

# LAIGA: LLM-assisted Interactive Genetic Algorithm for Preference-incorporated Multi-objective Optimization

Zhiwei Zhang and Chi Jiao

**Abstract**—Large Language models (LLMs) have demonstrated strong logical reasoning capabilities in evolutionary computation, particularly in generating new solutions, automatically designing algorithms, and evaluating the quality of solutions. Interactive evolutionary computation (IEC) is an approach that allows the decision maker (DM) to intervene in the algorithm execution process to influence the direction of evolution. In traditional interactive evolutionary processes, it is usually necessary to model the DM's preferences so that the algorithm can understand these preferences and guide the iterative process on behalf of the DM, thereby reducing the DM's workload. Although many studies have adopted different methods to construct preference models based on various types of preference information, whether LLM can serve as an auxiliary model in IEC remains unexplored. Therefore, this paper proposes a new algorithm, LAIGA, which is based on the genetic algorithm (GA) and integrates an LLM-assisted module to allow the DM to provide preferences in natural language form. The LLM then assists the algorithm in automatically adjusting the weight vector of the preference model. Finally, we designed experiments to evaluate the algorithm's performance on the DTLZ2 problem in dimensions 6–12 under three different types of DM preference models, confirming its effectiveness. The experimental results show that the proposed method achieves performance comparable to mainstream approaches, while reducing the number of DM interactions and cognitive burden, and ensuring that the final solution obtained is sufficiently satisfactory to the DM. This work provides new insights into the application of LLM in the field of IEC.

**Index Terms**—Interactive Evolutionary Computation, Multi-Objective Optimization, Large Language Model.

## I. INTRODUCTION

**R**EAL-WORLD optimization problems often involve multiple conflicting objective functions, which are referred to as multi-objective optimization problems (MOPs). In the process of solving multi-objective optimization problems, we obtain a number of Pareto frontier solutions rather than a single solution set. Mathematically, without additional information, each solution in the theoretical Pareto frontier solution set is equally good. However, from the subjective judgment of the Decision-Maker (DM), there are still differences in the superiority and inferiority among these frontier solutions. This makes it necessary for the DM to intervene in the optimization

process to obtain the most satisfactory solutions from a large number of frontier solutions, which are called Most Preferred Solutions (MPS).

Evolutionary computation (EC), as a core technology in the field of intelligent optimization, draws its design inspiration from the basic principles of biological evolution in nature. It solves complex optimization problems by simulating mechanisms such as heredity, mutation, and selection. EC encodes the solutions to a problem as individuals, which form a population. During the iteration process, new individuals are generated through genetic operations such as crossover and mutation, and the quality of individuals is evaluated based on the fitness function. Excellent individuals are retained to enter the next generation, thereby enabling the population to evolve towards a better solution space. Beyond continuous benchmark problems, evolutionary and population-based metaheuristics have been applied to a broad range of discrete engineering optimization tasks, including multi-objective disassembly line balancing [40], [42] and high-dimensional feature selection [41]. Learning-based and multi-objective combinatorial formulations have likewise proven effective, spanning knowledge-based hyper-heuristics for distributed scheduling [45], stochastic multi-objective optimization under operational uncertainty [46], and reinforcement-learning-based line balancing [47].

On the basis of EC, Interactive Evolutionary Computation (IEC) has been developed. As shown in Fig.1, IEC can incorporate the user's subjective preferences to make the algorithm evolve in the direction that interests the DM. Since the subjective preferences in IEC are relative and discrete, they can guide the search process of the evolutionary algorithm. This makes IEC particularly suitable for tasks where relationships are difficult to quantify mathematically or performance is hard to evaluate by computers. As a result, IEC has been widely applied in fields such as artistic design [1], [2] and engineering optimization [3]. Over the past 20 years, scholars' research on IEC has generally focused on areas such as the application of IEC in specific fields, enriching the interactivity of IEC, reducing the DM's fatigue during interaction, and combining IEC with multi-objective optimization.

Despite the proven effectiveness of IEC through its wide applications in relevant fields, there are still some limitations that need to be addressed. For instance, existing IEC methods usually require DMs to have clear cognitive awareness of their preferences and professional domain knowledge. In particular, DMs face significant cognitive challenges when adjusting reference points, weight vector of value functions, and other such processes [4]. In traditional interaction modes, whether DMs manually adjust the evolutionary direction to align with

Copyright: ©2026 by the authors. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

Manuscript received August 28, 2026; revised September 2, 2026; accepted October 7, 2026. This article was recommended for publication by Associate Editor Shujin Qin upon evaluation of the reviewers' comments.

Z. Zhang is with the Information and Control Engineering, Liaoning Petrochemical University, Fushun 113001, China (e-mail: zhangzw@lnpu.edu.cn).

C. Jiao is with the Artificial Intelligence and Software, Liaoning Petrochemical University, Fushun 113001, China (e-mail: jiaochi@stu.lnpu.edu.cn).

Corresponding author: Chi Jiao.

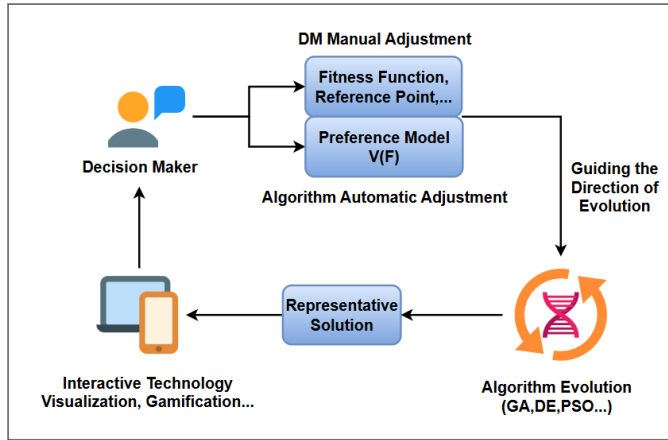


Fig. 1. DM Interacts with Algorithms.

their intentions, or DMs screen numerous offspring solutions while the algorithm automatically selects the direction, DMs need to have a clear understanding of their own subjective intentions and the complex mathematical relationships within the solution set. Otherwise, DM fatigue will seriously affect the algorithm's iteration in the desired direction. Although studies have long been conducted to reduce the burden through visual interfaces [5], optimization of interaction frequency [6], guess the hidden objectives [7], and provision of multiple preference processing methods with the freedom for DMs to choose [8], few studies have considered leveraging the advantages of natural language interaction. Cinelli et al. also mentioned that future research might include more types of preference information, including various linguistic terms [9].

Large Language Models (LLMs) have demonstrated astonishing capabilities in various fields. Due to their ability to absorb and process massive amounts of information, LLMs can effectively capture human knowledge, endowing them with strong reasoning and decision-making abilities. This also enables them to possess data processing capabilities that are hard for the human brain to surpass. In recent years, scholars have found that LLMs are not only adept at solving natural language processing problems but also show great potential in assisting optimization algorithms, especially evolutionary algorithms (EAs) [10]–[14], [43], [44], [48].

Inspired by the existing research mentioned above, this work is the first to incorporate LLM as an auxiliary model into IEC. Specifically, we use GA as the base algorithm and simultaneously set up three value functions (also referred to as utility functions) for quantifying the preferences of DMs with different styles. These serve as the DM's preference model to guide the evolutionary direction of GA. Subsequently, we construct a surrogate model framework that integrates two different types of preference information and submits them to the LLM for processing. This framework records the representative solutions generated during the algorithm iteration process and the DM's evaluations of these representative solutions. Finally, the LLM captures the DM's preference information based on the integrated information, thereby assisting the algorithm in preference modeling. In this way, at the algorithm

level, we transform the multiple objectives of the optimization problem into a single DM preference objective for solution.

Finally, we conducted comparative experiments to verify the performance of the proposed algorithm on the test optimization problems. The results indicate that LLM holds great potential in assisting GA for interactive evolutionary computation, as it can ensure the DM's satisfaction with the final solution under a limited number of DM interactions. The main contributions of this work are as follows:

- 1) A surrogate model based on LLM is proposed to parse two different types of preference information provided by users. It leverages the reasoning capability of LLM to analyze the semantic content in the preference information for weight vector prediction, without the need for training.
- 2) Based on genetic algorithm (GA), a new algorithmic framework named LAIGA is proposed by introducing the LLM-based surrogate model (including two variants for handling different types of preference information: D-LAIGA and S-LAIGA). This innovative framework allows DMs to use natural language either completely or partially when interacting with the algorithm. By doing so, it can reduce the DM's cognitive burden while lowering the algorithm's implementation difficulty and solution load.
- 3) To verify the effectiveness of LAIGA, our experiments constructed simulated DMs with three different preference models respectively to provide two distinct types of preference information. Meanwhile, the impacts of unclear expression and decision-making errors of real-world DMs are considered to enhance the practical relevance of the experiments.

## II. RELATED WORK

### A. Interaction mode between DM and algorithm

Interaction modes determine the collaboration efficiency between DMs and algorithms. Existing modes can be categorized into three types: a priori, a posteriori, and interactive. A priori methods require DMs to preset preferences before optimization (such as providing weight vector or preset reference points), but DMs often struggle to make reasonable judgments without information about solutions. A posteriori methods allow DMs to select the optimal solution from the finally generated Pareto frontier, yet they need to bear the cost of evaluating a large number of solutions. Interactive methods, on the other hand, alternate between solution generation and preference feedback during iterations (e.g., IEMO/D [15]). Through the cycle of "interaction-adjustment-evolution", they gradually focus on the regions of interest to DMs, and significantly reduce the cognitive load required for each interaction by decreasing the number of solutions presented to DMs.

Although interactive evolutionary algorithms allow DMs to gradually clarify their "intentions", traditional interaction methods mostly rely on structured inputs (such as adjustment of reference points or weights, scoring or comparison of solutions), requiring DMs to be sufficiently sensitive to the numerical values of objectives. In some complex problems

with multiple objectives, this may lead to DMs' vacillating decisions and thus result in fatigue [16]. While many researchers have attempted means such as visualization and gamification to reduce interaction burden, they have not yet fully utilized the flexibility of natural language. As the most natural form of human expression, natural language can intuitively convey intuitive and vague preferences, which, however, are difficult to directly map to the quantitative parameters required by algorithms. In recent years, the logical reasoning ability demonstrated by LLMs has shown great potential, which also provides a simple and easy-to-use solution for the mapping from semantics to quantitative parameters and offers important insights for this research.

### B. Preference Information Form in IEC

In the field of multi-objective optimization, IEC guides the evolutionary direction by incorporating the DM's subjective preferences, and the effective acquisition and expression of preference information constitute its core link. In existing research, the forms of preference information in IEC are mainly divided into two categories: direct preferences and indirect preferences.

Direct preferences require DMs to explicitly specify parameter values that reflect their expectations, such as reference points, weight vector, or trade-off relationships. Early preference-driven evolutionary multi-objective optimization algorithms (EMOAs) mostly relied on such forms. For example, MOGA guides the search through reference points (composed of expected objective values) [17]; R-NSGA-II [18] and R-NSGA3 [19] support multiple reference points to cover a wider range of preference regions; while RD-NSGA-II [20] and LBS-NSGA-II [21] define search directions through reference directions (vectors starting from reference points). As a typical form of direct preferences, weights can either represent the importance coefficients of objectives (such as the precise weights in MOEA-PROM [22]) or reflect the substitution rate between objectives (such as the weights used for weighted Euclidean distance calculation in R-NSGA-II). Some methods also support interval weights (such as MOEA/D/O [23]).

