Transferable Adaptive Differential Evolution for Many-Task Optimization

Sheng-Hao Wu, *Student Member, IEEE*, Zhi-Hui Zhan, *Senior Member, IEEE*, Kay Chen Tan, *Fellow, IEEE*, and Jun Zhang, *Fellow, IEEE*

***Abstract*—The evolutionary multitask optimization (EMTO) algorithm is a promising approach to solve many-task optimization problems (MaTOPs), in which similarity measurement and knowledge transfer (KT) are two key issues. Many existing EMTO algorithms estimate the similarity of population distribution to select a set of similar tasks and then perform KT by simply mixing individuals among the selected tasks. However, these methods may be less effective when the global optima of the tasks greatly differ from each other. Therefore, this article proposes to consider a new kind of similarity, namely, *shift invariance*, between tasks. The shift invariance is defined that the two tasks are similar after linear shift transformation on both the search space and the objective space. To identify and utilize the shift invariance between tasks, a two-stage transferable adaptive differential evolution (TRADE) algorithm is proposed. In the first evolution stage, a task representation strategy is proposed to represent each task by a vector that embeds the evolution information. Then, a task grouping strategy is proposed to group the similar (i.e., shift invariant) tasks into the same group while the dissimilar tasks into different groups. In the second evolution stage, a novel successful evolution experience transfer method is proposed to adaptively utilize the suitable parameters by transferring successful parameters among similar tasks within the same group. Comprehensive experiments are carried out on two representative MaTOP benchmarks with a total of 16 instances and a real-world application. The comparative results show that the proposed TRADE is superior to some state-of-the-art EMTO algorithms and single-task optimization algorithms.**

Manuscript received 21 June 2022; revised 23 August 2022 and 1 November 2022; accepted 30 December 2022. Date of publication 8 February 2023; date of current version 17 October 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62176094; in part by the Guangdong Natural Science Foundation Research Team under Grant 2018B030312003; and in part by the National Research Foundation of Korea (NRF) under Grant 2022H1D3A2A01093478. This article was recommended by Associate Editor L. Celentano. *(Corresponding authors: Zhi-Hui Zhan; Jun Zhang.)*

Sheng-Hao Wu and Zhi-Hui Zhan are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China, (e-mail: zhanapollo@163.com).

Kay Chen Tan is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong SAR (e-mail: kctan@polyu.edu.hk).

Jun Zhang is with Zhejiang Normal University, Jinhua 321004, China, and also with Hanyang University, Ansan 15588, South Korea.

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TCYB.2023.3234969.

Digital Object Identifier 10.1109/TCYB.2023.3234969

***Index Terms*—Adaptive, differential evolution (DE), evolutionary computation (EC), evolutionary multitasking optimization, knowledge transfer (KT), many-task optimization problem (MaTOP), shift invariance, similarity measurement.**

I. Introduction

RECENTLY, an emerging research topic, called evolutionary transfer optimization [1], [2], that combines transfer learning [3], [4], [5] and evolutionary computation (EC) [6], [7] (including evolutionary algorithms [8], [9] and swarm intelligence [10], [11], [12]) has become attractive. In the traditional EC paradigm, the optimization tasks are solved one by one without considering the relatedness and similarity among different tasks [13], [14]. However, it is observed that optimization tasks seldom exist independently in practice. For example, some distinct tasks may have similarities in the function landscape or problem structure. Therefore, it is motivated that evolutionary transfer optimization can use the optimization experience or the domain knowledge in solving some tasks (i.e., source tasks) to improve the search performance on other similar tasks (i.e., target tasks). The technique that reuses information from source tasks to help solve target tasks is called knowledge transfer (KT) [15], [16].

Among different types of optimization problems, the multitask optimization problem is one of the most representative problems related to evolutionary transfer optimization. In the multitask optimization problem, multiple optimization tasks are required to be solved simultaneously with the assumption that there are similarities between the tasks to some extent. Therefore, a new paradigm called evolutionary multitask optimization (EMTO) has emerged to solve multitask optimization problems [17]. Different from traditional EC algorithms, an EMTO algorithm not only contains an EC algorithm as the base solver but also contains a KT method. Some successful designs of KT methods have helped EMTO algorithms to achieve superior performance in solving multitask optimization problems [18], [19]. Moreover, these EMTO algorithms have also shown effectiveness and efficiency in solving real-world application problems [20], [21].

Based on the success of the EMTO algorithm, researchers begin to consider a kind of more complex optimization problem with many tasks (i.e., more than three tasks) to be solved, which is called the many-task optimization problem (MaTOP). The MaTOP is a challenging problem because the many tasks may contain some unrelated or misleading tasks. If unrelated tasks are mistakenly selected to perform KT between tasks, the performance of EMTO algorithms may deteriorate.

Like the multitask optimization problem [22], [23], two important issues need to be considered in solving MaTOP.

Issue 1: How to measure the similarity between the source tasks and the target task. There are multiple available source tasks in MaTOP that can be used to transfer knowledge to the target task. This is more challenging than MTOP which has only one or two source tasks. However, the similarities between these many source tasks and the target task are not known in advance. Then, how to measure the similarities between the source tasks and the target task to help find out which is the most suitable source task for transferring knowledge is an important issue.

Issue 2: How to transfer knowledge from source tasks to help the search process on the target task. Given that the source task and the target task are similar according to a similarity measurement, how to perform effective KT between the two tasks is still important. If the KT between tasks is not properly designed, the KT may lead to a negative effect called *negative transfer*. The negative transfer indicates that the interaction between tasks deteriorates the optimization performance on the target task compared to the independent search process.

To address these two issues, some EMTO algorithms have been proposed in the literature to solve MaTOP [24], [25]. Moreover, these EMTO algorithms have been successfully applied to solve real-world MaTOP, such as diversified robotic morphology designs [26]; fuzzy cognitive maps [27]; and robotic arm control [28], [29]. However, there are two remaining challenges and research gaps in solving MaTOP.

First, although some similarity measurements have been proposed in [30] and [31], they mainly consider the similarity/dissimilarity between the population distributions in a common search space of the source task and the target task. That is, if the two populations are closely distributed in the unified search space, the two tasks are regarded as similar. We denote this kind of population distribution-based similarity measurement (PDSM). If the global optimal solutions of the two tasks are highly similar (e.g., intersections in the global optimum), PDSM works well since the elite individuals of one task can be easily reused to improve the fitness of other tasks. However, in real-world problems, the global optimal solutions of different tasks may differ greatly from each other. Moreover, PDSM relies on the population distribution at the current generation. However, due to the randomness of the evolution process, the similarity measurement based on population distribution estimation at the only current generation may be unstable and unreliable. Furthermore, due to the limited size of the sample (i.e., the population size), there may exist an estimation error in the population distribution. Therefore, PDSM may not always be useful.

Second, although existing EMTO frameworks mainly use multiple populations for solving MaTOP and allow multiple populations to use different EC algorithms, many researchers adopt the EC algorithm with fixed parameter settings as the base solver for all populations. Differential evolution (DE) is a well-known EC algorithm which has many improved variants [32], [33] and applications in complex problems, such as large-scale [34], [35]; multimodal [36], [37]; many-objective [38]; and real-world problems [39], [40]. Hence, we use DE as the base solver in this article. However, previous studies [41], [42] have shown that the parameters (e.g., F and Cr) of the DE algorithm are sensitive to the problem to be solved. For example, when using DE as the base solver, different problems may require different parameter settings of F and Cr to achieve the best optimization results. However, it is unknown in advance what kind of problem we are facing and, therefore, it is difficult to set up suitable parameters. Fortunately, when dealing with MaTOP, there are some similarities between the tasks. Therefore, it is motivated that we can set up different parameters for different populations to solve different tasks, and then we can observe the performance of different parameters so as to transfer the well-performing parameters as successful evolution experiences (i.e., the parameter setting of F and Cr) between the similar tasks. This way, the parameters of DE can be adaptively adjusted to distinguish better F and Cr. To the best of our knowledge, although some efforts have been proposed in the literature to transfer solution knowledge or meta-knowledge among the tasks [18], [19], no research efforts have been paid to study the transfer of successful evolution experience (e.g., EC algorithm parameters) among the tasks to solve MaTOP.

To address the above challenges and fill the research gap, we propose a two-stage transferable adaptive DE (TRADE) to solve MaTOP effectively and efficiently. The main contributions of this article are as follows.

First, different from PDSM which merely considers the similarity in population distributions, we propose to consider a new kind of similarity between tasks, namely, *shift invariance*. The shift invariance means that the two tasks are similar after linear shift transformation on both the search space and the objective space. The proposed similarity measurement is called shift invariance-based similarity measurement (SISM). To the best of our knowledge, no studies have been taken to study utilizing the shift invariance between tasks to effectively solve MaTOP.

Second, to identify and capture the shift invariance between tasks, we propose a novel task representation strategy (TRS) together with a task grouping strategy (TGS) which are carried out in the *first evolution stage*. Specifically, TRADE first uses the same EC algorithm in all the populations to correspondingly solve all tasks for a few generations to collect evolution information for representing the tasks. That is, the populations of all the tasks use the same EC algorithm but evolve independently without any KT. Afterward, the TRS uses the obtained evolution information to represent each task as a feature vector. Then, the TGS divides the tasks into multiple groups based on the task feature vectors, where the tasks within the same group are regarded to be similar (i.e., shift invariant) in the function landscapes.

