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

Anonymous Author(s)

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. Conventional Random Mating Probability (*rmp*) and knowledge transfer techniques cannot handle a large number of tasks with high dimension efficiently. To address these limitations, this paper proposes a new algorithm called Similar Mating Multifactorial Evolutionary Algorithm (SM-MFEA) for multitasking optimization. This algorithm employs a novel approach for determining the optimal knowledge transfer probability of each task pair by leveraging previous successful transfers. In addition, a Dimension-aware Selection (DaS) strategy is proposed to improve knowledge transfer efficiency by selecting dimensions for transferring based on the similarity in each dimension between two tasks. Experimental results on benchmark datasets show that the proposed algorithm is superior to other state-of-the-art evolutionary many-task algorithms. Our further analyses also confirm that the proposed DaS strategy can be easily integrated into different existing algorithms to significantly improve their performance.

CCS CONCEPTS

• **Theory of computation → Computability**; **Bio-inspired optimization**; • **Mathematics of computing → Bio-inspired optimization**.

KEYWORDS

Knowledge Transfer, Many-task Optimization, Multifactorial Evolutionary Algorithm

ACM Reference Format:

Anonymous Author(s). 2023. Efficient Knowledge Transfer with Similar Mating Probability and Dimension-aware Selection for Many-Task Optimization. In *Proceedings of The Genetic and Evolutionary Computation Conference 2023 (GECCO '23).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Recently, cloud computing has been playing a pivotal role in modern life as technology advances. In real-world situations, requests from the client side, also known as tasks, are frequently intertwined, and would require a great deal of time and resources if processed independently. Therefore, handling a large number of client requests concurrently in cloud computing demands an efficient optimization algorithm. 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 (EMO) 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. Specifically, Gupta et al. [10] 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.

Due to its prominent attributes, MFEA excels in a wide range of real-world applications [3, 4, 12, 13, 20], including complex combinatorial optimization problems [18, 20]. In [9], experimental results show that MFEA outperformed other works regarding solution quality, convergence time and resource consumption in bi-level optimization problems. Subsequently, Gupta et al. [11] 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 of study [10], a variety of models were produced and yielded superior results. For expensive optimization problems, Ding et al. proposed MCEEA [7], a multifactorial optimization method that improves convergence speed by transferring knowledge from cheap optimization tasks to expensive ones.

Despite its widespread application and development, MFEA has two 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 of the current algorithms rely on *rmp* to control the degree of knowledge transfer and by adjusting *rmp* , they can prevent negative transfer between low-similarity tasks. Eventually, the probability of random mating between two tasks in these algorithms converges to a very small value as the number of tasks increases significantly. More specifically, it can be seen that the probability of an individual of the j^{th} task being selected for mating with an individual of the i^{th} task is $\frac{1}{K}$, where K is the number of tasks. In addition, the probability of random mating between two individuals depends on *rmp* . Thus, the probability that a task transfers its knowledge to another task through random mating is $\frac{rmp}{K}(rmp \leq 1)$ with the maximum value of $\frac{1}{K}$.

Secondly, many multitasking algorithms treat every dimension equally while performing crossover. Hence, these methods are easily prone to negative transfer because in most cases, positive transfer only occurs in some dimensions.

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

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

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 Simmilar Mating Probability (*smp*) parameter to determine the effective mating probability between tasks. The introduction of *smp* allows our algorithm to enhance positive transfer between similar tasks.

Beside SM-MFEA, we propose a new strategy, Dimension-aware Selection (DaS) that uses KL - Divergence to measure tasks similarity in every dimension. Subsequently, dimensions with high similarity between tasks have higher transfer probability. In this paper, we applied this strategy to the Simulated Binary Crossover (SBX) operator to create a new operator called DaS-SBX.

To evaluate the effectiveness of SM-MFEA and DaS-SBX, 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. In addition, when applied to other algorithms, DaS-SBX also demonstrates superior results compared to SBX. Our contributions are summarized as follows:

- Develop 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 then compare them with state-of-the-art algorithms.

The rest of the paper is organized as follows. Section 2 covers related works. In section 3, SM-MFEA and DaS strategy is described in detail. Section 4 contains the experimental results and comparison with other algorithms. Finally, the conclusion and future works are presented in Section 5.

2 RELATED WORKS & OVERVIEW

2.1 Related works

Because of its capabilities in solving many-task simultaneously, MFEA has gained a great deal of 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 [10]. 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 [1, 15, 16] and maximizing positive transfer [5, 14, 19].