In addition, although some studies have utilized linguistic variables to describe direct opinions about solutions [24] or the dominance relationships between different solutions [25] as preference information for optimization, these methods essentially perform optimization based on predefined mapping rules from linguistic variables to weights. While asking the DM to select preset linguistic variables to evaluate solutions reduces their cognitive burden to some extent, it also makes it difficult to handle complex and varied human language inputs.

The limitations of direct preference information are very obvious. Even if the DM is sufficiently professional, it is difficult for them to correctly express the ideal point to the algorithm at the beginning. More commonly, DMs prefer to express the differences between a solution and their ideal solution after obtaining some representative solutions.

Indirect preferences are obtained indirectly through the DM's overall judgment on solutions. Common forms include pairwise comparisons (e.g., NEMO [26]), selecting the

most/least preferred solutions from a candidate set (e.g., iT-DEA, iMOEA/D [27]), and conducting pairwise comparisons, ranking, or classification of solutions (e.g., XIMEA-DRSA [28], PINSGA2 [29]). Since direct preferences are highly dependent on the DM's professionalism, while indirect preferences are more in line with the DM's cognitive habits and can significantly reduce cognitive burden, indirect preferences have become a research hotspot in recent years. However, converting indirect preference information into parameters recognizable by algorithms (such as weights) still faces challenges: although algorithms using indirect preferences do not require DMs to provide explicit optimization directions, the information provided by indirect preferences is a kind of complex implicit information. This also leads to the need for more interaction times and more complex processing mechanisms for algorithms to obtain sufficient information to accurately fit the DM's preferences, which poses challenges in algorithm design and the DM's interaction burden.

Table I presents several existing algorithms for handling different types of preference information, along with our proposed algorithm.

Considering the high cognitive burden of direct preference information and the complexity of preference modeling for indirect preference information, this work employs LLMs as an auxiliary model for interactions between DMs and algorithms. Leveraging the language processing capabilities and logical reasoning of LLMs, DMs are allowed to provide two new forms of preference information: 1) Using pure natural language to vaguely describe their views on different objectives. 2) Ranking solutions while using natural language to vaguely describe the degree of difference between different solutions.

Essentially, these two types of preference information still belong to direct preference and indirect preference respectively. However, integrating LLM to incorporate semantic information enables the algorithm to obtain more information from less evaluation time and representative solutions, thereby significantly reducing the user's interaction burden while also lowering the implementation difficulty of the algorithm.

### C. Selection of Preference Models

Preference models serve as a bridge connecting DMs' preferences and solution evaluation in IEC. Existing models mainly fall into three categories: scoring functions, binary relations, and decision rules. Among them, scoring functions generate a comprehensive score by aggregating objective values and are widely used in evolutionary algorithms due to their simplicity [30].

Value functions are a type of scoring function, with their main forms being additive value functions featuring linear, piecewise linear, or generally monotonic marginal per-objective functions. These functions capture the relative quality of solutions and compare objective values without involving a predefined scale for the intensity of DMs' preferences. In addition, there are other types of value functions such as Choquet integrals, Gaussian functions, quasi-convex functions, and polynomial functions. They differ in terms of

TABLE I  
RESEARCH CONSIDERING DIFFERENT PREFERENCE INFORMATION

| Algorithm                         | Preference Information Provided                                     | Preference Type |
|-----------------------------------|---------------------------------------------------------------------|-----------------|
| MOGA [17]                         | Reference point                                                     | Direct          |
| R-NSGA-II [18], R-NSGA3 [19]      | Multiple reference points                                           | Direct          |
| RD-NSGA-II [20], LBS-NSGA-II [21] | Reference search directions                                         | Direct          |
| MOEA-PROM [22]                    | Exact weights                                                       | Direct          |
| MOEA/D/O [23]                     | Fuzzy interval weights                                              | Direct          |
| FPMOM [25]                        | Evaluate the solution using predefined semantic variables           | Direct          |
| FLMOEA [24]                       | Compare the solutions using predefined semantic dominance relations | Indirect        |
| NEMO [26]                         | Pairwise comparison of solutions                                    | Indirect        |
| iMOEA/D [27]                      | Selecting the most preferred or despised solutions                  | Indirect        |
| XIMEA-DRSA [28]                   | Ranking or classifying solutions                                    | Indirect        |
| PI-NSGA2 [29]                     | Ranking or pairwise comparison solutions                            | Indirect        |
| LAIGA                             | Direct evaluation or ranking of solutions using natural language    | Direct/Indirect |

distinguishing performance perceptions within and between objectives, compensation levels, or representing interactions between objectives [31].

Among them, two mainstream value functions are particularly suitable for initial integration into the algorithm as preference models, namely the simple additive value function and the Chebyshev value function, whose mathematical formulas are shown in (1) and (2). These two value functions play an important role in simulating DMs [32] representing two different styles of DMs respectively: one with a utilitarian tendency that emphasizes “the comprehensive performance of all objectives” and the other with an averaging tendency that emphasizes “all objectives being as good as possible”. In addition, various forms of polynomial functions have also been employed in a wide range of studies.

$$V_{addition}(f(x)) = \sum_{i=1}^k w_i(f_i(x)) \quad (1)$$

$$V_{chebyshev}(f(x)) = \max_{i=1, \dots, k} (w_i |f_i(x) - z_i^*|) \quad (2)$$

Furthermore, study [33] have pointed out that the performance of interactive algorithms is highly correlated with the satisfaction degree of DMs. However, due to the subjectivity of human evaluations and the complex differences between individuals, having real-world DMs evaluate the merits of solutions provided by algorithms will lead to a loss of fairness in experiments and make them irreproducible. In most studies, the simulated DMs used are represented by a simple linear or non-linear function. Such idealized simulated DMs are equivalent to relaxing the requirements for the algorithm, making it lose some practical significance to a certain extent. Therefore, it is necessary to add certain disturbances to the simulated DMs to simulate real-world scenarios.

#### D. The Potential of LLM in Evolutionary Computation

Genetic Algorithm (GA) is a classic evolutionary algorithm. To this day, it remains one of the most widely used evolutionary algorithms, and a series of algorithms with excellent performance have been developed based on it, such as the most widely recognized NSGA2 [34]. IGA integrates human subjective evaluation by introducing a fitness function that reflects the subjective preferences of human evaluators in

GA. By adjusting the fitness function with the preference information provided by users, the algorithm can evolve in the direction that conforms to users’ preferences [35]. However, in the practical application of IGA, the fitting of the preference function has long been confronted with two major challenges. On the one hand, traditional methods usually rely on using additional optimizers to process preference information, which not only increases the complexity of algorithm design and application but also incurs extra computational costs. On the other hand, the traditional preference function fitting process mainly depends on structured data (such as rankings or scores) provided by users, making it difficult to directly identify and integrate unstructured preference information expressed in natural language. This limits the flexibility and user-friendliness of IGA in human-computer interaction.

In recent years, the application of LLMs in evolutionary computation has shown great potential. Their strong capabilities in semantic understanding, data processing, and logical reasoning can support complex tasks, such as algorithm design [10], solution generation [11], [12], simulation of crossover operators [13], and evaluation of solutions to replace expensive objective functions [14].

These pioneering studies have verified the feasibility of integrating LLMs into the EC framework and inspired us to explore their application in the IEC framework. If we can fully leverage the technical advantages of LLMs in low-cost function approximation and natural language processing, it is expected to break through the limitations of traditional IGA methods and further lower the threshold for their practical application.

### III. LLM-ASSISTED IEC ALGORITHM

This section first introduces two basic algorithm frameworks of LAIGA, namely Direct Language Evaluation-LAIGA (D-LAIGA) and Sorting and Language Evaluation-LAIGA (S-LAIGA). Both frameworks are based on GA and use LLM-assisted algorithms to guide the evolutionary direction. The difference lies in that the two frameworks are respectively used to process two different types of preference information. Subsequently, it will elaborate on how to use prompts to enable the LLM to map users’ preference information to parameters usable by the algorithm, including specific implementation details. Finally, the algorithm flow of IGA is redesigned to facilitate the integration of LLM as a surrogate model into it.

### A. Related Definitions

To facilitate understanding, we provide the explanations of the terms that will be mentioned later in advance as follows:

**Definition 1 (Real Value Function).** The real value function is defined as

$$V^*(x) = g(f_1(x), f_2(x), \dots, f_m(x); w^*),$$

where  $f_i(x)$  is a weighted aggregation function of multiple objectives maps the objective vector into a scalar score.  $V^*(x)$  reflects the DM's intrinsic preference for solution  $x$ . It is assumed to be monotonic with respect to each objective. In practice, the complete form of  $V^*(x)$  is unobservable to the algorithm, and it is used only as a conceptual model of the DM's true judgments.

**Definition 2 (Value Function).** A value function is defined as

$$V(x) = \hat{g}(f_1(x), f_2(x), \dots, f_m(x); w),$$

where  $\hat{g}(\cdot)$  is an explicit or learned approximation of the true aggregation function  $g(\cdot)$ . The overall form of  $V(x)$  is predefined, while its parameters  $w$  are unspecified and subject to learning or adaptation. The scalar  $V(x)$  is called the *utility* or *preference degree* of solution  $x$ . The goal of the algorithm is to construct  $V(x)$  such that  $V(x) \approx V^*(x)$  for all  $x$  of interest.

**Definition 3 (Weight Vector).** A weight vector is denoted as  $\mathbf{w} = (w_1, \dots, w_m)$ , encoding the relative importance assigned to each objective.

In this work, we distinguish between:

- $w^* = (w_1^*, \dots, w_m^*)$ : the *true weight vector* embedded in the DM's true preference function  $V^*$ . It reflects the genuine subjective importance the DM assigns to each objective. In this work, we assume that  $\mathbf{w}^* = [0.3, 0.5, 0.2]$ .
- $w = (w_1, \dots, w_m)$ : the *estimated weight vector* employed in the value function  $V$  constructed by the algorithm. It is iteratively adjusted (possibly via LLM assistance) to approximate  $\mathbf{w}^*$  as closely as possible.

**Definition 4 (Most Preferred Solution, MPS).** The *Most Preferred Solution (MPS)* is the feasible solution that maximizes the DM's true preference function. Formally,

$$x^{\text{MPS}} = \arg \max_{x \in X} V^*(x).$$

**Definition 5 (Ideal Point).** The *ideal point* represents the outcome that the Decision Maker (DM) hopes a solution can achieve. In general, this point may be either attainable or unattainable within the feasible space. In this work, the ideal point is defined as the objective vector corresponding to the maximum of the DM's true preference function:

$$\mathbf{z}^{\text{ideal}} = f(x^{\text{MPS}}), \quad x^{\text{MPS}} = \arg \max_{x \in X} V^*(x).$$

The DM cannot directly provide this point to the algorithm; instead, the algorithm must gradually infer it through interactive evaluations with the DM.

