Efficient Knowledge Transfer with Similar Mating Probability and Dimension-aware Selection Strategy for Many-Task Optimization

Huynh Thi Thanh Binh *Member, IEEE*, Do Tuan Anh, Le Trung Kien, Dinh Tan Minh, Ban Ha Bang, Dao Van Tung, Su Nguyen

***Abstract*—Knowledge transfer is a crucial element of multitasking optimization algorithms and has received significant attention in the literature. Although many algorithms have been proposed to improve the effectiveness of knowledge transfer in multitasking optimization, negative transfer remains a key challenge. In addition, conventional random mating probability and knowledge transfer techniques cannot efficiently handle many tasks with high dimensions. To address these limitations, this paper proposes a new algorithm called Similar Mating Multifactorial Evolutionary Algorithm for multitasking optimization. This algorithm employs a novel approach for determining the knowledge transfer probability of each task pair by leveraging previous successful transfers to avoid restrictive positive knowledge transfer in traditional methods. A Dimension-aware Selection strategy is also proposed to improve knowledge transfer efficiency by adaptively selecting dimensions with high similarity between each pair of tasks. The experiment results demonstrate the superior performance of the proposed algorithm in terms of solution quality, convergence speed, and computational time. For the 10-tasks benchmark dataset, the proposed algorithm only needs less than half of the number of evaluations to achieve the best solutions obtained by other state-of-the-art algorithms. Our further analyses also confirm that the proposed Dimension-aware Selection strategy performs exceptionally well in the benchmark datasets with a large number of tasks.**

***Index Terms*—Knowledge Transfer, Many-task Optimization, Multifactorial Evolutionary Algorithm**

I. Introduction

RECENTLY, cloud computing has played a pivotal role in modern life as technology advances. In real-world situations, client requests (or tasks) are often interrelated and processing them independently would be time-consuming and resource-intensive. Therefore, handling a large number of client requests concurrently in cloud computing demands efficient optimization algorithms. For the past few years, Multitasking Optimization (MTO), which simultaneously solves many tasks, has gained attention among scholars. Inspired by the classical Evolutionary Algorithm (EA), Evolutionary Multitasking Optimization (EMTO) employs evolutionary-based search to solve multiple problems. This process allows knowledge transfer across overlapping tasks, which significantly improves solution quality and reduces convergence time. In their seminal work, Gupta et al. [1] proposed Multifactorial Evolutionary Algorithm (MFEA) based on multifactorial biological inheritance to solve multiple optimization problems simultaneously using a single population of individuals in which the interaction of genetic and cultural factors is responsible for the transmission of complex developmental traits to offspring. MFEA suggests a population representation called Unified Search Space (USS), which enables the transfer of traits between individuals of different cultural backgrounds. The USS facilitates the exchange of genetically valuable materials within various optimization problems.

Due to its prominent attributes, MFEA excels in a wide range of real-world applications [2]–[6], including complex combinatorial optimization problems [6], [7]. Subsequently, Gupta et al. [8] have employed MFEA in Multiobjective Optimization (MOO) in which implicit parallelism offered by a population enables simultaneous convergence toward the Pareto front. Based on the original idea in [1], a variety of models were produced and yielded superior results. For expensive optimization problems, Ding et al. proposed MCEEA [9], a multifactorial optimization method that improves convergence speed by transferring knowledge from cheap optimization tasks to expensive ones.

With its wide application and development, MFEA is being enhanced to address various challenges. One of these challenges is the many-tasks problem, which has a large number of tasks to be completed. As the number of tasks increases, producing high-quality solutions with small computational effort. When dealing with many tasks, MFEA has two main limitations: (1) the random mating probability of every task pair is bounded and cannot fully take advantage of knowledge transfer between tasks, and (2) there has been no technique for managing knowledge transfer on each dimension.

Firstly, most current algorithms rely on Random Mating Probability (*rmp*) to control the degree of knowledge transfer by adjusting *rmp* and can prevent negative transfer between low-similarity tasks. However, the probability of random mating between two tasks in these algorithms can become very small as a large number of tasks are considered. This issue may prevent useful knowledge transfer and the effectiveness of EMTO algorithms. These limitations prevent MFEA from finding quality solutions

H. T. T. Binh, D. T. Anh, L. T. Kien, D. T. Minh, B. H. Bang, D. V. Tung are with the School of Information and Communication Technology, Hanoi University of Science and Technology, Vietnam (e-mail: binhht@soict.hust.edu.vn, anhdt@soict.hust.edu.vn, kien.lt194084@sis.hust.edu.vn, minh.dt194116@sis.hust.edu.vn, bangbh@soict.hust.edu.vn, tung.dv204932@sis.hust.edu.vn).

Su Nguyen is with RMIT University, Australia (email: su.nguyen@rmit.edu.au)

This research was funded by Vingroup Innovation Foundation (VINIF) under project code VINIF.2022.DA00183.

efficiently in USS.

Secondly, many multitasking algorithms treat every dimension equally while performing crossover. This approach may be effective if tasks benefit from knowledge transfer across all dimensions. If useful knowledge (positive) transfer only happens in certain dimensions, treating every dimension equally can lead to negative transfer and waste the computational budget. To the best of our knowledge, there is no study that explicitly investigates the impact of dimensions in knowledge transfer in the literature of EMTO.

To address the above limitations of conventional MFEA, this paper proposes Similar Mating Multifactorial Evolutionary Algorithm (SM-MFEA) for many-task optimization problems. This algorithm uses a Similar Mating Probability (*smp*) parameter to determine the effective mating probability between tasks. The use of *smp* enables effective knowledge transfer by selecting crossovers based on all tasks rather than specific pairs used in existing MFEAs.

Apart from SM-MFEA, we propose a new strategy, called Dimension-aware Selection (DaS), which is based on identifying task similarities across independent dimensions to govern knowledge transfer. With its feature, DaS outperforms using a simple model and can be applied to many different algorithms, not limited to the one we proposed.

To evaluate the effectiveness of SM-MFEA and DaS, we compare its results against other state-of-the-art algorithms on benchmark datasets. Experimental results show that SM-MFEA outperforms others in terms of solution quality, convergence trend, and computation time. Furthermore, when combined with other algorithms, DaS produces better outcomes on the benchmark datasets compared to using those algorithms independently.

Our contributions are summarized as follows:

- Propose SM-MFEA algorithm to effectively utilize knowledge transfer between each task pair.
- Develop a Dimension-aware Selection (DaS) strategy that improves the efficiency of knowledge transfer by selecting dimensions for transferring based on KL-Divergence.
- Conduct experiments to evaluate the effectiveness of SM-MFEA and DaS strategy on many-task benchmark datasets and compare with state-of-the-art algorithms.

The rest of the paper is organized as follows. Section II covers related works. In section III, SM-MFEA and DaS strategy is described in detail. Section IV contains the experimental results and comparison with other algorithms. Finally, the conclusion and future works are presented in Section V.

II. Literature reviews

A. Related works

Because of its capabilities in solving many-task simultaneously, MFEA has garnered significant attention among researchers. The knowledge transfer mechanism of MFEA is implicit transfer through assortative mating and vertical cultural transmission. However, the performance of this algorithm mainly depends on the similarity between tasks. More specifically, knowledge transfer in closely related pairs of tasks is positive transfer [1]. Meanwhile, negative transfer generally occurs in exchanges between low-similarity pairs. Therefore, most of the current approaches in MFEA focus on addressing two main issues: minimizing negative transfer [10]–[14] and maximizing positive transfer [15]–[18].