Third, to effectively reuse the knowledge from similar tasks within the same group to improve the search efficiency, we propose a novel KT method, called successful evolution experience transfer (SEET), in the *second evolution stage*. Specifically, the populations of the tasks in the second evolution stage of TRADE use EC algorithms with different parameters and are evolved with the SEET method to transfer knowledge of successful evolution experience (i.e., successful parameters). To perform the SEET method, we first propose an evolution quality analysis strategy to distinguish which populations of the tasks evolve well or poorly within each group. Then, the successful parameter settings of the well-evolved populations identified by evolution quality analysis are regarded as knowledge of successful evolution experience and are transferred to the poorly evolved populations to produce promising offspring within each group.

The remainder of this article is organized as follows. Section II gives the introduction of the related work on MaTOP, DE, and the motivation of this article. Section III introduces the definition of the SISM and the details of the proposed TRADE algorithm. Section IV carries out experimental studies to show the effectiveness of the proposed TRADE. The conclusion is given in Section V.

II. Preliminary

The notations with their descriptions used in this article are given in Table S.I in the supplemental material.

A. Many-Task Optimization Problem

1) Problem Formulation: Suppose there are NT single-objective optimization tasks in a MaTOP and the task k ($k = 1, \ldots, NT$) denoted as T_k can be formulated as

$$\begin{aligned} \min\ & y_k = f_k(x_k) \\ \text{s.t.}\ & x_k \in \mathcal{X}_k, \mathcal{X}_k \subseteq \mathbb{R}^{D_k} \end{aligned} \tag{1}$$

where $f_k(\cdot)$ is the objective function of T_k, $\mathcal{X}_k$ is the search space of T_k, and D_k is the dimensionality of the search space.

The MaTOP is the extension of the multitask optimization problem that contains more than three tasks ($NT > 3$) to be solved. The output of an EMTO algorithm for solving MaTOP contains NT optimized solutions denoted as $\{x_1^*, \ldots, x_{NT}^*\}$ for all the NT tasks. In this article, we consider the optimization tasks in the continuous search space bounded by a box constraint. The lower bound and upper bound of dimension d ($d = 1, \ldots, D_k$) of T_k are denoted as $LB_{k,d}$ and $UB_{k,d}$. Since the search spaces of these tasks may be different, the solutions (i.e., x_k of T_k) are encoded into a *unified search space* $\mathcal{U} \subseteq [0, 1]^{D_U}$ where $D_U = \max\{D_1, \ldots, D_{NT}\}$. The encoded solution of T_k is denoted as u_k and calculated as

$$u_{k,d} = \left(x_{k,d} - LB_{k,d}\right)/\left(UB_{k,d} - LB_{k,d}\right) \tag{2}$$

where $u_{k,d}$ is the dth dimension of u_k. Conversely, when a solution u_k on T_k is to be evaluated, it should be decoded to obtain the solution x_k in the original search space $\mathcal{X}_k$. If $D_k < D_U$, only the first D_k dimensions of u_k are decoded to obtain the solution x_k.

2) Multipopulation Framework for MaTOP: Although there are EMTO algorithms using a single population, most of the EMTO algorithms for MaTOP in the literature are generally implemented in a multipopulation framework [25], [30]. The basic multipopulation framework for MaTOP (MP4MaT) in the literature is summarized and shown in Algorithm 1. The input of the MP4MaT framework contains a task set $\mathcal{T}$ containing tasks to be solved, a base solver set $\mathcal{A}$ containing EC algorithms (EAs) with different parameter settings, and the maximum number of fitness evaluations (MAXNFEs) for optimization of all the tasks. The number of the base solvers in $\mathcal{A}$ is denoted as NS. The EMTO algorithm based on MP4MaT first initializes a population for each task in the unified search space $\mathcal{U} \subseteq [0, 1]^{D_U}$ and assigns a base solver from the base solver set $\mathcal{A} = \{\text{EA}_i\}_{i=1}^{NS}$ for each task (lines 3–6). The assigned index for T_k is denoted as $\tau_k \in \{1, \ldots, NS\}$. Then, the evolution process of every task begins. In a generation, MP4MaT should decide whether to perform KT on each task. If T_k ($k = 1, \ldots, NT$) is going to perform KT, it first selects source tasks for KT based on a similarity measurement (lines 9 and 10). Then, the offspring of T_k are produced by a KT method based on the selected source tasks (line 11). Otherwise, the offspring of T_k is produced using the evolution operators of the assigned EA_{τ_k} (line 13). Afterward, the population for each task undergoes the fitness evaluation and selection process (lines 15 and 16). The stopping condition is when the number of fitness evaluations (NFEs) reaches $MAXNFE$.

Algorithm 1 MP4MaT Framework

```
Input:  $\mathcal{T} = \{T_k\}_{k=1}^{NT}$ : Task set;
        $\mathcal{A} = \{EA_i\}_{i=1}^{NS}$ : Base solver set;
        $MAXNFE$: Maximum number of fitness evaluations;
Output: $X^* = \{x_k^*\}_{k=1}^{NT}$: The best solutions for each task;
1:  Begin
2:  $NFE = 0$; //Number of fitness evaluations
3:  For $k = 1$ to $NT$ Do
4:      Initialize population $POP_k$ for task $k$ in $\mathcal{U} \subseteq [0, 1]^{D_U}$;
5:      Assign a base solver $EA_{\tau_k}(\tau_k \in \{1, \ldots, NS\})$ for task $k$;
6:  End For
7:  While $NFE < MAXNFE$ Do
8:      For $k = 1$ to $NT$ Do
9:          If target task $k$ will perform KT
10:             Select source tasks for KT based on a similarity measurement;
11:             Produce offspring by KT method from source tasks;
12:         Else
13:             Produce offspring by $EA_{\tau_k}$;
14:         End If
15:         Evaluate $POP_k$ and update $NFE$ and $x_k^*$;
16:         Undergo selection process and update $POP_k$;
17:     End For
18: End While
19: End
```

The MP4MaT framework contains two main components which are the task similarity measurement and the KT method for transferring knowledge between tasks. The major difference between different EMTO algorithms in the literature lies in the design of the task similarity measurement and the KT method.

B. Motivation and Contribution

It should be noted that although the MP4MaT framework allows multiple populations use different base solvers, most of the existing works use a unified base solver for all the tasks in MaTOP. That is, the base solver set $\mathcal{A} = \{\text{EA}_i\}_{i=1}^{NS}$ ($NS = 1$) only contains one EA associated with its fixed parameter settings. However, many studies show that the optimization performance of EA is highly related to its parameter settings and the suitable parameter setting is highly related to the problem being solved [41], [42]. Since the problem to be solved is a black-box optimization problem and there is

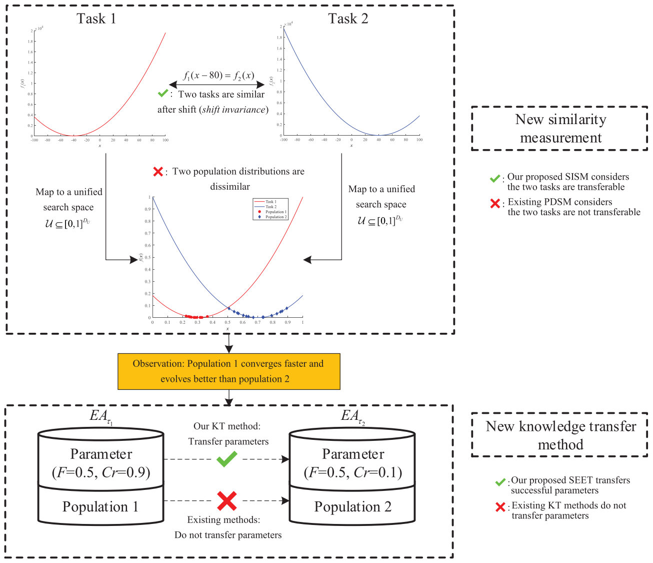

Task 1
Task 2
$f_1(x-80)=f_2(x)$
✓: Two tasks are similar after shift (*shift invariance*)
✗: Two population distributions are dissimilar
Map to a unified search space $\mathcal{U}\subseteq[0,1]^{D_U}$
Map to a unified search space $\mathcal{U}\subseteq[0,1]^{D_U}$
Task 1
Task 2
Population 1
Population 2
New similarity measurement
✓: Our proposed SISM considers the two tasks are transferable
✗: Existing PDSM considers the two tasks are not transferable
Observation: Population 1 converges faster and evolves better than population 2
EA_{τ_1}
EA_{τ_2}
Parameter (F=0.5, Cr=0.9)
Population 1
Our KT method: Transfer parameters
Existing methods: Do not transfer parameters
Parameter (F=0.5, Cr=0.1)
Population 2
New knowledge transfer method
✓: Our proposed SEET transfers successful parameters
✗: Existing KT methods do not transfer parameters

Fig. 1. Motivated example for SISM and SEET.

no prior knowledge, finding the suitable parameter setting of EA for solving the problem may require heavy computational effort on parameter tuning. Then, it is motivated that when multiple similar tasks are solved together, the parameter settings of EA that work well on the source tasks can be reused to improve the search efficiency of the target task.