There are many algorithms proposed for minimizing negative transfer between tasks. Bali et al. [1] 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. As a result, in extreme cases, the intra-task crossover is more favorable, which will significantly curtail inter-task crossover and prevent knowledge transfer. Additionally, MFEA-II has immense time complexity when solving a large number of tasks. A recent publication of Wang et al. [15], Multitask Evolutionary Algorithms based on Anomaly Detection (MTEA-AD), has mitigated 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 model is strongly affected by the experience of only one previous generation. Recently, Thang et al. [14] 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*. However, the *rmp* in this algorithm only governs the extent of knowledge transfer between pairs of tasks instead of among all of the tasks. In [2], 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. EME-BI has shown better experimental results compared to existing algorithms.

Maximizing positive knowledge transfer is also a prominent approach. Zheng et al. [19] proposed a Self-regulated Evolutionary Multitask Optimization algorithm (SREMTO) to tackle the deficient knowledge transfer scheme of MFEA. The SREMTO 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. [5] 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.

Furthermore, another approach maximizes positive transfer by treating each dimension individually. In [17], Sheng-Hao Wu et al. proposed CTM strategy and 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.

2.2 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 offsprings 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.

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

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

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

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

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* 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.

3 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 following a uniform distribution from 0 to 1, 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 recording the total percentage improvement of the offspring relative to their parent. Additionally, we also create a matrix C counting the number of children born between pairs of tasks in each generation in line 12. Both the C and Δ matrices are updated. Finally, in lines 13 to 16, 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 17 and select the best individuals for the next generation in line 18. The population size is linearly decreased after each iteration to facilitate exploitation capacity [14]. Finally, PCD, C and Δ are updated in line 19 and 20, 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

1 **Input:** K optimization tasks $T_1, ..., T_K$ with maximum number of evaluations MAXEVALS.
2 **Output:** The best solutions of all tasks $X_k^*, 1 \leq k \leq K$.
3 $N_{min} \leftarrow$ the minimum population size of each task.
4 Initialize population $P(0)$ with $K \times N$ individuals.
5 Assign the fittest task for each individual.
6 Initialize probability vectors $smp^1, smp^2, ..., smp^K$.
7 Initialize count evaluation $evals \longleftarrow 0$.
8 Set $t \longleftarrow 0$.
9 **while** $evals \leq$ MAXEVALS **do**
10 Offsprings population $O(t) \leftarrow \emptyset$.
11 Improvement percent matrix $\Delta \leftarrow [0]_{K \times (K+1)}$.
12 The number of offsprings matrix $C \leftarrow [0]_{K \times (K+1)}$.
13 **while** $|O(t)| < |P(t)|$ **do**
14 $o_a, o_b =$ SM_recombination$(P(t), \Delta, C)$. ▷ See algorithm 2
15 $O(t) = O(t) \cup [o_a, o_b]$.
16 $evals \leftarrow evals + 2$.
17 $P(t+1) = P(t) \cup O(t)$.
18 $P(t+1) \leftarrow$ Select $\left(N + (N_{min} - N) * \frac{evals}{\text{MAXEVALS}}\right)$ best individuals from each sub-population.
19 Update_DaS$(P(t+1))$ ▷ See algorithm 5
20 Update_SMP(Δ, C) ▷ See algorithm 3
21 $t = t + 1$.

3.1 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:

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

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 miniscule, 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 vectors. Hence, the probability model smp allows α_i^k 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

1 **Input:**
2 - Parent's population P
3 - The Improvement Percent matrix Δ.
4 - Count evaluation matrix C.
5 **Output:** Offsprings o_a, o_b
6 $\tau_a \leftarrow$ Randomly select a task from list $\{1, ..., K\}$
7 $\tau_b \leftarrow$ Select a second task follow wheel selection with probability smp^{τ_a}
8 Set skill factor for offsprings $\tau_{o_a} = \tau_{o_b} = \tau_a$
9 **if** $\tau_b = \tau_a$ **then**
10 $p_a, p_b \leftarrow$ Randomly select two individuals from sub-population P_{τ_a}.
11 $o_a, o_b \leftarrow$ Intra-Crossover$(p_a, p_b, \tau_{o_a}, \tau_{o_b})$
12 **else if** $\tau_b = K+1$ **then**
13 $p_a, p_b \leftarrow$ Randomly select individuals from sub-population P_{τ_a}
14 $o_a, o_b \leftarrow$ Mutate p_a, p_b
15 **else**
16 $p_a, p_b \leftarrow$ Randomly select an individual from sub-population P_{τ_a} and P_{τ_b} respectively
17 $o_a, o_b \leftarrow$ Inter-Crossover$(p_a, p_b, \tau_{o_a}, \tau_{o_b})$
18 Evaluate o_a, o_b by their assigned skill factors
19 Δ_{τ_a,τ_b} += $\max(\frac{f_{p_a} - f_{o_a}}{f_{p_a}}, 0)^2 + \max(\frac{f_{p_a} - f_{o_b}}{f_{p_a}}, 0)^2$
20 C_{τ_a,τ_b} += 2