There are many algorithms proposed for minimizing negative transfer between tasks. Bali et al. [11] proposed MFEA with Online Transfer Parameter Estimation (MFEA-II), an algorithm that relies on a self-adapting mixture matrix of the *rmp*. MFEA-II tries to minimize negative transfer by generating offspring with probability distribution similar to that of their parents. In extreme cases where the intra-task crossover is more favorable, MFEA-II will significantly curtail inter-task crossover and prevent knowledge transfer, leading to slower convergence. Additionally, MFEA-II has immense time complexity for learning *rmp* matrix online by minimizing the gap between two probability distributions averaged across all optimization tasks, especially when solving a large number of tasks. Recently, Wang et al. [12] proposed Multitask Evolutionary Algorithms based on Anomaly Detection (MTEA-AD) to mitigate the deficiency of *rmp*. In this algorithm, each task has a population and the anomaly detection model is used to detect the relationship among individuals in different tasks online. It provides explicit knowledge transfer by moving candidate individuals from one task to another. The candidates are selected using an anomaly detection model to eliminate individuals with negative knowledge. Nevertheless, the parameter of this model is strongly affected by the experience of only one previous generation. The algorithm is destabilized by this approach, as it only considers a single generation rather than the trend across multiple generations. Consequently, while the algorithm is well-motivated, its results are not remarkable. Recently, Thang et al. [13] introduced two algorithms LSA-MFEA and SA-MFEA, which adapt the *rmp* parameter to reduce negative transfers based on the historical memory of successful *rmp*. In [14], the authors proposed a many-task evolutionary algorithm called EME-BI which aims to combat negative transfer by adapting *rmp* to regulate both the frequency and direction of transfer. Besides, the algorithm is equipped with an ensemble of multiple search operators in which the computation resources are allocated dynamically based on their effectiveness. SA-MFEA, LSA-MFEA, and EME-BI have demonstrated superior experimental outcomes when compared to other existing algorithms. However, as rmp-based algorithms, they are still subject to a particular limitation regarding the probability aspect of such algorithms, which will be described in section II-B.

Maximizing positive knowledge transfer is also a prominent approach. Zheng et al. [15] proposed a Self-regulated Evolutionary Multitask Optimization (SREMTO) algorithm to tackle the deficient knowledge transfer scheme of MFEA. The SREMTO algorithm uses ability vectors to fully reflect each individual's capability of solving different tasks and enhance the mechanism of automatically adapting the intensity of knowledge transfer between related ones. During the evolutionary process, the degree of task relatedness will determine the intensity of knowledge transfer. Additionally, Chen et al. [16] suggested a many-task evolutionary algorithm

(MaTEA), in which a task is accommodated by an assisted task selected through the similarity measurement mechanism and the adaptive reward strategy. Despite showing promising experimental results, this algorithm is not suitable for solving many-task problems due to its wide range of hyperparameters. Similarly, Liaw and Ting proposed an algorithm named Evolution of Biocoenosis through Symbiosis (EBS) [17] for multi-task problems and an improved version known as Symbiosis in Biocoenosis Optimization (SBO) [18]. These frameworks estimate the degree of symbiosis in biocoenosis to determine the inter-task knowledge transfer probability. While this strategy can control knowledge transfer between tasks reduce the occurrence of negative transfer, it still faces the issue of "aimless" knowledge transfer, which can result in a waste of computational resources.

Furthermore, another approach maximizes positive transfer by treating each dimension individually. In [19], Sheng-Hao Wu et al. proposed a cross-task mapping (CTM) strategy and an orthogonal transfer (OT) method. The CTM strategy solves the dimension discrepancy by mapping individuals from their dimension to that of other tasks' dimensions while the OT method selects which dimensions can facilitate positive knowledge transfer from other tasks to the current task. Since knowledge transfer is guaranteed between every task pairs, the algorithm cannot suppress negative transfer, leading to enormous convergence time and exaggerated resource consumption. Additionally, due to its large computational time and algorithm complexity, this algorithm is not optimal for many-task optimization problems.

B. Disadvantages of mixture models in the context of rmp

During the evolutionary process, MFEA divides the original population into several different groups, each of which is responsible for solving a distinct task determined by their skill factor. Through "assortative mating and vertical cultural transmission," knowledge learned from one task is implicitly transferred to help other tasks learn better. The degree of transfers depends on *rmp* designed to regulate knowledge transfer between each task pair.

Without loss of generality, for all $k \in \{1, 2, ..., K\}$, the offspring sub-population associated with the k^{th} task is produced by transferring knowledge between the k^{th} sub-population and every other $(K-1)$ sub-populations. We denote the probability distribution of the k^{th} sub-population at generation t is $p^k(x,t)$ while that of their offspring is $q^k(x,t)$. Through knowledge transfer and crossover operators, the distribution $q^k(x,t)$ is considered to be a mixture of all K available distributions [11].

$$q^k(x,t) = \alpha_k^k \cdot p^k(x,t) + \textstyle\sum_{j \neq k} \alpha_j^k \cdot p^j(x,t), \text{with } \sum_{j=1}^{K} \alpha_j^k = 1 \quad (1)$$

We denote T_k as the k^{th} task, then α_j^k $(j \in \{1, 2, ..., K\})$ represents the probability that T_k received knowledge from T_j.

Let rmp_j^k be the random mating probability between T_k and T_j. From a probabilistic point of view, the distribution of offspring k^{th} at generation t is

$$q^k(x,t) = (1 - \sum_{j \neq k} \frac{rmp_j^k}{K}) \cdot p^k(x,t) + \sum_{j \neq k} \frac{rmp_j^k}{K} \cdot p^j(x,t) \quad (2)$$

where: $\alpha_k^k = 1 - \sum_{j \neq k} \frac{rmp_j^k}{K}$; and $\alpha_j^k = \frac{rmp_j^k}{K}$ with: $j \neq k$

From equation (2), it is evident that the algorithm can reduce negative knowledge transfer, correspondingly, $rmp_j^k = 0$ or $\alpha_j^k = 0$. As a result, knowledge transfer from T_j to T_k will not occur.

Because the transfer probability from T_j to T_k (α_j^k) has an upper bound at $\frac{1}{K}$, the computing resource reserved for transferring knowledge is also limited. As a result, it cannot fully utilize the positive knowledge from T_j to T_k. This issue will escalate as the number of tasks increases (for example, the transfer probability drop to just 2% with K = 50).

Because of its simplicity, *rmp* has been adopted in many algorithms in the literature to determine transfer probability. However, past studies have not yet overcome the overarching limitations of *rmp* (as mentioned in the Introduction section). Meanwhile, algorithms without *rmp* [15], [17]–[19] are too complex and not suitable for many-task problems.

This study develops SM-MFEA to address the above limitation of models based on *rmp* while being capable of efficiently solving many-task problems. Additionally, in *D*-dimensional unified search space, since every dimension has a different degree of importance and similarity, we proposed DaS strategy to determine the knowledge transfer probability of each dimension between every pair of tasks.

III. Proposed Method

The pseudo-code of the proposed algorithm SM-MFEA is presented in Algorithm 1. First, we initialize the population with $K \times N$ individuals, with N being the size of each sub-population in line 4. We then assign a skill-factor to each individual in the population in line 5. After that, K vectors smp^i $(1 \leq i \leq K)$, representing the crossover probability of i^{th} sub-population relative to others, are created in line 6. After that, we iteratively execute the following steps until the termination conditions are met. In lines 10 and 11, we initialize an empty offspring population and matrix Δ $(\in \mathbb{R}^{K \times (K+1)})$ recording the total square percentage improvement of the offspring relative to their parent. Furthermore, in line 12, we create a matrix C $(\in \mathbb{R}^{K \times (K+1)})$ that stores the number of offspring for each pair of tasks produced in each generation. Finally, in lines 13 to 17, we generate new individuals by performing recombination until the offspring population is equal to the parent population in size. We also integrate the DaS strategy into the recombination step, in which, Probability Crossover Dimension(PCD) parameter represents the similarity score on each dimension of two sub-populations. Subsequently, we merge two populations in line 18 and select the best individuals for the next generation in line 19. To facilitate exploitation capacity, after the selection process, the population size is linearly decreased , which is an effective strategy adopted in previous studies [13], [20]. Finally, *PCD* and smp^i vector $(1 \leq i \leq K)$ are updated in lines 20 and 21,

respectively. In the rest of this section, we will provide the details of each key component in the proposed algorithm.

