Electronic Research Archive

https://www.aimspress.com/journal/era

ERA, 34(6): 4216–4247.
DOI: 10.3934/era.2026189
Received: 06 March 2026
Revised: 03 May 2026
Accepted: 12 May 2026
Published: 19 May 2026

Research article

Multitask differential evolution optimization based on an expert library module

Yuehui Zhang[1,2], **Xuewen Xia**[1,2,*], **Yi Zeng**[1,2] **and Fenglin Lin**[1,2]

[1] College of Physics and Information Engineering, Minnan Normal University, Zhangzhou 363000, China

[2] Key Lab of Intelligent Optimization and Information Processing, Minnan Normal University, Zhangzhou 363000, China

* **Correspondence:** Email:xwxia@whu.edu.cn.

Abstract: Utilizing evolutionary algorithms (EAs) to solve multitask optimization problems (MTOPs) simultaneously has gained significant attention because many real applications usually share similar properties or have a certain degree of correlations. In this new research area in the EA community, namely multitask optimization (MTO), a key challenge is transfering useful knowledge among different tasks efficiently. In this paper, a multitask differential evolution (DE) algorithm is proposed in which a simple variant of the canonical DE is adopted as the basic optimizer. Furthermore, an expert library module (ELM), which can extract useful knowledge from the historical optimization experiences of DE, is introduced as a complement for the DE to generate offspring. In the proposed algorithm, named MTDE-ELM, when the DE performs the mutation operator, if all individuals chosen to perform the mutation have the same skill factor, intratask knowledge transfer can be realized. Conversely, when individuals with different skill factors are adopted to execute the mutation, intertask knowledge transfer can be realized. In the ELM, evolutionary data of different tasks are used to train the feedforward neural networks (FNNs). After that, the trained FNN can realize potential knowledge transfer among different tasks. Moreover, based on the trained FNN, each individual can query its promising evolutionary direction, which can be regarded as an effective supplement for the DE to generate offspring. The comprehensive performance of the MTDE-ELM is demonstrated through comparisons with five other MTO algorithms. Furthermore, distinct properties of the newly introduced strategies are also confirmed by experimental analysis.

Keywords: evolutionary algorithms; multitask optimization; knowledge transfer; differential evolution; feedforward neural network

1. Introduction

Evolutionary algorithms (EAs) [1] have gained popularity in optimizing complex problems in recent years by simulating Darwinian principles of "survival of the fittest" [2]. Due to their simple implementation and robust characteristics, various EAs have been successfully applied in different optimization problems in numerous study domains, such as astronomy [3], physics [4], bio-informatics [5], and other complicated scientific applications [6, 7]. Generally, these optimization problems can be classified into two types, single-objective [8] and multi-objective [9]. No matter the type of problem to be optimized, traditional EAs often treat these problems as isolated entities, disregarding potential correlations among them. However, many real applications usually share some similar properties or have a certain degree of correlation [10, 11]. Intuitively, if the properties and correlations of different tasks are utilized efficiently, EAs can optimize the tasks simultaneously. With this in mind, multitask optimization (MTO) [12, 13] has emerged as a new research area in the EA community, aiming to solve multitask optimization problems (MTOPs), in which multiple interrelated tasks must be optimized simultaneously.

A great challenge in an MTO algorithm is how to transfer helpful knowledge among different tasks in an MTOP effectively [14, 15]. In recent years, various MTO algorithms have highlighted the critical role of knowledge transfer through various strategies [16–18]. Intuitively, a simple knowledge transfer can be realized by a random crossover operation on two individuals who focus on distinct tasks. However, a random crossover has a certain blindness for knowledge transfer. On the contrary, others focus on leveraging historical experiential knowledge to guide the subsequent evolutionary process [19, 20]. A representative method is a multifactorial evolutionary algorithm (MFEA) [21], in which individuals within a single population have their skill factors according to their performance on different tasks. Thus, when two individuals with different skill factors are used to generate offspring via crossover operators, knowledge transfer can be implicitly carried out from one task to the other. In a further refinement, Zhou et al. [22] proposed a multifactorial evolutionary algorithm with adaptive knowledge transfer (MFEA-AKT), which incorporates a novel mechanism for more effectively utilizing historical information. The crossover operator in an MFEA-AKT can dynamically adjust the intensity of knowledge sharing based on real-time data, which can improve the efficiency of the knowledge transfer process.

Furthermore, the integration of learning mechanisms with evolutionary optimization has attracted increasing attention in recent years. In particular, learning-aided evolutionary computation [23] has been extensively studied to enhance search efficiency by leveraging historical optimization experience. Representative works by Zhan et al. [24] have proposed a meta-knowledge transfer-based differential evolution (MKTDE) that introduces the concept of meta-knowledge, defined as the knowledge of how to evolve high-quality solutions, enabling more general and flexible knowledge transfer across heterogeneous tasks. These studies suggest that learning transferable search patterns, rather than directly transferring candidate solutions, can significantly improve optimization performance. Nevertheless, effectively modeling such knowledge and mitigating negative transfer in complex multitask scenarios remains a challenging problem, which motivates the proposed method.

These studies verify that utilizing historical knowledge reasonably is conducive to obtaining correlations among different tasks in an MTOP. Although substantial evolutionary data, such as population distribution and fluctuations of individuals' fitness, can be acquired during the

optimization process, mere data acquisition is insufficient to guarantee its effective utilization. In other words, the acquired data has little value if useful knowledge cannot be extracted from it. Thus, to enhance the performance of an MTO algorithm, we need to establish a knowledge base system in which valuable knowledge can be obtained based on organically integrated evolutionary data.

To illustrate this concept, the optimization scenario is presented in Figure 1, in which two tasks (i.e., Task1 and Task2) are considered. In the figure, individual a_o in Task1 is regarded as a parent, and a_1 and a_2 are potential offspring of a_o. The evaluation results indicate that both of the offspring are superior to the parent. As a result, both evolutionary directions of a_o (i.e., move to the left and move to the right in Figure 1) are promising.

According to the principles of survival of the fittest, retaining a_1 for survival seems promising because it demonstrates higher fitness than a_2. However, it may hinder the discovery of the global optimum a_{min}, potentially resulting in convergence at a local optimum. However, if the gradient information of Task2 can be utilized, a_2 will be saved to the next generation. In this case, the probability of finding the global optima of the two tasks (i.e., a_{min} and b_{min}) will be increased, although a_2 is worse than a_1. This phenomenon illustrates that, even if individuals like a_2, which has potential advantages, are identified during the iteration process, their merits may be ignored if there is no helpful knowledge to guide their subsequent search directions. This observation confirms that it is feasible that unearthing correlations between different tasks and reasonably utilizing them can help an algorithm efficiently solve the tasks simultaneously.

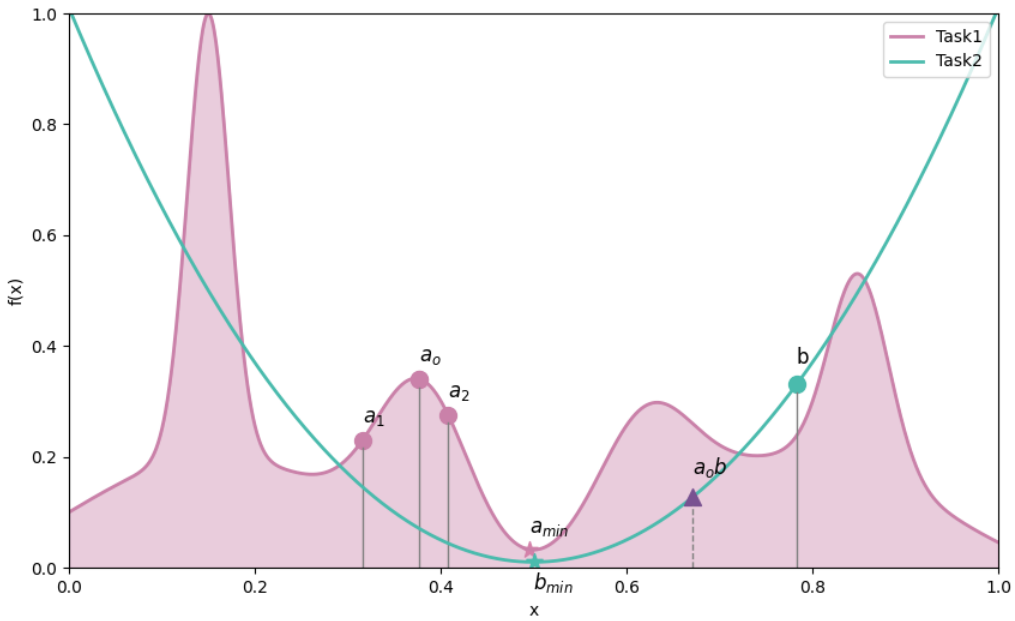

Figure 1. An example of two-task knowledge association: a_o represents the parent, and a_1 and a_2 are potential successful individuals, both belonging to Task1. a_1 outperforms a_2, but selecting a_1 as the new parent may cause offspring to move further away from the optimal solution $a_{\min}$. b represents an individual from Task2. By leveraging knowledge from b, Task1 can potentially generate offspring $a_o b$, which is more likely to enhance the convergence rate and accuracy of the Task1 optimization process.

Although substantial data and successful experiences have been accumulated in each task, they often remain segregated across tasks, impeding mutual learning and integration. As a result, the efficiency of an MTO algorithm is constrained by the lack of effective integration with existing knowledge in different tasks. Therefore, the study of the MTO should not solely focus on acquiring relevant knowledge but also effectively transferring the knowledge among different tasks.

In recent years, neural networks (NNs) [25] have demonstrated strong capability in modeling complex nonlinear relationships by learning patterns directly from data without relying on predefined rules. In the EA community, existing studies have shown that NNs can effectively extract useful knowledge from the historical search process of populations and further guide subsequent evolutionary search [26–28].

Furthermore, the integration of NNs and deep learning models with MTO has attracted increasing attention [29, 30]. In particular, recent studies have explored learning assisted evolutionary optimization frameworks that leverage prior optimization experience to improve search efficiency. These approaches introduce knowledge adaptation mechanisms to enable the reuse and refinement of historical search information, thereby guiding the evolutionary process across tasks. Existing research mainly focuses on data-driven paradigms, including surrogate-assisted MTO, deep representation learning, and NN-based multitask learning models [31, 32].These methods aim to facilitate knowledge sharing across tasks and improve optimization efficiency.Nevertheless, effectively controlling knowledge transfer and mitigating negative transfer, particularly in low-similarity scenarios, remain a significant challenges, which motivates the development of more robust learning-assisted MTO approaches.

In this paper, inspired by the aforementioned studies, a multitask differential evolutionary (DE) algorithm based on an expert library module (ELM), named MTDE-ELM (mutlitask differential evolutionary expert library module), is proposed, aiming to properly integrate and utilize knowledge elements across multiple tasks. In the MTDE-ELM, a variant of the DE algorithm [33] is adopted as the basic optimizer, and the ELM is introduced as a supplementary tool of the DE. The contributions of the MTDE-ELM can be summarized as follows.

1) **Intra-task knowledge transfer.** When utilizing the DE to optimize each task, all individuals adopted to perform the mutation process have the same skill factor. In this condition, different individuals' search experiences about the same task can be exchanged, and then the intratask knowledge transfer can be realized. Based on the knowledge transfer, the performance of the MTDE-ELM on each task can be highlighted.

2) **Inter-task knowledge transfer.** To deal with an MTOP, an efficient knowledge exchange between different individuals assigned to different tasks is crucial and indispensable. In the MTDE-ELM, when an individual in the DE performs the mutation operator, individuals with different skill factors are selected. In this case, the individuals can realize the intertask knowledge transfer because the generated offspring incorporates knowledge about various tasks. Relying on the knowledge transfer, an individual's experience on a task can help another individual optimize another task.

3) **Potential knowledge transfer.** In the ELM, which is a supplementary tool of the DE, a single feedforward neural network (FNN) is used to extract helpful knowledge from individuals' experience. In the FNN, evolutionary data of different tasks are adopted as the input and output data. In this condition, the trained FNN can learn the mapping relationship between individuals' positions and their

search directions among different tasks and then realize the potential knowledge transfer.

The rest of the paper is organized as follows. Section 2 provides an overview of the proposed methodology. Section 3 details the MTDE-ELM framework. Section 4 presents an experimental study on the benchmark problem. Finally, Section 5 summarizes the research work and outlines our outlook for future work.

2. Preliminaries

In this section, we present the background knowledge about MTO. Then, some representative studies of the MFEA are discussed.

2.1. Multitask optimization

Although various EAs have been proposed to solve single-objective and multi-objective problems, real-world applications often require EAs to handle multiple problems simultaneously [34, 35]. Inspired by the ability of computers to handle parallel tasks, many scholars began to pay much attention to MTO [36], in which two or more related tasks in an MTOP need to be optimized simultaneously.

Generally, there are k different tasks, denoted as T_t $(1 \leq t \leq k)$, involved in an MTOP. Without loss of generality, all tasks are minimization problems. Thus, the goal of the MTO is to determine optimal solutions $\mathbf{X}^*(t)$, defined as Eq (2.1), for each task T_t:

$$\mathbf{X}^*(t) = \arg\min f_t(\mathbf{X}(t)), \tag{2.1}$$

where $f_t(x)$ denotes the objective function of x on T_t. Due to that different tasks may have various search spaces [37, 38], that is different dimensions and ranges, operating directly within their original search spaces may hinder efficient knowledge sharing among different tasks. To address this issue, researchers have proposed unified search spaces for all tasks. Specifically, the search spaces of different tasks are first expanded to the maximum dimension $D_{max} = max(D_1, D_2, \ldots, D_k)$ of all the tasks, where D_t $(1 \leq t \leq k)$ is the number of variable dimensions of T_t. Then, normalization is applied to restrict each solution $\mathbf{X}_i(t)$ within its corresponding task's upper bound $\mathbf{UB}(t)$ and lower bound $\mathbf{LB}(t)$. Finally, each decision variable is mapped to a unified search space $(0, 1)^{D_{max}}$. Let $y_{i,j}(t)$ denote the original decision variable of the jth dimension for task t and $x_{i,j}(t)$ be the mapped decision variable of the jth dimension; then a simple linear transformation between $y_{i,j}(t)$ and $x_{i,j}(t)$ can be defined as Eq (2.2).

$$x_{i,j}(t) = lb_j(t) + (ub_j(t) - lb_j(t)) \cdot y_{i,j}(t), \tag{2.2}$$

where $x_{i,j}(t)$, $ub_j(t)$, and $lb_j(t)$ are the jth dimension values of $\mathbf{X}_i(t)$, $\mathbf{UB}(t)$, and $\mathbf{LB}(t)$, respectively. The unification of the search spaces allows an MTO algorithm to transfer knowledge between tasks with different variables and search spaces, and then helpful knowledge of a task can be shared by other tasks. In other words, knowledge sharing and collaboration among different tasks can be achieved.

2.2. Related works

The introduction of the MFEA marked a significant breakthrough in MTO field. The unique innovation of the MFEA lies in its integration of multifactorial genetic theory within a biocultural

modeling framework. In the MFEA, some new metrics introduced to evaluate individuals are defined as follows.

- Factorial cost ψ_i^t is used to evaluate the fitness of $\mathbf{X}_i$ on the task T_t. Generally, ψ_i^t is equal to an optimization problem's objective value if the problem is unconstrained. This metric represents the aggregate difference between an individual's actual objective values and the ideal objective values across all tasks, providing a comprehensive evaluation of the individual's overall performance.
- Factorial rank r_i^t is simply the index of $\mathbf{X}_i$ in the list of population members sorted in ascending order with respect ψ_i^t. The metric reflects the ranking of individuals' fitness for a specific task, indicating their relative performance in the task.
- Scalar fitness ϕ_i is the minimum of the inverse of factorial rank values r_i^t. A higher ϕ_i indicates better performance of an individual in the respective task T_t.
- Skill factor τ_i of $\mathbf{X}_i$ is the one task, amongst all other tasks in MTO, on which the individual is most effective.