Algorithm 2 depicts how offsprings are produced during the evolutionary process. Initially, task τ_a is selected randomly while task 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 τ_a is the recipient of knowledge transfer from task τ_b. Next, we randomly select two parents (p_a and p_b) to undergo crossover. In the case of $\tau_b = K+1$, two random individuals from P_{τ_a} are chosen to mutate. Two offsprings (denoted o_a and o_b) are generated. They are then evaluated according to skill-factor values of their parents. Finally, we update the matrix Δ_{τ_a,τ_b}, and matrix C to calculate smp for next generation.

Algorithm 3: Update_SMP

1 **Input:**
2 - The Improvement Percent matrix Δ.
3 - Count evaluation 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**
/* $\oslash$ is element-wise division */
7 $t = \Delta_i \oslash C_i$
8 new_smp $= \frac{1-\mu}{\sum t} \times t$
9 $smp^i = (smp^i - \frac{\mu}{K+1}) \times (1 - lr) + \text{new\_smp} \times lr + \frac{\mu}{K+1}$

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. After each iteration from lines 5 to 9, if the improvement amount of each task 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 improvement amount on each dimension is calculated, and it is used to update the new_smp value. The presence of the parameter μ allows the transfer probability between a pair of tasks to be more than zero. The speed of the update process is determined by a learning rate lr ($0 \leq lr \leq 1$) in line 9. If lr tends to 0, smp stagnates; however, with lr tending to 1, the current smp value is calculated by bypassing its previous value during the update process. Hence, a good value of lr maintains stability in the update process of the smp value.

3.2 Dimension-aware Selection

This section covers the DaS strategy and its integration into SBX, a new crossover operator called DaS-SBX (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

Algorithm 4: DaS-SBX Crossover

Input:
- 2 parents p_a and p_b
- 2 skill factors of offspring τ_{o_a}, τ_{o_b}

Output: 2 offsprings o_a and o_b

Get probability crossover of dimensions PCD_{τ_a,τ_b}

Crossover p_a and p_b to create o_a and o_b:

for $i \leftarrow 1$ *to* D **do**
  if $rand(0, 1) \leq PCD_{\tau_a,\tau_b}[i]$ **then**
    $o_a[i], o_b[i] = \text{SBX\_crossover}(p_a[i], p_b[i])$
  else
    if $\tau_{o_a} == \tau_a$ **then**
      $o_a[i] = p_a[i]$
    else
      $o_a[i] = p_b[i]$
    if $\tau_{o_b} == \tau_a$ **then**
      $o_b[i] = p_a[i]$
    else
      $o_b[i] = p_b[i]$

Algorithm 5: Update_DaS

Input:
- Population $P = \{P_k, 1 \leq k \leq K\}$
- PCD vectors

Output: PCD vectors after update

for *each* P_τ *in* P **do**
  $m_\tau = \frac{1}{|P_\tau|} \times \sum_{i=1}^{|P_\tau|} x^{(i)}$
  $std_\tau = \sqrt{\frac{1}{|P_\tau|} \times \sum_{i=1}^{|P_\tau|} (x^{(i)} - m_\tau)^2}$

`/* Update PCD vectors */`

for $\tau_a \leftarrow 1$ *to* K **do**
  for $\tau_b \leftarrow 1$ *to* K **do**
    $kl\_divergence = \log(\frac{std_{\tau_a}}{std_{\tau_b}}) + \frac{std_{\tau_b}^2 + (\mu_{\tau_a} - \mu_{\tau_b})^2}{2 \cdot std_{\tau_a}^2} - \frac{1}{2}$
    $PCD_{\tau_a,\tau_b} = e^{-\eta \times kl\_divergence}$

random value in the range of 0 to 1. If this value is smaller than the $PCD_{\tau_a,\tau_b}[i]$ value, the crossover between two tasks occurs at this dimension. Otherwise, the crossover is not allowed.