Algorithm 1: Framework of SM-MFEA

Input: K optimization tasks $T_1, ..., T_K$ with maximum number of evaluations MAXEVALS
Output: The best solutions of all tasks $X_k^*, 1 \leq k \leq K$
$N_{min} \leftarrow$ the minimum population size of each task
Initialize population $P(0)$ with $K \times N$ individuals
Assign the fittest task for each individual
Initialize probability vectors $smp^1, smp^2, ..., smp^K$
Initialize count evaluation $evals \longleftarrow 0$
Set $t \longleftarrow 0$
while $evals \leq MAXEVALS$ **do**
  Offspring population $O(t) \leftarrow \emptyset$
  Improvement percent matrix $\Delta \leftarrow \big[0\big]_{K\times(K+1)}$
  The number of offspring matrix $C \leftarrow \big[0\big]_{K\times(K+1)}$
  while $|O(t)| < |P(t)|$ **do**
    $o_a, o_b = \text{SM\_recombination}\big(P(t), \Delta, C\big)$. ▷ See algorithm 2
    $O(t) = O(t) \cup [o_a, o_b]$
    $evals \leftarrow evals + 2$
  end
  $P(t+1) = P(t) \cup O(t)$
  $P(t+1) \leftarrow$ Select $\left(N + (N_{min} - N) * \frac{evals}{\text{MAXEVALS}}\right)$ best individuals from each sub-population.
  Update_DaS($P(t+1)$) ▷ See algorithm 5
  Update_SMP(Δ, C) ▷ See algorithm 3
  $t = t + 1$
end

A. Similar Mating Probability

To address the limitation of *rmp* on positive transfer maximization, we proposed a probability model based on probability vector *smp* to control the degree of knowledge exchange across multiple tasks:

$$smp^k = \big[smp_1^k, smp_2^k, ..., smp_K^k, p_{\text{mutation}}^k\big] + \frac{\mu}{K+1} * [1]^{K+1} \quad (3)$$

such that:

$$\sum_{i=1}^{K} smp_i^k + p_{\text{mutation}}^k + \mu = 1$$

With the knowledge transfer probability from T_i to T_k being:

$$\alpha_i^k = smp_i^k + \frac{\mu}{K+1} \quad (\forall i \neq k) \quad (4)$$

and the intra-transfer probability in T_k being:

$$\alpha_k^k = smp_k^k + p_{\text{mutation}}^k + \frac{2\mu}{K+1} \quad (5)$$

From equation (4) and (5), we see that $\frac{\mu}{K+1} \leq \alpha_i^k \leq 1$.

We denoted smp^k as the probability vector for T_k where its element, smp_i^k, decides the probability that T_k receives knowledge from T_i. In addition, p_{mutation}^k represents the mutation probability of T_k. A hyperparameter μ is added to prevent the value of smp_i^k from becoming minuscule, which reduces the chance of knowledge transfer. The ability to transfer knowledge from other tasks to T_k can be controlled by adjusting the smp^k vector. Hence, the probability model based on probability vector *smp* allows, to reach the maximum value of 1 instead of $\frac{1}{K}$. Due to no prior knowledge in the initial generation, we initialize $smp_j^k = p_{\text{mutation}}^k = \frac{1-\mu}{K+1}$.

Algorithm 2: SM_recombination

Input:
- Parent's population P
- The Improvement Percent matrix Δ
- Count offsprings matrix C
Output: Offspring o_a, o_b
$\tau_a \leftarrow$ Randomly select an index of task from list $\{1, ..., K\}$
$\tau_b \leftarrow$ Select a second index of task follow roulette wheel selection with probability smp^{τ_a}
Set skill factor for offspring $\tau_{o_a} = \tau_{o_b} = \tau_a$
if $\tau_b = \tau_a$ **then**
  $p_a, p_b \leftarrow$ Randomly select two individuals from sub-population P_{τ_a}
  $o_a, o_b \leftarrow$ Intra-Crossover($p_a, p_b, \tau_{o_a}, \tau_{o_b}$)
else if $\tau_b = K + 1$ **then**
  $p_a, p_b \leftarrow$ Randomly select individuals from sub-population P_{τ_a}
  $o_a, o_b \leftarrow$ Mutate p_a, p_b
else
  $p_a, p_b \leftarrow$ Randomly select an individual from sub-population P_{τ_a} and P_{τ_b} respectively
  $o_a, o_b \leftarrow$ Inter-Crossover($p_a, p_b, \tau_{o_a}, \tau_{o_b}$)
  $o_a \leftarrow$ DaS_strategy(o_a, p_a, τ_b)
  $o_b \leftarrow$ DaS_strategy(o_b, p_a, τ_b)
end
Evaluate o_a, o_b by their assigned skill factors
Δ_{τ_a,τ_b} += SPI(f_{o_a}, f_{p_a}) + SPI(f_{o_b}, f_{p_a})
C_{τ_a,τ_b} += 2

Algorithm 2 depicts how offspring are produced during the evolutionary process. Initially, task T_{τ_a} is selected randomly while task T_{τ_b} is chosen from the wheel selection with probability vector smp^{τ_a}. Then, each offspring is assigned skill-factor τ_a, indicating that task T_{τ_a} is the recipient of knowledge transfer from task T_{τ_b}. In lines 9 to 20, we outline the method for generating two offspring. When both parents share the same skill factor, Intra-Crossover is employed to produce offspring. If τ_b is equal to $K+1$, a pair of random individuals from P_{τ_a} are selected for mutation. In other cases, p_a and p_b undergo Inter-Crossover to create two offspring, after which the DaS strategy is applied to enhance their traits. They are then evaluated according to their skill-factor. Next, in line 22, we update Δ_{τ_a,τ_b} for computing the *smp* vector in later. Each Δ_{τ_a,τ_b} (where $\tau_a \in \{1, 2, ..., K\}$ and $\tau_b \in \{1, 2, ..., K+1\}$) represent the sum of the improvement level of the offspring compared to its parent, and we utilize Square Percentage Improvement (SPI) to measure the degree of improvement. If τ_b equal $K+1$, $\Delta_{\tau_a,K+1}$ measures the

overall improvement resulting from mutations in task T_{τ_a}. SPI is calculated using the formula below:

$$\text{SPI}(f_o, f_p) = \max(\frac{f_p - f_o}{f_p}, 0)^2 \quad (6)$$

where f_p and f_o represent the factorial cost of the parent and offspring, respectively, the SPI calculates the improvement as a percentage squared if the offspring performs better than the parent. However, if the parent outperforms its offspring, the SPI returns a value of 0. The purpose of squaring in the SPI formula is to highlight the extent of improvement when the offspring is better than the parent and to amplify the contrast between substantial and minor improvements, resulting in a more precise assessment.

Similarly, C_{τ_a,τ_b} is used to count the number of offspring generated when individuals from task T_{τ_a} are crossover with individuals from task T_{τ_b} or undergo mutation. This algorithm involves producing two offspring every time a recombination is executed, which results in an increment of two in the value of C_{τ_a,τ_b} in line 23.