### B. Algorithm Framework

The algorithm flow of LAIGA is illustrated in Figure 2, where part of the flow varies depending on the different types of preferences provided by the DM. This section will introduce the two variants of this framework respectively to address different types of DM preference information.

1) *The Algorithm Framework of D-LAIGA:* When this framework acts as D-LAIGA and is applied to process direct preference information, the algorithm process starts from the initialization phase, where the initial parameters and conditions for algorithm operations are set. Subsequently, a parent population is generated to serve as the starting group for evolutionary iterations. After the generation of the parent population, the next-generation population is formed through operations such as reproduction and mutation, as well as selection based on the fitness function. Meanwhile, elite solutions are screened and archived. This process iterates continuously until the preset termination condition is met, after which the MPS is output.

In this scenario, the main task of LLM is to interact with the DM and continuously adjust the weight vector of the fitness function, enabling the function to gradually adapt to the DM's real preference model. After the generation of each population, the DM will evaluate the individual with the highest fitness in the current elite archive using pure natural language. LLM will comprehensively consider the DM's current evaluation and historical evaluation data, and finally adjust the fitness function appropriately to align with the DM's preferences for each objective. In this work, we set the fitness function directly as  $V(x)$  purely for the sake of experimental simplicity, although more sophisticated alternatives could be employed in practice.

2) *The Algorithm Framework of S-LAIGA:* When the framework serves as S-LAIGA and is used to process indirect preference information, operations such as initialization, reproduction, and mutation are basically the same as those in D-LAIGA. The difference lies in that S-LAIGA does not update the fitness function after each interaction with the DM. Instead, it maintains a weight pool during the iteration process, and only when the DM intends to stop the interaction will the optimal weight be selected from the weight pool to update the fitness function and complete the subsequent iterations.

In this case, the main task of LLM is to interact with the DM and infer the weight vector of the fitness function, enabling it to fit the real preference model of the DM. After generating each generation of the population, the algorithm screens out different solutions from the current population, submits them to the DM for ranking, and attaches natural language descriptions. Then, LLM will infer the weight vector based on the DM's ranking and descriptions, and the inferred weights will be temporarily stored in the weight pool for subsequent weight screening.

### C. Using LLM as surrogate model

To enable the LLM to function smoothly as an intermediary between the DM and the GA, a series of data processing steps are required: constructing prompt templates, data preprocessing, prompt concatenation, interacting with the LLM, and data postprocessing. In addition, for the two different types of

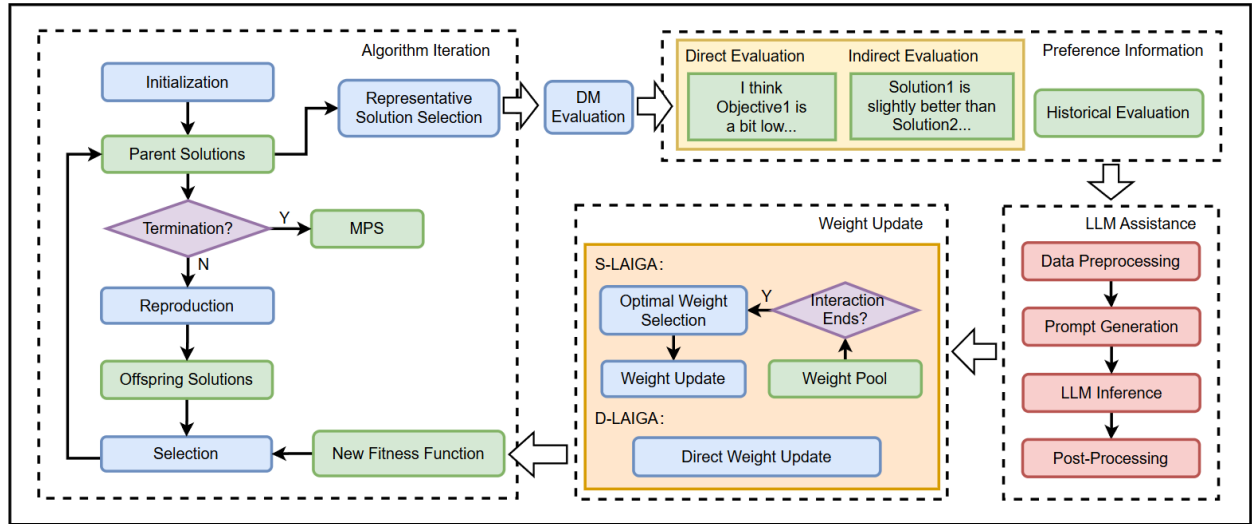


Fig. 2. Algorithm Framework of LAIGA.

preference information considered in this work, adjustments need to be made in different aspects during the processing, which will be elaborated in the following sections.

1) *Prompt Generation*: To ensure that LLMs can correctly parse the DM's evaluation information and make decisions by comprehensively considering historical evaluation data, it is necessary to preprocess the evaluation data and construct a set of clear task description templates. In real-world optimization problems, the measurement scales between different objectives usually vary significantly. Since LLMs are unaware of the value ranges of these objective values, this will affect their judgment of evaluation data and weights. Therefore, normalization is required to scale all objective values to the range [0,1], and the normalization formula is shown in (3).

$$f'_i = \frac{f_i}{f_{i,max}} \quad i = 1, \dots, n \quad (3)$$

It is worth noting that the normalization process is only applied to the input data to be presented to the LLM. However, what the DM sees during evaluation is the original data, because DMs are generally not adept at processing normalized values and are more concerned with the original objective values.

Subsequently, we constructed a prompt template to integrate all the information required by the LLM. When using the LLM to process direct preferences described in pure natural language, we integrated the following information: normalized objective values; corresponding DM evaluation data; current weights; and historical evaluation and weight adjustment records. The final prompt template adapted to this task is shown in Fig.3, which mainly includes the following parts: task overview, input and output constraints, adjustment rule constraints, historical evaluation data, and current evaluation data. We require the LLM to output in a specified format to facilitate subsequent extraction and processing of the adjusted weight information. By using this prompt template, the LLM will attempt to quantitatively analyze the DM's preference tendencies based on historical information and the semantic intensity of the DM's expressions.

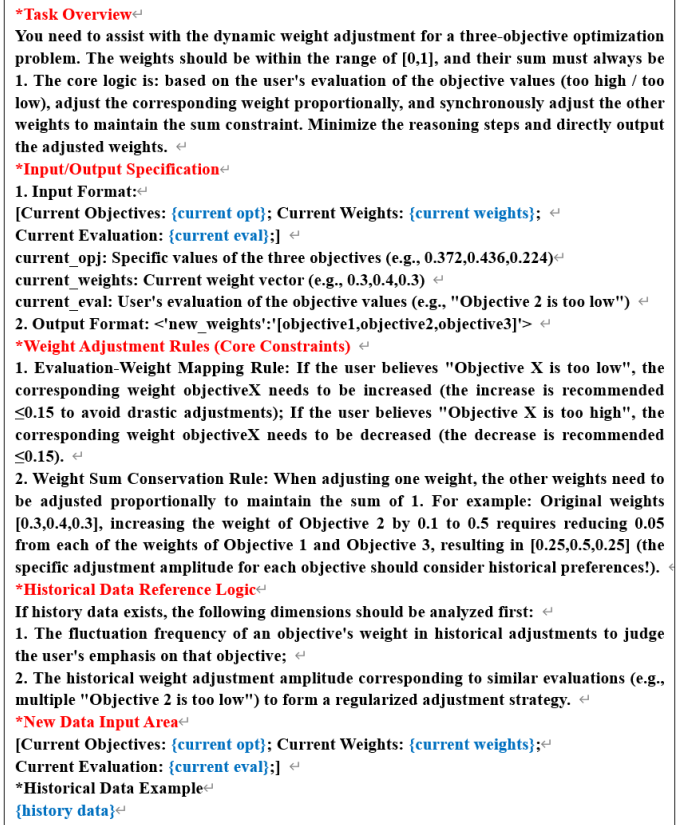


Fig. 3. Prompt Template for D-LAIGA.

When using the LLM to process indirect preferences expressed through solution ranking combined with linguistic descriptions, we integrate the following information: the user's ranking of each group of solutions along with linguistic descriptions of the degree of difference; current weights; and historical ranking records. The final prompt template adapted to this task is shown in Fig.4, which mainly includes the following parts: task overview, core background, weight

**\*Task Overview:**<sup>⌚</sup>  
Please adjust the values of the initial weights in the value function and infer the values of the unknown weights based on the user's preference rankings and linguistic evaluations of the solutions in the current and historical data.<sup>⌚</sup>

**\*Core Background:**<sup>⌚</sup>  
1. The optimization problem includes 3 objective values: F1, F2, F3.<sup>⌚</sup>  
2. It is assumed that users evaluate the quality of solutions through a value function: {value function form}.<sup>⌚</sup>  
3. Known conditions:<sup>⌚</sup>  
1)The known weight {w1} = {w1 value} is a fuzzy value and needs to be adjusted by you;<sup>⌚</sup>  
2)The weight {w2} and weight {w3} are unknown values and need to be inferred by you;<sup>⌚</sup>  
3)It must be satisfied that weight1 + weight2 + weight3 = 1, and all weights are ≥ 0.<sup>⌚</sup>

**\*Weight Inference Rules(Core Constraints):**<sup>⌚</sup>  
1. The weights you infer must satisfy that the ranking of all solutions' V values calculated based on these weights is completely consistent with the ranking given by the user. That is, for each group of solutions, the ranking of values calculated according to your weights from high to low must be exactly the same as the user's evaluation of "1st place → 2nd place → 3rd place".<sup>⌚</sup>  
2. Please ensure that your weights can accurately reproduce the preference order shown by the user, which is the only criterion for inferring the weights.<sup>⌚</sup>  
3. Your weights should also try to conform to the rankings in the historical data, but this is not mandatory.<sup>⌚</sup>  
4. When adjusting the specific values of the weights, you can refer to the user's linguistic descriptions of the degree of superiority or inferiority of the solutions (such as "much better" or "not much different") to make the difference in values match the intensity of the evaluation.<sup>⌚</sup>

**\*Historical Evaluation Information (For Reference):**<sup>⌚</sup>  
{historical evaluation data}<sup>⌚</sup>

**\*Current Solution Rankings (From Best to Worst):**<sup>⌚</sup>  
First group of solutions(F2 values are close, F3 values differ significantly): {rank of group1} User's evaluation: {evaluation1}<sup>⌚</sup>  
Second group of solutions\*(F3 values are close, F2 values differ significantly): {rank of group2} User's evaluation: {evaluation2}<sup>⌚</sup>

**\*Output Requirements:**<sup>⌚</sup>  
Only return the weight results in the specified format, no additional explanations are needed. Format: <new\_weights':[w1, w2, w3]> <sup>⌚</sup>

Fig. 4. Prompt Template for S-LAIGA.

inference rules, current ranking data, historical ranking data, and output constraints.

By providing ranking information of two sets of solutions with different directions of difference, this prompt template enables the LLM to more easily find and verify a suitable set of parameters for fitting. Meanwhile, the intensity of the DM's semantic evaluations will help the LLM more quickly determine the magnitude of the parameters.