In the rest of this paragraph, we describe the crossover DaS-SBX (presented in Algorithm 4), created by integrating DaS strategy into SBX crossover. At the beginning of Algorithm 4, $PCD_{\tau_a,\tau_b}[i]$ is the probability in which SBX crossover will be applied to the i^{th} dimension of the two parents. Otherwise, each offspring will inherit the i^{th} "allele" from the parents with the same skill factor.

Algorithm 5 illustrates the update mechanism of the PCD vectors, which occurs at the end of each generation. First, we assume that the values of the sub-populations in the dimensions follow a Gaussian distribution, the KL-Divergence value is computed using the mean (m_τ) and standard deviation (std_τ) of all individuals in P_τ at each dimension. Then, we normalize the KL-Divergence value to the range from 0 to 1 using the equation e^{-x}. Therefore, we can measure the similarity in terms of dimension based on the KL-divergence 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.

4 EXPERIMENTS AND RESULTS

4.1 Experimental Settings

To evaluate the efficiency of the proposed algorithm, we conduct experimental studies on two benchmark datasets: CEC'17 [5, 6] and GECCO20 [8], for many-task optimization.

CEC'17 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 functions 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 algorithm SM-MFEA and SM-MFEA with DaS-SBX (SM-MFEA-DaS) is compared to seven state-of-the-art many-tasks algorithms: EME-BI, LSA, MaTGA, MTEA-AD, SBGA, EBSGA, and MFEA. 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 of SM-MFEA: $lr = 0.1$ and $\mu = 0.1$ (using Friedman Test - Table 1)
- Parameter of DaS-SBX: $\eta = 3$ (using Friedman Test - Table 2)

Table 1: 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

4.2 Experimental Scenario

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

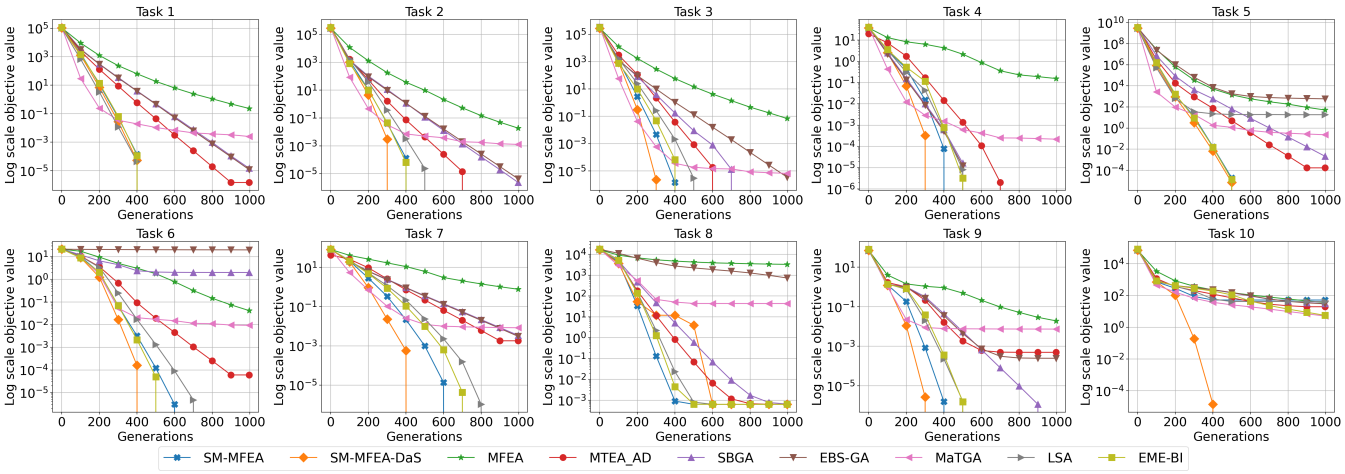

Figure 1: Convegence trend of algorithms on the CEC'17 dataset

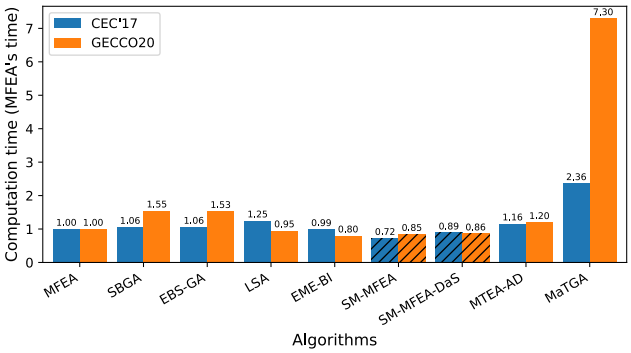

Figure 2: Computation time

Table 2: 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 3: 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 +, ≈, 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	≈

Table 4: The statistic values obtained by conducting Wilcoxon-signed rank test with $\alpha = 0.05$ on 50-task GECCO20 benchmarks (SM-MFEA-DaS is the control algorithm). Decisions +, ≈, 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	412	0	88	0	+
EBS-GA	425	0	75	0	+
MaTGA	349	0	151	0	+
LSA	301	113	86	0	+
EME-BI	146	159	195	0.28	≈
SM-MFEA	147	212	141	0	+

- 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-SBX against SBX when applying on LSA, MFEA, and SM-MFEA.
- Experiment 4: Evaluate the influence of DaS-SBX on knowledge transfer in each dimension and its impact on the solution quality.