To facilitate the effective transfer of knowledge across distinct tasks,the MFEA incorporates two mechanisms: an assortative mating mechanism and a vertical cultural transmission mechanism. Initially, the assortative mating mechanism permits individuals that have diverse skill factors in different tasks to perform a crossover operator. Based on the mechanism, the offspring inherit the combined skill sets of their progenitors and are equipped to selectively adopt the skills of one parent, thereby enhancing their evolutionary potential. Subsequently, the vertical cultural transmission mechanism is conducted, in which the offspring compete with their peers within the task for the opportunity to propagate their genetic traits. The synergistic interplay between the two mechanisms enables the MFEA to efficiently mediate the transfer of knowledge across tasks from parent to offspring. Based on the above discussions, the MFEA can be expressed as Algorithm 1.

Algorithm 1 MFEA.

Input: $T_1, T_2, \ldots, T_k$,the MTOPs with k tasks;
$G_{\max}$, maximum number of iterations;
N,population size of each task;

Initialize generation counter $g = 1$;
Generate an initial population Pop^g;
Evaluate each individual in Pop^g;
Calculate the skill factor τ for each individual in Pop^g;
while $g \leq G_{\max}$ **do**
    $Off^g \leftarrow$ Offspring Generation according to assortative mating;
    Evaluate individuals in Off^g by vertical cultural transmission;
    $Pop^g \cup Off^g \leftarrow$ Combine Pop^g and Off^g;
    $[\phi, \tau] \leftarrow$ Update scalar fitness and skill factors in $Pop^g \cup Off^g$;
    $Pop^{g+1} \leftarrow$ Select N best individuals in $Pop^g \cup Off^g$;
    $g = g + 1$;
Output: $X_1^*, X_2^*, \ldots, X_k^*$,the best solution for tasks;
end while

In the MFEA, individuals oriented to different tasks are placed in the same population. During the

optimization process, knowledge transfer is realized by a fixed crossover on aligned dimensions of different individuals. Furthermore, considering that different crossover operators have a unique bias in generating offspring, Zhou et al. [22] proposed the MFEA-AKT, in which multiple crossover operators are adopted. During the optimization, individuals adaptively select a proper crossover operator for knowledge transfer based on the information collected along the evolutionary process. However, these approaches may still suffer from negative transfer when the relationships between tasks are weak or misleading. To address this issue,the multitask evolutionary algorithm with anomaly detection (MTEA-AD) [39] introduced an anomaly detection mechanism to identify and filter out harmful knowledge during the transfer process. By adaptively regulating the knowledge-sharing behavior, MTEA-AD improves the robustness of the MTO.

Without prior knowledge about the intertask synergies, negative knowledge transfer may appear in a random crossover in the MFEA. Thus, to allay such fear, Bali et al. [36] extended the MFEA to the MFEA-II, in which a novel data-driven approach is used to capture intertask similarities. Then, a transfer parameter matrix, adapted online during the course of the multitasking search, is used to determine the extent of knowledge transfers. This mechanism enables the MFEA-II to respond to real-time feedback and flexibly adjust its search process. Although these adaptive methods improve the effectiveness of knowledge transfer, negative knowledge transfer between uncorrelated tasks also appears in many cases. To alleviate this issue, Bali et al. [40] proposed a linearized domain adaptation strategy, which can transform the search space of a simple task into a search space similar to its constitutive complex task. This high-order representative space resembles a high correlation with its constitutive task and provides a platform for efficient knowledge transfer via crossover. Generally, there is a common phenomenon that the solutions of distinct tasks usually obey different distributions, which may cause individuals after intertask learning to be unsuitable for the original task. Thus, Wang et al. [41] designed a domain adaptation-based mapping strategy to reduce the difference across solution domains and extract more genetic traits to enhance the overall performance of information interactions.

The studies in [40, 41] indicate that getting a better insight of correlations between two heterogeneous tasks' fitness landscape is helpful for the positive knowledge transfer among the tasks. Hence, Han et al. [42] proposed a multitask particle swarm optimization (MTPSO-HDA), in which an adaptive kernel function is utilized to construct a nonlinear mapping between the source and target tasks. Then, a multisource domain adaptive strategy based on fitness landscape similarity is designed to achieve positive knowledge transfer among heterogeneous tasks.

During the last decades, various multipopulation mechanisms have displayed very favorable performance on different EAs [13, 43] because they have an excellent ability to maintain population diversity. Thus, many scholars began to introduce multipopulation mechanisms into the MTO field. In [44], Liang used tasks' similarity, measured by the maximum mean discrepancy, to divide them into multiple groups, and a multisource knowledge transfer mechanism was introduced to determine the probabilities of self-evolution within each task and knowledge transfer among tasks. In 2023, Wu et al. [45] proposed another metric to define the similarity of different tasks, namely a shift invariance. Based on obtained similarity, different tasks can be categorized into different groups. Thus, successful knowledge can efficiently transfer among similar tasks within the same group. It can be found that these research strategies utilize the information generated by individuals or populations to guide the search process. These algorithms all show excellent performance in MTOPs.

Although existing methods have demonstrated the benefits of knowledge transfer, their knowledge utilization mechanisms mainly rely on direct information sharing and simple accumulation strategies. Current methods primarily focus on immediate knowledge transfer between tasks, whereas the potential of systematic knowledge mining and deep cross-domain pattern analysis remains fully explored [46, 47]. A more comprehensive knowledge analysis and association framework can further improve the effectiveness of evolutionary optimization in complex multitasking scenarios. To achieve this, we propose a mechanism for the acquisition, association, and application of historical knowledge designed to guide individuals in evolving toward a more rational direction.

2.3. Motivation

Research on knowledge acquisition is increasingly gaining attention in the field of the MTO. However, efficient integration and utilization of historical knowledge, extraction of valuable insights, and rapid and effective dissemination of this knowledge remain key challenges to be addressed. In the cycle of population evolution, individual data characterization plays a dual role: It not only preserves valuable evolutionary patterns but also serves as a bridge connecting various task domains. This dual functionality suggests that effective knowledge representation and transfer mechanisms are crucial for cross-domain optimization. Notably, capturing, associating, and utilizing knowledge patterns across domains is essential for enhancing optimization performance.

Motivated by these observations, this paper presents the MTDE-ELM, which systematically addresses knowledge management challenges in MTO. It integrates mechanisms for adaptive data acquisition, deep pattern analysis, dynamic knowledge association, and evolutionary trajectory prediction. This integrated approach demonstrates superior capability in identifying beneficial features and evolutionary directions during the data collection phase. Its predictive mechanism enables accurate forecasting of population evolutionary trajectories while facilitating comprehensive knowledge integration across diverse population domains. Experimental results show that this complex knowledge integration process greatly enhances the convergence properties of the algorithm and the stability of the solution in complex MTO environments.

3. Multitask differential evolutionary expert library module

This section presents the details of the proposed MTDE-ELM. First, the DE algorithm is briefly described because the framework MTDE-ELM is based on it. Next, the MTDE-ELM algorithm framework is briefly outlined to guide population evolution effectively. Finally, the operational process of the MTDE-ELM algorithm is described comprehensively.

3.1. Canonical DE algorithm

A DE algorithm is a population-based stochastic optimization technique. During the last few years, many DE variants have displayed favorable performance in different fields, beneficial for the powerful global searching ability and the higher robustness of the DE [48–50]. Similar to the genetic algorithm (GA) [51, 52], a typical DE consists of four basic operators, that is, initialization, mutation, crossover, and selection. The details of the operators in a single task are introduced as follows.

3.1.1. Initialization

An initial population $g = 1$ can be denoted as $Pop^g = \{\mathbf{X}_1^g, \mathbf{X}_2^g, ..., \mathbf{X}_N^g\}$, in which each individual $\mathbf{X}_i^g = \left(x_{i,1}^g, x_{i,2}^g, ..., x_{i,D}^g\right)$ is randomly generated in a feasible space, where D is the number of variables, and N is the population size. After that, the population consecutively applies mutation, crossover, and selection operators to generate offspring in each generation.

3.1.2. Mutation

After initialization, a differential mutation operation is performed among individuals in the population to generate a mutant vector $\mathbf{V}_i^g = \left(v_{i,1}^g, v_{i,2}^g, ..., v_{i,D}^g\right)$ for a target vector $\mathbf{X}_i^g$. The most common and the simplest mutation strategy, named "DE/rand/1", is defined in Eq (3.1).

$$\mathbf{V}_i^g = \mathbf{X}_{r1}^g + F \times (\mathbf{X}_{r2}^g - \mathbf{X}_{r3}^g), \tag{3.1}$$

where $\mathbf{X}_{r1}^g$, $\mathbf{X}_{r2}^g$, and $\mathbf{X}_{r3}^g$ are three individuals. Note that $r1$, $r2$, and $r3$ are mutually exclusive integers randomly selected in the set $\{1, 2, \ldots, N\} \setminus \{i\}$. The parameter F represents the scaling factor that controls the amplitude of the differential variation.

3.1.3. Crossover

After mutation, the mutant vector $\mathbf{V}_i^g$ undergoes a binomial crossover operation with the current individual $\mathbf{X}_i^g$ to generate a new candidate solution $\mathbf{U}_i^g = \left(u_{i,1}^g, u_{i,2}^g, ..., u_{i,D}^g\right)$. Specifically, it is represented as Eq (3.2).

$$u_{i,j}^g = \begin{cases} v_{i,j}^g & \text{if } rand \le CR \text{ or } j = j_r \quad (j \in D); \\ x_{i,j}^g & \text{otherwise,} \end{cases} \tag{3.2}$$

where $rand$ is a random number uniformly distributed in the range [0, 1], CR is the crossover probability, and j_r is an integer randomly generated in the range [1, D].

3.1.4. Selection

Finally, the objective evaluation function value is used to compare the newly generated candidate solution $\mathbf{U}_i^g$ with the current individual $\mathbf{X}_i^g$, selecting the better-performing individual to proceed to the next generation. If the candidate's solution is superior to the current solution, the candidate replaces the current individual. The selection operator is defined as Eq (3.3).

$$\mathbf{X}_i^{g+1} = \begin{cases} \mathbf{X}_i^g & \text{if } f(\mathbf{X}_i^g) < f(\mathbf{U}_i^g); \\ \mathbf{U}_i^g & \text{otherwise,} \end{cases} \tag{3.3}$$

where $f(x)$ is the fitness of the solution x.

3.2. MTDE-ELM framework

In the MTDE-ELM, there are two primary components, an offspring generation module (OGM) and the ELM. In the OGM, a simple DE algorithm is improved to deal with MTOPs. During the optimization process, the improved the DE algorithm can realize both intratask and intertask

knowledge transfer. The role of the ELM is to extract helpful knowledge from individuals' evolution, which can realize potential knowledge transfer within different tasks and then enhance the effectiveness of the OGM. Based on the three knowledge transfers, that is intratask, intertask, and potential knowledge transfer, MTDE-ELM can exhibit more efficient and promising performance. The details of two modules are introduced in the next sections.The pseudocode for the entire procedure of the MTDE-ELM is summarized in Algorithm 2, the basic framework flow of the MTDE-ELM is illustrated in Figure 2.

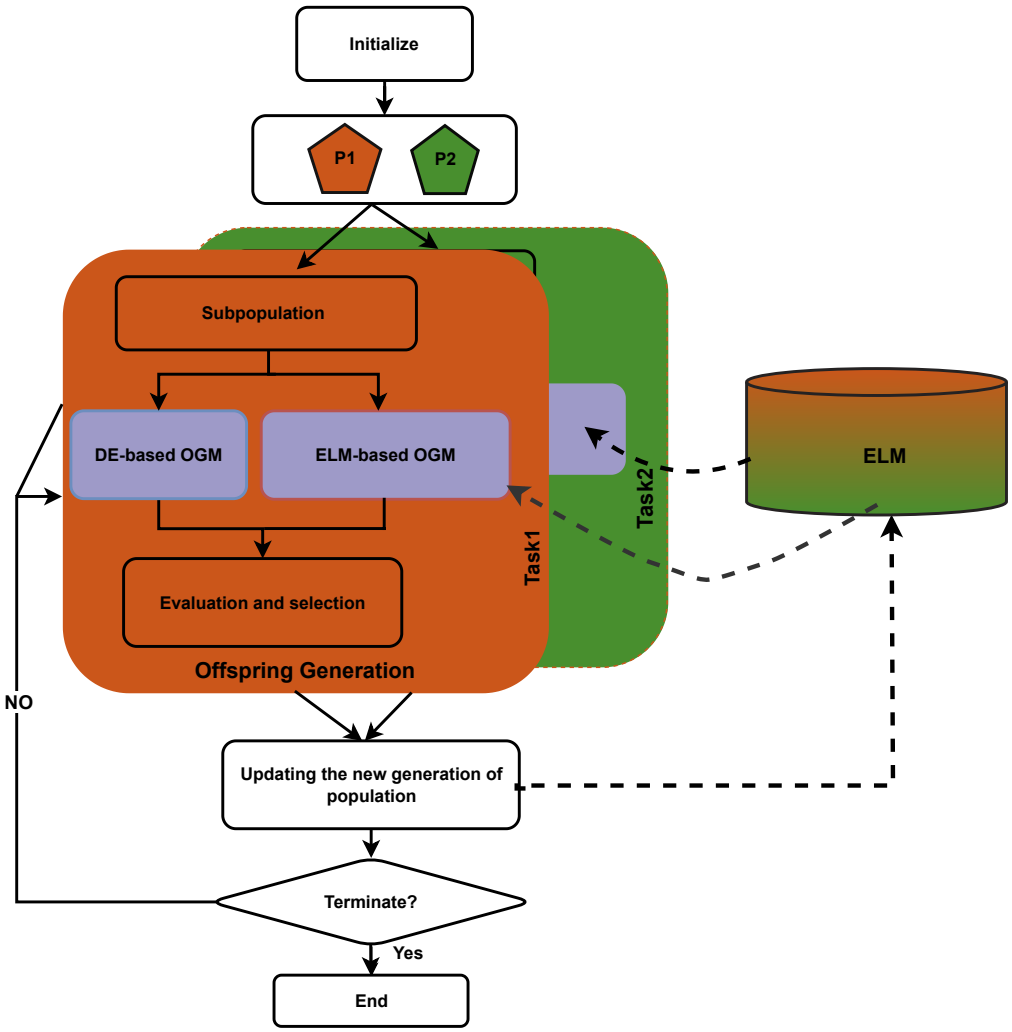

Figure 2. Flow chart of the MTDE-ELM for solving MTOPs with two tasks.

3.3. Expert library module

Although fitness landscapes can be highly complex in global search spaces, their variations are often relatively smooth within local neighborhoods. Therefore, promising search directions and step-size patterns obtained from previous generations can provide useful guidance for the current evolutionary process. In this study, "expert knowledge" is defined as the implicit search experience learned from historical evolutionary processes. Specifically, it includes effective search directions,

Algorithm 2 MTDE-ELM.

Input: $T_1, T_2, \ldots, T_k$; $MaxFES$;
Begin
Set parameters: $g = 1$, $fes = 0$, $trained = 0$, p_u, p_m, F, CR;
Initial Population $Pop^g = \left\{\mathbf{X}_1^g(\tau_1), \mathbf{X}_2^g(\tau_2), ..., \mathbf{X}_N^g(\tau_N)\right\}$;
while $fes \leq MaxFES$ **do**
  for $i = 1$ to N **do**
    $\mathbf{X}_i^{g+1}(\tau_i) = \text{OGM}(\mathbf{X}_i^g(\tau_i), p_u, p_m, F, CR, Pop^g, trained)$;
  end for
  $\mathcal{D}_{in} \leftarrow \left\{\mathbf{X}_1^g(\tau_1), \mathbf{X}_2^g(\tau_2), ..., \mathbf{X}_N^g(\tau_N)\right\}$; // see Eq (3.4)
  $\mathcal{D}_{out}^g \leftarrow \left\{\mathbf{Y}_1^g, \mathbf{Y}_2^g, ..., \mathbf{Y}_N^g\right\}$; // see Eq (3.5)
  Training the FNN in the ELM; // see Section 3.3.2
  $trained = 1$;
  $g = g + 1$; $fes = fes + N$;
end while
Output: $\mathbf{X}_1^*, \mathbf{X}_2^*, \ldots, \mathbf{X}_k^*$, the best solution for tasks

step-size adaptation patterns, and potential relationships among tasks that contribute to generating high-quality solutions. To model and utilize such knowledge, an ELM is introduced. The ELM is implemented using an FNN, which serves as a learning and representation tool rather than the knowledge itself. By training on historical evolutionary data, the FNN learns latent patterns of successful search behaviors and transforms them into a reusable representation. Based on the learned representations, the extracted expert knowledge is used to guide the offspring generation process, thereby improving both convergence speed and solution quality. The detailed mechanism will be introduced in Section 3.4.