Algorithm 3: Update_SMP

1 **Input:**
2 - The Improvement Percent matrix Δ
3 - Count offsrpings matrix C
4 **Output:** New vectors $smp^1, smp^2, ..., smp^K$ for new generations
5 **for** $i \leftarrow 1$ *to* K **do**
6 **if** $\sum \Delta_i > 0$ **then**
  `/* ⊘ is element-wise division */`
7 $t = \Delta_i \oslash C_i$
8 $new\_smp = \frac{1-\mu}{\sum t} \times t$
9 $smp^i = (smp^i - \frac{\mu}{K+1}) \times (1 - lr) + new\_smp \times lr + \frac{\mu}{K+1}$
10 **end**
11 **end**

Algorithm 3 describes the update process of the *smp* values. The efficiency of knowledge transfer is evaluated based on the improvement amount of offspring relative to its parent. Δ_i is a vector in which each element represents the sum of the percentage improvement of task T_i when performing crossover or mutation throughout a generation, and $\sum \Delta_i$ is the sum of all elements in the vector. After each iteration from lines 5 to 11, if the improvement amount of each task ($\sum \Delta_i$) in the current generation is more than zero (line 6), the *smp* value is updated. Otherwise, its value remains unchanged. In lines 7 and 8, the average improvement amount on each dimension is calculated, and it is used to create the new_smp value. The purpose of calculating vector t is to determine the average improvement percentage of a generation through the use of the improvement percent matrix Δ and count offsprings matrix C. The presence of the parameter μ allows the transfer probability between a pair of tasks to be more than zero (according to equation 3). The smp^i vector is updated in line 9 by accumulating changes and the speed of this update is regulated by the learning rate lr $(0 \leq lr \leq 1)$:

$$smp^i = (smp^i - \frac{\mu}{K+1}) \times (1 - lr) + new\_smp \times lr + \frac{\mu}{K+1} \quad (7)$$

When the *lr* approaches 0, the impact of new_smp on the value of smp^i becomes minimal, leading to smp^i changing very little over a small range. Conversely, as the *lr* approaches 1, the value of smp^i becomes unstable due to being heavily influenced by the latest new_smp measurement, which is taken only one generation ago. Hence, a good value of *lr* maintains stability in the update process of the smp^i vector.

B. Dimension-aware Selection

Algorithm 4: DaS_strategy

1 **Input:**
2 - Individual o, original individual p_o and their skill factor τ_p
3 - Skill factor knowledge from τ_{knwl}
4 **Output:** New individual o'
5 Get probability crossover of dimensions $PCD_{\tau_p,\tau_{knwl}}$
6 $isTransfered \leftarrow false$
7 **for** $i \leftarrow 1$ *to* D **do**
8 **if** *rand(0, 1)* $\leq PCD_{\tau_p,\tau_{knwl}}[i]$ **then**
9 $o'[i] \leftarrow o[i]$
10 $isTransfered \leftarrow true$
11 **else**
12 $o'[i] \leftarrow p_o[i]$
13 **end**
14 **end**
15 **if** *not isTransfered* **then**
16 $j \leftarrow randint(1, D)$
17 $o'[j] \leftarrow o[j]$
18 **end**

Algorithm 5: Update_DaS

1 **Input:**
2 - Population $P = \{P_k, 1 \leq k \leq K\}$
3 - PCD vectors
4 **Output:** PCD vectors after update
5 **for** *each* P_τ *in* P **do**
6 $m_\tau = \frac{1}{|P_\tau|} \times \sum_{i=1}^{|P_\tau|} x^{(i)}$
7 $std_\tau = \sqrt{\frac{1}{|P_\tau|} \times \sum_{i=1}^{|P_\tau|} (x^{(i)} - m_\tau)^2}$
8 **end**
  `/* Update PCD vectors */`
9 **for** $\tau_a \leftarrow 1$ *to* K **do**
10 **for** $\tau_b \leftarrow 1$ *to* K **do**
11 $kl\_divergence = \log(\frac{std_{\tau_a}}{std_{\tau_b}}) + \frac{std^2_{\tau_b} + (m_{\tau_a} - m_{\tau_b})^2}{2 \cdot std^2_{\tau_a}} - \frac{1}{2}$
12 $PCD_{\tau_a,\tau_b} = e^{-\eta \times kl\_divergence}$
13 **end**
14 **end**

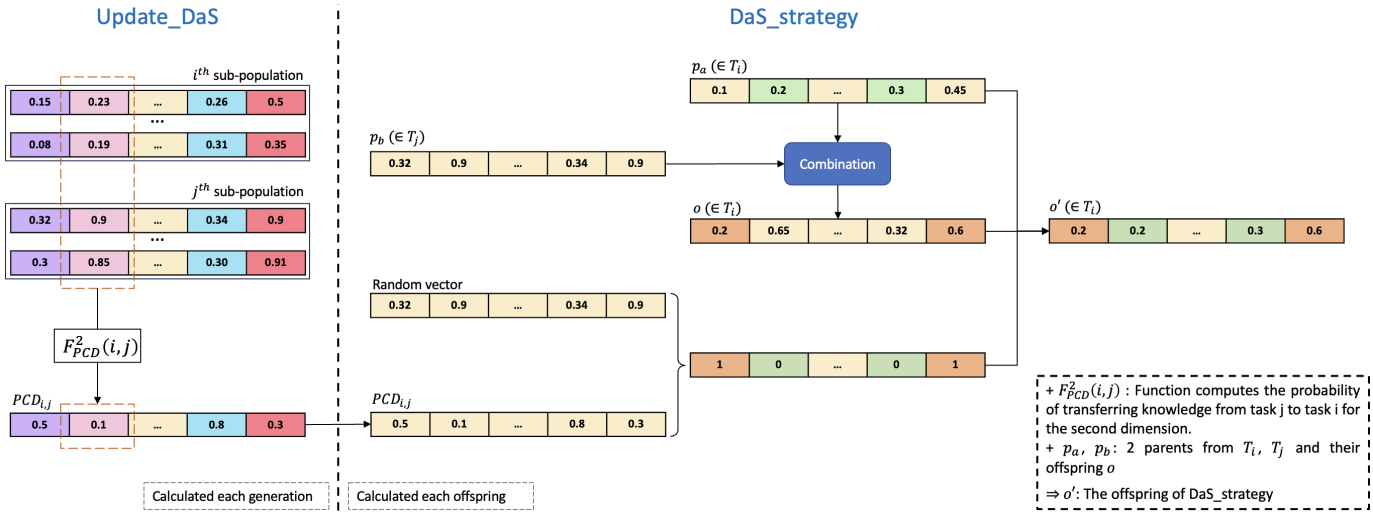

Fig. 1: The overview of our Dimension-aware Selection Process. There are two main components: DaS_strategy performed when creating offspring and Update_DaS performed in each generation. The Update_DaS modifies the vector $PCD_{i,j}$, which indicates the likelihood of knowledge being transferred from task T_j to T_i. In each generation, after an offspring o is generated from two parents p_a and p_b, the vector $PCD_{i,j}$ is utilized to select the dimensions to produce the offspring o' in DaS_strategy. The vector $PCD_{i,j}$ is compared with a random vector to differentiate between transferred dimensions (orange) and non-transferred dimensions (green). For the transferred dimensions, o' adopts the corresponding values from o. For the non-transferred dimensions, o' retains the corresponding value from parent p_a of the same task.

This section covers the DaS strategy, presented in Algorithm 4. The DaS strategy determines the knowledge transfer probability in every dimension separately. To achieve this goal, we employ the vector PCD_{τ_a,τ_b} (Probability Crossover Dimension) within the range $[0,1]^D$ to represent the similarity across each dimension of two sub-populations with skill-factor τ_a, τ_b and to determine the crossover probability. More specifically, the $PCD_{\tau_a,\tau_b}[i]$ corresponds to the knowledge transfer probability in the i^{th} dimension between two sub-populations.

For dimensions with a high degree of similarity, the chance of knowledge transfer between tasks in terms of those dimensions is increased. In contrast, for dimensions with a low degree of similarity, the opportunity for knowledge transfer in those dimensions is low. To decide a dimension for the knowledge transfer, we pick a random value in the range of 0 to 1. If this value is smaller than the $PCD_{\tau_a,\tau_b}[i]$ value, the knowledge transfer between two tasks occurs at this dimension. Otherwise, it is not permitted.

Algorithm 5 illustrates the update mechanism of the *PCD* vectors, which occurs at the end of each generation. In the initial generation, since we do not have precise knowledge of the sub-population's distribution, we assign a value of 1 to the parameter *PCD* in all dimensions. This implies that in the first generation, knowledge transfer takes place across all dimensions.

The KL-Divergence value indicates the extent of difference between two distributions, with a range of values from 0 to $+\infty$, where a higher value signifies a greater degree of divergence. We assume that the values of the sub-populations in the dimensions follow a Gaussian distribution [11], [12], [19]. Using this assumption, KL divergence value is computed in equation 8.

$$KL(p,q) = \log\frac{\sigma_q}{\sigma_p} + \frac{\sigma_p^2 + (m_p - m_q)^2}{2\cdot\sigma_q^2} - \frac{1}{2} \tag{8}$$

with p and q is two Gauss distribution with mean m_p, m_q and standard deviation σ_p, σ_q.