4.3 Results and Discussions

4.3.1 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 in terms of solution quality, convergence rate, and computational efficiency. The results are shown in Table 3 and 4 and are characterized as "better," "equal," or "worse" to indicate the superiority of SM-MFEA's

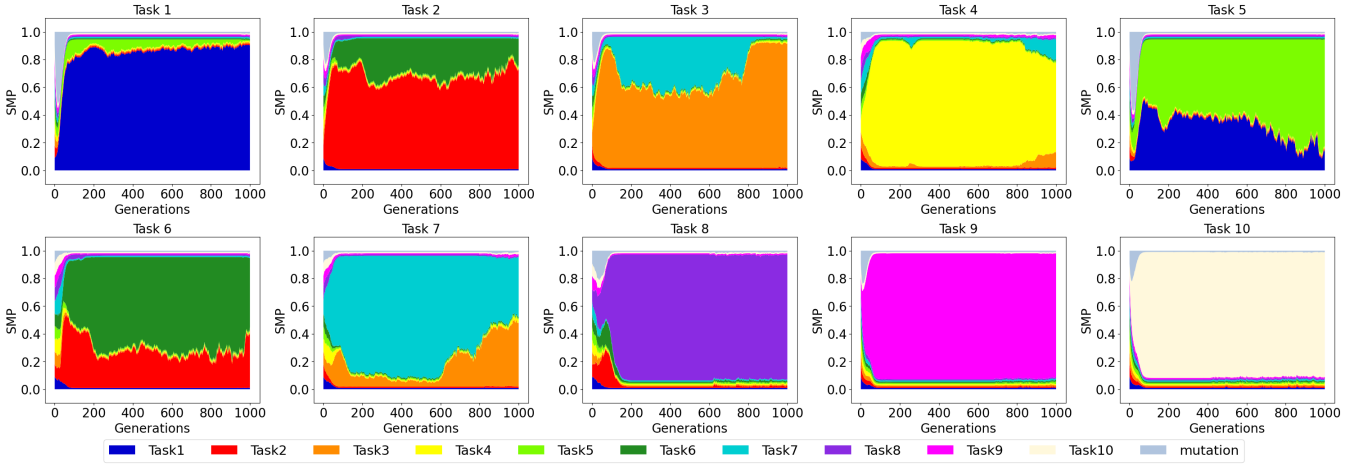

Figure 3: The learned SMP vectors in SM-MFEA

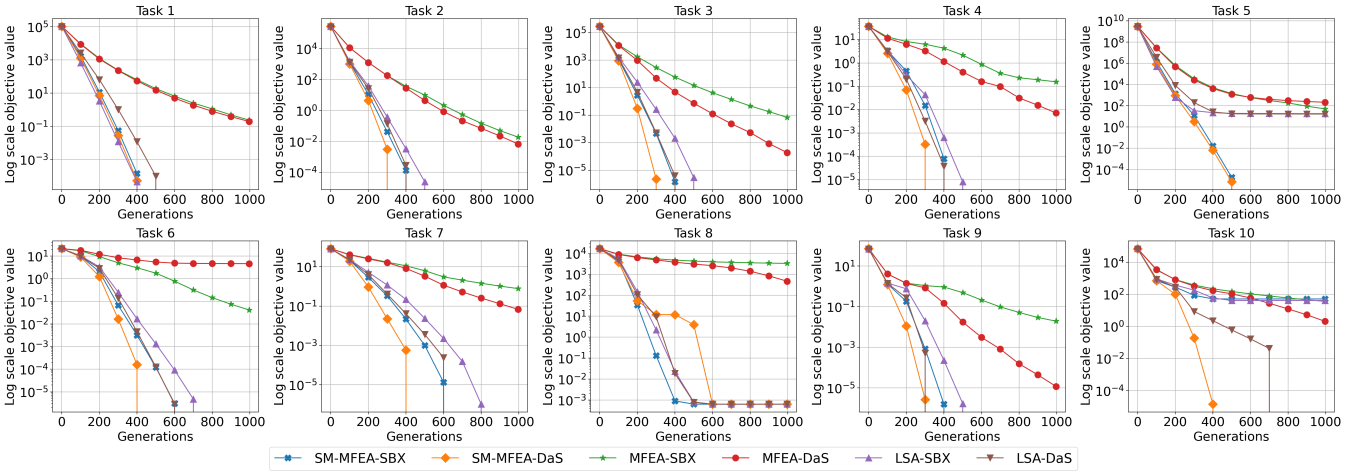

Figure 4: Convergence trends of SM-MFEA, LSA and MFEA using DaS-SBX or SBX

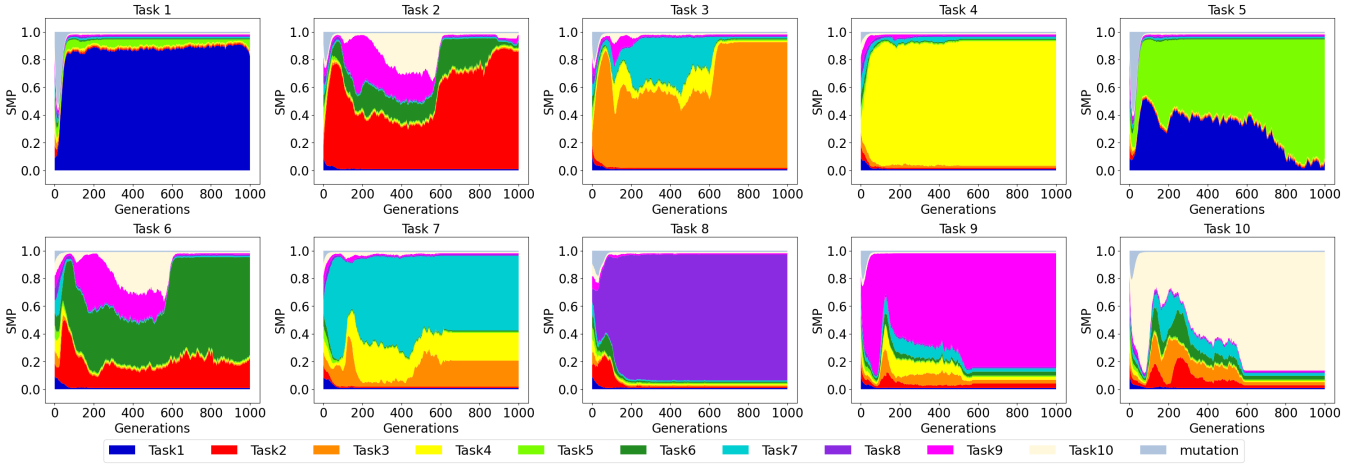

Figure 5: The learned SMP vectors in SM-MFEA-DaS

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 results show that SM-MFEA and SM-MFEA-DaS remarkably outperform LSA, MaTGA, MTEA-AD, SBGA, EBSGA, and MFEA or at least as well as EME-BI. The results are statistically significant because the p-value is less than 0.05.

The convergence trends of all algorithms in the CEC'17 dataset are depicted in Figure 1. 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. Compared to EME-BI, SM-MFEA obtains comparable results. Moreover, Figure 1 demonstrates that on average SM-MFEA-DaS only requires 45.6% × MAXEVALS evaluations to attain the optimal solutions. Markedly, our algorithms surpass other algorithms in terms of convergence rate.

Figure 2 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, SBGA, LSA, and MaTGA are in Java, MTEA-AD in Matlab, and SM-MFEA in Python), we record the ratio of these algorithms' runtime to MFEA's runtime in same programming language and benchmark dataset. The average experimental results in Figure 2 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.

4.3.2 *Analysis of smp effect on knowledge transfer.*

We use an area chart in Figure 3 to analyze the *smp*'s impact on SM-MFEA on the CEC'17 dataset. In the figure, each chart depicts the probability of knowledge transfer from other tasks to a task.

By utilizing the *smp*, knowledge transfer probability is improved among the tasks. In Figure 3, 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 1 to 5, task 3 to 7, task 7 to 3, task 2 to 6, and task 6 to 2 over 0.1 (K=10) compared to the other algorithms based on *rmp*. It shows that transfer probability is enhanced significantly.

4.3.3 *Comparison of the performance of state-of-the-art algorithms when applying DaS-SBX.*

In this section, we evaluate the performance of the DaS-SBX independently from SM-MFEA. More specifically, we replace the SBX with DaS-SBX on several well-known algorithms (LSA, MFEA, and SM-MFEA) and record their solution quality and convergence rate.