In recent years, various neural networks have exhibited distinct properties [28, 53, 54], such as robustness and predictive capabilities, in different study communities. In this study, the FNN [55, 56], which can autonomously extract hidden associations from complex data for effective decision-making, is adopted as a fundamental module of the ELM. Concretely, we intend for the FNN to extract promising search behaviors and potential correlations among different tasks after training and learning. Based on the results of the FNN, the potential knowledge transfer among different tasks can be realized, and successful search directions can be obtained to guide the population.

3.3.1. Input data and output data of FNN

One of the FNN's main objectives in this study is to learn and store the knowledge about the relationship between individuals' positions and their successful evolutionary directions. After training, an individual can query for the knowledge by feeding its current position to the FNN and then obtain the output of the FNN as its evolution direction.

When the FNN performs its training process, two primitive types of data are indispensable, the input data and the predicted output data. Specially, in the MTDE-ELM, two data repositories named $\mathcal{D}_{in}$ and $\mathcal{D}_{out}$ are used to save the input data and the expected output data, respectively.

Concretely, the data repository $\mathcal{D}_{in}$ is used to record parent individuals' position, while the data repository $\mathcal{D}_{out}$ is applied to store successful search directions of the individuals, that is the differential value between offspring's position and the parent individuals' positions. The definitions of $\mathcal{D}_{in}$ and $\mathcal{D}_{out}$ at generation g can be defined as Eqs (3.4) and (3.5), respectively.

$$\mathcal{D}_{in}^{g} = \left\{\mathbf{X}_1^g, \mathbf{X}_2^g, ..., \mathbf{X}_N^g\right\}, \tag{3.4}$$

$$\mathcal{D}_{out}^{g} = \left\{\mathbf{Y}_1^g, \mathbf{Y}_2^g, ..., \mathbf{Y}_N^g\right\}, \tag{3.5}$$

where $\mathbf{Y}_i^g = \mathbf{X}_i^{g+1} - \mathbf{X}_i^g$ $(1 \le i \le N)$.

The principle of "survival of the fittest", as described by Eq (3.3), manifests that offspring are always better than (or at least not inferior to) their parents. Thus, $\mathbf{Y}_i^g$ in Eq (3.5) can be regarded as "successful" search direction of $\mathbf{X}_i$ at generation g.

Another objective of the FNN is to realize the potential knowledge transfer between different tasks. If the data in $\mathcal{D}_{in}$ and $\mathcal{D}_{out}$ are coming from the same task, the knowledge learned by the FNN can help an individual perform a promising search in the same task. In other words, an individual aiming to optimize another task cannot obtain useful knowledge from the FNN because the other task may have different characteristics. Thus, to realize the potential knowledge transfer between different tasks, only one FNN is used in the MTDE-ELM. In the FNN, evolutionary data of multiple tasks are adopted as input data and output data. Specially, the data in $\mathcal{D}_{in}$ and $\mathcal{D}_{out}$ are obtained from different tasks rather than a single task. In this condition, the trained FNN can not only learn the mapping relationship between positions and directions experience but also realize the potential knowledge transfer. Based on the FNN, an individual, no matter which task is optimized by it, the individual can obtain a promising search direction.

3.3.2. Training of the FNN

The FNN training relies on two core mechanisms: forward propagation and backpropagation. These mechanisms are utilized at different evolutionary stages throughout the entire process. It consists of an input layer, hidden layers, and output layer, an information flows between layers to reflect knowledge associations. The input layer corresponds to $\mathcal{D}_{in}$, representing positions within the search space, with input dimensions consistent with the decision variables of the problem. The output layer corresponds to $\mathcal{D}_{out}$, maintaining dimensional consistency with the inputs.

Generally, the complexity of the input data determines the number of hidden layers. While a single hidden layer can approximate any continuous function, complex multitask problems require multiple hidden layers to capture intricate data associations effectively. Increasing the number of hidden layers enhances representational capacity and increases computational costs and training complexity. In this study, a two-hidden-layer structure balances representational capacity and computational complexity, improving adaptability while managing computational and training demands through adaptive mechanisms. Figure 3 shows the architecture of the ELM based on OGM.

In Figure 3, AF denotes the activation function. Therefore, in the FNN architecture, the linear activation function is adopted in the output layer, while a sigmoid activation function is selected in the hidden layers. During the training process of the FNN, for the input data $\mathbf{X}_i^g$, the error backpropagation algorithm is adopted. Concretely, the error between the predicted output of the FNN, denoted as $\tilde{\mathbf{Y}}_i$, and the expected output, that is $\mathbf{Y}_i^g$ in Eq (3.5), is calculated by a mean squared error (MSE) loss

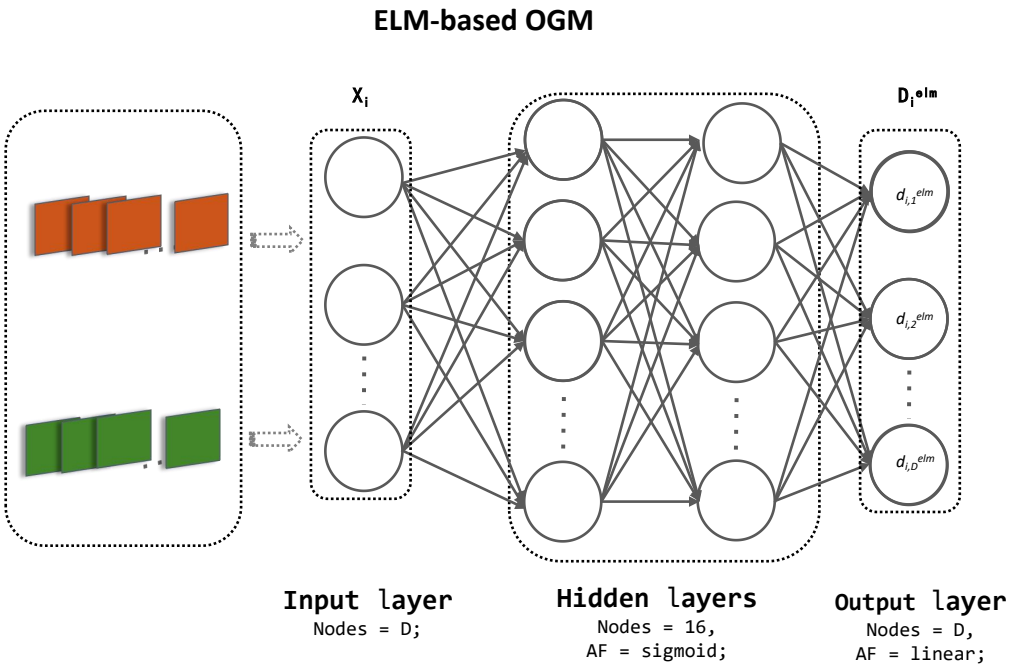

Figure 3. The architecture of ELM-based OGM. The input layer has an input dimension of D. The two hidden layers consist of 16 nodes with sigmoid activation functions, and the output layer consists of D nodes with linear activation functions.

function [57]. Based on the error backpropagated inside the FNN, the weights and biases of the FNN can be adjusted.

During the training process, the epoch parameter represents the number of times the training dataset is processed by the FNN. Each epoch consists of a complete forward and backward pass, in which the FNN processes all training data and updates the weights.

Network training begins with an initial learning rate of 0.1 to facilitate weight convergence. Pattern-based guidance is provided upon successful training during subsequent evolutionary iterations, helping offspring converge toward optimal regions in the search space. In the MTDE-ELM, the epoch is set to 50, which means that the FNN will perform data processing and weight adjustment 50 times during the training process. Multiple trainings will gradually optimize the weights and improve the model's performance. After initialization, the flag will be set to one to indicate that the network is ready for the task.

Although the training data is constructed based on successful offspring, this strategy does not significantly impair the global search ability. This is because the FNN is employed as an auxiliary guidance mechanism rather than a dominant search operator. The evolutionary process is still primarily driven by stochastic operators such as mutation and crossover, which continuously introduce diversity into the population.

Furthermore, the learned knowledge is applied in a probabilistic manner, preventing the search from being overly biased toward previously successful patterns. In addition, the MTO framework enables implicit information exchange across tasks, which further enhances exploration capability. As a result, the proposed training strategy maintains a balance between exploitation and exploration, reducing the

risk of premature convergence.

3.4. Offspring generation module

In the MTDE-ELM, there are two basic components integrated in the OGM. the first is a DE-based offspring generation strategy, which promotes large variations between individuals in the population, facilitating extensive exploration of the global search space. Particularly in the early stages of the optimization process, the strategy can increase the likelihood of identifying potentially optimal solutions. The second basic strategy is an ELM-based offspring generation strategy, which allows the MTDE-ELM to focus on local search regions, leveraging historical successes to refine solutions and enhance the solution quality. Details of the two strategies are introduced as follows.

3.4.1. DE-based offspring generation

In the MTDE-ELM, the basic optimizer is the DE algorithm introduced in Section 3.1. However, the canonical DE focuses on optimizing a single task. Thus, the skill factor is introduced as an additional feature for each individual. Concretely, an individual in the canonical DE is recorded as $\mathbf{X}_i^g(\tau_i)$, where τ_i is the skill factor of $\mathbf{X}_i$ at generation g. Thus, based on the parameter τ_i, the DE variant performs an intratask knowledge transfer or an intertask knowledge transfer to generate offspring, and the DE structure is shown in Figure 4.

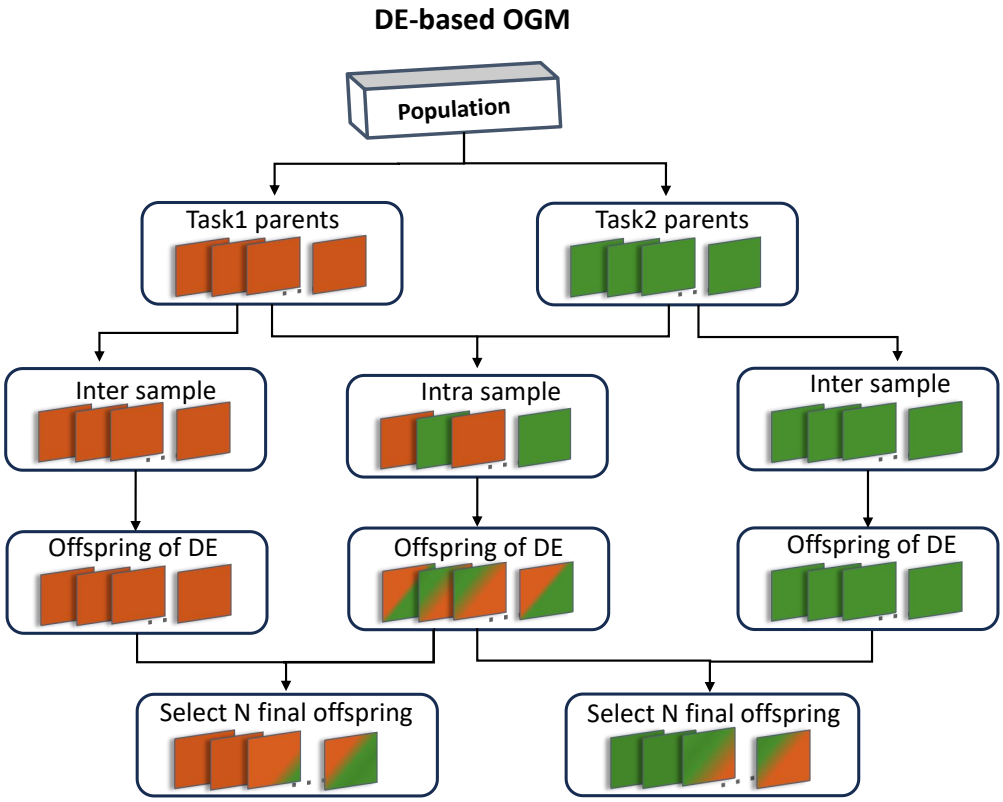

Figure 4. The architecture of DE-based OGM. The DE variant performs an intratask or an intertask knowledge transfer.

When performing the intratask knowledge transfer, a target individual $\mathbf{X}_i^g(\tau_i)$ in the DE only selects individuals who have the same skill factor as the target individual to conduct the mutation and crossover operators. In this condition, only helpful search knowledge for the same task is shared by the individuals, which is why we call it the "intratask knowledge transfer". On the contrary, to realize the intertask knowledge transfer, the target individual in the DE chooses those individuals who have different skill factors from itself to execute the mutation and crossover operators. In this case, the target individual can obtain different tasks' information from other individuals' search experience. In fact, it is through the "intertask knowledge transfer" that an algorithm can optimize different problems simultaneously.

From the above discussions, we see that the intratask knowledge transfer focuses on improving an algorithm's performance on a single task, which is named as the exploitation ability in this study. By contrast, the intertask knowledge transfer is able to leverage the advantage of different tasks and then enhance an algorithm's capability on various tasks, which is called the exploration ability in this study.

To balance the exploitation and exploration of the MTDE-ELM, a preset mating probability, named p_m, is used to determine when to perform the intertask knowledge transfer or the intertask knowledge transfer. Concretely, if a randomly generated number is less than p_m, the intertask knowledge transfer is performed to promote the exploration ability of the MTDE-ELM. Otherwise, the intratask knowledge transfer is conducted to improve exploitation ability of its.

3.4.2. ELM-based offspring generation

When the algorithm selects the knowledge learned by ELM to generate offspring, the current parent individual $\mathbf{X}_i^g(\tau_i)$ is input into the trained FNN for simulation. Then, the output of the FNN, denoted as a vector $\mathbf{D}_i^{elm}$, contains promising search directions and steps for the parent individual $\mathbf{X}_i^g(\tau_i)$. Subsequently, as shown in Eq (3.6), an offspring can be generated based on $\mathbf{X}_i^g(\tau_i)$ and $\mathbf{D}_i^{elm}$.

$$\mathbf{X}_i^{g+1}(\tau_i) = \mathbf{X}_i^g(\tau_i) + 2 \times rand \times \mathbf{D}_i^{elm}, \tag{3.6}$$

where *rand* is a random value generated in the interval [0,1], which controls the strength of the influence of $\mathbf{D}_i^{elm}$ on the offspring.

In summary, when using the ELM-based offspring generation, the specialist knowledge learned by the FNN can guide an individual to adjust its search direction and step, thereby accelerating its search for the optimal solution. In other words, the ELM is like a skilled navigator, helping the algorithm to be more directional and efficient in a complex search space. When choosing the above mentioned strategies to generate offspring, the MTDE-ELM uses a predefined probability, named p_u, to determine which strategy can be adopted. Concretely, if a randomly generated value *rand* is lower than p_u, the DE-based offspring generation strategy is adopted. Conversely, the ELM-based strategy is applied to guide offspring generation. Thus, based on the above discussions, the OGM can be detailed as Algorithm 3.

3.4.3. Complexity analysis

The computational cost of the MTDE-ELM in each generation mainly consists of three parts: DE operations, fitness evaluations, and FNN-based knowledge transfer. The DE operators require $O(KND)$ operations. Fitness evaluations cost $O(KNC_f)$. For the FNN based transfer, each forward propagation

Algorithm 3 OGM.