Afterward, we normalize the KL-Divergence value to the range from 0 to 1 using the equation $F(x) = e^{-\eta\cdot x}$. The value of function $F(x)$ obtained indicates the degree of similarity between the two distributions p and q, with a higher value indicating a greater similarity. Combining the calculations, the i^{th} of the PCD_{τ_a,τ_b} vector can be expressed through equation (9).

$$F^i_{PCD}(\tau_a,\tau_b) = \exp\left(-\eta\cdot\left(\log\frac{\sigma_b^i}{\sigma_a^i} + \frac{{\sigma_a^i}^2 + (m_a^i - m_b^i)^2}{2\cdot{\sigma_b^i}^2} - \frac{1}{2}\right)\right) \tag{9}$$

where variables m_a^i, m_b^i and σ_a^i, σ_b^i represent the means and standard deviations of sub-populations P_{τ_a} and P_{τ_b} in i^{th} dimension, respectively.

Therefore, we can measure the similarity in terms of dimension based on *PCD* and suppress negative transfer between different tasks while preserving the original Intra-Crossover process. Specifically, the proposed algorithm yields $PCD_{k,k} = [1]^D$, meaning crossover always occurs if the parents have the same skill-factor. Moreover, because PCD_{τ_a,τ_b} represents the degree of knowledge transfer of τ_b to τ_a so $PCD_{\tau_a,\tau_b} \neq PCD_{\tau_b,\tau_a}$. Suppose that knowledge from T_{τ_a} is advantageous for T_{τ_b}, but it is uncertain if the reverse is also true. An example of the Dimension-Aware Selection Process is depicted in Figure 1.

IV. Experiments and Results

A. Experimental Settings

To evaluate the efficiency of the proposed algorithm, we conduct experimental studies on two benchmark datasets: CEC'17 [21] and GECCO20 [22], for many-task optimization.

CEC'17, described in table I, comprises ten tasks with different properties. They are divided into two categories: E for easy tasks (from task 1 to task 4) and C for complicated tasks (from task 5 to task 10). In this dataset, C tasks can obtain positive knowledge transfer from their respective E tasks to accelerate the search process and improve solution quality. For example, T_1 is an assisted task of T_5; hence, solving them simultaneously can achieve better results than doing so separately.

The GECCO20 dataset has been used as a benchmark for GECCO 2020 and CEC 2022 competition in Multitask Optimization. This dataset consists of 50 complex optimization problems with many different shift and rotation operators to increase the difficulty of finding optimal solutions. Therefore, researchers use GECCO20 to evaluate the algorithm's performance in solving complex tasks simultaneously.

Our proposed algorithms, SM-MFEA and SM-MFEA with DaS strategy (SM-MFEA-DaS), are compared to 7 state-of-the-art many-tasks algorithms: EME-BI [14], LSA [13], MaTGA (MaTEA [16] with genetic algorithm solver), MTEA-AD [12], SBGA [18], EBSGA [17], and MFEA [1]. We run every algorithm 30 times with different random seeds on the same machine. For a fair comparison, we keep the hyperparameters of each algorithm to their value in the original paper. The hyperparameters are selected as below:

- Minimum population size of each task N_{min} is 30.
- Maximum number of evaluations is *K* x 100,000 with *K* being the number of tasks.
- Parameter nc = 2 for Simulated Binary Crossover (SBX)
- Parameter nm = 5 for Polynomial Mutation
- Parameter of SM-MFEA: $lr = 0.1$ and $\mu = 0.1$ (using Friedman Test - Table II)
- Parameter of DaS strategy: $\eta = 3$ (using Friedman Test - Table III)

B. Experimental Scenario

In this section, the efficacy of our proposed algorithm is evaluated in several different experiments:

- Experiment 1: Evaluate the effectiveness of SM-MFEA and SM-MFEA-DaS against other algorithms regarding solution quality, convergence trends, and running time.
- Experiment 2: Evaluate the effect of *smp* on knowledge transfer.
- Experiment 3: Evaluate the efficiency of DaS strategy when applying on MFEA, SBGA, MaTGA, LSA, EME-BI, and SM-MFEA.
- Experiment 4: Evaluate the influence of DaS on knowledge transfer in each dimension and its impact on the solution quality.

C. Results and Discussions

1) Comparison of SM-MFEA and SM-MFEA-DaS with state-of-the-art algorithms:

In this section, we evaluate the performance of SM-MFEA and SM-MFEA-DaS against state-of-the-art algorithms regarding solution quality, convergence rate, and computational efficiency.

The convergence trends of all algorithms in the CEC'17 dataset are depicted in Figure 2. The average objective value from 30 separate runs is plotted on the *y*-axis, and the generation number is represented on the *x*-axis. Evidently, SM-MFEA and SM-MFEA-DaS converge to better solutions than other state-of-the-art algorithms. In addition, SM-MFEA surpasses LSA, MaTGA, MTEA-AD, SBGA, EBSGA, and MFEA in 9 of 10 tasks, and with the addition of DaS, it excels in all 10 tasks. Among the various algorithms, only SM-MFEA-DaS can fully solve task T_{10} and achieving optimal results on the CEC'17 benchmark. Moreover, Figure 2 demonstrates that SM-MFEA-DaS achieves optimal solutions with an average of only $45.6\% \times$ `MAXEVALS` evaluations. Markedly, our algorithms surpass other algorithms in terms of convergence rate.

The results are shown in Table IV and V and are characterized as "better", "equal", or "worse" to indicate the superiority of SM-MFEA's solutions in each instance. To statistically evaluate the efficiency of the algorithms, we use the Wilcoxon signed-rank test with α = 0.05 on the GECCO20 dataset. The statistical results show that SM-MFEA and SM-MFEA-DaS remarkably outperform LSA, MaTGA, MTEA-AD, SBGA, EBSGA, and MFEA, with all p-values being less than 0.05. Compared with EME-BI, it shows that SM-MFEA reaches the same results, while SM-MFEA-DaS demonstrates superior performance.

Results show that both SM-MFEA and SM-MFEA-DaS represent a significant performance when compared to others. This is a surprisingly good result because our model solely introduces a novel probability model (SM-MFEA) and a dimension strategy (DaS), and has not integrated with other complex techniques (such as DE or search operators as seen in EME-BI).

Figure 3 compares the running time of SM-MFEA, SM-MFEA-DaS, and other algorithms on the benchmark datasets: CEC'17 and GECCO20. For a fair comparison, we run every algorithm on the same device and `MAXEVALS`. However, because they are not implemented in the same language (EME-BI, EBS-GA are in Java; MTEA-AD is in Matlab; SBGA, LSA, MaTGA, and SM-MFEA are in python), we record the ratio of these algorithms' runtime to MFEA's one in same programming language and benchmark dataset (show in equation 10).

$$Computation\_time(A) = \frac{Time_{pl}(A)}{Time_{pl}(MFEA)} \tag{10}$$

where:

- A is the algorithm that needs to caculate the computation time.