A motivating example of the proposed methods is shown in Fig. 1. Suppose that there are two optimization tasks (i.e., T_1 and T_2) to be solved. Following the procedure of MP4MaT framework, the search spaces of the two tasks are first mapped to a unified search space $\mathcal{U} \subseteq [0, 1]^{D_U}$ where $D_U = \max\{D_1, D_2\}$ to allow KT. The optimal solutions (i.e., $x = 0.3$ and $x = 0.7$) of the two tasks are far from each other in the unified search space. It is assumed that the two tasks are shift invariant. That is, the objective function $f_1(x)$ of T_1 after linear transformation, denoted as $f_1(x–80)$, is highly similar to the objective function $f_2(x)$ of T_2. For a detailed definition of shift invariance, refer to Section III-A. The base solver EA_{τ_1} for T_1 is DE with parameter settings of $F = 0.5$ and $Cr = 0.9$ while the base solver EA_{τ_2} for T_2 is DE with parameter settings of $F = 0.5$ and $Cr = 0.1$. The population distributions of the two tasks at generation g are also plotted in Fig. 1. It can be observed that the distance between population 1 and the optimal solution ($x = 0.3$) of T_1 is smaller than the distance between population 2 and the optimal solution ($x = 0.7$) of T_2 in the unified search space. Since the two tasks are regarded to be very similar (i.e., shift invariant), it is rational to think that population 1 which uses EA_{τ_1} with parameter settings of $F = 0.5$ and $Cr = 0.9$ is more successful and can solve these two tasks better than that uses EA_{τ_2} with parameter settings of $F = 0.5$ and $Cr = 0.1$. Therefore, in this article, we consider transferring these successful evolution parameters as the successful evolution experience between tasks to facilitate a more efficient search for MaTOP. To the best of our knowledge, no research attention has been paid to using different parameter settings of EA as base solvers in MP4MaT framework and making use of the shift invariance-based similarity between tasks by transferring parameter settings among different tasks.

Moreover, we highlight the contribution of our proposed KT methods by the example in Fig. 1. In the case of Fig. 1, the existing PDSM considers that the two tasks are dissimilar since the population distributions of T_1 and T_2 are different. However, these two tasks are actually similar by simple shift transformation, which can be properly identified by our proposed SISM method.

C. Differential Evolution

A DE algorithm mainly contains four processes: 1) initialization; 2) mutation; 3) crossover; and 4) selection. In the initialization process, the dth dimension of the ith individual, denoted as $x_{i,d}$, ($i = 1, \ldots, NP$) where NP is the population size is initialized as

$$x_{i,d} = LB_d + (UB_d - LB_d) \cdot \text{rand} \tag{3}$$

where rand denotes a randomly sampled number within [0, 1], and LB_d and UB_d are the lower bound and upper bound of the dth dimension. After initialization, the iteration of DE begins. In a generation, the population first undergoes the mutation process. In the mutation process, each individual generates a mutant vector. An advanced mutation operator called DE/current-to-$p$best/1 in [43] is

$$\vec{v}_i = \vec{x}_i + F \cdot \left(\vec{x}_{\text{pbest}} - \vec{x}_i\right) + F \cdot \left(\vec{x}_{r_1} - \widehat{x}_{r_2}\right) \tag{4}$$

where $\vec{x}_{\text{pbest}}$ is a randomly selected individual from the top $p\%$ individuals in the population. F is the scaling factor of the difference vector, r_1 is an index randomly selected from $\{1, \ldots, NP\}$, and $\widehat{x}_{r2}$ is a randomly selected individual from the union of the population $POP = \{\vec{x}_1, \ldots, \vec{x}_{NP}\}$ and an *arc*hive ARC storing historical solutions until generation g. The archive ARC is empty in the initialization process of DE. Without loss of generality, we only consider the optimization problem with simple box constraints in this article and, therefore, the generated vectors are clipped by LB_d and UB_d to satisfy the box constraints.

After mutant vectors are generated, the crossover operator is carried out on each mutant vector to produce a trial vector. We introduce the binomial crossover here. The dth dimension of the ith trial vector at generation g, denoted as $u_{i,d}$, is set as

$$u_{i,d} = \begin{cases} v_{i,d}, & \text{if rand } < Cr \text{ or } d == d_r \\ x_{i,d}, & \text{otherwise} \end{cases} \tag{5}$$

where Cr is a crossover parameter for the ith individual and d_r is a randomly selected dimension before the crossover process on the ith mutant vector.

After trial vectors are generated, the trial vector is evaluated and undergoes the selection process to update the population POP and the archive ARC for a minimization problem as

$$\vec{x}_i = \begin{cases} \vec{u}_i, \text{ if } f\left(\vec{u}_i\right) < f\left(\vec{x}_i\right) \\ \vec{x}_i, \text{ otherwise} \end{cases} \tag{6}$$

$$ARC = \begin{cases} ARC \cup \left\{\vec{x}_i\right\}, & \text{if } f\left(\vec{u}_i\right) < f\left(\vec{x}_i\right) \\ ARC, & \text{otherwise.} \end{cases} \tag{7}$$

If the size of ARC exceeds NP, then ($|ARC|–NP$) individuals will be randomly selected and removed from ARC.

Similar to [43], we generate F and Cr for the ith individual from a Gaussian distribution with parameters mF and mCr as

$$F = \text{Gaussian}(mF, 0.1) \tag{8}$$

$$Cr = \text{Gaussian}(mCr, 0.1) \tag{9}$$

In this article, we use different DEs with different settings of mF and mCr as the base solvers for different tasks. In this way, we can set mF and mCr with different values to study their effects for solving different tasks.

III. Proposed TRADE Algorithm

A. Shift Invariance-Based Similarity Measurement

The concept of shift invariance originates from the pattern recognition task in the computer vision area [44]. Let an image with spatial resolution $H \times W$ and C channels be represented by $X \in \mathbb{R}^{H\times W\times C}$. According to [44], the shift invariance is represented as

$$\widetilde{\mathcal{F}}(X) = \widetilde{\mathcal{F}}\big(\text{Shift}_{\Delta h,\Delta w}(X)\big) \quad \forall(\Delta h, \Delta w) \tag{10}$$

where $\widetilde{\mathcal{F}}(X)$ is the output function for the input X, Δh and Δw are the shift transformation of the pixel in vertical and horizontal directions in the image, and the shift operation is defined as

$$\big[\text{Shift}_{\Delta h,\Delta w}(X)\big]_{h,w,c} = X_{(h-\Delta h)\%H,(w-\Delta w)\%W,c} \tag{11}$$

where % is the modulus operator. This shift transformation is called circular shift when the pixels after the shift hit the edge of the image, they are rolled to the other edge. Similarly, we can define the shift invariance-based similarity between optimization tasks in MaTOP in the following.

Definition 1: Two optimization tasks (i.e., T_1 and T_2) with the same dimensionality D of the search space are said to be shift invariant if there exists a linear transformation with parameters $\Delta x \in \mathbb{R}^D$ and $\Delta y \in \mathbb{R}$, a connected region $\mathcal{X}^D \subseteq \mathbb{R}^D$, and a small positive value ε such that

$$\frac{1}{V}\int_{x\in\mathcal{X}^D} \left\|\text{Shift}_{\Delta x,\Delta y}(f_1(x)) - f_2(x)\right\| dx < \varepsilon \tag{12}$$

where $V > 0$ is the hypervolume of $\mathcal{X}^D$ and the shift operation on a fitness function is defined as

$$\text{Shift}_{\Delta x,\Delta y}(f(x)) = f(x+\Delta x) + \Delta y \tag{13}$$

where Δx is a D-dimensional vector representing the shift in the search space and Δy is a scalar representing the shift in the objective space. The major difference between SISM and PDSM is that SISM aims to capture the landscape similarity between two tasks while PDSM aims to capture the global optimum similarity. Therefore, the SISM is more general to identify the similarity of the tasks, as the example illustrated in Fig. 1.

B. Framework of TRADE

The framework of TRADE for solving MaTOP is plotted in Fig. 2. The framework contains two evolution stages.

In the first evolution stage, the initialization process is carried out for each task. Specifically, a population of size NP is created for each task with uniform random sampling in the

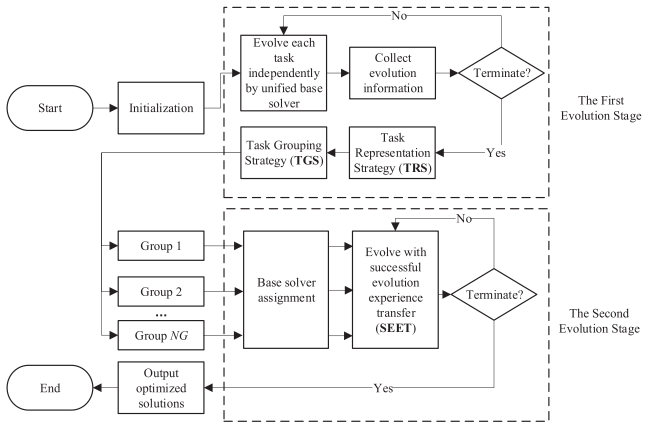

Start
Initialization
Evolve each task independently by unified base solver
Collect evolution information
Terminate?
No
Yes
Task Grouping Strategy (TGS)
Task Representation Strategy (TRS)
The First Evolution Stage
Group 1
Group 2
...
Group NG
Base solver assignment
Evolve with successful evolution experience transfer (SEET)
Terminate?
No
Yes
The Second Evolution Stage
End
Output optimized solutions

Fig. 2. Framework of two-stage TRADE.

original search space according to (3). Moreover, a unified EA (EA$_u$) associated with its predefined parameter setting is assigned for all the tasks. After initialization, all the populations of the tasks are evolved independently by the same evolution operators of EA$_u$ for a small number of generations (G_1) in the first evolution stage. Note that in this stage, the KT process is not performed. Through the evolution process in the first evolution stage, the information of the tasks is collected. After the stopping condition of the evolution process of the first evolution stage is satisfied, the TRS is carried out to represent each task as a vector according to the evolution information extracted from the search history. Then, the TGS can divide the tasks into NG groups based on the representations extracted by TRS. After TGS, the tasks that are divided into the same group are considered shift invariant. The goal of the first evolution stage is to identify and capture the shift invariance between similar tasks based on SISM for the MaTOP. The details of the first evolution stage including TRS and TGS of TRADE will be introduced in Section III-C.