Input: $\mathbf{X}_i^g(\tau_i)$, p_u, p_m, F, CR, Pop^g, *trained*;
if *rand* $< p_u$ and *trained* $= 1$ **then**
  /* ELM-based offspring generation */
  Adopt $\mathbf{X}_i^g(\tau_i)$ as the input data of the trained FNN; $\mathbf{D}_i^{elm} \leftarrow$ Output of the trained FNN;
  $\mathbf{X}_i^{g+1}(\tau_i) = \mathbf{X}_i^g(\tau_i) + 2 \times rand \times \mathbf{D}_i^{elm}$; // see Eq (3.6)
else
  /* DE-based offspring generation */
  $\mathbf{X}_{r1}^g(\tau_{r1}) \leftarrow$ Randomly select an individuals from Pop^g, where $\tau_{r1}=\tau_i$;
  if *rand* $< p_m$ **then**
    /* Intra-task knowledge transform */
    $[\mathbf{X}_{r2}^g(\tau_{r2}), \mathbf{X}_{r3}^g(\tau_{r2})] \leftarrow$ Randomly select two individuals from population Pop^g, where $i \neq r1 \neq r2 \neq r3$, $\tau_i=\tau_{r2}=\tau_{r3}$;
  else
    /* Inter-task knowledge transform */
    $[\mathbf{X}_{r2}^g(\tau_{r2}), \mathbf{X}_{r3}^g(\tau_{r2})] \leftarrow$ Randomly select two individuals from Pop^g, where $i \neq r1 \neq r2 \neq r3$, $\tau_i \neq \tau_{r2}$ and $\tau_i \neq \tau_{r3}$;
    $\tau_i \leftarrow$ Randomly select a task skill from $\{\tau_{r2}, \tau_{r3}\}$;
  end if
  $\mathbf{V}_i = \mathbf{X}_{r1}^g(\tau_{r1}) + F \times (\mathbf{X}_{r2}^g(\tau_{r2}) - \mathbf{X}_{r3}^g(\tau_{r3}))$; // see Eq (3.1)
  $\mathbf{U}_i \leftarrow$ Crossover $\mathbf{V}_i$ and $\mathbf{X}_i^g(\tau_i)$; // see Eq (3.2)
  if $f(\mathbf{U}_i) < f(\mathbf{X}_i^g(\tau_i))$ **then**
    $\mathbf{X}_i^{g+1}(\tau_i) = \mathbf{U}_i$; // see Eq (3.3)
  end if
end if
Output: $\mathbf{X}_i^{g+1}(\tau_i)$;

costs $O(D \cdot H)$, where H is the number of hidden neurons. With E training epochs, the total cost of the learning module is $O(K \cdot E \cdot N \cdot D \cdot H)$. Because E and H are small constants determined empirically ($E = 50$, $H = 16$ in this study), the overall time complexity per generation simplifies to $O(KNC_f + KND)$. The space complexity is $O(KND + DH)$, which remains linear in D and N.This indicates that the memory requirement grows linearly with the problem scale, ensuring the algorithm's scalability.

4. Experimental studies

In this section, the over all performance of the MTDE-ELM is evaluated by a set of experiments. Moreover, sensitivity of the proposed strategies in the MTDE-ELM is also analyzed through extensive experiments.

4.1. Benchmark functions

In the experimental section, we use the CEC2017 [58] test suite to comprehensively assess the performance of the MTDE-ELM.

Based on the degree of the tasks', the test functions are classified into three categories: complete intersection (CI), partial intersection (PI), and no intersection (NI). Additionally, according to tasks' similarity, the functions can be further divided into high similarity (HS), medium similarity (MS), and low similarity (LS). Thus, the set of test functions exhibits various attributes and characteristics, such as dimensionality and landscape, among others. Additional details about these functions can be found in [59].

The pairings based on task similarity and intersection degree result in nine MTOP instances. Based on task similarity and crossover, nine MTOP instances are created, each with two tasks, yielding a total of 18 test tasks. All experiments were performed with a computer with an Intel Core i9-12900K 3.20 GHz 8-core, CPU 3.20 GHz, and 64.0 GB of RAM.

4.2. Parameter settings

First, in the experiments conducted on the MTOP benchmark, the proposed MTDE-ELM algorithm is compared with five state-of-the-art MTO algorithms, that is MFEA [21], evolutionary multitasking with explicit autoencoding (EMEA) [60], multitasking evolutionary algorithm with an adaptive solver (MTEA-SaO) [61], block-level knowledge transfer based on differential evolution (BLKT-DE) [62], the MTEA-AD [39], and multitask differential evolution with multiple knowledge types and transfer adaptation (MTDE-MKTA) [63]. The parameter settings for the MTDE-ELM and the comparison algorithms are summarized as follows:

1) Population size: $N = 100 \times k$, where k is the number of the tasks.

2) Maximum function evaluations: MaxFEs = 100,000.

3) Simulated binary crossover [64] and polynomial mutation [65] in the MTDE-MKTA and EMEA: $\eta_c = 2$ and $\eta_m = 5$.

4) DE parameters for the MTDE-ELM: $F = 0.5$, $CR = 0.2$.

5) Random mating probability p_m, knowledge utilization probability p_u in the MTDE-ELM: p_m= 0.5, p_u= 0.1.

6) Parameter settings in the FNN: learning rate $lr = 0.1$ and training iteration times $epoch = 50$.

7) Parameters in each peer algorithm not explicitly mentioned are set to the optimal values provided in the corresponding papers.

In the experiments, each algorithm is run independently 30 times to obtain reliable experimental results and minimize the possibility of error. The error value is used as a performance measure, calculated as the gap between the best fitness obtained by the algorithm and the actual best fitness. The Wilcoxon ranksum test [66] with a 5% significance level is employed to statistically evaluate the experimental results. The symbols "+ / - / ≈" indicate that the results of the MTDE-ELM are "significantly better than/significantly worse than/equal to" those of the comparative algorithms based on the Wilcoxon ranksum test.

4.3. Overall performance of the MTDE-ELM

The experimental results, in terms of mean objective values and standard deviations obtained by 30 independent runs, of the MTDE-ELM, MFEA, EMEA, MTEA-SaO, BLKT-DE, MTEA-AD, and MTDE-MKTA on the nine benchmark problems are listed in Table 1. The best experimental result is marked in boldface for each task. Note that the results in each cell of the table denote a mean

objective value and a standard deviation. For example, "$(1.29\times10^{-4}(5.41\times10^{-4}))$" in Table 1 indicates that the mean objective value is 1.29×10^{-4} , and the standard deviation is 5.41×10^{-4}. According to the Wilcoxon ranksum test, the MTDE-ELM achieves competitive overall performance across the 18 tasks. Specifically, it outperforms the MFEA, EMEA, MTEA-SaO, BLKT-DE, MTEA-AD, and MTDE-MKTA in 15, 15, 14, 16, 12, and 10 tasks, respectively, and being inferior in only a few cases. This demonstrates that the MTDE-ELM maintains a clear advantage over both classical and recent MTO methods.

The convergence behaviors of all algorithms are illustrated in Figure 5. For high-similarity tasks (CI-HS, PI-HS, and NI-HS), the MTDE-ELM consistently achieves faster convergence and better final solution quality than the competing methods. This indicates that the proposed approach can effectively exploit strong intertask relationships and leverage transferable knowledge to accelerate the search process. For moderate-similarity tasks (CI-MS, PI-MS, and NI-MS), the MTDE-ELM remains competitive, although its advantage becomes less pronounced. In several cases, it shows slower progress in the early generations but gradually surpasses other algorithms as the evolution proceeds. Nevertheless, in some tasks (e.g., PI-MS-T2), methods such as BLKT-DE and MTEA-AD achieve comparable or slightly better final performance, suggesting that the effectiveness of the knowledge transfer becomes more sensitive to task relationships in such scenarios. For low-similarity tasks (CI-LS, PI-LS, and NI-LS), the performance differences among algorithms are reduced. Although the MTDE-ELM still performs competitively in most cases, it is outperformed in certain tasks (e.g., NI-LS-T1), where methods such as the EMEA, MTEA-SaO, MTEA-AD, and MTDE-MKTA achieve better results. This is mainly due to the limited or misleading transferable knowledge across tasks, which increases the risk of negative transfer and reduces the effectiveness of the learned mappings.

Overall, the results indicate that the MTDE-ELM can effectively exploit useful intertask knowledge and achieve strong performance across diverse optimization scenarios. Its advantage is particularly evident in tasks with high or moderate similarity, whereas in low-similarity scenarios, the benefit of the knowledge transfer diminishes, leading to more competitive performance among different methods.

Table 1. Mean objective values and standard deviations obtained by the MTDE-ELM, MFEA, EMEA, MTEA-SaO, BLKT-DE, MTEA-AD, and MTDE-MKTA on the nine benchmark MTO problems.

Problem	Task	MTDE-ELM	MFEA	EMEA	MTEA-SaO	BLKT-DE	MTEA-AD	MTDE-MKTA
F1:CI+HS	T1	1.29×10^{-4} (5.41×10^{-4})	3.19×10^{-2} (1.86×10^{-2}) -	3.43×10^{-2} (1.16×10^{-2}) -	2.66×10^{-2} (9.70×10^{-3}) -	7.17×10^{-2} (5.93×10^{-2}) -	1.33×10^{-2} (7.24×10^{-3}) -	1.7451×10^{-3} (4.22×10^{-3}) -
	T2	1.70×10^{-1} (6.33×10^{-1})	1.47×10^{2} (3.77×10^{1}) -	4.35×10^{2} (1.95×10^{1}) -	1.58×10^{2} (9.82×10^{1}) -	1.27×10^{2} (4.61×10^{1}) -	1.55×10^{1} (3.26×10^{1}) -	6.4007×10^{0} (1.64×10^{1}) -
F1:CI+MS	T1	3.78×10^{-5} (5.96×10^{-6})	1.23×10^{0} (6.83×10^{-1}) -	4.55×10^{-1} (3.58×10^{-1}) -	3.91×10^{-1} (3.74×10^{-1}) -	4.05×10^{-1} (2.89×10^{-1}) -	1.78×10^{-2} (6.60×10^{-3}) -	6.8890×10^{-2} $(2.28\times10^{-1})\approx$
	T2	8.84×10^{-7} (2.75×10^{-7})	1.55×10^{2} (3.97×10^{1}) -	4.44×10^{2} (1.85×10^{1}) -	7.76×10^{1} (8.65×10^{1}) -	1.23×10^{2} (4.63×10^{1}) -	1.87×10^{-1} (1.40×10^{-1}) -	1.0926×10^{1} $(3.93\times10^{1})\approx$
F1:CI+LS	T1	1.88×10^{1} (5.99×10^{0})	2.03×10^{1} (8.05×10^{-2}) -	2.12×10^{1} $(1.96\times10^{-1})\approx$	2.12×10^{1} $(3.45\times10^{-2})\approx$	2.12×10^{1} (1.21×10^{-1}) -	1.78×10^{1} $(7.40\times10^{0})\approx$	2.0148×10^{1} $(1.22\times10^{-1})\approx$
	T2	4.54×10^{3} (2.13×10^{3})	2.37×10^{3} (3.86×10^{2}) +	1.17×10^{4} (4.23×10^{2}) -	2.23×10^{3} (3.66×10^{2}) +	6.59×10^{3} (1.69×10^{3}) -	4.11×10^{3} $(5.65\times10^{3})\approx$	3.1518×10^{3} (6.16×10^{2}) -
F2:PI+HS	T1	2.99×10^{2} (9.54×10^{1})	2.99×10^{2} $(6.77\times10^{1})\approx$	2.29×10^{2} (8.81×10^{1}) +	2.15×10^{2} (8.39×10^{1}) +	1.67×10^{2} (9.10×10^{1}) +	2.28×10^{2} (4.96×10^{1}) +	1.5059×10^{2} (4.17×10^{1}) +
	T2	2.32×10^{-3} (6.42×10^{-3})	1.21×10^{-2} (1.18×10^{-2}) -	4.36×10^{-1} (1.09×10^{-1}) -	7.05×10^{-2} (4.11×10^{-2}) -	4.49×10^{-1} (4.08×10^{-1}) -	7.82×10^{-3} (3.14×10^{-3}) -	3.4704×10^{-1} (8.46×10^{-1}) -
F2:PI+MS	T1	7.36×10^{-4} (1.27×10^{-4})	6.00×10^{-1} (7.10×10^{-1}) -	3.31×10^{-1} (3.06×10^{-1}) -	3.74×10^{-1} (2.94×10^{-1}) -	3.78×10^{-1} (2.28×10^{-1}) -	4.21×10^{-2} (3.07×10^{-2}) -	4.8974×10^{-2} $(1.95\times10^{-1})\approx$
	T2	8.80×10^{1} (3.47×10^{0})	1.12×10^{2} (3.48×10^{1}) -	1.38×10^{3} (6.54×10^{2}) -	1.75×10^{2} (1.38×10^{2}) -	6.60×10^{1} (1.73×10^{1}) +	8.78×10^{1} $(2.74\times10^{0})\approx$	8.8206×10^{1} $(4.16\times10^{0})\approx$
F2:PI+LS	T1	1.10×10^{-1} (2.10×10^{-1})	1.56×10^{1} (7.98×10^{0}) -	4.37×10^{-1} (3.23×10^{-1}) -	5.17×10^{-1} (4.04×10^{-1}) -	4.99×10^{0} (7.27×10^{0}) -	9.92×10^{-2} $(1.45\times10^{-1})\approx$	9.4918×10^{-1} (1.29×10^{0}) -
	T2	3.40×10^{-2} (2.52×10^{-2})	1.47×10^{1} (7.72×10^{0}) -	3.44×10^{-1} (4.72×10^{-1}) -	7.91×10^{-1} (4.37×10^{-1}) -	2.47×10^{0} (2.33×10^{0}) -	2.60×10^{-1} (1.47×10^{-1}) -	5.1855×10^{-1} (7.24×10^{-1}) -
F3:NI+HS	T1	6.53×10^{1} (2.04×10^{1})	1.96×10^{2} (5.92×10^{1}) -	8.77×10^{2} (1.30×10^{3}) -	2.35×10^{2} (6.07×10^{1}) -	1.47×10^{2} (9.40×10^{1}) -	1.58×10^{2} (6.78×10^{1}) -	1.4496×10^{2} (1.13×10^{2}) -
	T2	7.42×10^{0} (9.68×10^{0})	1.73×10^{2} (5.09×10^{1}) -	4.30×10^{2} (2.26×10^{1}) -	2.05×10^{2} (5.45×10^{1}) -	1.39×10^{2} (4.78×10^{1}) -	1.40×10^{2} (1.29×10^{2}) -	1.0402×10^{2} (4.14×10^{1}) -
F3:NI+MS	T1	9.52×10^{-4} (8.97×10^{-4})	3.40×10^{-2} (1.86×10^{-2}) -	3.30×10^{-2} (1.51×10^{-2}) -	3.45×10^{-2} (1.73×10^{-2}) -	1.05×10^{-1} (6.05×10^{-2}) -	1.94×10^{-2} (8.35×10^{-3}) -	6.6461×10^{-3} $(1.05\times10^{-2})\approx$
	T2	1.19×10^{0} (6.93×10^{-1})	2.12×10^{1} (5.81×10^{0}) -	3.04×10^{1} (1.18×10^{1}) -	1.32×10^{1} (5.49×10^{0}) -	3.39×10^{0} (9.65×10^{-1}) -	3.43×10^{0} (1.09×10^{0}) -	1.0016×10^{1} (2.81×10^{0}) -
F3:NI+LS	T1	3.36×10^{2} (8.25×10^{1})	3.10×10^{2} $(7.34\times10^{1})\approx$	1.82×10^{2} (6.83×10^{1}) +	2.00×10^{2} (7.39×10^{1}) +	2.00×10^{2} (7.63×10^{1}) +	2.36×10^{2} (1.25×10^{1}) +	1.6896×10^{2} (4.00×10^{1}) +
	T2	7.71×10^{2} (2.90×10^{2})	2.31×10^{3} (3.24×10^{2}) -	1.16×10^{4} (5.12×10^{2}) -	2.34×10^{3} (4.30×10^{2}) -	1.92×10^{3} (6.39×10^{2}) -	1.24×10^{3} (3.53×10^{2}) -	3.6322×10^{3} (6.98×10^{2}) -
+/-/≈		Base	1/ 15 / 2	2 / 15 / 1	3 / 14 / 1	2 / 16 / 0	2 / 12 / 4	2 / 10 / 6

4.4. Sensitivity of proposed strategies in the MTDE-ELM

In order to evaluate the effectiveness of the MTDE-ELM, two aspects of its performance are investigated in this section.