Table 5, and Figure 4 present the results of LSA, MFEA, and SM-MFEA when using DaS-SBX and SBX over 30 independent runs on the CEC'17 and GECCO20 benchmarks. In Table 5, the better, equal, and worse columns imply that DaS-SBX is better, equal, or worse than SBX on the Wilcoxon signed-rank test. The results demonstrate that the algorithms using DaS-SBX obtain better solutions than those using SBX. Additionally, to investigate the performance of DaS-SBX in many-task optimization, we conduct experiments on GECCO20 50-task benchmark dataset. The result (presented in table 5) on the Wilcoxon signed-rank test confirms the dominance of using DaS-SBX.

Table 5: The statistic values obtained by conducting Wilcoxon-signed rank test with α = 0.05 on 50-task GECCO20 benchmarks (compare between DaS-SBX and SBX). Decisions +, ≈, and − indicate DaS-SBX is significantly better, similar, and worse than SBX

Algorithms	Better	Equal	Worse	p-value	Decision
MFEA	388	0	112	0	+
LSA	315	4	181	0	+
SM-MFEA	147	212	141	0	+

Furthermore, in Figure 4, we also compare results and investigate the convergence trends of the state-of-the-art algorithm when applying DaS-SBX 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 DaS-SBX integrated algorithms obtain better results and tend to converge faster than the original ones without using DaS-SBX.

4.3.4 *Analysis of the effect of DaS-SBX on SM-MFEA.*

Figure 5 illustrates the value of *smp* among tasks with SM-MFEA using DaS-SBX crossover. It shows that crossover on some dimensions of strategy facilitates more positive knowledge transfer among pairs of tasks. Therefore, it can help SM-MFEA to escape local optima in some cases. Specifically, when applying SBX (crossover for all dimensions), task 2 can only receive knowledge transfer from task 6. On the contrary, if we employ DaS-SBX, knowledge transfer on some dimensions from task 9 and 10 are also beneficial to task 2. Hence, DaS-SBX suppresses negative transfer in some dimensions and stimulates positive transfer in others.

5 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 in terms of solution quality, convergence trend, and running time. The experiments also indicate the impressive efficiency of the DaS strategy.

In future works, we extend the DaS strategy to allow crossover of multiple parents on each dimension simultaneously, thus providing a more effective knowledge transfer mechanism for multi-population optimization problems.

REFERENCES

[1] Kavitesh Kumar Bali, Yew-Soon Ong, Abhishek Gupta, and Puay Siew Tan. 2020. Multifactorial Evolutionary Algorithm With Online Transfer Parameter Estimation: MFEA-II. *IEEE Transactions on Evolutionary Computation* 24, 1 (2020), 69–83. https://doi.org/10.1109/TEVC.2019.2906927

[2] Huynh Thi Thanh Binh, Le Van Cuong, Ta Bao Thang, and Nguyen Hoang Long. 2022. Ensemble Multifactorial Evolution with Biased Skill-Factor Inheritance for Many-task Optimization. *IEEE Transactions on Evolutionary Computation* (2022), 1–1. https://doi.org/10.1109/TEVC.2022.3227120

[3] Huynh Thi Thanh Binh, Ta Bao Thangy, Nguyen Binh Long, Ngo Viet Hoang, and Pham Dinh Thanh. 2020. Multifactorial Evolutionary Algorithm for Inter-Domain Path Computation under Domain Uniqueness Constraint. In *2020 IEEE Congress on Evolutionary Computation (CEC)*. 1–8. https://doi.org/10.1109/CEC48606.2020.9185701

[4] Rohitash Chandra, Abhishek Gupta, Yew Ong, and Chi-Keong Goh. 2016. Evolutionary Multi-task Learning for Modular Training of Feedforward Neural Networks, Vol. 9948. 37–46. https://doi.org/10.1007/978-3-319-46672-9_5

[5] Yongliang Chen, Jinghui Zhong, Liang Feng, and Jun Zhang. 2020. An Adaptive Archive-Based Evolutionary Framework for Many-Task Optimization. *IEEE Transactions on Emerging Topics in Computational Intelligence* 4, 3 (2020), 369–384. https://doi.org/10.1109/TETCI.2019.2916051

[6] Bingshui Da, Yew-Soon Ong, Liang Feng, A Kai Qin, Abhishek Gupta, Zexuan Zhu, Chuan-Kang Ting, Ke Tang, and Xin Yao. 2017. Evolutionary multitasking for single-objective continuous optimization: Benchmark problems, performance metric, and baseline results. *arXiv preprint arXiv:1706.03470* (2017).