#### D. Interaction with LLM

Once the prompt template is determined, we can use an algorithm to cyclically process the information related to the DM's evaluations, splice it into the template to form a prompt, then interact with the LLM to obtain response information, and perform post-processing on the response information to get the new parameters inferred by the LLM. Algorithm1 describes this process. The input parameters vary when the LLM performs different tasks, but the output is exactly the same.

In actual interactions, LLMs may occasionally fail to strictly comply with requirements, returning incorrect strings or weights. This is more common when using LLMs with low parameter counts, so the post-processing step is particularly important. After obtaining the response, it is necessary to verify its validity and filter out abnormal outputs from the LLM. Therefore, the post-processing function should include a check on the weight relationship as shown in (4). This ensures that the LLM does not excessively adjust weights due

#### Algorithm 1 Interacting with LLM

**Require:**  $S$  (currently evaluated solution),  
 $LLM$  (Large Language Model),  
 $eval_{cur}$  (current evaluation data),  
 $eval_{his}$  (historical evaluation data),  
 $tmpl$  (prompt template),  
 $prompt$  (final prompt).  
**Ensure:**  $w_i$  (corresponding weights of each objective  $F$ )

- 1:  $prompt \leftarrow \emptyset$ ;
- 2:  $S \leftarrow \text{Normalization}(S)$ ;
- 3:  $prompt \leftarrow \text{GeneratePrompt}(S, eval_{cur}, eval_{his}, tmpl)$ ;
- 4:  $response \leftarrow \text{Inference}(LLM, prompt)$ ;
- 5:  $w_i \leftarrow \text{PostProcessing}(response)$ ;
- 6:  $eval_{his} \leftarrow eval_{his} \cup (S, eval_{cur} \cup w_i)$ ;
- 7: **return**  $w_i$

to abnormal objective values or evaluation errors by the DM. In subsequent experiments, the value of  $\alpha$  is always fixed at 0.2.

$$w_i \geq 0, \sum w_i = 1, \Delta w_i = |w_i^{old} - w_i^{new}| \leq \alpha \quad (4)$$

$$\forall i = 1, \dots, k$$

This section has elaborated on how to use LLMs as surrogate models in IEC. The subsequent chapters will detail how to integrate the GA as the base algorithm with the surrogate model.

#### E. LLM - assisted IEC

After the LLM is adapted to the task of parsing DM preferences, the algorithm still needs to be adjusted to fit the LLM. This chapter will elaborate on the algorithmic framework of the two LAIGA frameworks. The algorithms use the GA as the base algorithm, and the operators and their parameters related to reproduction, mutation, and other operations not mentioned in this section will be detailed in the subsequent experimental section.

1) *Algorithm Framework of D-LAIGA:* The pseudocode of D-LAIGA is presented in Algorithm2. The following is an explanation of the main steps of the algorithm:

- 1) Initialization: Sampling is performed using the Latin Hypercube Sampling (LHS) method [36] to obtain an initial population  $P$  with  $N$  individuals. Weights  $w_i$  are initialized for fitness calculation, an archive  $B$  is initialized to record the best individuals generated during iterations, and an archive  $P_u$  is initialized to temporarily store candidate representative solutions. The representative solutions  $S$  to be evaluated by the DM will be selected from  $P_u$ .
- 2) DM involvement in evolution: During iterations, if interaction with the DM is required, the representative solutions  $S$  with the highest fitness are selected from the archive  $P_u$  based on the current  $w_i$ , and the DM evaluates these individuals through the interaction interface.

- 3) Weight update: The DM's evaluation information 'eval' undergoes a series of processing (integrating solutions, evaluations, historical evaluations, etc.) and is then submitted to the LLM for updating  $w_i$ . This step is detailed in 1 mentioned earlier.
- 4) Population screening: The fitness function is adjusted according to the latest  $w_i$ , and the mixed population of the current population  $P$  and its offspring is screened. The  $N$  individuals with the highest fitness proceed to the next iteration.
- 5) Update of candidate representative solution archive:  $P$  and  $P_u$  are merged, and diverse solution sets are selected using crowding distance and non-dominated sorting [34] and stored in the archive  $P_u$ .
- 6) Update of optimal individual: The fitness of  $B$  is recalculated using the current weights  $w_i$ , and  $B$  is updated by comparing it with the individual with the highest fitness in the current population  $P$ . The algorithm will continue to loop through the above process until a preset termination condition is met or a solution sufficiently satisfying the DM is found, leading to manual termination.

The algorithm will continue to loop through the above process until a preset termination condition is met or a solution sufficiently satisfying the DM is found, leading to manual termination.

This section elaborates on the specific process of integrating LLM into IGA. In the next section, we will present the performance testing, experimental results, and analysis of the limitations of the LAIGA algorithm.

2) *Algorithm Framework of S-LAIGA*: The pseudocode of S-LAIGA is presented in Algorithm3. Compared with D-LAIGA, which only requires evaluating the absolute merits of solutions, the DM in S-LAIGA needs to judge the relative numerical superiority-inferiority relationships between different solutions. This requires representative solutions to have sufficient differences; otherwise, the DM's cognitive burden will increase. Therefore, S-LAIGA does not simply use a fitness function to screen offspring. Instead, it prioritizes maintaining population diversity during interactive iterations and only activates the fitness function after all interactions are completed to enable rapid population convergence. The following is an explanation of the main steps of the algorithm:

- 1) Initialization: Sampling is conducted using the Latin Hypercube Sampling (LHS) method [36] to obtain an initial population  $P$  consisting of  $N$  individuals. Weights  $w_i$  are initialized for fitness calculation, and a weight pool  $W$  is initialized to record the weights generated during iterations.
- 2) DM involvement in evolution: During iterations, if interaction with the DM is required, two groups of differentiated representative solutions are selected from the current population. Specifically, for objective  $t_i$ , the average Euclidean distance of all solutions in the current population is calculated using (5). The  $n$  solutions with the smallest distances are selected to form a similar subset  $S'_i$ . Then, uniform sampling is performed on  $S'_i$  to obtain a group of solutions that are similar

---

#### Algorithm 2 D-LAIGA

---

**Require:**  $N$  (population size),  
 $LLM$  (Large Language Model),  
 $w_i$  (weights),  
 $B$  (optimal solution),  
 $P$  (current population),  
 $P_u$  (candidate representative solutions),  
 $S$  (representative solution),  
 $eval$  (evaluation).  
**Ensure:**  $x^*$  (optimal solution)

```

1:  $P \leftarrow \text{Initialize}(N)$ ;
2:  $w_i \leftarrow \text{Initialize}(w_i)$ ;
3:  $B \leftarrow \text{Max}(\text{Fitness}(P, w_i))$ ;
4:  $P_u \leftarrow \emptyset$ ;
5:  $S \leftarrow \emptyset$ ;
6: while Gens < MaxGens do
7:   if NeedInteraction then
8:      $S \leftarrow \text{highestFitness}(P_u, w_i)$ ;
9:      $eval \leftarrow \text{interactive}(DM, S)$ ;
10:     $w_i \leftarrow \text{getWeights}(LLM, eval)$ ;
11:   end if
12:    $P \leftarrow \text{reproduction}(P) \cup P$ ;
13:    $P \leftarrow \text{selection}(w_i, P, N)$ ;
14:    $P_u \leftarrow \text{crowdingRanking}(P \cup P_u, N/2)$ ;
15:   if  $\text{Fitness}(B, w_i) < \text{Max}(\text{Fitness}(P, w_i))$  then
16:      $B \leftarrow \arg \max_{x \in P} \text{Fitness}(x, w_i)$ ;
17:   end if
18: end while
19:  $x^* \leftarrow B$ ;
20: return  $x^*$ 

```

---

in one objective but significantly different in another. Similarly, by swapping the roles of the two objectives, two complementary groups of representative solutions  $S$  are obtained. Finally, the DM evaluates  $S$  through the interaction interface.

- 3) Weight pool update: Data is integrated through Algorithm1 to obtain the weights inferred by the LLM, and the newly obtained weights are then added to the weight pool. When users subsequently evaluate the representative solutions, the weights in the weight pool will also be used to rank the advantages and disadvantages of these representative solutions. The degree of consistency between the rankings calculated by these weights and the rankings given by users will determine the scores of these weights.
- 4) Screening of population: Before reaching the maximum number of interactions, the crowded distance and non-dominated sorting [34] is used to screen the mixed population of the current population  $P$  and its offspring, and the surviving population will enter the next iteration.
- 5) Screening of optimal weights: When the maximum number of interactions is reached, the weight with the highest score is directly selected as the optimal weight. Alternatively, multiple weights with the highest scores in the weight pool can be used to calculate the current population separately, so as to obtain different optimal

**Algorithm 3** S-LAIGA

---

**Require:**  $N$  (population size),  
 $LLM$  (Large Language Model),  
 $w_i$  (weights),  
 $B$  (optimal solution),  
 $P$  (current population),  
 $W$  (weight pool),  
 $S$  (representative solution),  
 $eval$  (evaluation data).  
**Ensure:**  $x^*$  (optimal solution)

```

1:  $P \leftarrow \text{Initialize}(N)$ ;
2:  $w_i \leftarrow \text{Initialize}(w_i)$ ;
3:  $W \leftarrow w_i$ ;
4:  $S \leftarrow \emptyset$ ;
5: while Gens < MaxGens do
6:   while Interaction < MaxInteraction do
7:     if NeedInteraction then
8:        $S \leftarrow \text{representativeSolutions}(P)$ ;
9:        $eval \leftarrow \text{interactive}(DM, S)$ ;
10:       $W \leftarrow \text{scoring}(W, eval)$ ;
11:       $w_i \leftarrow \text{getWeights}(LLM, eval)$ ;
12:       $W \leftarrow W \cup w_i$ ;
13:    end if
14:     $P \leftarrow \text{reproduction}(P) \cup P$ ;
15:     $P \leftarrow \text{crowdingRanking}(P, N)$ ;
16:  end while
17:   $P \leftarrow \text{reproduction}(P) \cup P$ ;
18:   $w_i \leftarrow \text{selectBestWeight}(W)$ ;
19:   $P \leftarrow \text{selection}(w_i, P, N)$ ;
20: end while
21:  $B \leftarrow \text{highestFitness}(P, w_i)$ ;
22:  $x^* \leftarrow B$ ;
23: return  $x^*$ 

```

---

schemes which are then presented to the DM. The weight corresponding to the scheme that the DM prefers the most is selected as the optimal weight.

- 6) Obtaining the optimal individual: After reaching the maximum number of interactions, the optimal weight  $w_i$  is used as the parameter of the fitness function to complete subsequent iterations and obtain the optimal solution  $B$ .

$$\bar{d}_k = \frac{1}{N-1} \sum_{\substack{j=1 \\ j \neq k}}^N |F_{k,t_i} - F_{j,t_i}| \quad (5)$$

This section introduces two different methods for integrating LLM into IGA. In the next section, we will design experiments to test the performance of the algorithm and conduct relevant discussions.