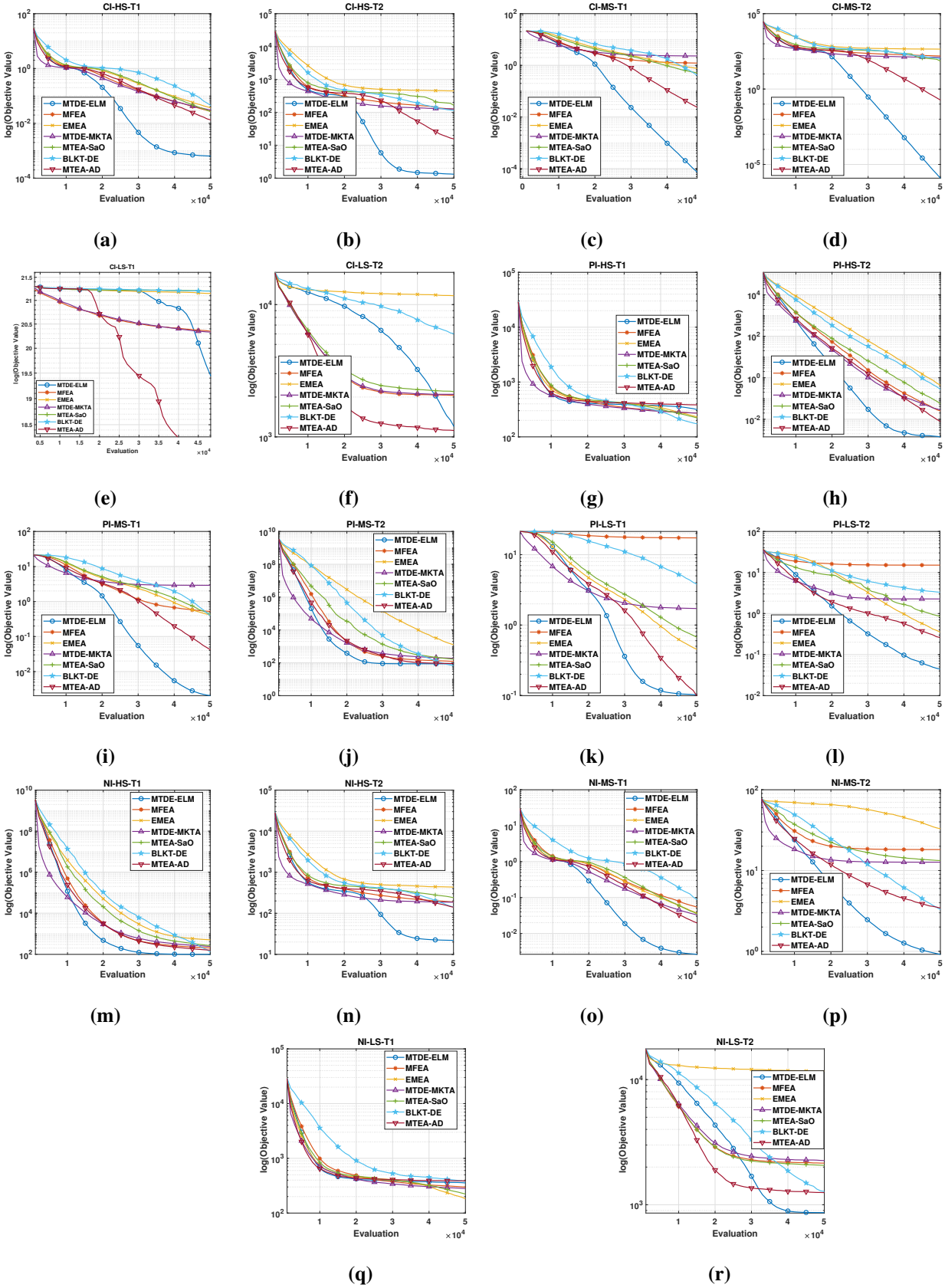

Figure 5. Average convergence trends of the MTDE-ELM, MFEA, EMEA, MTEA-SaO, BLKT-DE, MTEA-AD and MTDE-MKTA on benchmark.

4.4.1. Performance of the ELM

To evaluate the effectiveness of the ELM in the MTDE-ELM, we conduct a comparison between the MTDE-ELM and its variant, named MTDE, in which the ELM is discarded. Concretely, the offspring generation in the MTDE only depends on the DE algorithm with the intratask and the intertask knowledge transfers. All experimental parameters remained consistent with prior configurations. Table 2 presents the average objective values and standard deviations obtained from 30 independent runs of the MTDE-ELM and MTDE on nine benchmark test suites in MTO, with the best results highlighted in bold.

As shown in Table 2, the MTDE-ELM outperforms MTDE on 18 tasks, demonstrating its superior performance. These results validate the effectiveness of incorporating an NN structure for knowledge acquisition, association, and application strategies. Therefore, the experimental findings confirm that processing historical successful experiences through an NN is essential for enhancing the performance of the MTDE. The convergence trends of the MTDE-ELM and MTDE for PI-MS and NI-LS are depicted in Figure 6. It can be observed from the figure that both algorithms exhibit similar convergence behavior during the initial evolutionary phase, as they both rely on stochastic knowledge to guide the generation of offspring. However, the convergence of the MTDE-ELM is significantly improved by leveraging historical evolutionary experience as the population evolves. It illustrates the effectiveness of the ELM in facilitating the search for the best solution to the task.

Table 2. Mean objective values and standard deviations obtained by the MTDE-ELM and MTDE on benchmarks.

Problem	Task	MTDE-ELM	MTDE
F1:CI-HS	T1	1.03×10^{-4} (1.91×10^{-3})	4.64×10^{-1} (1.20×10^{-1}) -
	T2	2.71×10^{-1} (8.73×10^{-1})	3.65×10^{2} (1.66×10^{1}) -
F1:CI-MS	T1	4.09×10^{-5} (1.02×10^{-5})	2.67×10^{0} (2.77×10^{-1}) -
	T2	6.16×10^{-6} (6.40×10^{-7})	3.59×10^{2} (1.83×10^{1}) -
F1:CI-LS	T1	1.76×10^{1} (5.94×10^{0})	2.12×10^{1} (4.07×10^{-2}) -
	T2	2.18×10^{3} (1.04×10^{3})	1.40×10^{4} (6.62×10^{2}) -
F2:PI-HS	T1	2.85×10^{2} (1.06×10^{2})	4.22×10^{2} (1.93×10^{1}) -
	T2	2.24×10^{-3} (1.89×10^{-2})	1.24×10^{1} (5.10×10^{0}) -
F2:PI-MS	T1	1.06×10^{-3} (2.31×10^{-3})	2.86×10^{0} (3.03×10^{-1}) -
	T2	7.72×10^{1} (6.50×10^{-1})	1.05×10^{3} (3.51×10^{2}) -
F2:PI-LS	T1	1.12×10^{-1} (2.20×10^{-1})	1.29×10^{1} (5.01×10^{0}) -
	T2	3.05×10^{-2} (2.85×10^{-2})	7.93×10^{0} (3.47×10^{0}) -
F3:NI-HS	T1	8.47×10^{1} (3.27×10^{1})	1.17×10^{3} (4.38×10^{2}) -
	T2	1.59×10^{1} (1.56×10^{1})	3.80×10^{2} (1.77×10^{1}) -
F3:NI-MS	T1	2.42×10^{-3} (3.15×10^{-3})	5.22×10^{-1} (9.03×10^{-2}) -
	T2	1.28×10^{0} (8.38×10^{-1})	8.35×10^{0} (8.99×10^{-1}) -
F3:NI-LS	T1	3.44×10^{2} (5.63×10^{1})	4.58×10^{2} (2.28×10^{1}) -
	T2	7.35×10^{2} (2.93×10^{2})	7.76×10^{3} (1.13×10^{3}) -
$+/-/\approx$		Base	0 / 18 / 0

4.4.2. Performance of the FNN in ELM

In the MTDE-ELM, a single FNN is adopted in the ELM aiming to realize the potential knowledge transfer. However, the question remains of whether introducing an FNN for each task can bring more outstanding performance. In order to address this question, a comparison is made between the MTDE-ELM and its variant,the MTDE-DOUBLE-ELM. In this variant, each task has its own FNN to learn the mapping relationship between positions and directions experience. This comparison aims to evaluate the advantage of the MTDE-ELM in predicting the evolution of the population by learning

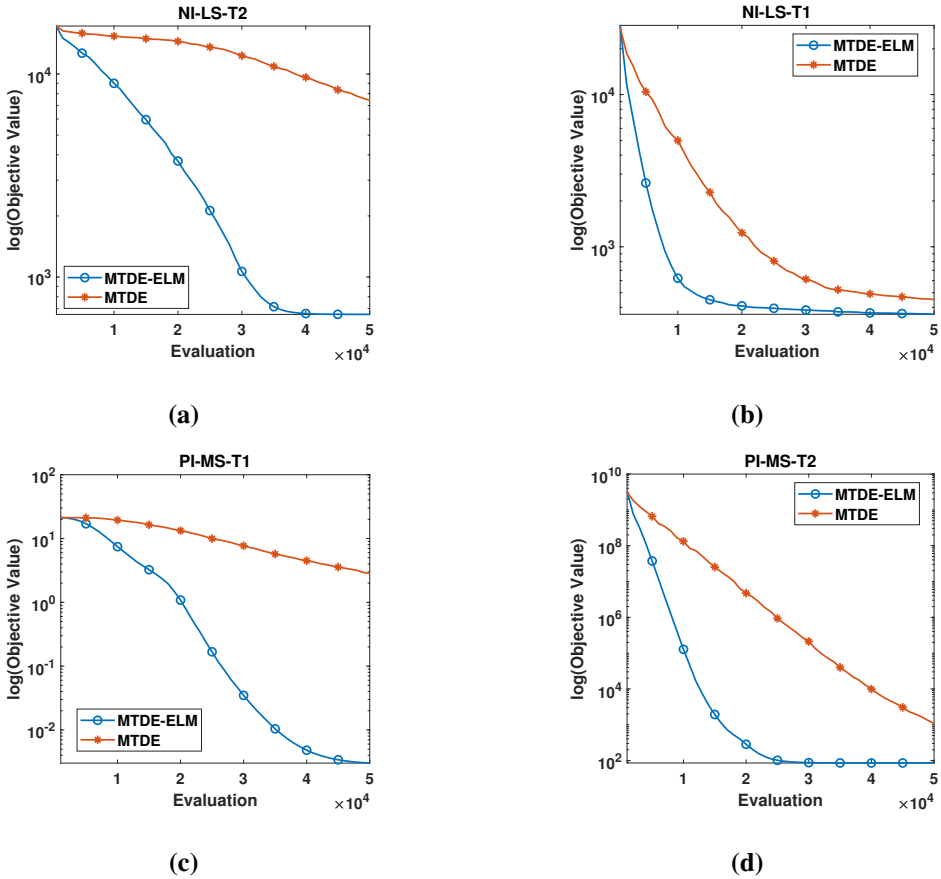

Figure 6. Average convergence trends of the MTDE-ELM and MTDE on benchmark.

the evolutionary patterns from the historical success experiences of different individuals across tasks. Table 3 presents the results of 30 independent runs of both algorithms on nine benchmark suites of the MTO. Under identical algorithm configurations, the MTDE-ELM exhibits superior performance across 17 tasks. Nevertheless, in the PI-HS test task, particularly for T1,the MTDE-DOUBLE-ELM demonstrates a marked advantage.

The convergence curves for the MTDE-ELM and MTDE-DOUBLE-ELM are shown in Figure 7. From the convergence plot of NI-HS, it can be observed that, although both algorithms exhibit similar convergence trends, the MTDE-ELM converges significantly faster than the MTDE-DOUBLE-ELM, indicating its stronger optimization capability throughout the entire process. In the PI-HS-T1 convergence plot, the MTDE-ELM converges more rapidly in the early stages, but as the population evolves, the MTDE-DOUBLE-ELM ultimately achieves a better solution. Conversely, in PI-HS-T2, the MTDE-ELM consistently outperforms the MTDE-DOUBLE-ELM across the entire iteration, demonstrating its stability and robust optimization ability in this task.

In contrast to the MTDE-DOUBLE-ELM, which trains separate NNs for each subpopulation's historical success experience, the MTDE-ELM integrates the historical success experiences of the entire population into a unified NN model. By consolidating the information from all subpopulations within a single model,the MTDE-ELM is able to learn the evolutionary patterns of other optimization problems and incorporate these successful experiences to guide the evolution of its own individuals.

Consequently, the MTDE-ELM, through the training of a common NN, not only enhances its ability for cross-task learning but also improves the robustness and adaptability of the algorithm, demonstrating its broad applicability in MTO.

Table 3. Mean objective values and standard deviations obtained by the MTDE-ELM and MTDE-DOUBLE-ELM on benchmarks.

Problem	Task	MTDE-ELM	MTDE-DOUBLE-ELM
F1:CI-HS	T1	1.31×10^{-4} (6.92×10^{-4})	1.50×10^{-3} (4.55×10^{-4}) -
	T2	1.81×10^{-1} (5.01×10^{-1})	3.55×10^{0} (1.26×10^{0}) -
F1:CI-MS	T1	1.09×10^{-4} (3.47×10^{-4})	1.43×10^{-2} (3.01×10^{-3}) -
	T2	7.79×10^{-6} (3.77×10^{-4})	1.20×10^{-1} (4.91×10^{-2}) -
F1:CI-LS	T1	1.99×10^{1} (4.25×10^{0})	2.12×10^{1} (4.72×10^{-2}) -
	T2	3.86×10^{3} (1.82×10^{3})	1.54×10^{4} (4.08×10^{2}) -
F2:PI-HS	T1	3.12×10^{2} (8.38×10^{1})	2.19×10^{2} (9.13×10^{1}) **+**
	T2	1.61×10^{-3} (4.10×10^{-3})	7.78×10^{3} (3.19×10^{2}) -
F2:PI-MS	T1	7.78×10^{-4} (2.11×10^{-4})	1.43×10^{-2} (3.42×10^{-3}) -
	T2	8.72×10^{1} (6.19×10^{-1})	1.19×10^{2} (1.71×10^{0}) -
F2:PI-LS	T1	1.47×10^{-1} (2.78×10^{-1})	3.71×10^{0} (1.23×10^{0}) -
	T2	3.63×10^{-2} (1.10×10^{-1})	4.91×10^{-2} (1.30×10^{-2}) -
F3:NI-HS	T1	7.89×10^{1} (4.13×10^{1})	6.45×10^{3} (1.52×10^{3}) -
	T2	1.12×10^{1} (1.03×10^{1})	2.88×10^{2} (9.61×10^{1}) -
F3:NI-MS	T1	1.26×10^{-3} (1.16×10^{-3})	1.48×10^{-3} (4.03×10^{-4}) -
	T2	9.58×10^{-1} (6.14×10^{-1})	3.14×10^{1} (7.17×10^{-1}) -
F3:NI-LS	T1	3.52×10^{2} (5.12×10^{1})	2.62×10^{4} (1.34×10^{3}) -
	T2	6.97×10^{2} (2.95×10^{2})	5.77×10^{3} (1.12×10^{3}) -
+/ - / ≈		Base	1 / 17 / 0

4.5. Sensitivity analysis

To validate the effectiveness and robustness of the adopted FNN configuration, we conduct a comprehensive sensitivity analysis on key architectural and training parameters, including network depth, network width, learning rate, training epochs, and the performance of p_m and p_u. All experiments are conducted on the same benchmark suite, and the statistical results are reported in terms of $+/-/\approx$ based on the Wilcoxon ranksum test.

4.5.1. Analysis of network depth

To investigate the impact of network depth, three configurations are evaluated: 1×16, 2×16, and 3×16. As shown in Table 4, the proposed the MTDE-ELM with a two-hidden-layer structure (2×16) demonstrates consistently competitive performance across the majority of tasks. Specifically, although deeper architectures (3×16) achieve superior performance on several individual tasks (F3:NI+HS and F3:NI+MS), they also exhibit noticeable performance degradation on other tasks (F1:CI+MS and F1:CI+LS), indicating a lack of stability. In contrast, the shallower structure (1×16) tends to show comparable or slightly inferior results, suggesting limited representation capability for modeling complex intertask relationships.

Overall, the 2×16 configuration achieves the best balance between performance and robustness, as evidenced by its largest number of statistically comparable results. This suggests that a moderate network depth is sufficient to capture useful knowledge transfer patterns while avoiding overfitting or instability introduced by overly complex architectures.