In the second evolution stage, a solver assignment process is carried out within each group. Specifically, each task (i.e., population) is assigned with a base solver associated with its parameter settings that is randomly selected from the base solver set $\mathcal{A} = \{\text{EA}_i\}_{i=1}^{NS}$. Then, each population optimizes its corresponding task by using this solver. To perform SEET, the evolution quality analysis is used to identify the populations that evolve better due to using suitable parameter settings. These suitable parameter settings are regarded as knowledge of successful evolution experience. Then, the successful parameters settings (i.e., the knowledge) are transferred from well-evolved populations to poorly evolved populations to produce promising offspring. With the SEET method, the populations of the tasks can propagate the suitable parameter settings of EA for solving similar tasks within a group. Note that an important reason that the SEET method can work well is that the tasks within the group are regarded to be similar after TGS in the first evolution stage. The details of the second evolution stage including the SEET method are introduced in Section III-D.

C. First Evolution Stage With TRS and TGS

1) TRS: We propose a simple yet efficient TRS for the tasks without introducing much computational cost to identify

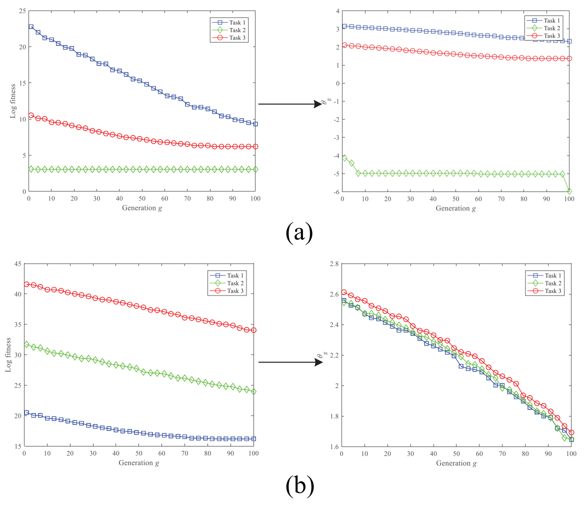


Fig. 3. Examples of TRS. (a) Three tasks are with heterogeneous function landscapes. (b) Three tasks are with shift-invariant function landscapes.

Algorithm 2 First Evolution Stage

```
Input:  $\mathcal{T} = \{T_k\}_{k=1}^{NT}$ : Task set;
        $EA_u$ : Unified base solver;
        $G_1$ : Maximum evolution generation for stage one;
Output: $gID$: Group indices of all the tasks;
        $NFE$: Number of fitness evaluations;
        $g$ : Current generation after stage one;
 1: Begin
 2: $NFE = 0$;
 3: For $k = 1$ to $NT$ Do
 4:     Evolve the population $POP_k$ using $EA_u$ for $G_1$ generations, collect
        $Y_k^* = \left\{y_{k,1}^*, \ldots, y_{k,G_1}^*\right\}$, and update $NFE$;
 5: End For
 6: Obtain the task representation $\{\theta_1, \ldots, \theta_{NT}\}$ by Eq. (14) and Eq. (15);
 7: $NG = \mathbf{CRP}(\{\theta_1, \ldots, \theta_{NT}\}, \alpha, \rho)$;
 8: $gID = \mathbf{kmeans}(\{\theta_1, \ldots, \theta_{NT}\}, NG)$;
 9: $g = G_1$;
10: End
```

and capture the shift invariance between the tasks. Suppose the evolution process for the first evolution stage lasts for G_1 generations. Then, T_k is represented as a G_1-dimensional vector $\theta_k \in \mathbb{R}^{G_1}$. The best-so-far fitness of each task in every generation is collected as the evolution information. Let $Y_k^* = \{y_{k,1}^*, \ldots, y_{k,G_1}^*\}$ be a set containing the best-so-far fitness in every generation for T_k in the first evolution stage. The lower bound of Y_k^* is calculated as

$$LBY_k = \text{mean}\left(Y_k^*\right) - 4 \cdot \text{std}\left(Y_k^*\right) \tag{14}$$

where mean$(\cdot)$ and std$(\cdot)$ denote functions for calculating mean and standard deviation for a set of scalars, respectively. Then, the dimension g ($g = 1, \ldots, G_1$) of vector θ_k can be calculated as

$$\theta_{kg} = \log\left(\max\left\{\left(y_{k,g}^* - LBY_k\right), \eta\right\}\right) \tag{15}$$

where log$(\cdot)$ is the logarithm function and $\eta = 1E-25$ is a small positive value to avoid an invalid logarithmic operation.

To illustrate how the TRS can identify and capture the shift invariance between tasks in MaTOP, an example is given in Fig. 3. In Fig. 3(a) and (b), the curves of logarithm of fitness versus generations on three tasks are plotted on the left side while the representations of three tasks by TRS are plotted on the right side. Note that all tasks are evolved by the same base solver EA$_u$. In Fig. 3(a), since the three tasks are with heterogeneous functions, task representations after TRS (i.e., the curves in the right figure) are rather different. In Fig. 3(b), the three tasks are shift invariant. That is, there exists a shift transformation on the search space and the objective space such that the transformed functions are similar to each other. Therefore, in this case, the task representations after TRS are highly similar which are shown in the right of Fig. 3(b). This experimental example shows that the TRS can well identify and capture the shift invariance between tasks in MaTOP.

2) TGS: Given the task representation, the TGS is carried out to divide the tasks so that similar tasks belong to the same group while the dissimilar tasks belong to different groups based on SISM. Herein, we use the simple Euclidean distance between θ_k of different tasks to measure the shift invariance-based similarity. In this article, the classical K-means clustering algorithm [45] is used to group the tasks after the first evolution stage. Note that the parameter K representing the number of clusters in K-means algorithm is a sensitive parameter that affects the quality of the grouping process. Specifically, if the number of groups NG (i.e., parameter K in K-means algorithm) is not set properly, some dissimilar tasks may be grouped into the same group, and transferring knowledge between them may lead to negative transfer. Therefore, we adopt the Chinese restaurant process to determine NG automatically [46].

The complete procedure of the first evolution stage in TRADE is shown in Algorithm 2. The **CRP**$(\cdot)$ in line 7 is an implemented function of the Chinese restaurant process with input parameters α and ρ. The default setting of these parameters will be introduced in Section IV-A. The **kmeans**$(\cdot)$ in line 8 is the K-means clustering algorithm [45]. The output of the first evolution stage includes the group indices for all the tasks (gID) and the NFEs. The unified base solver EA$_u$ in this article is the DE algorithm with parameter settings of $mF = 0.5$ and $mCr = 0.5$.

3) Time Complexity Analysis: The computation in the first evolution stage mainly comes from TRS and TGS. For TRS, since each representation vector of the task is G_1-dimensional and there are NT tasks, it has a complexity of $O(NT\times\ G_1)$. The time complexity of **CRP**$(\cdot)$ is $O(NT^2 \times G_1)$ and the time complexity of **kmeans**$(\cdot)$ algorithm performed on NT tasks vectors is $O(T \times NG \times NT \times G_1)$ where T denotes the number of iterations and be considered as a constant. NG is the returned result of **CRP**$(\cdot)$ which is normally much smaller than NT and can be considered a constant. That is, the major time complexity in the first evolution stage is $O(NT^2 \times G_1)$.