TABLE I: Details of the CEC'17 dataset

Task	Function	Search Space	Global Optima	USS Optima	Category	Assisted Tasks
T_1	$Sphere_1$	$[-100, 100]^{50}$	0^{50}	0.5^{50}	E-task	None
T_2	$Sphere_2$	$[-100, 100]^{50}$	80^{50}	0.9^{50}	E-task	None
T_3	$Sphere_3$	$[-100, 100]^{50}$	-80^{50}	0.1^{50}	E-task	None
T_4	$Weierstrass_1$	$[-0.5, 0.5]^{25}$	-0.4^{25}	$0.1^{25} \times [0, 1]^{25}$	E-task	None
T_5	$Rosenbrock$	$[-50, 50]^{50}$	0^{50}	0.5^{50}	C-task	T_1
T_6	$Ackley$	$[-50, 50]^{50}$	40^{50}	0.9^{50}	C-task	T_2
T_7	$Weierstrass_2$	$[-0.5, 0.5]^{50}$	-0.4^{50}	0.1^{50}	C-task	T_3, T_4
T_8	$Schewefel$	$[-500, 500]^{50}$	420.9687^{50}	0.9209687^{50}	C-task	None
T_9	$Griewank$	$[-100, 100]^{50}$	$-80^{25} \times 80^{25}$	$0.1^{25} \times 0.9^{25}$	C-task	T_4
T_{10}	$Rastrigin$	$[-50, 50]^{50}$	$40^{25} \times -40^{25}$	$0.9^{25} \times 0.1^{25}$	C-task	None

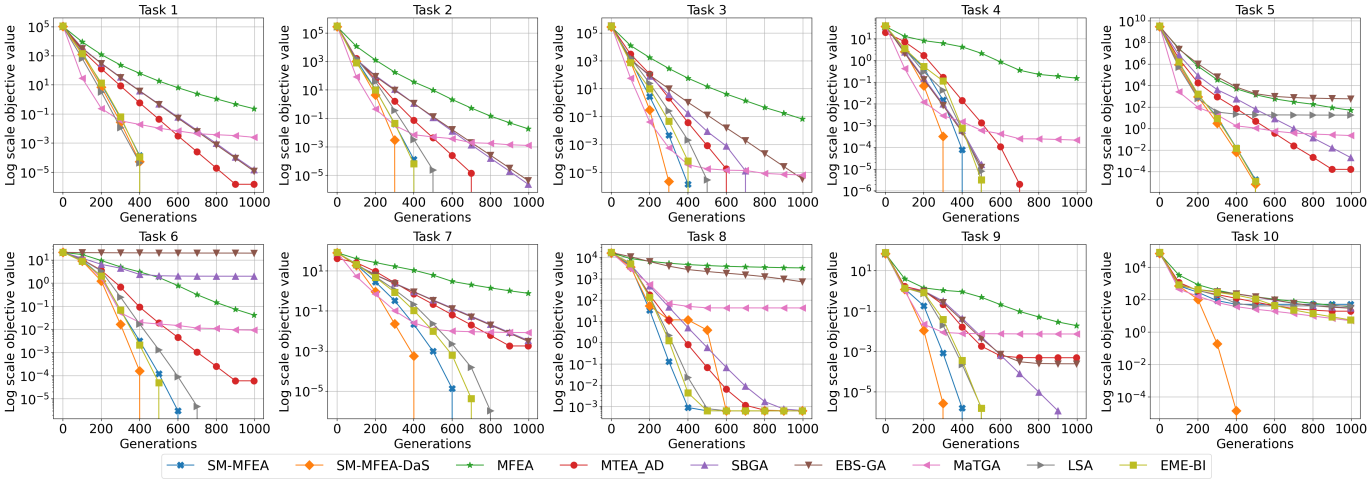

Fig. 2: Convegence trend of algorithms on the CEC'17 dataset

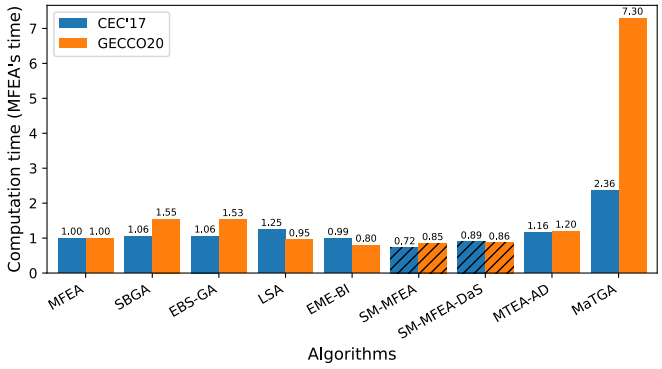

Fig. 3: Computation time

TABLE II: Result of the Friedman test on different with different lr values

Param	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
μ	**1.25**	3.05	3.15	3.95	4.70	6.40	7.10	7.10	8.30
lr	**5.15**	8.20	9.35	10.35	9.10	9.30	6.85	7.85	9.15

TABLE III: Result of the Friedman test on different with different η values

Param	0	1	2	3	4
η	6.60	4.40	4.40	**3.70**	4.20

TABLE IV: The statistic values obtained by conducting Wilcoxon-signed rank test with $\alpha = 0.05$ on 50-task GECCO20 benchmarks (SM-MFEA is the control algorithm). Decisions $+, \approx$, and $-$ indicate that SM-MFEA is significantly better, similar, and worse than other algorithms

Algorithms	Better	Equal	Worse	p-value	Decision
MFEA	500	0	0	0	$+$
MTEA-AD	500	0	0	0	$+$
SBGA	390	0	110	0	$+$
EBS-GA	426	0	74	0	$+$
MaTGA	331	0	169	0.01	$+$
LSA	288	115	97	0	$+$
EME-BI	133	159	208	0.21	$\approx$

- $Time_{pl}(X)$ is the running time of the algorithm X implemented in the language pl.

The average experimental results in Figure 3 show that SM-MFEA and SM-MFEA-DaS consume less time than EBS-GA, SBGA, LSA, MaTGA, and MTEA-AD on both benchmark datasets. In comparison with EME-BI, the runtime of SM-MFEA and SM-MFEA-DaS are the same.

2) Analysis of smp effect on knowledge transfer:

We use an area chart in Figure 4 to analyze the smp's impact on SM-MFEA on the CEC'17 dataset. In the figure, each chart shows the probability of transfer that a task received from another. More precisely, for the i^{th} chart, the horizontal axis indicates the generations of the i^{th} sub-population while