#### IV. EXPERIMENT AND RESULTS

In the experiments introduced in this section, we will test the solution results of the algorithm on the DTLZ2 problem with convex boundaries. Generally speaking, multi-objective test problems are usually minimization problems. However,

since the purpose of our algorithm is to maximize the DM's satisfaction, this will lead to conflicts at the code level. If we modify the algorithm settings such that a smaller weight  $w_i$  represents a more important objective  $f_i$  and set the algorithm's selection mechanism to prioritize smaller values, the optimal solution will stay at the endpoint where the objective value corresponding to the smallest weight is 1 and the values of all other objectives are 0. This will cause the problem to degenerate from quantitative analysis to qualitative analysis. Therefore, if the DM does not care about the endpoint values, it is necessary to modify the DTLZ2 problem into a maximization problem. Considering the above factors and the requirements of this experiment, we set the form of DTLZ2 as shown in (6).

$$\begin{aligned} \text{Max } f(\mathbf{x}) &= \begin{cases} (1 + g(\mathbf{x})) \cos\left(\frac{\pi}{2}x_1\right) \cos\left(\frac{\pi}{2}x_2\right) \\ (1 + g(\mathbf{x})) \cos\left(\frac{\pi}{2}x_1\right) \sin\left(\frac{\pi}{2}x_2\right) \\ (1 + g(\mathbf{x})) \sin\left(\frac{\pi}{2}x_1\right) \end{cases} \quad (6) \\ \text{s.t. } 0 \leq x_i \leq 1, & \text{ for } i = 1, \dots, n, \\ \text{where } g(\mathbf{x}) &= \sum_{i=3}^n (x_i - 0.5)^2. \end{aligned}$$

After modifying DTLZ2 into a maximization problem, while maintaining its complexity, the radius of the Pareto spherical boundary varies with the dimension of the solution space. For instance, when the dimension of decision variables is 12, its Pareto front becomes a convex surface that satisfies the condition of  $f_1^2 + f_2^2 + f_3^2 = 3.5^2$  [29]. In practical experiments, we scale the objective values of solutions under different dimensions according to (3), so as to unify the Pareto fronts of this problem across different dimensions.

$$\begin{aligned} \text{Max } V^*(f(x)) &= \sum_{i=1}^k w_i^*(f_i(x)) \quad (7) \\ \text{s.t. } w_i^* &\geq 0 \quad \forall i = 1, \dots, k \quad \sum_{i=1}^k w_i^* = 1 \end{aligned}$$

For the DM of egalitarianism represented by the Chebyshev value function, we set its reference point as  $z_i^* = \sup_{x \in D_i} f_i(x)$  (i.e., the ideal maximum value of all objectives without considering the trade-off relationship), which indicates that the DM hopes all objectives can be as good as possible when unaware of the trade-off relationship. Meanwhile, to align the optimization purpose of the Chebyshev function with that of the test problem, the Chebyshev value is negated during the iteration and evaluation process, making it an indicator where a larger value is better. Based on these modifications, the problem is ultimately simplified to the one shown in (8).

$$\begin{aligned} \text{Max } V^*(f(x)) &= 1 - \max_{i=1, \dots, k} (w_i^* |f_i(x) - \sup_{x \in D_i} f_i(x)|) \quad (8) \\ \text{s.t. } w_i^* &\geq 0 \quad \forall i = 1, \dots, k \quad \sum_{i=1}^k w_i^* = 1 \end{aligned}$$

Furthermore, an additional nonlinear value function is incorporated herein, as illustrated in (9). This function is continuous and smooth, and it is employed to simulate the nonlinear and gradual preference changes of the DM, thereby facilitating an extended comparative analysis.

$$\begin{aligned} \text{Max } V^*(f(x)) &= \sum_{i=1}^k w_i^* f_i(x)^3 + \prod_{i=1}^k f_i(x) \\ \text{s.t. } w_i^* &\geq 0 \quad \forall i = 1, \dots, k \quad \sum_{i=1}^k w_i^* = 1 \end{aligned} \quad (9)$$

When different types of  $V^*(f(x))$  reach their maximum values, the corresponding solutions are the MPS under the DM of that type.

#### A. Simulation of DM for LAIGA

Using real-world DM evaluations would lead to inconsistencies and non-reproducibility in the feedback provided during the experiment. While involving a large number of DMs for evaluations can mitigate this risk, it would also render the experimental costs unaffordable [33]. To avoid the instability caused by human subjective factors and improve the reliability of experimental results, it is necessary to construct simulated DMs to interact with the algorithm instead of real DMs. Despite the fact that existing studies have proposed simulated DM libraries for testing, due to the differences in the preference information handled in this research, two simulated DMs are independently constructed herein. Each DM is designed to provide different types of preference information: D-DM and S-DM. D-DM tends to directly evaluate the absolute strengths and weaknesses of solutions using pure natural language, while S-DM prefers to assess the relative strengths and weaknesses of solutions through ranking combined with natural language descriptions. In addition, a disturbance mechanism is established to simulate human decision-making errors in real-world scenarios.

1) *Automatic evaluation model*: Given that this work involves natural language evaluation, we designed different automatic evaluation models for two types of simulated DMs. First, we defined the ideal points  $z^{ideal}$  of the model under (7), (8) and (9) as  $[0.48666, 0.81111, 0.32443]$ ,  $[0.54120, 0.65410, 0.52846]$ , and  $[0.56957, 0.74169, 0.35423]$ , respectively. Meanwhile, based on these ideal values, threshold ranges with deviations of  $\pm 5\%$ ,  $\pm 10\%$ , and  $\pm 15\%$  were defined.

For D-DM, regarding the objective values of each solution, the simulated DM will calculate the deviation from the ideal values and generate semantic feedback with different intensities according to the threshold ranges. For example, if a certain objective value is more than 15% lower than  $z_i^{ideal}$ , the feedback will be "Objective i is too low!".

Since real-world DMs are accustomed to prioritizing the performance of the most important or worst-performing objectives, when generating feedback, the simulated DM will prioritize objectives based on both the severity of their deviation (classified into levels 1-4, with level 4 being the most severe) and their importance (based on the relative magnitude of  $w^*$ ), and only select the top 1-2 feedback items with the highest

priority. Specifically, a two-level ranking mechanism is applied to the representative solutions: first, objectives are sorted in descending order of deviation severity; for objectives with the same deviation severity, they are further sorted in descending order of importance. In addition, for relatively less important objectives, if their deviation is excessively large, their priority will be appropriately elevated. If all objective values fall within the 5% deviation range from the ideal values, the feedback will be "All objectives are perfect!".

For S-DM, the simulated DM will calculate the representative solutions using the three aforementioned value functions based on the real weights  $w_i$ , sort them according to their values, and provide vague descriptions of the sorted solutions based on the thresholds. For example, if the value gap between a pair of solutions is less than 5%, the feedback will be "They are not much different, but Solution 1 is slightly better".

2) *Interference during the evaluation process*: In the real world, DMs often make evaluation errors due to various disturbance factors. Therefore, it is necessary to incorporate these factors into the simulated DMs to make the preference functions imperfect. Some scholars [37] have proposed mathematical methods to simulate interference factors, such as objective omission, mixing, and noise. Building on the methods presented therein, we incorporated the following interferences into the simulated DM model:

- 1) *Impact of objective omission*: If the DM omits  $q$  objectives during evaluation, these  $q$  objectives will be set to fixed values (such as default values or mean values) in the evaluation, and only  $m - q$  objectives will be retained for calculation.
- 2) *Impact of objective mixing*: The objectives actually involved in the evaluation may be linear combinations of the real objectives, which represents that the DM has confused the cognition of certain objectives, i.e.,  $(\hat{z}_k = (1 - \gamma)z_{c_k} + \gamma z_{c_{k+1}})$  (where  $\gamma$  is the mixing parameter). Therefore, the imperfect preference function needs to be calculated based on the mixed objectives  $\hat{z}_k$ .
- 3) *Impact of noise*: Gaussian noise  $\epsilon \sim \mathcal{N}(0, \sigma^2)$  should be added to the final value of the solution to simulate the uncertainty in decision-making.

In the subsequent chapters, we will discuss the scenarios with and without interference respectively, and specifically mention the settings of parameters related to interference.

#### B. Comparative experiment

For the purpose of comparison, two algorithms as follows are introduced to conduct comparative experiments. Their abbreviations and corresponding explanations are presented below:

- 1) *PI-NSGA2*: The PI-NSGA2 algorithm PI NSGA2 2010 is used for iteration. It adopts either a linear value function or a strictly monotonic product-type polynomial function to fit the user's actual preferences. This algorithm periodically requires the Decision Maker (DM) to rank six representative solutions and readjust the preference function for subsequent iterations.

2) R-NSGA3: The R-NSGA3 algorithm [19] is used for iteration, which requires the DM to preset the desired ideal point  $z^{ideal}$ . In this experiment, the value of this reference point is set by increasing each objective value of the actual MPS by 5%, which corresponds to the DM having a slightly overly high expectation for the solution.

All the above algorithms are implemented based on the framework provided by the python-pymoo library [38]. The main parameters in the experiment are fixed as follows: the crossover operator adopts SBX (probability = 0.9, distribution index = 15); the mutation operator uses PM (probability = 0.1, distribution index = 20); the population size is 50; the maximum number of iterations is 200; for algorithms requiring interaction with the DM, the maximum number of interactions is fixed at 8; LAIGA does not predefine initial weights, whereas S-LAIGA only predefines the initial weight of Objective 1 as 0.25; all other parameters not mentioned use the default parameters of the algorithms in the pymoo library. The LLM used is DeepSeek-V3-0324-API(Non local deployment).

Table II presents a comparison of the solution results of different algorithms under three value functions. The values therein represent the real values ( $V^{real}$ ) of the optimal solutions finally obtained by different algorithms. Among these, the optimal solutions obtained by PI-NSGA2 and R-NSGA3 are selected from multiple individuals in the final generation based on the real preference function, while the optimal solution of LAIGA is the only individual in the optimal solution archive after the algorithm iteration is completed. All results are calculated based on the F-values normalized by (3), and 5 independent experiments were conducted for each different algorithm. Furthermore, to facilitate the observation of differences, we calculated the relative error for all data in each experiment using (10), which is presented as a percentage in the subsequent tables.

$$\frac{V^*(z^{ideal}) - V^*(x)}{V^*(z^{ideal}) - V^*(x_{worst})} \quad (10)$$

Among them,  $V^*(z^{ideal})$  represents the actual value of the theoretical MPS under a specific value function;  $V^*(x)$  denotes the actual value of the MPS obtained by a certain algorithm; and  $V^*(x_{worst})$  stands for the actual value of the MPS with the worst value among all solutions obtained by the algorithms in all experimental groups under a specific value function. In this way, the relative differences between different algorithms can be intuitively demonstrated: the smaller the percentage value, the closer the solution obtained by the algorithm is to the theoretically optimal MPS under the corresponding value function.

It is readily observable that under most circumstances, the optimal value achievable by LAIGA across multiple experiments outperforms that of other algorithms. R-NSGA3 exhibits the most stable performance due to its pre-defined fixed preference information; however, this is often difficult to achieve in practice if the DM lacks sufficient expertise. PI-NSGA relies on rigorous mathematical logic for fitting, and when the number of DM interactions is small, the dif-

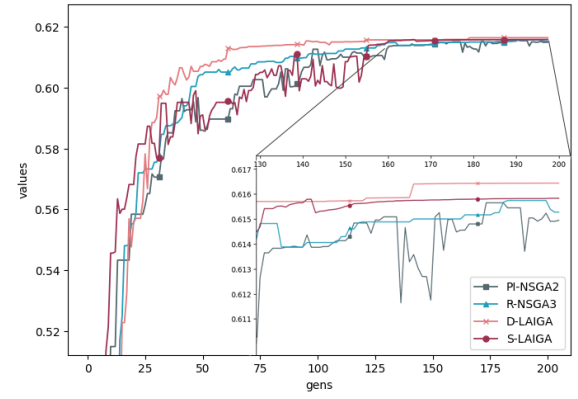


Fig. 5. Iteration of Algorithms under Additive-Value Functions.