Take a representative PDSM method that uses Kullback–Leibler divergence as an example [25], the time complexity is $O(MAXGEN\times\ NT^2 \times D_U)$. Note that we have $G_1 < MAXGEN$. That is, the complexity of TRS and TGS is not dependent on D_U compared to existing PDSM methods that calculate the similarity between populations of every pair of tasks in every generation. Moreover, G_1 can be specified manually to balance accuracy and computational cost. The parameter investigation

Algorithm 3 Second Evolution Stage

```
Input:  $\mathcal{T} = \{T_k\}_{k=1}^{NT}$: Task set;
        $\mathcal{A} = \{EA_i\}_{i=1}^{NS}$: Base solver set;
        $\{POP_k\}_{k=1}^{NT}$: Evolved populations after stage one.
        $\{ARC_k\}_{k=1}^{NT}$: Evolved archives after stage one.
        $Y_k^* = \left\{y_{k,1}^*, \ldots, y_{k,G_1}^*\right\}$: The set containing best-so-far fitness at
            every generation after stage one.
        $gID = \{gID_k\}_{k=1}^{NT}$: Group indices of all the tasks;
        $NFE$: Number of fitness evaluations after stage one;
        $MAXNFE$: Maximum number of fitness evaluations;
        $g$: Current generation after stage one;
Output: $X^* = \left\{x_k^*\right\}_{k=1}^{NT}$: The best solutions for each task;
1:  Begin
2:  Assign each $T_k$ an $EA_{\tau_k}$ along with its $mF$ and $mCr$ that is randomly
    selected from $\mathcal{A} = \{EA_i\}_{i=1}^{NS}$;
3:  $count$ = zeros($NT$, $NS$);   // counter of the used times of different EA
                                     for each task
4:  $countSuc$ = zeros($NT$, $NS$); // counter of the successful times of
                                     different EA for each task
5:  While $NFE < MAXNFE$
6:      Perfom EQA and calculate $EQ_k$ according to Eq. (16);
7:      Sort the tasks in each group based on $EQ_k$ and obtain their rank;
8:      Calculate $p_{\text{SEET}}$ according to Eq. (17);
9:      For $k = 1$ to $NT$ Do
10:         For $i = 1$ to $NP$ Do
11:             If rand$_1 < p_{\text{SEET}}$ and rand$_2 > 1/\text{rank}(T_k)$ and $gSize(gID_k) >$
                1// SEET
12:                 Randomly select a source task $T_s$ from the tasks whose ranks
                    are smaller than $gSize(gID_k)/NS$;
13:                 Select an EA index $\tau_s$ according to Eq. (18);
14:                 $count(k, \tau_s) = count(k, \tau_s) + 1$;
15:                 Set $mF$ and $mCr$ as the parameters of $EA_{\tau_s}$;
16:                 Sample $F$ and $Cr$ with $mF$ and $mCr$ by Eq. (8) and Eq. (9);
17:             Else // Use base solver $EA_{\tau_k}$
18:                 Set $mF$ and $mCr$ as the parameters of $EA_{\tau_k}$;
19:                 $count(k, \tau_k) = count(k, \tau_k) + 1$;
20:                 Sample $F$ and $Cr$ with $mF$ and $mCr$ by Eq. (8) and Eq. (9);
21:             End If
22:             Undergo mutation and crossover according to Eq. (4) and Eq. (5)
                to obtain an offspring individual;
23:             Evaluate individual and update $NFE$, $x_k^*$, and $y_{k,g}^*$;
24:             Update population $POP_k$ by selection and archive $ARC_k$ accord-
                ing to Eq. (6) and Eq. (7);
25:             If parent is replaced by offspring produced by $EA_{\tau_s}$
26:                 $countSuc(k, \tau_s) = countSuc(k, \tau_s) + 1$;
27:             Else If parent is replaced by offspring produced by $EA_{\tau_k}$
28:                 $countSuc(k, \tau_k) = countSuc(k, \tau_k) + 1$;
29:             End If
30:         End For
31:         $Y_k^* = Y_k^* \cup y_{k,g}^*$;
32:     End For
33:     $g = g + 1$;
34: End While
35: End
```

of G_1 is carried out in Section IV-D. Therefore, the proposed TRS and TGS are more scalable.

D. Second Evolution Stage With SEET

The complete procedure of the second evolution stage in TRADE is shown in Algorithm 3. The input mainly includes the evolved populations $\{POP_k\}_{k=1}^{NT}$, evolved archives $\{ARC_k\}_{k=1}^{NT}$, the best-so-far fitness of all the tasks, and the group indices (gID) of all the tasks after the first evolution stage. Note that all base solvers in the base solver set $\mathcal{A} = \{\text{EA}_i\}_{i=1}^{NS}$ are implemented as DEs with different parameter settings of mF and mCr. The main components in the second evolution stage include solver assignment, the evolution quality analysis, and the offspring production process.

For the solver assignment process, we simply assign each task an EA with its mF and mCr that is randomly selected from the base solver set $\mathcal{A} = \{\text{EA}_i\}_{i=1}^{NS}$ at the beginning of the second evolution stage (line 2).

For the evolution quality analysis process, we evaluate the evolution quality of each population at the beginning of a generation (line 6). The evolution quality reflects how well an EA with its parameter settings performs on a task. Suppose that the current generation is $g > G_1$ after the first evolution stage and let $Y_k^* = \{y_{k,1}^*, \ldots, y_{k,G_1}^*, \ldots, y_{k,g}^*\}$ be a set containing the best-so-far fitness for T_k from generation 1 to g. Then, evolution quality EQ_k of T_k in solving a minimization problem is calculated as

$$EQ_k = \left(y_{k,G_1}^* - y_{k,g}^*\right) / \left(y_{k,1}^* - y_{k,g}^* + \eta\right) \tag{16}$$

where $\eta = 1E-25$ is a very small value to avoid zero. Since different tasks may use different base solvers in the second evolution stage (i.e., $g > G_1$), a larger EQ_k indicates that EA_{τ_k} with its successful mF and mCr achieves larger improvement on the global best fitness of T_k. That is, if $EQ_1 > EQ_2$, it is considered that EA_{τ_1} optimizes its task (i.e., T_1) more successfully than EA_{τ_2} (i.e., solving T_2). After calculating EQ_k for all the tasks, we sort the tasks in each group according to their EQ_k in descending order (line 7). Then, the rank denoted as rank(T_k) of each task T_k based on EQ_k is obtained. A smaller rank(T_k) indicates T_k is with a larger EQ_k and EA_{τ_k} is more successful.

For the offspring production process, the offspring of a task can be produced by either the base solver EA or the proposed SEET method. The occurrence of the SEET process is determined based on two parameters probabilistically (line 11). If an offspring individual is determined to be produced by the base solver EA, it follows the procedure described in Section II-C (lines 18–20). Otherwise, the offspring individual is generated by the SEET method (lines 12–16). Specifically, to decide whether SEET is used to produce an offspring individual, two random numbers (i.e., rand$_1$ and rand$_2$) within [0,1] are independently generated. Then, they are compared with two parameters: the probability of SEET (p_{SEET}) and 1/rank(T_k), respectively. p_{SEET} that controls the occurrence of SEET is calculated as (line 8)

$$p_{\text{SEET}} = 0.5 + 0.5 \cdot (g - G_1)/(MAXGEN - G_1) \tag{17}$$

where g is the current generation and $MAXGEN$ is the maximum number of generations. It can be seen that p_{SEET} gradually increases to 1 as the evolution proceeds. This is because we want to keep the evolution process by different EAs independently to distinguish the successfully evolved tasks in the early stage. In the later stage, these successful parameters are encouraged to transfer to poorly evolved tasks to improve the search performance. Moreover, the task T_k with a larger rank(T_k) is considered to be a poorly evolved task and will have a larger probability to learn from the successfully evolved tasks. Note that if the group size of T_k denoted as $gSize(gID_k)$ is 1, the only task in the group will evolve independently.

If an offspring individual is determined to be produced by SEET, a source task denoted as T_s is first selected randomly based on the evolution quality analysis (line 12). Afterward, an EA index denoted as $\tau_s \in \{1, \ldots, NS\}$ associated with its parameter setting is selected based on the evolution experience of the selected source task T_s (line 13). Finally, the offspring individual is produced by crossover and mutation operators of the selected EA (denoted as EA_{τ_s}) (lines 15 and 16). The main components of SEET are evolution experience representation, source task selection, and the solver (i.e., EA) selection.

For the evolution experience representation, we define two variables: *count* and *countSuc*, for counting the used times and successful times of different EA for each task. At the beginning of the second evolution stage, *count* and *countSuc* are initialized as $NT \times NS$ zero-matrix (lines 3 and 4). Every time a new individual is produced, the element of the task and the selected EA in *count* is increased by 1 (lines 14 and 19). If the produced offspring individual by a selected EA survives through selection, the corresponding *countSuc* will be updated (lines 25–28). In this way, the success rates of different EAs on a task can be calculated.

For the source task selection, a source task T_s is randomly selected from the tasks whose ranks are smaller than $gSize(gID_k)/NS$ in the group of T_k (line 12). This is to encourage T_k to learn from the successful tasks in the group. Note that if $gSize(gID_k)/NS$ is smaller than 1, we simply randomly select a task for T_k from the group to learn.

For the solver selection, a solver index τ_s is selected based on the evolution experience of the source task T_s as (line 13)

$$\tau_s = \underset{i \in \{1, \ldots, NS\}}{\text{argmax}} \{countSuc(s, i)/count(s, i)\}. \quad (18)$$

That is, we select the parameter setting of EA from the evolution experience of T_s with the highest success rates. Afterward, F and Cr are generated by (8) and (9) using the parameter settings of mF and mCr of EA_{τ_s} (lines 15 and 16). The rest of the offspring producing process is executed in the same way as (4) and (5).

After the entire offspring population is produced, the evolution processes including fitness evaluation, selection, and archive update are executed (lines 23 and 24). The entire process is repeated until the stopping condition (e.g., when $NFE >= MAXNFE$) is met.