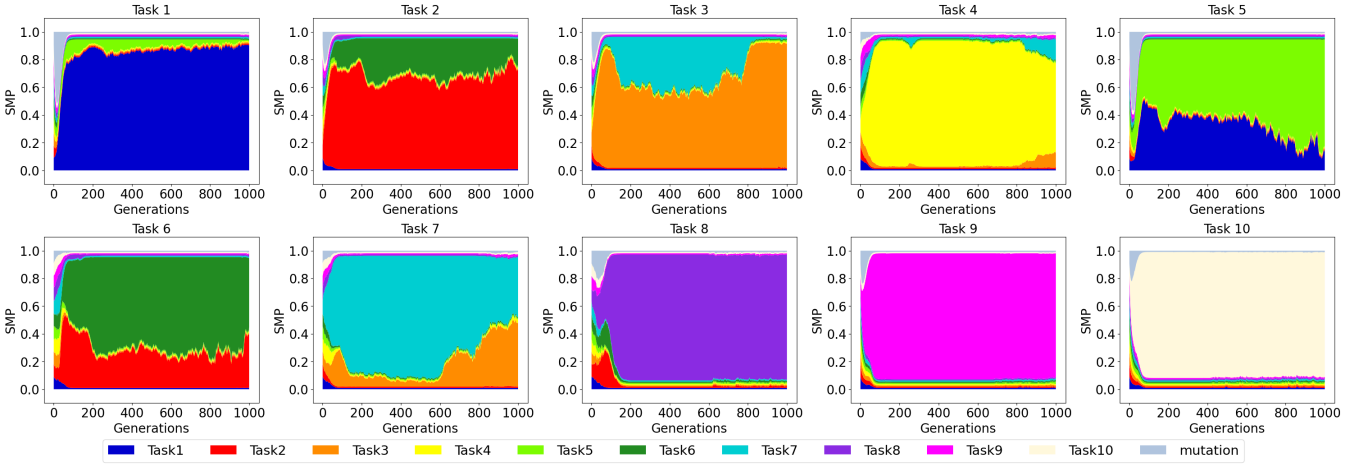

Fig. 4: The learned SMP vectors in SM-MFEA

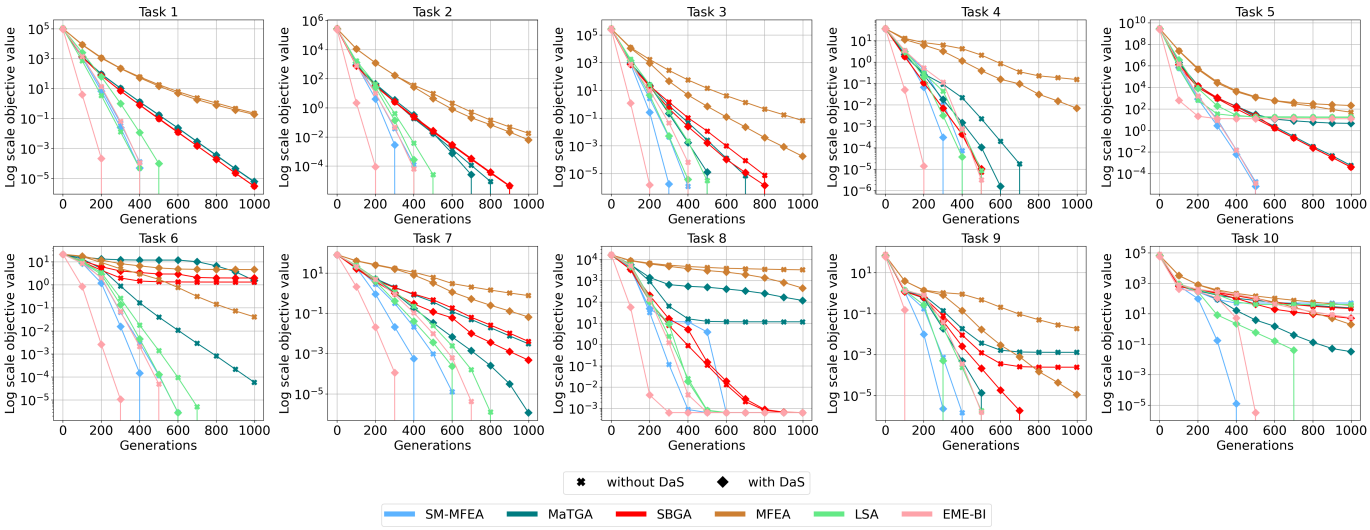

Fig. 5: Comparison of convergence trends of algorithms when using the DaS strategy.

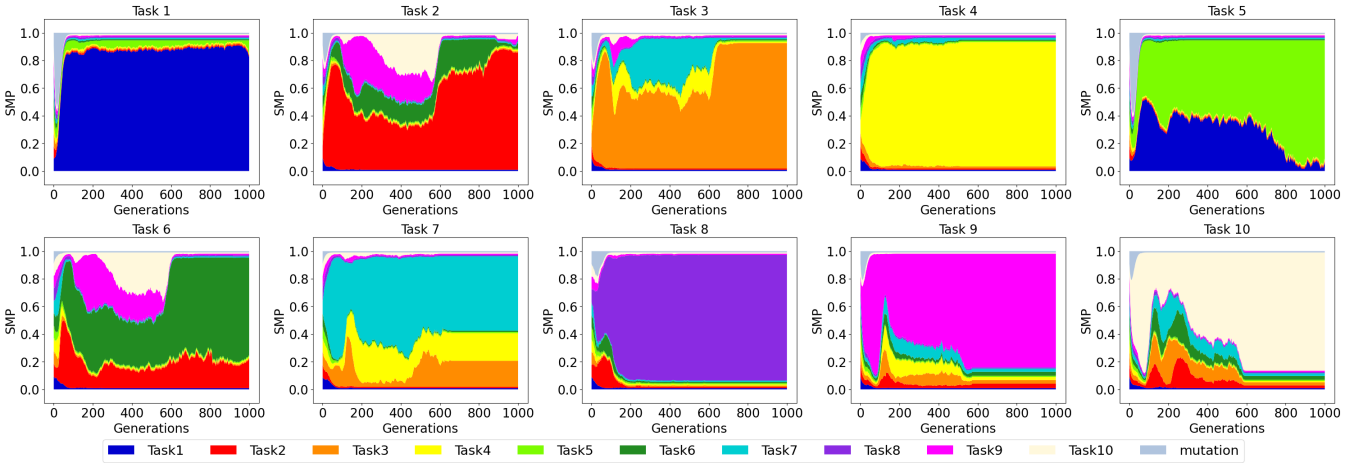

Fig. 6: The learned SMP vectors in SM-MFEA-DaS

TABLE V: The statistic values obtained by conducting Wilcoxon-signed rank test with α = 0.05 on 50-task GECCO20 benchmarks (SM-MFEA-DaS is the control algorithm). Decisions $+, \approx$, and $-$ indicate SM-MFEA-DaS is significantly better, similar, and worse than other algorithms

Algorithms	Better	Equal	Worse	p-value	Decision
MFEA	500	0	0	0	+
MTEA-AD	500	0	0	0	+
SBGA	435	0	65	0	+
EBS-GA	431	0	69	0	+
MaTGA	365	0	135	0	+
LSA	284	147	69	0	+
EME-BI	178	167	155	0	+
SM-MFEA	232	152	116	0	+

TABLE VI: The statistic values obtained by conducting Wilcoxon-signed rank test with α = 0.05 on 50-task GECCO20 benchmarks (compare algorithm effectiveness with/without DaS). Decisions $+, \approx$, and $-$ indicate using DaS is significantly better, similar, and worse than not using DaS

Algorithms	Better	Equal	Worse	p-value	Decision
MFEA	388	0	112	0	+
LSA	315	4	181	0	+
SBGA	306	0	196	0.009	+
MaTGA	269	0	231	0.038	+
EME-BI	244	118	138	0	+
SM-MFEA	147	212	141	0	+

the vertical axis indicates the value of the smp^i vector during the evolutionary process.

By utilizing the smp vector, knowledge transfer probability is improved among the tasks. In Figure 4, the area chart indicates that knowledge transfer probability between tasks can be increased over $\frac{1}{K}$. Specifically, the probability of knowledge transfer from task T_1 to T_5, task T_3 to T_7, task T_7 to T_3, task T_2 to T_6, and task T_6 to T_2 is substantially higher than 0.1, which cannot be obtained by using algorithms based on rmp. Combined with the results obtained in section IV-C1, it demonstrates the effectiveness of the smp-based algorithm. This accomplishment is attributed to its ability to surpass the maximum probability threshold of the *rmp*-based algorithms, as evidenced in section III-A.

3) Comparison of the performance of state-of-the-art algorithms when applying DaS strategy:

In this section, we evaluate the performance of the DaS independently from SM-MFEA. More specifically, we apply DaS strategy to several well-known algorithms (MFEA, LSA, SBGA, MaTGA, EME-BI and SM-MFEA) and record their solution quality and convergence rate. Figure 5 and Table VI present the results of algorithms with and without DaS over 30 independent runs on the CEC'17 and GECCO20 benchmarks.

In Table VI, the better, equal, and worse columns imply that using the DaS is better, equal, or worse than without using it on GECCO20 50-task benchmark dataset according to the Wilcoxon signed-rank test. The table's results confirm that the DaS strategy has significantly improved the effectiveness of experimental algorithms.

Furthermore, in Figure 5, we compare results and investigate the convergence trends of algorithms when applying DaS strategy on the CEC'17. In the Figure, the *y*-axis is the average objective value over 30 independent runs, while the *x*-axis is the generation count. It is evident that the DaS strategy integrated algorithms obtain better results and tend to converge faster than the original ones without using the DaS strategy.

4) Analysis of the effect of DaS on knowledge transfer: Figure 6 illustrates the value of *smp* vectors among tasks with SM-MFEA using DaS strategy on CEC'17 dataset. It shows that knowledge transfer on some dimensions of strategy facilitates more positive knowledge transfer among pairs of tasks. Therefore, it can help algorithms to reach near optimal solutions significantly faster in most cases (presented in figure 5).