[7] Jinliang Ding, Cuie Yang, Yaochu Jin, and Tianyou Chai. 2019. Generalized Multitasking for Evolutionary Optimization of Expensive Problems. *IEEE Transactions on Evolutionary Computation* 23, 1 (2019), 44–58. https://doi.org/10.1109/TEVC.2017.2785351

[8] K. Q. Liang Feng, Y. Y. Abhishek Gupta, Y.-S. O. Eric Scott, and X. Chi. July 2020. New mto benchmarks for gecco 2020 competition on evolutionary multi-task optimization. (July 2020). http://www.bdsc.site/websites/MTO/index.html.

[9] Abhishek Gupta, Jacek Mańdziuk, and Yew Ong. 2015. Evolutionary multitasking in bi-level optimization. *Complex Intelligent Systems* 1 (12 2015), 83–95. https://doi.org/10.1007/s40747-016-0011-y

[10] Abhishek Gupta, Yew-Soon Ong, and Liang Feng. 2016. Multifactorial Evolution: Toward Evolutionary Multitasking. *IEEE Transactions on Evolutionary Computation* 20, 3 (2016), 343–357. https://doi.org/10.1109/TEVC.2015.2458037

[11] Abhishek Gupta, Yew-Soon Ong, Liang Feng, and Kay Chen Tan. 2017. Multiobjective Multifactorial Optimization in Evolutionary Multitasking. *IEEE Transactions on Cybernetics* 47, 7 (2017), 1652–1665. https://doi.org/10.1109/TCYB.2016.2554622

[12] Abhishek Gupta, Lei Zhou, Yew Soon Ong, Zefeng Chen, and Yaqing Hou. 2021. Half a Dozen Real-World Applications of Evolutionary Multitasking, and More. *IEEE Computational Intelligence Magazine* 17 (2021), 49–66.

[13] Ramon Sagarna and Yew-Soon Ong. 2016. Concurrently searching branches in software tests generation through multitask evolution. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. 1–8. https://doi.org/10.1109/SSCI.2016.7850040

[14] Ta Bao Thang, Tran Cong Dao, Nguyen Hoang Long, and Huynh Thi Thanh Binh. 2021. Parameter adaptation in multifactorial evolutionary algorithm for many-task optimization. *Memetic Computing* 13 (12 2021), 1–14. https://doi.org/10.1007/s12293-021-00347-4

[15] Chao Wang, Jing Liu, Kai Wu, and Zhaoyang Wu. 2022. Solving Multitask Optimization Problems With Adaptive Knowledge Transfer via Anomaly Detection. *IEEE Transactions on Evolutionary Computation* 26, 2 (2022), 304–318. https://doi.org/10.1109/TEVC.2021.3068157

[16] Yu-Wei Wen and Chuan-Kang Ting. 2017. Parting ways and reallocating resources in evolutionary multitasking. In *2017 IEEE Congress on Evolutionary Computation (CEC)*. 2404–2411. https://doi.org/10.1109/CEC.2017.7969596

[17] Sheng-Hao Wu, Zhi-Hui Zhan, Kay Chen Tan, and Jun Zhang. 2023. Orthogonal Transfer for Multitask Optimization. *IEEE Transactions on Evolutionary Computation* 27, 1 (2023), 185–200. https://doi.org/10.1109/TEVC.2022.3160196

[18] Yuan Yuan, Yew-Soon Ong, Abhishek Gupta, Puay Siew Tan, and Hua Xu. 2016. Evolutionary multitasking in permutation-based combinatorial optimization problems: Realization with TSP, QAP, LOP, and JSP. In *2016 IEEE Region 10 Conference (TENCON)*. 3157–3164. https://doi.org/10.1109/TENCON.2016.7848632

[19] Xiaolong Zheng, A. K. Qin, Maoguo Gong, and Deyun Zhou. 2020. Self-Regulated Evolutionary Multitask Optimization. *IEEE Transactions on Evolutionary Computation* 24, 1 (2020), 16–28. https://doi.org/10.1109/TEVC.2019.2904696

[20] Lei Zhou, Liang Feng, Jinghui Zhong, Yew-Soon Ong, Zexuan Zhu, and Edwin Sha. 2016. Evolutionary multitasking in combinatorial search spaces: A case study in capacitated vehicle routing problem. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. 1–8. https://doi.org/10.1109/SSCI.2016.7850039