IV. Experimental Studies

A. Experimental Setup

1) Benchmark Problem: In the experiments, we use two representative MaTOP benchmarks, namely: 1) CEC19MaTOP [47] and 2) GECCO20MaTOP [48] to test the effectiveness of the proposed algorithm. Both CEC19MaTOP and GECCO20MaTOP use some basic functions with rather different global optima to act as the component tasks. These basic functions are Sphere, Rosenbrock, Rastrigin, Ackley, Griewank, Weierstrass, and Schwefel, which have heterogeneous function landscapes. For example, Sphere function is a smooth and single-modal function while Rastrigin is a multimodal function that contains many local optima. The CEC19MaTOP benchmark contains six 50-task problems and the GECCO20MaTOP benchmark contains ten 50-task problems. The numbers of different basic functions used in these benchmarks are listed in Table S.II in the supplemental material. Specifically, all the problems in the CEC19MaTOP benchmark only contain one type of basic function while some problems in the GECCO20MaTOP benchmark contain different types of basic functions. The only difference between the tasks that use the same basic function is that their function landscapes are shifted by different biases. Hence, the tasks that use the same basic function can be considered shift invariant according to Definition 1. Therefore, the problems (i.e., problems 4–10 in GECCO20MaTOP benchmark) that contain multiple types of basic functions are considered more difficult and challenging.

2) Parameter Settings: The parameter settings of TRADE are listed in Table S.III in the supplemental material. In this article, we use a DE variant with a *current-to-pbest* mutation strategy which has shown good ability in balancing exploration and exploitation. Then, DE with different parameter settings of mF and mCr in (8) and (9) is used as the base solvers. Specifically, mF of all the base solvers is fixed as 0.5, and mCr can have three available settings from {0.1, 0.5, 0.9}. Therefore, the base solver set $\mathcal{A}$ contains three base solvers, denoted as DE/0.1, DE/0.5, and DE/0.9, with DE/0.5 as the unified base solver EA_u. The population size of each task (NP) is set as 100. The $MAXGEN$ is set as 1000. Since there are $NT = 50$ tasks in each problem, the maximum NFEs ($MAXNFEs$) is set as $NT \times NP \times MAXGEN = 50\,00\,000$. The parameter of maximum generations for the first evolution stage (G_1) is set as 100. The parameters α and ρ of CRP in TSG are 0.05 and 10, respectively. The algorithms are implemented in MATLAB and the experiments are conducted on a computer cluster with processors Intel Xeon E5-2699 v3.

3) Compared Algorithms: To test the effectiveness of the TRADE algorithm, we carry out comparisons between TRADE, EMTO algorithms, and single-task optimization algorithms. All the compared EMTO algorithms are implemented based on the MP4MaT framework. To enable a fair comparison, they all use the same base solver set (i.e., $\mathcal{A} = \{$DE/0.1, DE/0.5, DE/0.9$\}$) as TRADE and the solver assignment strategy in the compared algorithms is randomly selecting a base solver from $\mathcal{A}$ for a given task at the beginning. The compared algorithms for solving MaTOP are EBS using DE as base solver (denoted as EBSDE) [24], MaTDE [25], MTEA-AD [28] using DE as base solver (denoted as MTDE-AD), AEMTO [29], and EMaTO [30]. The reasons for choosing these algorithms for comparison are as follows. First, all the compared EMTO algorithms are flexible in using different EAs as base solvers. Hence, despite that some EMTO algorithms such as EMaTO use other EAs such as GA as base solvers in the original paper, the DE can still be seamlessly used in these EMTO algorithms. Second, most of these algorithms adopt PDSM for measuring the similarity between tasks while TRADE adopts SISM. Moreover, the compared algorithms perform KT by transferring solutions while TRADE transfer EA parameters. Hence, the comparison can validate the effectiveness of the proposed SISM and KT methods.

Different from the EMTO algorithms, the single-task optimization algorithms solve the tasks independently without

KT. The compared single-task algorithms are STDE/0.1 [42], STDE/0.5 [42], STDE/0.9 [42], STDE/r, and STcDE [49]. Herein, the ST stands for "single-task" and is added as the prefix of the algorithm name. Specifically, the STDE/0.1, STDE/0.5, and STDE/0.9 use the parameter setting of $mCr = 0.1$, $mCr = 0.5$, and $mCr = 0.9$, respectively. To enable a fair comparison between TRADE and these STDEs with different parameter settings, these STDEs are implemented in a two-stage manner as TRADE. That is, in the first evolution stage, STDE uses the same unified base solver as TRADE to evolve and in the second evolution stage, STDE uses the base solver with its predefined parameter settings (i.e., mCr) to evolve. STDE/r is an STDE variant with "r" meaning "random." Specifically, in the second evolution stage of STDE/r, an EA associated with its parameter setting of mCr is randomly selected from $\mathcal{A}$ to evolve. Through the comparison between TRADE and the above STDEs, we can show that our TRADE can gradually adapt to the most suitable parameter setting for solving a set of similar tasks by SEET. Furthermore, STcDE is a single-task optimization algorithm using an adaptive DE with different parameters. Besides, there are other works [23] that use adaptive DE with different parameters to solve multitask optimization problems. The major difference between TRADE and these adaptive DEs is that TRADE uses cross-task knowledge to adjust parameters (e.g., mCr) while these adaptive DEs use intratask knowledge to adjust parameters. Therefore, comparing TRADE with STcDE can show the effectiveness of SEET in parameter adaptation.

4) Performance Metric: To reduce the statistical error caused by the randomness of the optimization algorithms, all the algorithms are run independently 30 times. The obtained best fitness in each run is collected and used for comparison. Afterward, we run a Wilcoxon rank sum test at a significance level of 0.05 between our TRADE algorithm and the compared algorithms on each task of the MaTOP. The symbols "W/T/L" indicate that our TRADE performs significantly better (Win), equal to (Tie), or significantly worse (Lose) than the compared algorithm on a task, respectively. Then, the number of tasks that TRADE is significantly better, equal to, or significantly worse than the compared algorithm is counted. TRADE is said to be better than the compared algorithm on a MaTOP if the number of "W" is larger than the number of "L" and the difference between W and L is at least larger than 5 (e.g., 10% of the 50 tasks). The comparative result on a problem will be given in the parenthesis by "+/=/−" meaning TRADE is better than, equal to, or worse than the compared algorithm. To further reduce statistical comparison error, we conduct the Friedman test [50] to compare performance among the multiple algorithms and report the average ranks, denoted as *AvgRank*. The smaller rank indicates better performance and the best result is marked in **boldface**. Moreover, the multiple comparison test [51] is adopted and the calculated p-values are given for each benchmark.

B. Results and Discussion

1) Comparison Between TRADE and EMTO Algorithms: The comparative results between TRADE and the EMTO algorithms are shown in Tables I and II. Moreover, the detailed

TABLE I
COMPARATIVE RESULTS BY WILCOXON RANK SUM TEST BETWEEN TRADE AND EMTO ALGORITHMS ON CEC19MATOP AND GECCO20MATOP BENCHMARKS

CEC19MaTOP					
TRADE vs	**AEMTO**	**EBSDE**	**EMaTO**	**MaTDE**	**MTDE-AD**
Problem	W/T/L	W/T/L	W/T/L	W/T/L	W/T/L
1	48/0/2(+)	49/1/0(+)	49/0/1(+)	50/0/0(+)	50/0/0(+)
2	13/12/25(−)	46/0/4(+)	4/32/14(−)	15/1/34(−)	50/0/0(+)
3	50/0/0(+)	50/0/0(+)	50/0/0(+)	50/0/0(+)	50/0/0(+)
4	27/23/0(+)	27/23/0(+)	7/38/5(=)	28/22/0(+)	44/6/0(+)
5	10/40/0(+)	10/40/0(+)	14/8/28(−)	6/44/0(+)	47/3/0(+)
6	0/0/50(−)	0/1/49(−)	24/26/0(+)	0/0/50(−)	34/16/0(+)
# of +/=/−	4/0/2	5/0/1	3/1/2	4/0/2	6/0/0
GECCO20MaTOP					
TRADE vs	**AEMTO**	**EBSDE**	**EMaTO**	**MaTDE**	**MTDE-AD**
Problem	W/T/L	W/T/L	W/T/L	W/T/L	W/T/L
1	50/0/0(+)	50/0/0(+)	24/24/2(+)	50/0/0(+)	50/0/0(+)
2	11/22/17(−)	25/18/7(+)	6/43/1(=)	24/21/5(+)	50/0/0(+)
3	50/0/0(+)	50/0/0(+)	50/0/0(+)	50/0/0(+)	50/0/0(+)
4	40/7/3(+)	40/7/3(+)	11/34/5(+)	42/5/3(+)	49/1/0(+)
5	38/12/0(+)	38/12/0(+)	29/16/5(+)	37/13/0(+)	49/1/0(+)
6	23/24/3(+)	23/23/4(+)	9/38/3(+)	17/30/3(+)	33/15/2(+)
7	49/1/0(+)	46/3/1(+)	27/14/9(+)	46/4/0(+)	50/0/0(+)
8	42/4/4(+)	33/14/3(+)	42/6/2(+)	33/13/4(+)	44/6/0(+)
9	30/20/0(+)	27/22/1(+)	39/11/0(+)	35/14/1(+)	37/12/1(+)
10	38/12/0(+)	29/21/0(+)	39/10/1(+)	32/18/0(+)	34/16/0(+)
# of +/=/−	9/0/1	10/0/0	9/1/0	10/0/0	10/0/0

TABLE II
COMPARATIVE RESULTS BY FRIEDMAN TEST AND MULTIPLE COMPARISON TEST AMONG TRADE AND EMTO ALGORITHMS ON CEC19MATOP AND GECCO20MATOP BENCHMARKS

