE International Conference on evolutionary computation*. IEEE, 1996, pp. 547–552.
[57] M. B. Schmid and J. R. Roth, "Genetic methods for analysis and manipulation of inversion mutations in bacteria," *Genetics*, vol. 105, no. 3, pp. 517–537, 1983.

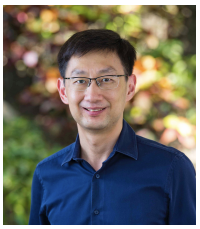
Yuxiao Huang received the B.E. degree and the Ph.D degree from College of Computer Science, Chongqing University, Chongqing, China, in 2018 and 2023, respectively. He is currently serving as a Postdoctoral Fellow with the Department of Data Science and Artificial Intelligence, The Hong Kong Polytechnic University, Hong Kong SAR. His current research interests include intelligent computing, evolutionary transfer optimization, combinatorial optimization, and GPU computing.

Shenghao Wu (Member, IEEE) received the B.S. and Ph.D. degrees in computer science and technology from the South China University of Technology, Guangzhou, China, in 2019 and 2023, respectively. He is currently a Postdoctoral Research Fellow with the Department of Data Science and Artificial Intelligence, The Hong Kong Polytechnic University, Hong Kong. His research interests mainly include computational intelligence, machine learning, and their applications in real-world problems.

Wenjie Zhang (Member, IEEE) received the B.Eng. degree from Huazhong University of Science and Technology, China, in 2015, and the Ph.D. degree from National University of Singapore (NUS), Singapore, in 2020. He was a visiting scholar in Stanford University in 2019. His current research interests include uncertainty quantification and deep learning in smart cities and smart grids.

Jibin Wu (Member, IEEE) received the B.E. and Ph.D degree in Electrical Engineering from National University of Singapore, Singapore in 2016 and 2020, respectively. Dr. Wu is currently an Assistant Professor in the Department of Data Science and Artificial Intelligence, The Hong Kong Polytechnic University. His research interests broadly include brain-inspired artificial intelligence, neuromorphic computing, computational audition, speech processing, and machine learning. Dr. Wu has published over 40 papers in prestigious conferences and journals in artificial intelligence and speech processing, including NeurIPS, ICLR, AAAI, TPAMI, TNNLS, TASLP, and IEEE JSTSP. He is currently serving as the Associate Editors for IEEE Transactions on Neural Networks and Learning Systems and IEEE Transactions on Cognitive and Developmental Systems.

Liang Feng received the Ph.D. degree from the School of Computer Engineering, Nanyang Technological University, Singapore, in 2014. He is currently a Professor with the College of Computer Science, Chongqing University, Chongqing, China. His research interests mainly include computational and artificial intelligence, machine learning, multi-agent system as well as transfer learning and optimization. Prof. Feng has been honored with the 2019 IEEE TEVC Outstanding Paper Award, the 2023 IEEE TETCI Outstanding Paper Award, and the 2024 IEEE CIM Outstanding Paper Award. He is Associate Editor of the IEEE Transactionss on Evolutionary Computation, IEEE Transactions on Emerging Topics in Computational Intelligence, IEEE Computational Intelligence Magazine, and IEEE Transactions on Cognitive and Developmental Systems. He is also the Founding Chair of the IEEE CIS Intelligent Systems Applications Technical Committee Task Force on Transfer Learning & Transfer Optimization.



Kay Chen Tan (Fellow, IEEE) received the B.Eng. degree (with First-Class Hons.) and the Ph.D. degree from the University of Glasgow, Glasgow, U.K., in 1994 and 1997, respectively. He is currently the Head and Chair Professor of Computational Intelligence with the Department of Data Science and Artificial Intelligence, The Hong Kong Polytechnic University, Hong Kong. Prof. Tan was the Editor-in-Chief of IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION from 2015 to 2020, and IEEE Computational Intelligence Magazine from 2010 to 2013, and currently serves as an Editorial Board member of 10+ journals. He served as the Vice-President (Publications) of the IEEE Computational Intelligence Society, USA, from 2021 to 2024, and currently serves as an Honorary Professor with University of Nottingham in U.K., and the Chief Co-Editor of Springer Book Series on Machine Learning: Foundations, Methodologies, and Applications.