The effectiveness of the proposed DaS Strategy is well demonstrated in the CEC'17 dataset. In this problem, tasks are expected to assist each other if they have the same objective or optimal point in Unified Search Space (USS). For instance, task T_2 is expected to assist task T_6 because they all be optima at 0.9^{50}. Hence, in challenging cases, it is anticipated that the task will receive no knowledge transfer because of the difference of optimal points in the USS. Using CEC'17 as an illustration, Table I shows that no task has the same optimal point in the USS as task T_{10} over 50 dimensions. Although several tasks share the same optimal point with task T_{10} while considering only the first 25 or last 25 dimensions (T_2 and T_6 for the former, and tasks T_3 and T_7 for the latter), task T_{10} will still receive negative knowledge transfers on the remaining dimensions when the crossover is performed with these tasks. Specifically, in SM-MFEA algorithm, the probabilities of knowledge transfer from the other tasks to task T_{10} tend to $\frac{\mu}{K+1}$ (shown in Figure 4). Consequently, task T_{10} cannot be solved not only in SM-MFEA but also in the other algorithms unless the DaS Strategy is utilized. (as depicted in Figure 5).

The previous analysis also suggests to overcome the issue of task T_{10}. The DaS strategy tries to only select dimensions with high similarity to transfer knowledge. In the case of task T_{10}, it is expected that tasks T_2, T_3, T_6, and T_7 then are evaluated as suitable to transfer knowledge to T_{10}. Upon considering task T_{10} in Figure 4 and Figure 6, it becomes apparent that the DaS strategy has had a positive impact on T_{10}'s evolution progress when it has received more knowledge from the appropriate tasks as previously analyzed (Tasks T_2, T_3, T_6, and T_7). The impact of this knowledge transfer is demonstrated in the results of task T_{10} in Figure 5 where the top 5 out of the 12 test cases are all "with DaS". Notably, 3 out of the 6 algorithms that are experienced with DaS achieved the global optimum, which was not attained by other algorithms without the use of DaS.

V. Conclusion

This paper introduces a novel approach to efficiently handle a large number of tasks with high dimensions for multitasking optimization. The novelty of our proposal is twofold. First, similar mating probability (*smp*) is used to leverage past successful transfers to determine the optimal knowledge transfer

probability for every pair of tasks. Second, a dimension-aware selection (DaS) strategy improves knowledge transfer efficiency on each dimension between two tasks. Experimental results on benchmark datasets show that the proposed SM-MFEA algorithm is much better than other state-of-the-art algorithms regarding solution quality, convergence trend, and running time. The experiments also indicate the impressive efficiency of the DaS strategy.

Many optimization problems in practice are modeled as many-task problems. Solving these problems is a significant interest for scientists. In this study, the experiments only demonstrate the efficiency of the proposed algorithms on the Benchmark datasets, which has yet to be utilized to solve real-world problems.

In future work, we will extend the DaS strategy to enable positive transfer on all dimensions, as well as optimize computing resource usage. It will be interesting to investigate the applications of the proposed algorithms in real-world problems such as network design, production scheduling, and vehicle routing.

REFERENCES

[1] A. Gupta, Y.-S. Ong, and L. Feng, "Multifactorial evolution: Toward evolutionary multitasking," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 3, pp. 343–357, 2016.

[2] A. Gupta, L. Zhou, Y. S. Ong, Z. Chen, and Y. Hou, "Half a dozen real-world applications of evolutionary multitasking, and more," *IEEE Computational Intelligence Magazine*, vol. 17, pp. 49–66, 2021.

[3] H. T. T. Binh, T. B. Thang, N. B. Long, N. V. Hoang, and P. D. Thanh, "Multifactorial evolutionary algorithm for inter-domain path computation under domain uniqueness constraint," in *2020 IEEE Congress on Evolutionary Computation (CEC)*, 2020, pp. 1–8.

[4] R. Chandra, A. Gupta, Y.-S. Ong, and C.-K. Goh, "Evolutionary multi-task learning for modular training of feedforward neural networks," vol. 9948, 10 2016, pp. 37–46.

[5] R. Sagarna and Y.-S. Ong, "Concurrently searching branches in software tests generation through multitask evolution," in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2016, pp. 1–8.

[6] L. Zhou, L. Feng, J. Zhong, Y.-S. Ong, Z. Zhu, and E. Sha, "Evolutionary multitasking in combinatorial search spaces: A case study in capacitated vehicle routing problem," in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2016, pp. 1–8.

[7] Y. Yuan, Y.-S. Ong, A. Gupta, P. S. Tan, and H. Xu, "Evolutionary multitasking in permutation-based combinatorial optimization problems: Realization with tsp, qap, lop, and jsp," in *2016 IEEE Region 10 Conference (TENCON)*, 2016, pp. 3157–3164.

[8] A. Gupta, Y.-S. Ong, L. Feng, and K. C. Tan, "Multiobjective multifactorial optimization in evolutionary multitasking," *IEEE Transactions on Cybernetics*, vol. 47, no. 7, pp. 1652–1665, 2017.

[9] J. Ding, C. Yang, Y. Jin, and T. Chai, "Generalized multitasking for evolutionary optimization of expensive problems," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 1, pp. 44–58, 2019.

[10] Y.-W. Wen and C.-K. Ting, "Parting ways and reallocating resources in evolutionary multitasking," in *2017 IEEE Congress on Evolutionary Computation (CEC)*, 2017, pp. 2404–2411.

[11] K. K. Bali, Y.-S. Ong, A. Gupta, and P. S. Tan, "Multifactorial Evolutionary Algorithm With Online Transfer Parameter Estimation: MFEA-II," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 1, pp. 69–83, 2020.

[12] C. Wang, J. Liu, K. Wu, and Z. Wu, "Solving multitask optimization problems with adaptive knowledge transfer via anomaly detection," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 2, pp. 304–318, 2022.

[13] T. B. Thang, T. C. Dao, N. H. Long, and H. T. T. Binh, "Parameter adaptation in multifactorial evolutionary algorithm for many-task optimization," *Memetic Computing*, vol. 13, pp. 1–14, 12 2021.

[14] H. T. T. Binh, L. V. Cuong, T. B. Thang, and N. H. Long, "Ensemble multifactorial evolution with biased skill-factor inheritance for many-task optimization," *IEEE Transactions on Evolutionary Computation*, 2022, DOI:10.1109/TEVC.2022.3227120.

[15] X. Zheng, A. K. Qin, M. Gong, and D. Zhou, "Self-regulated evolutionary multitask optimization," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 1, pp. 16–28, 2020.

[16] Y. Chen, J. Zhong, L. Feng, and J. Zhang, "An adaptive archive-based evolutionary framework for many-task optimization," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 3, pp. 369–384, 2020.

[17] R.-T. Liaw and C.-K. Ting, "Evolutionary many-tasking based on biocoenosis through symbiosis: A framework and benchmark problems," in *2017 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2017, pp. 2266–2273.

[18] ——, "Evolutionary manytasking optimization based on symbiosis in biocoenosis," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 4295–4303.

[19] S.-H. Wu, Z.-H. Zhan, K. C. Tan, and J. Zhang, "Orthogonal transfer for multitask optimization," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 1, pp. 185–200, 2023.

[20] R. Tanabe and A. S. Fukunaga, "Improving the search performance of shade using linear population size reduction," in *2014 IEEE Congress on Evolutionary Computation (CEC)*, 2014, pp. 1658–1665.

[21] B. Da, Y.-S. Ong, L. Feng, A. K. Qin, A. Gupta, Z. Zhu, C.-K. Ting, K. Tang, and X. Yao, "Evolutionary multitasking for single-objective continuous optimization: Benchmark problems, performance metric, and baseline results," *arXiv preprint arXiv:1706.03470*, 2017.

[22] K. Q. L. Feng, Y. Y. A. Gupta, Y.-S. O. E. Scott, and X. Chi, "New MTO benchmarks for GECCO 2020 competition on evolutionary multi-task optimization," July 2020. [Online]. Available: http://www.bdsc.site/websites/MTO/index.html.