CEC19MaTOP						
	TRADE	**AEMTO**	**EBSDE**	**EMaTO**	**MaTDE**	**MTDE-AD**
AvgRank	**62.69**	76.02	102.15	76.71	85.53	139.90
p-value	/	0.02	2.07E-08	0.01	9.71E-07	2.07E-08
GECCO20MATOP						
	TRADE	**AEMTO**	**EBSDE**	**EMaTO**	**MaTDE**	**MTDE-AD**
AvgRank	**48.78**	94.38	98.60	80.31	97.35	123.58
p-value	/	2.07E-08	2.07E-08	2.07E-08	2.07E-08	2.07E-08

results of the EMTO algorithms on 50 tasks in a representative MaTOP (i.e., GECCO20MaTOP5) are given in Table S.IV in the supplemental material. According to the numbers of +/=/− in Table I and the *AvgRank* in Table II, we can observe that our TRADE significantly outperforms AEMTO, EBSDE, EMaTO, MaTDE, and MTDE-AD on most of the problems in the CEC19MaTOP benchmark and the GECCO20MaTOP benchmark. Specifically, in the CEC19MaTOP benchmark that all the problems contain homogeneous tasks using the same basic function such that all the tasks can be considered shift invariant, our proposed TRADE based on SISM is more effective than the compared algorithms based on PDSM. This is because when the global optima in different tasks in a problem differ greatly, the PDSM may become less effective. On the contrary, the TRADE utilizes a new kind of similarity, namely, shift invariance, which can still be useful when the global optima of the tasks differ greatly. Moreover, in the more complex GECCO20MaTOP benchmark where the MaTOP (e.g., GECCO20MaTOP5) contains heterogeneous tasks using different basic functions, our TRADE still outperforms the compared EMTO algorithms. This indicates that TRADE can handle more difficult MaTOP with heterogeneous tasks than the compared EMTO algorithms.

2) Comparison Between TRADE and Single-Task Algorithms: The summarized comparative results between TRADE and the single-task optimization algorithms are

TABLE III
SUMMARIZED COMPARATIVE RESULTS BY WILCOXON RANK SUM TEST BETWEEN TRADE AND THE SINGLE-TASK OPTIMIZATION ALGORITHMS ON CEC19MATOP AND GECCO20MATOP BENCHMARKS

CEC19MATOP					
TRADE vs	**STDE/0.1**	**STDE/0.5**	**STDE/0.9**	**STDE/r**	**STcDE**
# of +/=/−	3/0/3	3/1/2	4/1/1	4/2/0	5/1/0
GECCO20MATOP					
TRADE vs	**STDE/0.1**	**STDE/0.5**	**STDE/0.9**	**STDE/r**	**STcDE**
# of +/=/−	10/0/0	3/4/3	9/1/0	10/0/0	7/0/3

TABLE IV
COMPARATIVE RESULTS BY FRIEDMAN TEST AND MULTIPLE COMPARISON TEST AMONG TRADE AND SINGLE-TASK OPTIMIZATION ALGORITHMS ON CEC19MATOP AND GECCO20MATOP BENCHMARKS

CEC19MaTOP						
	TRADE	**STDE/0.1**	**STDE/0.5**	**STDE/0.9**	**STDE/r**	**STcDE**
AvgRank	**61.25**	102.16	84.37	115.18	100.85	79.21
p-value	/	2.07E-08	5.96E-07	2.07E-08	2.07E-08	2.83E-04
GECCO20MATOP						
	TRADE	**STDE/0.1**	**STDE/0.5**	**STDE/0.9**	**STDE/r**	**STcDE**
AvgRank	**65.30**	114.01	89.07	105.00	102.15	67.49
p-value	/	2.07E-08	2.07E-08	2.07E-08	2.07E-08	0.99

TABLE V
NUMBER OF BEST RESULTS OBTAINED BY DIFFERENT EAS WITH DIFFERENT PARAMETER SETTINGS ON CEC19MATOP AND GECCO20MATOP BENCHMARKS

	CEC19MaTOP						**GECCO20MaTOP**									
Problem	1	2	3	4	5	6	1	2	3	4	5	6	7	8	9	10
STDE/0.1	0	0	50	5	8	50	0	0	50	0	19	19	17	11	16	22
STDE/0.5	0	50	0	22	42	0	50	9	0	38	25	7	33	27	26	25
STDE/0.9	50	0	0	23	0	0	0	41	0	12	6	24	0	12	8	3

shown in Tables III and IV. The detailed comparative results are shown in Tables S.V and S.VI in the supplemental material. Moreover, the number of obtained best results of different single-task EAs with different parameter settings (mCr) on the tested benchmark are shown in Table V. From Tables III–V, we can obtain several important observations and conclusions.

First, on the CEC19MaTOP benchmark with homogeneous tasks, the performances of EAs with different parameter settings are rather different on different problems. For example, Table V shows that STDE/0.9 obtains the best results on 50 tasks in problem 1 that only contains Rosenbrock function while STDE/0.5 obtains the best results on 50 tasks in problem 2 that only contains Ackley function.

Second, although STDEs with different parameter settings have their corresponding advantages on different problems, the proposed TRADE can achieve generally better results than these STDE variants in terms of average rank. This is because the proposed TRADE with SEET method can adaptively transfer successful parameters for solving a set of shift invariant tasks and reduce the effect of harmful parameters in the search process. STDE/r represents the expected performance of a DE algorithm without knowing which parameter among $mCr = 0.1$, $mCr = 0.5$, and $mCr = 0.9$ is best in advance. The TRADE gives encouraging results that TRADE outperforms STDE/r on most of the problems in the CEC19MaTOP benchmark. This indicates that TRADE is effective in the case that we have no prior knowledge about which parameter works better on a black-box problem.

Third, the proposed TRADE generally outperforms STDE/0.1, STDE/0.5, STDE/0.9, STDE/r on the more complex GECCO20MaTOP benchmark that contains heterogeneous tasks. Specifically, TRADE obtains more $+$ than $-$ and achieves the smallest average rank on two benchmarks. The results indicate that TRADE is capable of handling more difficult MaTOP with heterogeneous tasks.

Fourth, the proposed TRADE that adapts parameters by transferring successful parameters from other tasks outperforms STcDE that adapts parameters by the knowledge within the task on 12 problems in the two benchmarks. STcDE also uses the same base solver set $\mathcal{A}$ as TRADE does and increases the usage of the successful parameters in the search process. In summary, TRADE can be regarded as using cross-task knowledge to adapt parameters while STcDE can be regarded as using intratask knowledge to adapt parameters. These results indicate the effectiveness of the SEET method that adapts parameters by KT.

C. Component Analysis

1) Effects of TRS and TSG in the First Evolution Stage: The first evolution stage in TRADE mainly contains the TRS and TGS. TRS serves as representing each task to measure shift invariance-based similarity between tasks while TGS serves as grouping similar tasks into the same group based on SISM.

To testify the effectiveness of TRS, we formulate a TRADE variant called TRADE-PDSM. In TRADE-PDSM, the population center after the first evolution stage is used to represent a task instead of the way TRS does. As can be seen from the name, TRADE-PDSM is based on PDSM to represent a task that tries to capture the similarity in the global optima between tasks. On the contrary, the proposed TRS can be used to capture the shift invariance between tasks. Ideally, after TRS and TGS in the first evolution stage, the tasks using the same basic functions will be grouped into the same group.

The comparative result between TRADE and TRADE-PDSM is given in Table S.VII in the supplemental material. TRADE behaves slightly worse than TRADE-PDSM on the CEC19MaTOP benchmark and outperforms TRADE-PDSM on the GECCO20MaTOP benchmark. This is because problems in CEC19MaTOP contain homogeneous tasks using the same basic functions and the grouping strategy on these homogeneous tasks does not take effect significantly. However, on MaTOPs that contain several heterogeneous tasks with different basic functions such as problems 5-8 in the GECCO20MaTOP benchmark, TRADE significantly outperforms TRADE-PDSM. This indicates that the proposed TRS based on SISM is more effective in solving more complex MaTOP compared to TRADE-PDSM based on PDSM.

To testify the effectiveness of TGS, we formulate a TRADE variant called TRADE-w/o-TGS. In TRADE-w/o-TGS, the TGS is removed. That is, after the first evolution stage, all the tasks are grouped into a single group. The comparative result between TRADE and TRADE-w/o-TGS is given in Table S.VII in the supplementary material. The results in Table S.VII in the supplementary material show

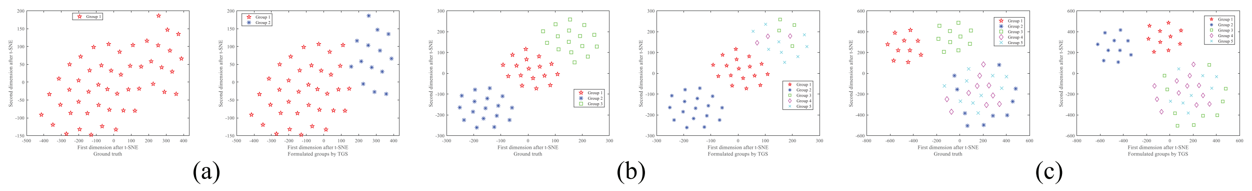


Fig. 4. Grouping results on the problems in GECCO20MaTOP benchmark based on TRS after the first evolution stage. Ground truth of the groups (left) and the formulated groups by TGS (right) after performing t-SNE dimensionality reduction method on the representations of tasks obtained by TRS after the first evolution stage on (a) GECCO20MaTOP1, (b) GECCO20MaTOP4, and (c) GECCO20MaTOP8.

that TRADE achieves similar performance on CEC19MaTOP benchmark and outperforms TRADE-w/o-TGS on 4 problems on GECCO20MaTOP benchmark. The advantage of TRADE is significant on MaTOPs (e.g., problem 5) that contain heterogeneous tasks. This indicates that TGS is effective by grouping similar tasks and transferring knowledge between these tasks to improve search efficiency.