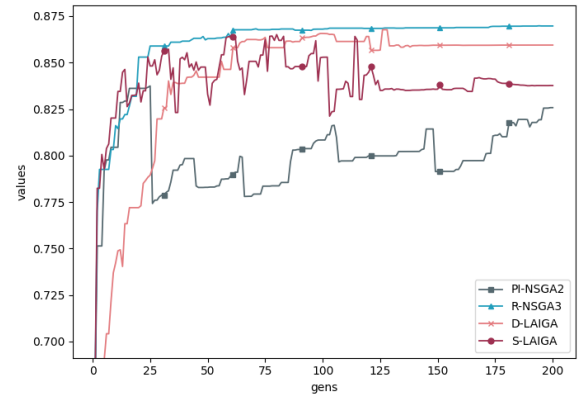


Fig. 6. Iteration of Algorithms under Chebyshev-Value Functions.

ficulty of its fitting increases significantly [29]. Leveraging the reasoning capability of LLM, LAIGA can understand the DM's preferences within a limited number of interactions. Moreover, when the DM provides only a small amount of evaluations, LAIGA's fitting accuracy is superior to that of PI-NSGA2. Nevertheless, LAIGA is dependent on the reasoning capability of large models. Without prior training, the results of the algorithm across multiple experiments exhibit considerable fluctuations and lack interpretability. This limitation will be further discussed in subsequent sections.

Fig.5, Fig.6 and Fig.7 respectively illustrate the variation trends of the theoretically optimal individuals in the population of each generation during the iteration process of different algorithms under the three value functions. Herein, the horizontal axis represents the number of iterations, and the vertical axis represents the value of the potential optimal individual in the corresponding generation. It should be noted that in the experiment, the potential optimal individual of each generation is automatically selected by the real preference function, which is actually difficult to select in practical problems. Here, it is only used to demonstrate the variation of elite individuals in the population. The iteration curves are exemplified by the scenario of the 12-dimensional problem.

It is not difficult to observe that when the DM has preferences with smooth changes (either linear or non-linear), D-LAIGA exhibits excellent convergence ability. In contrast, S-

TABLE II  
COMPARISON OF ALGORITHM UNDER THREE DIFFERENT VALUE FUNCTIONS

| Function<br>(6n,12n) | PI-NSGA2                |                          |            | R-NSGA3                 |                         |            | D-LAIGA                |                          |            | S-LAIGA                |                         |            |
|----------------------|-------------------------|--------------------------|------------|-------------------------|-------------------------|------------|------------------------|--------------------------|------------|------------------------|-------------------------|------------|
|                      | Best                    | Worst                    | $\sigma_V$ | Best                    | Worst                   | $\sigma_V$ | Best                   | Worst                    | $\sigma_V$ | Best                   | Worst                   | $\sigma_V$ |
| 6n                   |                         |                          |            |                         |                         |            |                        |                          |            |                        |                         |            |
| addition             | 6.1640e-1<br>(0.8231%)  | 6.1175e-1<br>(94.1779%)  | 1.7912e-3  | 6.1639e-1<br>(1.0239%)  | 6.1615e-1<br>(5.8422%)  | 8.6351e-5  | 6.1640e-1<br>(0.8231%) | 6.1146e-1<br>(100.0000%) | 1.7842e-3  | 6.1643e-1<br>(0.2208%) | 6.1260e-1<br>(77.1130%) | 1.8731e-3  |
| chebyshev            | 8.5316e-1<br>(10.6673%) | 7.0505e-1<br>(100.0000%) | 6.9774e-2  | 8.7024e-1<br>(0.3655%)  | 8.6842e-1<br>(1.4632%)  | 6.3010e-4  | 8.7029e-1<br>(0.3354%) | 8.2792e-1<br>(25.8909%)  | 1.5974e-2  | 8.7008e-1<br>(0.4620%) | 8.0330e-1<br>(40.7404%) | 2.2266e-2  |
| non-linear           | 2.2718e-1<br>(2.0229%)  | 2.2532e-1<br>(13.3561%)  | 6.7179e-4  | 2.2749e-1<br>(0.1340%)  | 2.2723e-1<br>(1.7183%)  | 8.6581e-5  | 2.2736e-1<br>(0.9262%) | 2.2443e-1<br>(18.7789%)  | 1.1501e-3  | 2.2728e-1<br>(1.4136%) | 2.2724e-1<br>(1.6573%)  | 1.3271e-5  |
| 12n                  |                         |                          |            |                         |                         |            |                        |                          |            |                        |                         |            |
| addition             | 6.1553e-1<br>(18.2895%) | 6.1184e-1<br>(92.3710%)  | 1.3776e-3  | 6.1554e-1<br>(18.0887%) | 6.1377e-1<br>(53.6238%) | 6.3283e-4  | 6.1615e-1<br>(5.8422%) | 6.1350e-1<br>(59.0444%)  | 8.5568e-4  | 6.1636e-1<br>(1.6262%) | 6.1201e-1<br>(88.9580%) | 2.0948e-3  |
| chebyshev            | 8.3003e-1<br>(24.6182%) | 7.1887e-1<br>(91.6645%)  | 3.7775e-2  | 8.6946e-1<br>(0.8360%)  | 8.6823e-1<br>(1.5778%)  | 4.6953e-4  | 8.6899e-1<br>(1.1194%) | 8.3063e-1<br>(24.2563%)  | 1.3168e-2  | 8.6825e-1<br>(1.5658%) | 8.0719e-1<br>(38.3942%) | 2.1446e-2  |
| non-linear           | 2.2698e-1<br>(3.2415%)  | 2.1110e-1<br>(100.0000%) | 6.0295e-3  | 2.2712e-1<br>(2.3885%)  | 2.2633e-1<br>(7.2020%)  | 3.1448e-4  | 2.2708e-1<br>(2.6322%) | 2.2463e-1<br>(17.5603%)  | 9.6553e-4  | 2.2726e-1<br>(1.5355%) | 2.2265e-1<br>(29.6247%) | 2.2405e-3  |

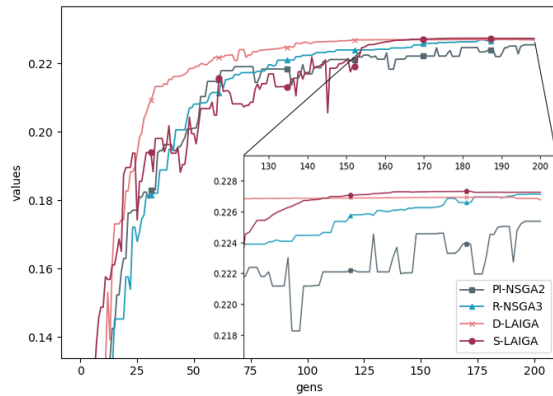


Fig. 7. Iteration of Algorithms under Non-Linear-Value Functions.

LAIGA demonstrates relatively poor convergence ability due to the need to maintain the diversity of the population in the early stage of iteration, but it can still achieve favorable results in the end. When there exists a breakpoint in the DM's preferences (Chebyshev), all methods except R-NSGA3, which is guided by the reference point, suffer from varying degrees of loss in convergence, while R-NSGA3 can maintain stable performance.

Table III demonstrates the performance of LAIGA under the scenario where the DM's evaluations are subject to disturbances. The perturbations we set are as follows: 1) setting the mixing parameter  $\gamma$  to 0.05, 0.1 and 0.15 to mix the three objectives; 2) fixing objective 3 at 0.5; 3) adding Gaussian noise interference with  $\sigma = 0.05, 0.1, 0.15$ .

It can be easily observed from the experimental results that in the presence of disturbances, S-LAIGA, which utilizes indirect preference information, incurs a relatively small loss. In contrast, D-LAIGA suffers a relatively larger loss due to the incorrect feedback from the DM. This phenomenon can be attributed to the fact that direct preference information relies on the DM's accurate cognition, whereas indirect preference information is inherently ambiguous. Consequently, algorithms that leverage indirect preference information generally exhibit stronger anti-interference capability.

Table IV presents the performance of LAIGA under differ-

TABLE III  
PERFORMANCE OF LAIGA UNDER DIFFERENT INTERFERENCES

| Function<br>(12n)              | D-LAIGA                 |                          |            | S-LAIGA                |                          |            |
|--------------------------------|-------------------------|--------------------------|------------|------------------------|--------------------------|------------|
|                                | Best                    | Worst                    | $\sigma_V$ | Best                   | Worst                    | $\sigma_V$ |
| $\gamma = 0.05, \sigma = 0.05$ |                         |                          |            |                        |                          |            |
| addition                       | 6.0540e-1<br>(5.2215%)  | 5.8869e-1<br>(13.1241%)  | 5.6852e-3  | 6.1271e-1<br>(1.7645%) | 5.9584e-1<br>(9.7427%)   | 6.5129e-3  |
| chebyshev                      | 8.1591e-1<br>(37.6671%) | 8.0001e-1<br>(48.5690%)  | 6.3610e-3  | 8.6382e-1<br>(4.8174%) | 8.0043e-1<br>(48.2811%)  | 2.0258e-2  |
| non-linear                     | 2.2509e-1<br>(21.2233%) | 2.2067e-1<br>(59.9544%)  | 1.5926e-3  | 2.2377e-1<br>(1.2443%) | 2.2237e-1<br>(45.0578%)  | 2.2132e-3  |
| $\gamma = 0.1, \sigma = 0.1$   |                         |                          |            |                        |                          |            |
| addition                       | 5.9744e-1<br>(8.9860%)  | 5.8545e-1<br>(14.6564%)  | 4.4233e-3  | 6.1248e-1<br>(1.8732%) | 5.6218e-1<br>(25.6613%)  | 1.7584e-2  |
| chebyshev                      | 8.0019e-1<br>(48.4456%) | 8.0003e-1<br>(48.5553%)  | 5.7270e-5  | 8.6125e-1<br>(6.5795%) | 7.2500e-1<br>(100.0000%) | 4.9297e-2  |
| non-linear                     | 2.2440e-1<br>(27.2695%) | 2.1750e-1<br>(87.7322%)  | 2.5629e-3  | 2.2728e-1<br>(2.0329%) | 2.2251e-1<br>(43.8311%)  | 2.2979e-3  |
| $\gamma = 0.15, \sigma = 0.15$ |                         |                          |            |                        |                          |            |
| addition                       | 5.9546e-1<br>(9.9224%)  | 5.7772e-1<br>(18.3120%)  | 6.2250e-3  | 6.1216e-1<br>(2.0246%) | 4.0499e-1<br>(100.0000%) | 7.9301e-2  |
| chebyshev                      | 8.0010e-1<br>(48.5073%) | 8.0003e-1<br>(48.5553%)  | 2.9265e-5  | 8.5671e-1<br>(9.6924%) | 8.0337e-1<br>(46.2652%)  | 2.0612e-2  |
| non-linear                     | 2.2393e-1<br>(31.3880%) | 2.1610e-1<br>(100.0000%) | 3.1332e-3  | 2.2742e-1<br>(2.3835%) | 2.2249e-1<br>(44.0063%)  | 2.0532e-3  |

ent DM interaction frequencies. It can be observed from the table that the interaction frequency has a negligible impact on the variation of LAIGA's results. On occasion, even due to the inherent randomness of the algorithm itself, the performance of the algorithm under a high interaction frequency may still outperform that under a low interaction frequency. We initially hypothesize that this is attributed to the limited processing capability of LLM. Even when historical data of interactions with the DM is provided, LLMs do not take all of this data into account.