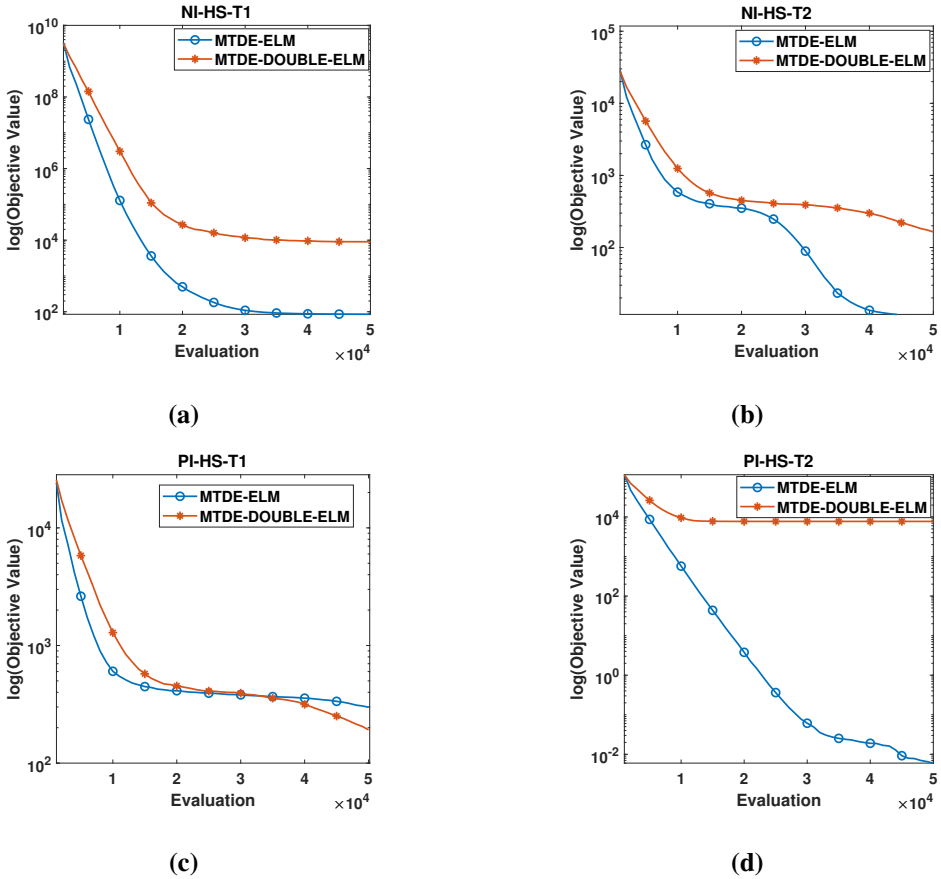

Figure 7. Average convergence trends of the MTDE-ELM and MTDE-DOUBLE-ELM on benchmark.

Table 4. Sensitivity analysis of FNN depth configurations in the MTDE-ELM.

Problem	Task	MTDE-ELM(2×16)	1×16	3×16
F1:CI+HS	T1	$5.07 \times 10^{-4}(9.20 \times 10^{-4})$	$4.58 \times 10^{-4}(1.11 \times 10^{-3})\approx$	$7.16 \times 10^{-5}(1.65 \times 10^{-4})\approx$
	T2	$6.19 \times 10^{-1}(8.52 \times 10^{-1})$	$3.59 \times 10^{-1}(6.90 \times 10^{-1})\approx$	$1.37 \times 10^{-1}(3.14 \times 10^{-1})\approx$
F1:CI+MS	T1	$4.40 \times 10^{-5}(8.21 \times 10^{-6})$	$2.45 \times 10^{-4}(8.93 \times 10^{-4})\approx$	$5.86 \times 10^{-4}(6.83 \times 10^{-5})$-
	T2	$1.29 \times 10^{-6}(4.66 \times 10^{-7})$	$4.94 \times 10^{-4}(2.21 \times 10^{-3})\approx$	$2.12 \times 10^{-4}(5.77 \times 10^{-5})$-
F1:CI+LS	T1	$1.99 \times 10^{1}(2.59 \times 10^{0})$	$1.97 \times 10^{1}(3.00 \times 10^{0})\approx$	$2.11 \times 10^{1}(5.88 \times 10^{-2})$-
	T2	$1.74 \times 10^{3}(1.10 \times 10^{3})$	$1.64 \times 10^{3}(1.28 \times 10^{3})\approx$	$5.53 \times 10^{3}(1.39 \times 10^{3})$-
F2:PI+HS	T1	$2.67 \times 10^{2}(1.07 \times 10^{2})$	$3.38 \times 10^{2}(4.98 \times 10^{1})$-	$2.99 \times 10^{2}(9.54 \times 10^{1})\approx$
	T2	$1.79 \times 10^{-3}(5.61 \times 10^{-3})$	$2.50 \times 10^{-2}(8.81 \times 10^{-2})\approx$	$8.37 \times 10^{-5}(2.37 \times 10^{-5})\approx$
F2:PI+MS	T1	$4.68 \times 10^{-4}(7.13 \times 10^{-4})$	$5.50 \times 10^{-3}(2.09 \times 10^{-2})\approx$	$2.57 \times 10^{-3}(4.34 \times 10^{-4})$-
	T2	$8.70 \times 10^{1}(4.42 \times 10^{-1})$	$8.72 \times 10^{1}(6.08 \times 10^{-1})\approx$	$8.47 \times 10^{1}(8.76 \times 10^{0})$+
F2:PI+LS	T1	$1.04 \times 10^{-1}(2.56 \times 10^{-1})$	$4.72 \times 10^{-2}(1.01 \times 10^{-1})\approx$	$9.69 \times 10^{-3}(2.20 \times 10^{-3})\approx$
	T2	$1.46 \times 10^{-2}(1.22 \times 10^{-2})$	$4.78 \times 10^{-2}(7.87 \times 10^{-2})\approx$	$1.51 \times 10^{-2}(6.86 \times 10^{-3})\approx$
F3:NI+HS	T1	$9.15 \times 10^{1}(4.13 \times 10^{1})$	$1.01 \times 10^{2}(3.42 \times 10^{1})\approx$	$6.00 \times 10^{1}(2.22 \times 10^{1})$+
	T2	$1.62 \times 10^{1}(1.57 \times 10^{1})$	$1.74 \times 10^{1}(1.36 \times 10^{1})\approx$	$1.15 \times 10^{1}(1.62 \times 10^{1})\approx$
F3:NI+MS	T1	$1.13 \times 10^{-3}(8.78 \times 10^{-4})$	$1.37 \times 10^{-3}(1.28 \times 10^{-3})\approx$	$4.86 \times 10^{-4}(1.04 \times 10^{-4})$+
	T2	$1.39 \times 10^{0}(8.31 \times 10^{-1})$	$1.12 \times 10^{0}(6.50 \times 10^{-1})\approx$	$4.48 \times 10^{-1}(8.06 \times 10^{-2})$+
F3:NI+LS	T1	$2.99 \times 10^{2}(9.32 \times 10^{1})$	$3.62 \times 10^{2}(2.33 \times 10^{1})$-	$3.66 \times 10^{2}(3.34 \times 10^{1})$-
	T2	$8.32 \times 10^{2}(2.95 \times 10^{2})$	$6.91 \times 10^{2}(2.32 \times 10^{2})\approx$	$6.96 \times 10^{2}(2.57 \times 10^{2})\approx$
+/-/ ≈		Base	0 /4 / 14	4 / 6 / 8

4.5.2. Analysis of network width

Further, three configurations: 2×8, 2×16, and 2×32, are evaluated to investigate the impact of network width, as reported in Table 5. The results show that increasing the number of neurons

does not lead to consistent performance improvement. Although the wider configuration (2×32) achieves superior results on several tasks (F1:CI+HS and F3:NI+HS), it also suffers from noticeable performance degradation on others (F2:PI+MS and F3:NI+LS). Similarly, the narrower configuration (2×8) reduces the model capacity and performs better in a few cases (F3:NI+LS-T1), but yields inferior results on several other tasks.

Overall, the 2×16 configuration achieves the most stable performance across different tasks, as evidenced by the largest number of statistically comparable results. This indicates that a moderate network width is sufficient to capture useful knowledge patterns while avoiding the instability introduced by overly small or excessively large models. These observations suggest that the performance of the MTDE-ELM is not highly sensitive to moderate variations in network width, but extreme configurations may introduce either underfitting or overfitting issues. Therefore, the adopted setting (2×16) provides a reasonable trade-off between representation capability, generalization, and computational efficiency.

Table 5. Sensitivity analysis of FNN width configurations in the MTDE-ELM.

Problem	Task	MTDE-ELM(2×16)	2×8	2×32
F1:CI+HS	T1	$5.07 \times 10^{-4}(9.20 \times 10^{-4})$	$4.41 \times 10^{-4}(6.54 \times 10^{-4})\approx$	$8.03 \times 10^{-5}(2.51 \times 10^{-4})$ +
	T2	$6.19 \times 10^{-1}(8.52 \times 10^{-1})$	$5.19 \times 10^{-1}(9.59 \times 10^{-1})\approx$	$1.84 \times 10^{-1}(5.25 \times 10^{-1})$ +
F1:CI+MS	T1	$3.34 \times 10^{-4}(1.02 \times 10^{-3})$	$2.12 \times 10^{-4}(7.79 \times 10^{-4})\approx$	$2.12 \times 10^{-4}(7.76 \times 10^{-4}) \approx$
	T2	$6.61 \times 10^{-4}(2.75 \times 10^{-3})$	$3.64 \times 10^{-4}(1.63 \times 10^{-3})\approx$	$3.93 \times 10^{-4}(1.75 \times 10^{-3}) \approx$
F1:CI+LS	T1	$1.76 \times 10^{1}(6.07 \times 10^{0})$	$1.95 \times 10^{1}(4.71 \times 10^{0})\approx$	$1.89 \times 10^{1}(4.68 \times 10^{0}) \approx$
	T2	$1.14 \times 10^{3}(9.25 \times 10^{2})$	$1.41 \times 10^{3}(8.33 \times 10^{2})\approx$	$1.65 \times 10^{3}(1.41 \times 10^{3}) \approx$
F2:PI+HS	T1	$2.57 \times 10^{2}(1.09 \times 10^{2})$	$2.62 \times 10^{2}(1.17 \times 10^{2})\approx$	$3.19 \times 10^{2}(8.94 \times 10^{1})$ -
	T2	$3.19 \times 10^{-4}(4.65 \times 10^{-4})$	$1.27 \times 10^{-2}(3.65 \times 10^{-2})\approx$	$4.60 \times 10^{-3}(1.77 \times 10^{-2}) \approx$
F2:PI+MS	T1	$5.88 \times 10^{-4}(8.14 \times 10^{-4})$	$8.40 \times 10^{-4}(1.71 \times 10^{-3})\approx$	$3.20 \times 10^{-3}(6.46 \times 10^{-3})$ -
	T2	$8.69 \times 10^{1}(3.76 \times 10^{-1})$	$8.72 \times 10^{1}(7.83 \times 10^{-1})\approx$	$8.77 \times 10^{1}(2.55 \times 10^{0}) \approx$
F2:PI+LS	T1	$5.23 \times 10^{-2}(1.07 \times 10^{-1})$	$1.95 \times 10^{-1}(4.32 \times 10^{-1})\approx$	$2.13 \times 10^{-1}(3.77 \times 10^{-1}) \approx$
	T2	$1.24 \times 10^{-2}(8.51 \times 10^{-3})$	$1.70 \times 10^{-1}(4.07 \times 10^{-1})$-	$4.44 \times 10^{-2}(5.79 \times 10^{-2})$ -
F3:NI+HS	T1	$9.90 \times 10^{1}(3.17 \times 10^{1})$	$9.62 \times 10^{1}(3.45 \times 10^{1})\approx$	$8.23 \times 10^{1}(3.26 \times 10^{1})$ +
	T2	$1.28 \times 10^{1}(7.17 \times 10^{0})$	$1.90 \times 10^{1}(1.53 \times 10^{1})\approx$	$1.76 \times 10^{1}(2.18 \times 10^{1}) \approx$
F3:NI+MS	T1	$1.79 \times 10^{-3}(1.90 \times 10^{-3})$	$1.07 \times 10^{-3}(1.16 \times 10^{-3})\approx$	$1.52 \times 10^{-3}(2.39 \times 10^{-3}) \approx$
	T2	$7.76 \times 10^{-1}(4.06 \times 10^{-1})$	$1.39 \times 10^{0}(6.35 \times 10^{-1})$-	$1.30 \times 10^{0}(7.31 \times 10^{-1})$ -
F3:NI+LS	T1	$3.19 \times 10^{2}(6.70 \times 10^{1})$	$2.34 \times 10^{2}(1.22 \times 10^{2})$+	$3.59 \times 10^{2}(3.00 \times 10^{1})$ -
	T2	$7.60 \times 10^{2}(2.76 \times 10^{2})$	$9.97 \times 10^{2}(2.89 \times 10^{2})$-	$7.99 \times 10^{2}(2.65 \times 10^{2}) \approx$
+/−/≈		Base	1 /3 / 14	3 / 5 / 10

4.5.3. Analysis of learning rate

The learning rate (lr) determines the step size of network weight updates. The learning rate is evaluated over the set in $\{0.05, 0.1, 0.15, 0.2, 0.25\}$, and the results are summarized in Table 6. The results show that extreme learning rates lead to performance degradation. When $lr = 0.05$, the model is undertrained within the limited number of epochs, resulting in inferior performance on most tasks. Conversely, a large learning rate ($lr = 0.25$) introduces instability due to overshooting and noisy updates. Moderate learning rates ($lr = 0.15$ and $lr = 0.2$) achieve better performance on some tasks but exhibit higher variability. In contrast, $lr = 0.1$ provides the most stable performance across tasks, indicating a desirable balance between convergence speed and training stability.

4.5.4. Analysis of training epochs

To justify the parameter settings of the integrated FNN, we investigate the impact of the number of training epochs ($epochs$) on the performance of the MTDE-ELM. Specifically, four configurations of $epochs \in \{10, 20, 50, 100\}$, are evaluated across all benchmark tasks. As illustrated in Table 7, the

Table 6. Sensitivity analysis of the learning rate of the MTDE-ELM.

Problem	Task	MTDE-ELM(lr = 0.1)	lr = 0.05	lr = 0.15	lr = 0.2	lr = 0.25
F1:CI+HS	T1	$2.60\times10^{-4}(8.14\times10^{-4})$	$6.63\times10^{-3}(3.30\times10^{-3})$-	$3.34\times10^{-6}(1.41\times10^{-6})\approx$	$1.03\times10^{-4}(2.11\times10^{-4})\approx$	$1.11\times10^{-3}(1.71\times10^{-3})$-
	T2	$2.62\times10^{-1}(5.85\times10^{-1})$	$8.40\times10^{0}(3.23\times10^{0})$-	$5.22\times10^{-3}(2.29\times10^{-3})\approx$	$1.37\times10^{-1}(2.58\times10^{-1})\approx$	$1.73\times10^{0}(2.75\times10^{0})$-
F1:CI+MS	T1	$4.12\times10^{-5}(6.43\times10^{-6})$	$3.00\times10^{-2}(3.31\times10^{-3})$-	$6.40\times10^{-4}(1.15\times10^{-4})$-	$2.41\times10^{-3}(2.75\times10^{-4})$-	$1.25\times10^{-2}(3.33\times10^{-3})$-
	T2	$1.10\times10^{-6}(3.88\times10^{-7})$	$4.77\times10^{-1}(1.06\times10^{-1})$-	$2.63\times10^{-4}(8.65\times10^{-5})$-	$3.67\times10^{-3}(1.04\times10^{-3})$-	$8.92\times10^{-2}(5.06\times10^{-2})$-
F1:CI+LS	T1	$1.98\times10^{1}(2.39\times10^{0})$	$2.11\times10^{1}(3.89\times10^{-2})$-	$2.12\times10^{1}(3.26\times10^{-2})$-	$2.11\times10^{1}(2.10\times10^{-1})$-	$2.11\times10^{1}(4.66\times10^{-2})$-
	T2	$1.31\times10^{3}(8.33\times10^{2})$	$1.08\times10^{4}(4.29\times10^{2})$-	$4.55\times10^{3}(1.23\times10^{3})$-	$4.32\times10^{3}(9.72\times10^{2})$-	$5.81\times10^{3}(1.02\times10^{3})$-
F2:PI+HS	T1	$3.22\times10^{2}(8.10\times10^{1})$	$3.86\times10^{2}(1.81\times10^{1})$-	$3.10\times10^{2}(9.12\times10^{1})\approx$	$3.08\times10^{2}(9.51\times10^{1})\approx$	$3.47\times10^{2}(7.28\times10^{1})\approx$
	T2	$1.06\times10^{-3}(2.38\times10^{-3})$	$1.73\times10^{-2}(4.04\times10^{-3})$-	$7.92\times10^{-5}(1.95\times10^{-5})\approx$	$8.80\times10^{-4}(2.24\times10^{-4})\approx$	$1.30\times10^{-2}(2.56\times10^{-3})$-
F2:PI+MS	T1	$4.09\times10^{-3}(1.25\times10^{-2})$	$6.90\times10^{-2}(8.17\times10^{-3})$-	$2.71\times10^{-3}(3.31\times10^{-4})$+	$8.86\times10^{-3}(1.11\times10^{-3})$-	$3.84\times10^{-2}(7.85\times10^{-3})$-
	T2	$8.73\times10^{1}(1.11\times10^{0})$	$6.10\times10^{1}(1.80\times10^{1})$+	$8.67\times10^{1}(4.95\times10^{-1})\approx$	$8.14\times10^{1}(1.25\times10^{1})$+	$7.12\times10^{1}(1.99\times10^{1})\approx$
F2:PI+LS	T1	$2.72\times10^{-1}(5.12\times10^{-1})$	$4.63\times10^{-1}(1.89\times10^{-1})\approx$	$9.97\times10^{-3}(1.93\times10^{-3})$+	$4.09\times10^{-2}(1.15\times10^{-2})$+	$2.37\times10^{-1}(9.76\times10^{-2})\approx$
	T2	$7.99\times10^{-2}(2.38\times10^{-1})$	$1.52\times10^{-1}(1.96\times10^{-1})$-	$1.53\times10^{-2}(3.26\times10^{-3})\approx$	$3.79\times10^{-2}(2.54\times10^{-2})\approx$	$1.63\times10^{-1}(2.48\times10^{-1})$-
F3:NI+HS	T1	$7.83\times10^{1}(3.47\times10^{1})$	$5.37\times10^{1}(2.31\times10^{0})$+	$5.12\times10^{1}(1.52\times10^{1})$+	$5.45\times10^{1}(1.82\times10^{1})$+	$5.50\times10^{1}(1.58\times10^{1})$+
	T2	$1.08\times10^{1}(1.08\times10^{1})$	$1.21\times10^{1}(5.07\times10^{0})\approx$	$3.81\times10^{0}(8.56\times10^{0})$+	$4.50\times10^{0}(9.63\times10^{0})$+	$5.63\times10^{0}(7.00\times10^{0})$+
F3:NI+MS	T1	$1.66\times10^{-3}(1.58\times10^{-3})$	$2.36\times10^{-2}(7.92\times10^{-3})$-	$4.90\times10^{-4}(2.08\times10^{-4})$+	$2.60\times10^{-3}(5.20\times10^{-4})$-	$1.63\times10^{-2}(5.43\times10^{-3})$-
	T2	$9.98\times10^{-1}(6.68\times10^{-1})$	$1.83\times10^{0}(2.22\times10^{-1})$-	$4.95\times10^{0}(1.94\times10^{-1})$-	$9.35\times10^{-1}(2.28\times10^{-1})\approx$	$1.67\times10^{0}(2.65\times10^{-1})$-
F3:NI+LS	T1	$3.34\times10^{2}(6.91\times10^{1})$	$4.02\times10^{2}(1.51\times10^{1})$-	$3.75\times10^{2}(1.41\times10^{1})$-	$3.79\times10^{2}(2.42\times10^{1})$-	$3.96\times10^{2}(1.45\times10^{1})$-
	T2	$7.81\times10^{2}(2.82\times10^{2})$	$1.10\times10^{3}(3.93\times10^{2})$-	$7.26\times10^{2}(2.56\times10^{2})\approx$	$8.67\times10^{2}(3.65\times10^{2})\approx$	$1.03\times10^{3}(2.77\times10^{2})$-
+/−/≈		Base	2/ 14/ 2	5 / 6 /7	4 / 7/ 7	2 / 13 / 3

optimization performance improves noticeably as the number of *epochs* increases from 10 to 50, indicating that a sufficient number of training iterations is necessary for the FNN to learn meaningful knowledge representations from the collected evolutionary data. When the quantity of *epochs* is relatively small (*epochs* = 10), the model tends to be undertrained, resulting in less effective guidance during the search process.

However, further increasing *epochs* beyond 50 leads to only marginal improvements, and even slight performance degradation on several tasks (F1:CI+MS and F2:PI+HS). This behavior suggests that excessive training may cause the model to overfit the limited and potentially noisy samples generated during evolution, thereby reducing its generalization capability for knowledge transfer. Furthermore, employing an excessively large value for *epochs* is computationally inefficient. Because the FNN tends to reach stable performance at *epochs* = 50, further increasing the training intensity provides diminishing returns in terms of optimization performance while increasing the overall computational cost of the evolutionary process

Table 7. Sensitivity analysis of training epochs parameters in the MTDE-ELM.

Problem	Task	MTDE-ELM(epochs = 50)	epochs = 10	epochs = 20	epochs = 100
F1:CI+HS	T1	$2.60\times10^{-4}(8.14\times10^{-4})$	$1.41\times10^{-3}(1.67\times10^{-3})$-	$9.80\times10^{-5}(9.13\times10^{-5})\approx$	$8.03\times10^{-6}(3.58\times10^{-6})\approx$
	T2	$2.62\times10^{-1}(5.85\times10^{-1})$	$1.62\times10^{0}(1.64\times10^{0})$-	$1.39\times10^{-1}(1.13\times10^{-1})\approx$	$1.16\times10^{-2}(5.49\times10^{-3})\approx$
F1:CI+MS	T1	$4.12\times10^{-5}(6.43\times10^{-6})$	$1.08\times10^{-2}(2.73\times10^{-3})$-	$2.81\times10^{-3}(5.61\times10^{-4})$-	$8.14\times10^{-4}(1.23\times10^{-4})$-
	T2	$1.10\times10^{-6}(3.88\times10^{-7})$	$6.91\times10^{-2}(2.83\times10^{-2})$-	$4.88\times10^{-3}(1.85\times10^{-3})$-	$4.16\times10^{-4}(1.38\times10^{-4})$-
F1:CI+LS	T1	$1.98\times10^{1}(2.39\times10^{0})$	$2.11\times10^{1}(4.38\times10^{-2})$-	$2.11\times10^{1}(4.11\times10^{-2})$-	$2.12\times10^{1}(3.44\times10^{-2})$-
	T2	$1.31\times10^{3}(8.33\times10^{2})$	$4.69\times10^{3}(1.01\times10^{3})$-	$5.90\times10^{3}(1.04\times10^{3})$-	$9.48\times10^{3}(7.25\times10^{2})$-
F2:PI+HS	T1	$3.22\times10^{2}(8.10\times10^{1})$	$3.14\times10^{2}(9.22\times10^{1})\approx$	$3.47\times10^{2}(7.92\times10^{1})\approx$	$3.72\times10^{2}(1.78\times10^{1})$-
	T2	$1.06\times10^{-3}(2.38\times10^{-3})$	$1.34\times10^{-2}(3.31\times10^{-3})$-	$1.09\times10^{-3}(3.12\times10^{-4})$-	$8.97\times10^{-5}(2.10\times10^{-5})\approx$
F2:PI+MS	T1	$4.09\times10^{-3}(1.25\times10^{-2})$	$3.14\times10^{-2}(6.21\times10^{-3})$-	$1.00\times10^{-2}(1.55\times10^{-3})$-	$3.11\times10^{-3}(4.37\times10^{-4})$-
	T2	$8.73\times10^{1}(1.11\times10^{0})$	$7.78\times10^{1}(1.80\times10^{1})\approx$	$7.88\times10^{1}(1.63\times10^{1})$+	$8.67\times10^{1}(4.13\times10^{-1})\approx$
F2:PI+LS	T1	$2.72\times10^{-1}(5.12\times10^{-1})$	$1.60\times10^{-1}(3.76\times10^{-2})$-	$4.30\times10^{-2}(1.47\times10^{-2})\approx$	$1.14\times10^{-2}(3.05\times10^{-3})$+
	T2	$7.99\times10^{-2}(2.38\times10^{-1})$	$1.25\times10^{-1}(1.06\times10^{-1})$-	$4.02\times10^{-2}(2.54\times10^{-2})$-	$1.41\times10^{-2}(3.51\times10^{-3})\approx$
F3:NI+HS	T1	$7.83\times10^{1}(3.47\times10^{1})$	$5.07\times10^{1}(9.52\times10^{0})$+	$4.75\times10^{1}(7.19\times10^{-1})$+	$5.56\times10^{1}(1.77\times10^{1})$+
	T2	$1.08\times10^{1}(1.08\times10^{1})$	$5.57\times10^{0}(8.95\times10^{0})$+	$1.51\times10^{0}(5.30\times10^{0})$+	$1.69\times10^{0}(2.38\times10^{0})$+
F3:NI+MS	T1	$1.66\times10^{-3}(1.58\times10^{-3})$	$1.59\times10^{-2}(3.86\times10^{-3})$-	$2.73\times10^{-3}(7.53\times10^{-4})$-	$5.71\times10^{-4}(1.88\times10^{-4})$+
	T2	$9.98\times10^{-1}(6.68\times10^{-1})$	$1.71\times10^{0}(2.39\times10^{-1})\approx$	$9.08\times10^{-1}(1.78\times10^{-1})$+	$4.43\times10^{-1}(4.78\times10^{-2})$+
F3:NI+LS	T1	$3.34\times10^{2}(6.91\times10^{1})$	$3.93\times10^{2}(1.88\times10^{1})$-	$3.81\times10^{2}(1.26\times10^{1})$-	$3.80\times10^{2}(1.26\times10^{1})$-
	T2	$7.81\times10^{2}(2.82\times10^{2})$	$9.06\times10^{2}(3.70\times10^{2})\approx$	$8.08\times10^{2}(3.60\times10^{2})\approx$	$5.98\times10^{2}(2.40\times10^{2})\approx$
+/−/≈		Base	2 / 12 /4	4/ 9/ 5	5/ 7/6

4.5.5. Performance of p_m and p_u

In the MTDE-ELM, the three distinct knowledge transfers (the intratask, the intertask, and the potential knowledge transfers) determine the three different offspring generation methods. Specifically, p_m and p_u are two key hyperparameters that balance the algorithm's exploitation and exploration capabilities, and their different value combinations will directly affect the performance of the MTDE-ELM in MTO. Thus, we design a set of control experiments to explore the optimal parameter configuration. We investigate the impact of these parameters and identify their optimal combination by comparing the algorithm's performance across p_m values of 0.1, 0.3, 0.5, 0.7, and 0.9 as well as p_u values of 0.1, 0.3, 0.5, 0.7, and 0.9. The comparative results for various parameter settings, measured by the Friedman test of 30 independent runs on each benchmark function, are visualized through a heatmap illustrated in Figure 8.

The experimental results in Figure 8 show a relatively lower p_u, which denotes that offspring generation is predominantly guided by the DE algorithm and can yield more favorable performance on different p_m values in terms of the Friedman test on all the benchmark functions. The results verify that the potential knowledge transfer generated by the ELM can exhibit a positive effect, though overreliance on it cannot guarantee favorable performance of the MTDE-ELM. For the parameter p_m, neither a larger value nor a smaller value exhibits promising performance. On the contrary, the MTDE-ELM displays the most favorable performance when p_m is set as 0.5. In other words, both the intratask and the intertask knowledge transfers, which focus on exploitation and exploration, respectively, play important roles in MTO community.

Based on the results, the recommended parameter settings for the MTDE-ELM are p_m= 0.5 and p_u = 0.1.

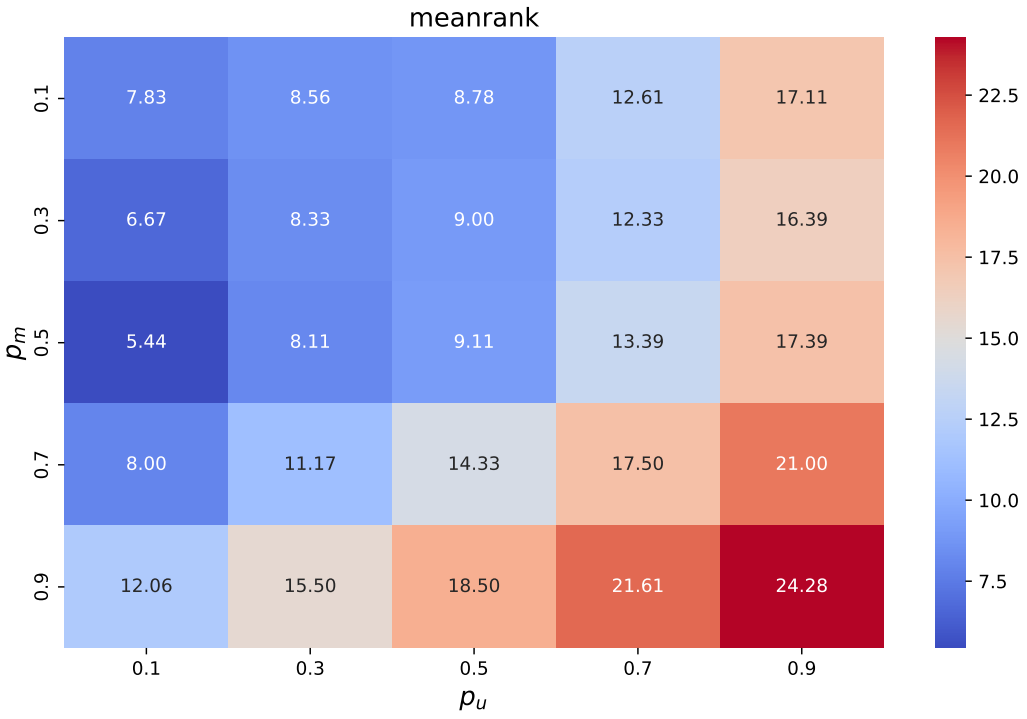

Figure 8. Heat map distribution for different parameter settings of p_m and p_u.

5. Conclusions

This study aims to address the limitations of knowledge transfer strategies in most MTO algorithms by proposing a multitask DE algorithm based on the ELM termed the MTDE-ELM. Unlike traditional MTO algorithms, the MTDE-ELM employs two distinct approaches for offspring generation. The one approach relies on a variant DE algorithm to guide offspring creation. The other approach utilizes the ELM, by which some helpful knowledge of the previous search process can be extracted to perform the offspring generation. Through these approaches, three distinct knowledge transfers can be realized.

A target individual can select other individuals with the same skill factor as himself to perform the mutation operator when adopting the DE-based offspring generation. Based on this method, the intratask knowledge transfer can be realized, thereby improving the exploitation ability of the MTDE-ELM. Conversely, the intertask knowledge transfer can be achieved when target individuals choose individuals with different skill factors to conduct the mutation operator. Thus, the exploration capability of the MTDE-ELM can be enhanced. Moreover, the potential task knowledge transfer is realized by the FNN before utilizing the ELM to generate offspring. In this process, the evolutionary data of multiple tasks are adopted as the input and output data of the FNN. After the training process, an individual can use the trained FNN to generate offspring. To evaluate the overall performance of the MTDE-ELM, we conduct comprehensive experiments on the widely used benchmark suite. Compared to several state-of-the-art MTO algorithms, it exhibits competitive performance in terms of solution accuracy and convergence. Furthermore, the characteristics of new proposed strategies and parameters are also examined by extensive experiments.

Although the proposed the MTDE-ELM demonstrates promising performance, several research directions are worth exploring. First, an adaptive mechanism could be designed to dynamically adjust the number of layers, neurons, and weight parameters based on the complexity of the optimization task and the characteristics of the population. This would enable the NN to better align with specific problem requirements. Second, adaptive strategies could be developed to dynamically determine when and how to perform various knowledge transfers. Finally, we envision extending the MTDE-ELM to broader optimization domains, such as large-scale optimization, multiobjective optimization, and multimodal optimization problems.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgment

This study was funded by the National Natural Science Foundation of Fujian Province (Grant nos.: 2025J01980, 2024J01822, 2023J01922).

Conflict of interest

The authors declare there is no conflicts of interest.