*1) Limitations and Future Research:* When evaluated using the three value functions, LAIGA exhibits excellent performance; however, it still has certain limitations. In the experimental verification of this work, the proxy model used has a large parameter size, and its strong semantic parsing and historical association capabilities are the prerequisites for LAIGA's excellent performance. However, the effectiveness of the LAIGA algorithm is highly dependent on the performance of LLMs: when using LLMs with a smaller parameter scale (such as 20B and below), their depth of semantic understanding, ability to associate historical data, and stability of logical reasoning may decline rapidly. As a result, the weight

TABLE IV  
PERFORMANCE OF LAIGA UNDER DIFFERENT INTERACTION TIMES

| Function<br>(12n)                   | D-LAIGA                 |                         |            | S-LAIGA                 |                          |            |
|-------------------------------------|-------------------------|-------------------------|------------|-------------------------|--------------------------|------------|
|                                     | Best                    | Worst                   | $\sigma_V$ | Best                    | Worst                    | $\sigma_V$ |
| interval=30, interactions=4 for DM  |                         |                         |            |                         |                          |            |
| addition                            | 6.1630e-1<br>(0.9771%)  | 6.1499e-1<br>(10.0547%) | 4.7827e-4  | 6.1592e-1<br>(3.6103%)  | 6.0201e-1<br>(100.0000%) | 5.5510e-3  |
| chebyshev                           | 8.6704e-1<br>(10.4457%) | 8.3467e-1<br>(99.2864%) | 1.1885e-2  | 8.6660e-1<br>(11.6533%) | 8.3473e-1<br>(99.1217%)  | 1.2168e-2  |
| non-linear                          | 2.2741e-1<br>(1.9760%)  | 2.2672e-1<br>(15.3429%) | 2.4401e-4  | 2.2726e-1<br>(4.8818%)  | 2.2235e-1<br>(100.0000%) | 1.9385e-3  |
| interval=10, interactions=12 for DM |                         |                         |            |                         |                          |            |
| addition                            | 6.1638e-1<br>(0.4227%)  | 6.0825e-1<br>(56.7598%) | 3.0503e-3  | 6.1593e-1<br>(3.5410%)  | 6.1026e-1<br>(42.8314%)  | 2.2483e-3  |
| chebyshev                           | 8.6245e-1<br>(23.0431%) | 8.3580e-1<br>(96.1851%) | 9.4511e-3  | 8.6464e-1<br>(17.0326%) | 8.3441e-1<br>(100.0000%) | 1.3832e-2  |
| non-linear                          | 2.2743e-1<br>(1.5885%)  | 2.2410e-1<br>(66.0984%) | 1.2098e-3  | 2.2729e-1<br>(4.3007%)  | 2.2729e-1<br>(4.3007%)   | 4.7160e-5  |

TABLE V  
THE NAME AND PARAMETER QUANTITY OF LLM IN THE STUDY

| Abbreviation | Full Name                           | Parameter Count                    |
|--------------|-------------------------------------|------------------------------------|
| deepseek-V3  | DeepSeek-V3-0324                    | Total: 680B-<br>Activated only:37B |
| grok-3       | Grok-3                              | -                                  |
| gpt-4o       | GPT-4o                              | 200B                               |
| mistral-3.1  | Mistral-Small-3.1-24B-Instruct-2503 | 24B                                |
| llama4       | Llama-4-Scout-17B-16E-Instruct      | 17B                                |
| gpt-4o-mini  | GPT-4o-mini                         | 8B                                 |
| deepseek-R1  | DeepSeek-R1-0528-Qwen3-8B           | 8B                                 |
| mata-llama3  | Meta-Llama-3-8B-Instruct            | 8B                                 |

adjustment process is prone to abnormal fluctuations — for example, being unable to accurately map feedback such as “excessively high target value” to corresponding weight reduction, or making unreasonable adjustments under the constraint that the sum of weights remains 1 (such as arbitrarily changing the weights of irrelevant targets). This will lead to a significant decrease in the stability of weight allocation, thereby affecting the accuracy of the algorithm in fitting DM’s preferences, and ultimately deviating from the ideal optimization direction.

To further explore the impact of LLM performance on the algorithm, we randomly selected prompts with three historical evaluations generated in several experiments. These prompts were used to pose questions to different LLMs. Finally, the Euclidean distance formula was employed to calculate the deviation between the weights provided by the tested LLMs and the actual weights, and the average value of these deviations was obtained. The test results are illustrated in Fig.8, where the abbreviations of the LLM models used are presented in TableV.

It can be observed that in the task of parsing pure natural language preference information, models with larger parameter sizes are more capable of stably handling semantic parsing and weight adjustment tasks. However, in the task of parsing preference information mainly based on ranking, the ability to guess weights depends on whether the large model has undergone specialized training for similar tasks, while the impact of parameter size is relatively small. This also indicates that the effectiveness of LAIGA is highly dependent on the base capabilities of LLMs. Future research can further improve performance through the following directions:

- 1) Enhancing the capability of LLMs as auxiliary models:

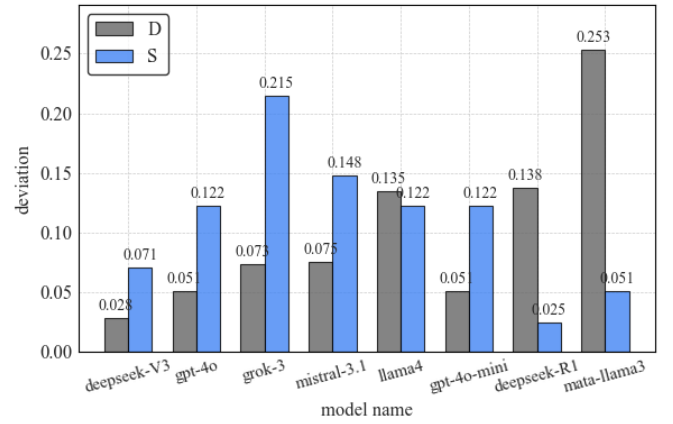


Fig. 8. Comparison of Deviation Rates for LLM.

Optimize prompt templates to enhance the constraint-following ability of small models, and introduce a fine-tuning mechanism to make LLMs specifically used for weight adjustment tasks.

- 2) Reducing the gap between simulated DMs and real scenarios: In the experiments, the feedback from the simulated DM is a fuzzy feedback method based on interval-based fixed rules plus simulated perturbations. However, in the real world, the ambiguity of human evaluations and the diversity of language styles will be further increased. If extended to real scenarios, LLMs need to deal with the ambiguity and diversity of natural language, which may require adding more refined ambiguity resolution rules in the prompt templates.

Moreover, beyond improving the robustness and generalization of the current framework, future research can further expand LAIGA in several promising directions. First, large language models can be employed as intelligent recommenders that assist in selecting appropriate optimization or decision-making algorithms based on the characteristics of the problem and the decision context. This idea resonates with recent advances in taxonomy-based decision support systems that automatically recommend suitable MCDM or interactive methods [9], providing a pathway toward more autonomous and adaptive decision frameworks.

Second, LAIGA can be extended to support group decision-making scenarios where multiple DMs are involved. In such cases, LLMs could act as mediators or facilitators, harmonizing conflicting preferences among DMs and incorporating affective or emotional considerations to create more human-centered decision processes. This direction aligns with the emerging trend of group decision support in multiobjective optimization, as discussed in recent systematic reviews [39]. Integrating such multi-agent and emotionally aware mechanisms would open new perspectives for collaborative and socially intelligent evolutionary decision-making systems.

## V. CONCLUSION

To address the issue of excessive cognitive burden on DMs in IEC, this work proposes a dynamic weight adjustment

method based on LLMs and constructs a novel interaction model. By integrating a prompt template with DMs' historical evaluation data, this method enables LLMs to parse DMs' natural language feedback and automatically optimize the weight allocation of multi-objective problems, thereby fitting DMs' preference information and effectively reducing the number of interactions and cognitive burden on DMs.

In the experiments, the proposed method (LAIGA) was tested against various comparative algorithms on the modified DTLZ2 problem. The results show that LAIGA exhibits the ability to approach the ideal MPS with a relatively low number of interactions under the three different types of value functions, verifying the effectiveness of this algorithmic framework.

In addition, the work explored the impact of LLM performance in handling different tasks, indicating that the performance of algorithms using LLMs as proxy models is currently limited to a certain extent by the performance of large models. However, with the rapid development of large model technology, the depth of semantic understanding, ability to associate historical data, and stability of logical reasoning of models will continue to improve, and it is believed that this limitation will be resolved soon. Meanwhile, this work also points out other scalable directions, including reducing the gap between simulated DMs and real scenarios, expanding problem dimensions, and improving base algorithms.

In summary, this work innovatively integrates LLMs into IGA, realizing the prediction of weight vector through interactions based on pure natural language or ranking. The idea of incorporating fuzzy natural language information into preference information provides a practical solution to alleviate the cognitive burden of DMs in IEC, reduce the computational pressure of algorithms, and improve the efficiency of multi-objective optimization. This research offers new insights for related fields and expands the application scenarios of LLMs in IEC.