References

1. S. Han, W. Pedrycz, C. Han, Nonlinear channel blind equalization using hybrid genetic algorithm with simulated annealing, *Math. Comput. Modell.*, **41** (2005), 697–709. https://doi.org/10.1016/j.mcm.2004.05.006
2. T. Back, U. Hammel, H. P. Schwefel, Evolutionary computation: Comments on the history and current state, *IEEE Trans. Evol. Comput.*, **1** (1997), 3–17. https://doi.org/10.1109/4235.585888
3. S. Sen, S. Agarwal, P. Chakraborty, K. P. Singh, Astronomical big data processing using machine learning: A comprehensive review, *Exp. Astron.*, **53** (2022), 1–43. https://doi.org/10.1007/s10686-021-09827-4
4. P. Charbonneau, Genetic algorithms in astronomy and astrophysics, *Astrophys. J. Suppl. Ser.*, **101** (1995), 309–334. https://doi.org/10.1086/192242
5. J. Caspermeyer, Researchers show nature conserves its most vital dna by multitasking, *Mol. Biol. Evol.*, **33** (2016), 2477–2478. https://doi.org/10.1093/molbev/msw113
6. X. Zhou, A. K. Qin, M. Gong, K. C. Tan, A survey on evolutionary construction of deep neural networks, *IEEE Trans. Evol. Comput.*, **25** (2021), 894–912. https://doi.org/10.1109/TEVC.2021.3079985
7. L. Wang, J. Li, X. Yan, A variable population size opposition-based learning for differential evolution algorithm and its applications on feature selection, *Appl. Intell.*, **54** (2024), 959–984. https://doi.org/10.1007/s10489-023-05179-y
8. T. Wei, S. Wang, J. Zhong, D. Liu, J. Zhang, A review on evolutionary multitask optimization: Trends and challenges, *IEEE Trans. Evol. Comput.*, **26** (2021), 941–960. https://doi.org/10.1109/TEVC.2021.3139437
9. A. Gupta, Y. S. Ong, L. Feng, K. C. Tan, Multiobjective multifactorial optimization in evolutionary multitasking, *IEEE Trans. Cybern.*, **47** (2016), 1652–1665. https://doi.org/10.1109/TCYB.2016.2554622
10. J. Ding, C. Yang, Y. Jin, T. Chai, Generalized multitasking for evolutionary optimization of expensive problems, *IEEE Trans. Evol. Comput.*, **23** (2019), 44–58. https://doi.org/10.1109/TEVC.2017.2785351
11. Z. G. Chen, Z. H. Zhan, S. Kwong, J. Zhang, Evolutionary computation for intelligent transportation in smart cities: A survey, *IEEE Comput. Intell. Mag.*, **17** (2022), 83–102. https://doi.org/10.1109/MCI.2022.3155330
12. K. C. Tan, L. Feng, M. Jiang, Evolutionary transfer optimization-a new frontier in evolutionary computation research, *IEEE Comput. Intell. Mag.*, **16** (2021), 22–33. https://doi.org/10.1109/MCI.2020.3039066
13. Y. Wang, Q. Zhang, G. G. Wang, Improving evolutionary algorithms with information feedback model for large-scale many-objective optimization, *Appl. Intell.*, **53** (2023), 11439–11473. https://doi.org/10.1007/s10489-022-03964-9
14. H. Zhao, X. Ning, X. Liu, C. Wang, J. Liu, What makes evolutionary multi-task optimization better: A comprehensive survey, *Appl. Soft Comput.*, **145** (2023), 110545. https://doi.org/10.1016/j.asoc.2023.110545

15. W. Yang, J. Liu, S. Tan, W. Zhang, Y. Liu, Evolutionary dynamic grouping based cooperative co-evolution algorithm for large-scale optimization, *Appl. Intell.*, **54** (2024), 4585–4601. https://doi.org/10.1007/s10489-024-05390-5

16. X. Zheng, A. K. Qin, M. Gong, D. Zhou, Self-regulated evolutionary multitask optimization, *IEEE Trans. Evol. Comput.*, **24** (2019), 16–28. https://doi.org/10.1109/TEVC.2019.2904696

17. X. Xia, L. Gui, F. Yu, H. Wu, B. Wei, Y. L. Zhang, et al., Triple archives particle swarm optimization, *IEEE Trans. Cybern.*, **50** (2019), 4862–4875. https://doi.org/10.1109/TCYB.2019.2943928

18. F. Gao, L. Huang, W. Gao, L. Li, S. Wang, M. Gong, et al., Transferring knowledge by budget online learning for multiobjective multitasking optimization, *Swarm Evol. Comput.*, **91** (2024), 101765. https://doi.org/10.1016/j.swevo.2024.101765

19. Y. Guo, G. Chen, M. Jiang, D. Gong, J. Liang, A knowledge guided transfer strategy for evolutionary dynamic multiobjective optimization, *IEEE Trans. Evol. Comput.*, **27** (2022), 1750–1764. https://doi.org/10.1109/TEVC.2022.3222844

20. Z. Cui, B. Zhao, T. Zhao, X. Cai, J. Chen, Adaptive multi-task evolutionary algorithm based on knowledge reuse, *Inf. Sci.*, **648** (2023), 119568. https://doi.org/10.1016/j.ins.2023.119568

21. A. Gupta, Y. Ong, L. Feng, Multifactorial evolution: Toward evolutionary multitasking, *IEEE Trans. Evol. Comput.*, **20** (2016), 343–357. https://doi.org/10.1109/TEVC.2015.2458037

22. L. Zhou, L. Feng, K. C. Tan, J. Zhong, Z. Zhu, K. Liu, et al., Toward adaptive knowledge transfer in multifactorial evolutionary computation, *IEEE Trans. Cybern.*, **51** (2021), 2563–2576. https://doi.org/10.1109/TCYB.2020.2974100

23. Z. H. Zhan, J. Y. Li, S. Kwong, J. Zhang, Learning-aided evolution for optimization, *IEEE Trans. Evol. Comput.*, **27** (2022), 1794–1808. https://doi.org/10.1109/TEVC.2022.3232776

24. J. Y. Li, Z. H. Zhan, K. C. Tan, J. Zhang, A meta-knowledge transfer-based differential evolution for multitask optimization, *IEEE Trans. Evol. Comput.*, **26** (2021), 719–734. https://doi.org/10.1109/TEVC.2021.3131236

25. R. Chandra, A. Gupta, Y. S. Ong, C. K. Goh, Evolutionary multi-task learning for modular knowledge representation in neural networks, *Neural Process. Lett.*, **47** (2018), 993–1009. https://doi.org/10.1007/s11063-017-9718-z

26. C. Qin, T. Zhu, K. Jiang, Y. Wu, J. Zhang, Neural-network-based safe learning control for non-zero-sum differential games of nonlinear systems with asymmetric input constraints, *Appl. Intell.*, **54** (2024), 7810–7828. https://doi.org/10.1007/s10489-024-05593-w

27. Z. F. Xue, Z. J. Wang, Z. H. Zhan, S. Kwong, J. Zhang, Neural network-based knowledge transfer for multitask optimization, *IEEE Trans. Cybern.*, **54** (2024), 7541–7554. https://doi.org/10.1109/TCYB.2024.3469371

28. Y. Jiang, Z. H. Zhan, K. C. Tan, J. Zhang, Knowledge learning for evolutionary computation, *IEEE Trans. Evol. Comput.*, **29** (2023), 16–30. https://doi.org/10.1109/TEVC.2023.3278132

29. Q. Rui, W. L. Liu, Y. Wu, J. Zhong, Nonlinear mapping meets multi-task bayesian optimization: A knowledge transfer perspective, in *2025 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, (2025), 1414–1419. https://doi.org/10.1109/SMC58881.2025.11343035

30. J. Guo, L. Li, H. Sun, M. Qin, H. Yu, T. Zhang, A min-max optimization framework for sparse multi-task deep neural network, *Neurocomputing*, **650** (2025), 130865. https://doi.org/10.1016/j.neucom.2025.130865

31. H. Han, B. Zhao, X. Wu, X. Li, Survey on multi-task optimization: Towards cross-domain and asynchronous multi-task, *Swarm Evol. Comput.*, **99** (2025), 102175. https://doi.org/10.1016/j.swevo.2025.102175

32. C. Wang, L. Li, L. Jiao, J. Zhao, F. Liu, S. Yang, Learning evolution via optimization knowledge adaptation, *IEEE Trans. Pattern Anal. Mach. Intell.*, **2026** (2026), 1–18. https://doi.org/10.1109/TPAMI.2026.3686919

33. R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, *J. Glob. Optim.*, **11** (1997), 341–359. https://doi.org/10.1023/A:1008202821328

34. L. Feng, Y. S. Ong, S. Jiang, A. Gupta, Autoencoding evolutionary search with learning across heterogeneous problems, *IEEE Trans. Evol. Comput.*, **21** (2017), 760–772. https://doi.org/10.1109/TEVC.2017.2682274

35. H. Li, Y. S. Ong, M. Gong, Z. Wang, Evolutionary multitasking sparse reconstruction: Framework and case study, *IEEE Trans. Evol. Comput.*, **23** (2018), 733–747. https://doi.org/10.1109/TEVC.2018.2881955

36. K. Bali, Y. Ong, A. Gupta, P. S. Tan, Multifactorial evolutionary algorithm with online transfer parameter estimation: Mfea-ii, *IEEE Trans. Evol. Comput.*, **24** (2020), 69–83. https://doi.org/10.1109/TEVC.2019.2906927

37. Z. Liang, J. Zhang, L. Feng, Z. Zhu, A hybrid of genetic transform and hyper-rectangle search strategies for evolutionary multi-tasking, *Exp. Syst. Appl.*, **138** (2019), 112798. https://doi.org/10.1016/j.eswa.2019.07.015

38. M. Gong, Z. Tang, H. Li, J. Zhang, Evolutionary multitasking with dynamic resource allocating strategy, *IEEE Trans. Evol. Comput.*, **23** (2019), 858–869. https://doi.org/10.1109/TEVC.2019.2893614

39. C. Wang, J. Liu, K. Wu, Z. Wu, Solving multitask optimization problems with adaptive knowledge transfer via anomaly detection, *IEEE Trans. Evol. Comput.*, **26** (2021), 304–318. https://doi.org/10.1109/TEVC.2021.3068157

40. K. K. Bali, A. Gupta, L. Feng, Y. S. Ong, Linearized domain adaptation in evolutionary multitasking, in *2017 IEEE Congress on Evolutionary Computation (CEC)*, (2017), 1295–1302. https://doi.org/10.1109/CEC.2017.7969454

41. X. Wang, Q. Kang, M. Zhou, S. Yao, A. Abusorrah, Domain adaptation multitask optimization, *IEEE Trans. Cybern.*, **53** (2023), 4567–4578. https://doi.org/10.1109/TCYB.2022.3222101

42. H. Han, X. Bai, Y. Hou, J. Qiao, Multitask particle swarm optimization with heterogeneous domain adaptation, *IEEE Trans. Evol. Comput.*, **28** (2024), 178–192. https://doi.org/10.1109/TEVC.2023.3258491

43. X. Xia, L. Gui, Y. Zhang, X. Xu, F. Yu, A fitness-based adaptive differential evolution algorithm, *Inf. Sci.*, **549** (2021), 116–141. https://doi.org/10.1016/j.ins.2020.11.015

44. Z. Liang, X. Xu, L. Liu, Y. Tu, Z. Zhu, Evolutionary many-task optimization based on multisource knowledge transfer, *IEEE Trans. Evol. Comput.*, **26** (2022), 319–333. https://doi.org/10.1109/TEVC.2021.3101697

45. S. H. Wu, Z. H. Zhan, K. C. Tan, J. Zhang, Transferable adaptive differential evolution for many-task optimization, *IEEE Trans. Cybern.*, **53** (2023), 7295–7308. https://doi.org/10.1109/TCYB.2023.3234969

46. T. Zhang, W. Gong, Y. Li, Multitask differential evolution with adaptive dual knowledge transfer, *Appl. Soft Comput.*, **165** (2024), 112040. https://doi.org/10.1016/j.asoc.2024.112040

47. X. Ma, M. Xu, Y. Yu, H. Liu, Y. Wang, L. Wang, et al., Enhancing evolutionary multitasking optimization by leveraging intertask knowledge transfers and improved evolutionary operators, *Knowl. Based Syst.*, **259** (2023), 110027. https://doi.org/10.1016/j.knosys.2022.110027

48. L. Feng, W. Zhou, L. Zhou, S. W. Jiang, J. H. Zhong, B. S. Da, et al., An empirical study of multifactorial PSO and multifactorial DE, in *IEEE Congress on Evolutionary Computation (CEC)*, (2017), 921–928. https://doi.org/10.1109/CEC.2017.7969407

49. Z. Tang, M. Gong, Y. Wu, W. Liu, Y. Xie, Regularized evolutionary multitask optimization: Learning to intertask transfer in aligned subspace, *IEEE Trans. Evol. Comput.*, **25** (2021), 262–276. https://doi.org/10.1109/TEVC.2020.3023480

50. G. Yuan, G. Sun, L. Deng, C. Li, G. Yang, A novel differential evolution algorithm based on periodic intervention and systematic regulation mechanisms, *Appl. Intell.*, **54** (2024), 11779–11803. https://doi.org/10.1007/s10489-024-05781-8

51. J. H. Holland, Genetic algorithms, *Sci. Am.*, **267** (1992), 66–73. https://doi.org/10.1038/scientificamerican0792-66

52. J. Liu, L. Tong, X. Xia, A genetic algorithm for vehicle routing problems with time windows based on cluster of geographic positions and time windows, *Appl. Soft Comput.*, **169** (2025), 112593. https://doi.org/10.1016/j.asoc.2024.112593

53. A. Sureyya Rifaioglu, T. Doğan, M. Jesus Martin, R. Cetin-Atalay, V. Atalay, Deepred: Automated protein function prediction with multi-task feed-forward deep neural networks, *Sci. Rep.*, **9** (2019), 7344. https://doi.org/10.1038/s41598-019-43708-3

54. M. Moller, Efficient training of feed-forward neural networks, in *Neural Network Analysis, Architectures and Applications*, CRC Press, (2024), 136–173. https://doi.org/10.1201/9781003572886-8

55. D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors, *Nature*, **323** (1986), 533–536. https://doi.org/10.1038/323533a0

56. S. Schmidgall, R. Ziaei, J. Achterberg, L. Kirsch, S. Hajiseyedrazi, J. Eshraghian, Brain-inspired learning in artificial neural networks: A review, *APL Mach. Learn.*, **2** (2024), 021501. https://doi.org/10.1063/5.0186054

57. Y. LeCun, Y. Bengio, G. Hinton, Deep learning, *Nature*, **521** (2015), 436–444. https://doi.org/10.1038/nature14539

58. B. Da, Y. S. Ong, K. C. Tan, A. K. Qin, A. Gupta, Z. Zhu, et al., Evolutionary multitasking for single-objective continuous optimization: Benchmark problems, performance metric, and baseline results, preprint, arXiv:1706.03470. https://doi.org/10.48550/arXiv.1706.03470

59. Y. Yuan, Y. S. Ong, L. Feng, A. K. Qin, A. Gupta, B. Da, et al., Evolutionary multitasking for multiobjective continuous optimization: Benchmark problems, performance metrics and baseline results, preprint, arXiv:1706.02766. https://doi.org/10.48550/arXiv.1706.02766

60. L. Feng, L. Zhou, J. Zhong, A. Gupta, Y. S. Ong, K. C. Tan, et al., Evolutionary multitasking via explicit autoencoding, *IEEE Trans. Cybern.*, **49** (2019), 3457–3470. https://doi.org/10.1109/TCYB.2018.2845361

61. Y. Li, W. Gong, S. Li, Multitasking optimization via an adaptive solver multitasking evolutionary framework, *Inf. Sci.*, **630** (2023), 688–712. https://doi.org/10.1016/j.ins.2022.10.099

62. Y. Jiang, Z. H. Zhan, K. C. Tan, J. Zhang, Block-level knowledge transfer for evolutionary multitask optimization, *IEEE Trans. Cybern.*, **54** (2024), 558–571. https://doi.org/10.1109/TCYB.2023.3273625

63. Y. Li, W. Gong, Multiobjective multitask optimization with multiple knowledge types and transfer adaptation, *IEEE Trans. Evol. Comput.*, **29** (2025), 205–216. https://doi.org/10.1109/TEVC.2024.3353319

64. K. Deb, R. B. Agrawal, Simulated binary crossover for continuous search space, *Complex Syst.*, **9** (1995), 115–148.

65. K. Deb, M. Goyal, A combined genetic adaptive search (geneas) for engineering design, *Comput. Sci. Inf. Syst*, **26** (1996), 30–45.

66. J. Carrasco, S. García, M. Rueda, S. Das, F. Herrera, Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review, *Swarm Evol. Comput.*, **54** (2020), 100665. https://doi.org/10.1016/j.swevo.2020.100665

AIMS Press