## REFERENCES

- [1] Y. Wang, Y. Pei, H. Shindo, Q. Liu, and H. P. Ren, "An interactive differential evolution method with human auditory perception for sound composition," *IEEE Trans. Cogn. Dev. Syst.*, vol. 16, no. 3, pp. 1134–1146, 2024.
- [2] T. Kowaliw, A. Dorin, and J. McCormack, "Promoting creative design in interactive evolutionary computation," *IEEE Trans. Evol. Comput.*, vol. 16, no. 4, pp. 523–536, 2012.
- [3] F. Castiglione, S. Corrente, S. Greco, P. Bianucci, A. Sordo-Ward, L. Garrote, et al., "Interactive multiobjective evolutionary optimization model for dam management support," *J. Hydrol.*, vol. 647, p. 132304, 2025.
- [4] K. Li, M. Liao, K. Deb, G. Min, and X. Yao, "Does preference always help? A holistic study on preference-based evolutionary multiobjective optimization using reference points," *IEEE Trans. Evol. Comput.*, vol. 24, no. 6, pp. 1078–1096, 2020.
- [5] M. K. Tomczyk and M. Kadziński, "Co-evolution improves the efficiency of preference learning methods when the decision maker's aspirations develop over time," in *Proc. ACM Genet. Evol. Comput. Conf. (GECCO)*, Lisbon, Portugal, 2023, pp. 759–767.
- [6] A. Sinha, P. Korhonen, J. Wallenius, and K. Deb, "An interactive evolutionary multiobjective optimization algorithm with a limited number of decision maker calls," *Eur. J. Oper. Res.*, vol. 233, no. 3, pp. 674–688, 2014.
- [7] S. M. Shavarani, M. López-Ibáñez, and R. Allmendinger, "Detecting hidden and irrelevant objectives in interactive multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 28, no. 2, pp. 544–557, 2024.
- [8] M. Ghaznavi, M. Ilati, and E. Khorram, "An interactive algorithm for solving multiobjective optimization problems based on a general scalarization technique," *Iran. J. Numer. Anal. Optim.*, vol. 6, no. 1, pp. 79–101, 2016.
- [9] M. Cinelli, M. Kadziński, G. Miebs, M. Gonzalez, R. Słowiński, et al., "Recommending multiple criteria decision analysis methods with a new taxonomy-based decision support system," *Eur. J. Oper. Res.*, vol. 302, no. 2, pp. 633–651, 2022.
- [10] N. v. Stein and T. Bäck, "LLaMEA: A large language model evolutionary algorithm for automatically generating metaheuristics," *IEEE Trans. Evol. Comput.*, vol. 29, no. 2, pp. 331–345, 2025.
- [11] S. Liu, C. Chen, X. Qu, K. Tang, and Y. S. Ong, "Large language models as evolutionary optimizers," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Yokohama, Japan, 2024, pp. 1–8.
- [12] F. Liu, X. Lin, S. Yao, Z. Wang, X. Tong, M. Yuan, et al., "Large language model for multiobjective evolutionary optimization," in *Proc. Springer Int. Conf. Evol. Multi-Criterion Optim. (EMO)*, Canberra, Australia, 2025, pp. 178–191.
- [13] E. Meyerson, M. J. Nelson, H. Bradley, A. Gaier, A. Moradi, A. K. Hoover, et al., "Language model crossover: Variation through few-shot prompting," *ACM Trans. Evol. Learn.*, vol. 4, no. 4, pp. 1–40, 2024.
- [14] H. Hao, X. Zhang, and A. Zhou, "Large language models as surrogate models in evolutionary algorithms: A preliminary study," *Swarm Evol. Comput.*, vol. 91, p. 101741, 2024.
- [15] M. K. Tomczyk and M. Kadziński, "Decomposition-based interactive evolutionary algorithm for multiple objective optimization," *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, pp. 320–334, 2019.
- [16] J. M. Kivikangas, E. Vilkkumaa, J. Blank, V. Harjunen, P. Malo, K. Deb, N. J. Ravaja, and J. Wallenius, "Effects of many conflicting objectives on decision-makers' cognitive burden and decision consistency," *Eur. J. Oper. Res.*, vol. 322, no. 1, pp. 182–197, 2025.
- [17] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization," in *Proc. 5th Int. Conf. Genet. Algorithms (ICGA)*, Urbana-Champaign, IL, USA, 1993, pp. 416–423.
- [18] K. Deb and J. Sundar, "Reference point based multi-objective optimization using evolutionary algorithms," in *Proc. ACM 8th Annu. Conf. Genet. Evol. Comput. (GECCO)*, Seattle, Washington, USA, 2006, pp. 635–642.
- [19] Y. Vesikar, K. Deb, and J. Blank, "Reference point based NSGA-III for preferred solutions," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Bangalore, India, 2018, pp. 1587–1594.
- [20] K. Deb and A. Kumar, "Interactive evolutionary multi-objective optimization and decision-making using reference direction method," in *Proc. ACM 9th Annu. Conf. Genet. Evol. Comput. (GECCO)*, London, UK, 2007, pp. 781–788.
- [21] K. Deb and A. Kumar, "Light beam search based multi-objective optimization using evolutionary algorithms," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Tokyo, Japan, 2007, pp. 2125–2132.
- [22] B. Rekiek, P. De Lit, F. Pellichero, T. L'Eglise, E. Falkenauer, et al., "Dealing with user's preferences in hybrid assembly lines design," *IFAC Proc. Vol.*, vol. 33, no. 17, 2000, pp. 989–994.
- [23] E. Fernández, N. Rangel-Valdez, L. Cruz-Reyes, C. G. Gomez-Santillan, and C. A. Coello-Coello, "Preference incorporation in MOEA/D using an outranking approach with imprecise model parameters," *Swarm Evol. Comput.*, vol. 72, p. 101097, 2022.
- [24] X. Shen, Y. Guo, Q. Chen, and W. Hu, "A multi-objective optimization evolutionary algorithm incorporating preference information based on fuzzy logic," *Comput. Optim. Appl.*, vol. 46, no. 1, pp. 159–188, 2010.
- [25] S. Ramakrishnan and Y. A. Hasan, "Fuzzy preference-based multi-objective optimization method," *Artif. Intell. Rev.*, vol. 39, no. 2, pp. 165–181, 2013.
- [26] J. Branke, S. Corrente, S. Greco, R. Słowiński, P. Zielniewicz, et al., "Using Choquet integral as preference model in interactive evolutionary multiobjective optimization," *Eur. J. Oper. Res.*, vol. 250, no. 3, pp. 884–901, 2016.
- [27] M. Gong, F. Liu, W. Zhang, L. Jiao, and Q. Zhang, "Interactive MOEA/D for multi-objective decision making," in *Proc. ACM 13th Annu. Conf. Genet. Evol. Comput. (GECCO)*, Dublin, Ireland, 2011, pp. 721–728.
- [28] S. Corrente, S. Greco, B. Matarazzo, and R. Słowiński, "Explainable interactive evolutionary multiobjective optimization," *Omega*, vol. 122, p. 102925, 2024.
- [29] K. Deb, A. Sinha, P. J. Korhonen, and J. Wallenius, "An interactive evolutionary multiobjective optimization method based on progressively approximated value functions," *IEEE Trans. Evol. Comput.*, vol. 14, no. 5, pp. 723–739, 2010.

- [30] A. Sinha and J. Wallenius, "MCDM, EMO and hybrid approaches: Tutorial and review," *Math. Comput. Appl.*, vol. 27, no. 6, p. 112, 2022.
- [31] M. Ehrgott, M. Köksalan, M. Kadziński, and K. Deb, "Fifty years of multi-objective optimization and decision-making: From mathematical programming to evolutionary computation," *Eur. J. Oper. Res.*, 2025.
- [32] M. Kadziński and M. K. Tomczyk, "Interactive evolutionary multiple objective optimization for group decision incorporating value-based preference disaggregation methods," *Group Decis. Negot.*, vol. 26, no. 4, pp. 693–728, 2017.
- [33] S. M. Shavarani, M. López-Ibáñez, and J. Knowles, "On benchmarking interactive evolutionary multiobjective algorithms," *IEEE Trans. Evol. Comput.*, vol. 28, no. 4, pp. 1084–1098, 2024.
- [34] K. Deb, A. Pratap, S. Agarwal, and T. A. M. T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, 2002.
- [35] I. Mata, G. Fadel, A. Garland, and W. Zanker, "Affordance-based interactive genetic algorithm (ABIGA)," *Des. Sci.*, vol. 4, p. e5, 2018.
- [36] M. D. Shields and J. Zhang, "The generalization of Latin hypercube sampling," *Reliab. Eng. Syst. Saf.*, vol. 148, pp. 96–108, 2016.
- [37] M. López-Ibáñez and J. Knowles, "Machine decision makers as a laboratory for interactive EMO," in *Proc. Springer Int. Conf. Evol. Multi-Criterion Optim. (EMO)*, Guimarães, Portugal, 2015, pp. 295–309.
- [38] J. Blank and K. Deb, "Pymoo: Multi-objective optimization in Python," *IEEE Access*, vol. 8, pp. 89497–89509, 2020.
- [39] J. Pajasmaa, K. Miettinen, and J. Silvennoinen, "Group decision making in multiobjective optimization: A systematic literature review," *Group Decis. Negot.*, vol. 34, no. 2, pp. 329–371, 2025.
- [40] S. Qin, F. Guo, J. Wang, L. Qi, J. Wang, X. Guo, and Z. Zhao, "Expanded discrete migratory bird optimizer for circular disassembly line balancing with tool deterioration and replacement," *Int. J. Artif. Intell. Green Manuf.*, vol. 1, no. 1, pp. 14–25, 2025.
- [41] Y. Qing and J. Zhao, "Enhanced gold rush optimizer for feature selection: Application in software fault prediction datasets," *Int. J. Artif. Intell. Green Manuf.*, vol. 1, no. 2, pp. 58–68, 2025.
- [42] H. Zhang, D. Pham, and Q. Kang, "Improved fruit fly algorithm for multi-objective disassembly line balancing problem considering learning effect," *Int. J. Artif. Intell. Green Manuf.*, vol. 1, no. 2, pp. 108–118, 2025.
- [43] X. Guo, Y. Feng, and M. Zhou, "Mixed-layout multi-type factory remanufacturing system optimization via LLM-TD3," *Int. J. Artif. Intell. Green Manuf.*, vol. 1, no. 4, pp. 184–194, 2025.
- [44] S. Qin, S. Dai, and B. Hu, "LLM-enhanced dueling DQN for multi-factory remanufacturing optimization with hybrid disassembly lines and drone delivery," *Int. J. Artif. Intell. Green Manuf.*, vol. 1, no. 4, pp. 195–206, 2025.
- [45] Y. Fu, Z. Zhang, M. Huang, X. Guo, and L. Qi, "Multi-objective integrated energy-efficient scheduling of distributed flexible job shop and vehicle routing by knowledge-and-learning-based hyper-heuristics," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 9, no. 3, pp. 2137–2150, 2025.
- [46] G. Xu, Z. Zhang, Z. Li, X. Guo, L. Qi, and X. Liu, "Multi-objective discrete brainstorming optimizer to solve the stochastic multiple-product robotic disassembly line balancing problem subject to disassembly failures," *Mathematics*, vol. 11, no. 6, p. 1557, 2023.
- [47] J. Wang, G. Xi, X. Guo, S. Liu, S. Qin, and H. Han, "Reinforcement learning for hybrid disassembly line balancing problems," *Neurocomputing*, vol. 569, p. 127145, 2024.
- [48] X. Guo, T. Zhang, J. Wang, S. Liu, S. Qin, L. Qi, Y. Ji, and J. Cao, "Data-driven disassembly line allocation via RoBERTa fine-tuning and reinforcement learning," *IEEE Trans. Ind. Informat.*, 2026.



**Chi Jiao** He received the bachelor's degree from Taishan University, Tai'an, China. He is currently pursuing the master's degree with the Artificial Intelligence and Software College, Liaoning Petrochemical University, Fushun 113001, P. R. China. His research interests include artificial intelligence algorithms and multi-objective optimization.



**Zhiwei Zhang** He received his M. S. degree from Liaoning Petrochemical University, Fushun, China. He is currently pursuing the Ph. D. degree in engineering management at Shenyang University of Technology, Shenyang, China. Meanwhile, he works at the School of Artificial Intelligence and Software, Liaoning Petrochemical University, Fushun, China. His current research interests include artificial intelligence, intelligent computing, and optimal scheduling.