To directly show the effect of TRS and TGS, we plot the grouping results after an independent run of TRADE on the GECCO20MaTOP benchmark in Fig. 4. The data after the first evolution stage is collected and the TRS is employed to represent each task. Based on these task representations, we adopt the t-distributed stochastic neighbor embedding (t-SNE) [52] to represent these tasks in 2-D space. In Fig. 4, the left figure in each subfigure is the ground truth for the grouping of the tasks. For example, problem 4 in GECCO20MaTOP benchmark uses three types of basic functions: Sphere, Rosenbrock, and Ackley. Hence, there are three groups of tasks and the tasks in the group that use the same basic function are considered shift invariant based on SISM. However, this information is not known in advance, and TRS and TGS are proposed with the aim of capturing the shift invariance. It can be seen that the right figure of each subfigure in Fig. 4 that TRS can represent tasks well. Specifically, the distances between similar tasks (i.e., from the same group) are small and the distances between dissimilar tasks (i.e., from different groups) are large after TRS. Then, based on representations obtained by TRS, the tasks can be grouped into different groups by TGS. For example, in the right figure of Fig. 4(b), the grouping result after TGS is very close to the ground truth of the groups. These results further indicate the effectiveness of the proposed TRS and TGS to measure shift invariance between tasks.

2) Effects of SEET in the Second Evolution Stage: To testify the effectiveness of the SEET method, we formulate three TRADE variants: TRADE/0, TRADE/0.5, and TRADE/1. The number after "TRADE/" represents the fixed parameter setting of p_{SEET}. For example, in the second evolution stage of TRADE/0, p_{SEET} is fixed to 0. Hence TRADE/0 represents the variants in that no KT method takes effect and each task evolves independently in the second evolution stage. The comparative results of TRADE and the compared variants are shown in Table S.VII in the supplementary material. TRADE significantly outperforms TRADE/0 on both two benchmarks. This indicates that the proposed SEET method is effective. Moreover, TRADE generally outperforms TRADE/0.5 and TRADE/1 on the GECCO20MaTOP benchmark. Since TRADE/0.5 and TRADE/1 use the fixed parameter settings of p_{SEET}, the results indicate the effectiveness of the design of adapting p_{SEET} when solving complex MaTOPs.

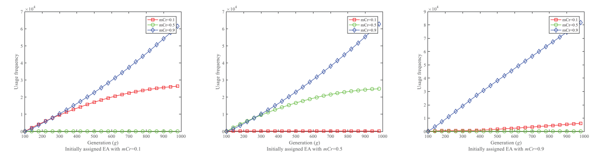


Fig. 5. Usage frequency of different EA parameters during the evolutionary process with different initially assigned EA of $mCr = 0.1$, $mCr = 0.5$, and $mCr = 0.9$, respectively, at the beginning of the second evolution stage in TRADE for solving CEC19MaTOP1.

Moreover, we formulate a TRADE variant called TRADE-self to testify the effectiveness of the parameter adaptation by SEET. TRADE-self is formulated based on the self-adaptive DE in [49] and only uses intratask information to select EA associated with its parameter settings adaptively to produce offspring. It should be noted that the major difference between TRADE and TRADE-self is that TRADE uses knowledge from other tasks to adapt parameters while TRADE-self uses knowledge from its task to adapt parameters independently. The comparative result between TRADE and TRADE-self is given in Table S.VII in the supplementary material. TRADE generally outperforms TRADE-self which further indicates that the parameter adaptation brought by the SEET method is effective.

To directly show the parameter adaptation behavior of TRADE, the usage frequency of different EA parameters with different initially assigned EA at the beginning of the second evolution stage in TRADE on CEC19MaTOP1 is plotted in Fig. 5. Note that all tasks in the problems of the CEC19MaTOP benchmark use the same basic function. Combining with Table V, we observe that in the left figure of Fig. 5, the initially assigned EA on a task of problem 1 in CEC19MaTOP is with a parameter setting of $mCr = 0.1$, which is not an ideal parameter for solving the task. However, by the SEET method, the successful parameter $mCr = 0.9$ which is a good parameter according to Table V is transferred to this task to help improve search efficiency. As a result, the usage of $mCr = 0.9$ gradually increases while the usage of $mCr = 0.1$ gradually decreases at a generation as the search process proceeds. In the right figure of Fig. 5, we observe that in the case that the initially assigned EA on a task is the most suitable parameter ($mCr = 0.9$), the usage of this parameter increases throughout the search process. This observation further indicates the effectiveness of the proposed SEET method.

D. Parameter Sensitivity Analysis

In this section, we investigate the parameter sensitivity of G_1 which controls the maximum generations used in the first evolution stage in TRADE. We run experiments on two benchmarks with different settings of $G_1 \in${20, 40, 60, 80,

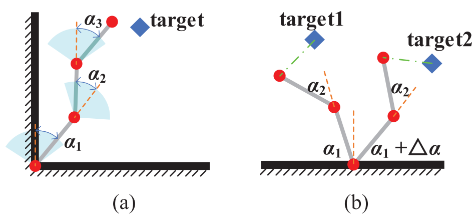


Fig. 6. (a) Example of robotic arm with three equal links. (b) Illustration of shift invariance in two robotic arm control tasks.

120, 140, 160, 180} in TRADE. Note that the default setting of TRADE is $G_1 = 100$. The TRADE variants are TRADE-20, TRADE-40, ..., TRADE-180, respectively. The comparative results between TRADE and the compared variants are shown in Table S.VIII in the supplemental material. The results indicate that a larger G_1 can lead to better search performance. This is because when G_1 is set to be a larger value, more evolution information is collected and used to represent the tasks by TRS. Then, TGS can group the tasks more accurately such that similar (i.e., shift invariant) tasks belong to the same group while dissimilar tasks belong to different groups. As a result, TRADE can improve search performance by sharing successful parameters among shift invariant tasks by the SEET method. However, setting a larger G_1 can lead to a higher computational cost since the computational time complexity of TGS is related to the dimensionality of the task representation. To strike a balance between accuracy and computational cost, we choose to set G_1 to a moderate value of 100.

E. Real-World Application Study

To study the effectiveness of the TRADE algorithm on real-world applications, we consider the robotic arm control problem [29]. In a robotic arm control task, the goal is to find the angle of each joint in the robotic arm such that the distance between the end position of the arm and the target position is minimized. A candidate solution to the task is represented as a D-dimensional vector $\overrightarrow{\alpha} = \{\alpha_1, \ldots, \alpha_D\}$ containing the angle of each joint. The end position of a solution is denoted as $P_{\overrightarrow{\alpha}}$ and the target position is denoted as P_{tar}. Hence, the objective is formulated as

$$f\left(\overrightarrow{\alpha}, P_{\text{tar}}\right) = \left\| P_{\overrightarrow{\alpha}} - P_{\text{tar}} \right\|. \tag{19}$$

Fig. 6(a) gives an example of a robotic arm control task with three joints. In our setting, the beginning position denoted as P_0 and the length of each joint of the robotic arm are fixed. Hence, we can generate different robotic arm control tasks by setting different target positions to construct a robotic arm control MaTOP.

An important property of this problem is that there exists shift invariance between tasks. Specifically, when the distances from two target positions of the two tasks to the beginning position P_0 is the same, the two tasks are shift invariance. We give simple proof here. Suppose there are two tasks T_1 and T_2 with target positions $P_{\text{tar},1}$ and $P_{\text{tar},2}$, respectively, such that $\|P_0 - P_{\text{tar},1}\| = \|P_0 - P_{\text{tar},2}\|$. This means that $P_{\text{tar},2}$ can be obtained by rotating the $P_{\text{tar},1}$ around the center P_0 with a certain angle denoted as $\Delta\alpha$. Then, for each solution denoted as $\overrightarrow{\alpha_1} = \{\alpha_{1,1}, \alpha_{1,2}, \ldots, \alpha_{1,D}\}$ of T_1, there exists a shift transformation with vector $\Delta\overrightarrow{\alpha} = \{\Delta\alpha, 0, \ldots, 0\}$ such that $f(\overrightarrow{\alpha_1}, P_{\text{tar},1}) = f(\overrightarrow{\alpha_1} + \Delta\overrightarrow{\alpha}, P_{\text{tar},2})$. According to Definition 1, the two tasks are shift invariant. An illustrating example of the shift invariance is given in Fig. 6(b) with two robotic arms with two joints. The two end positions of the two arms have the same distances to their corresponding target positions. Moreover, the solution of one arm only needs to change the first dimension α_1 with the angle $\Delta\alpha$ to obtain the solution of the other arm.

TABLE VI
COMPARATIVE RESULTS BETWEEN TRADE AND OTHER EMTO ALGORITHMS ON ROBOTIC ARM CONTROL MATOP

Robotic Arm Control MaTOP					
TRADE vs	**AEMTO**	**EBSDE**	**EMaTO**	**MaTDE**	**MTDE-AD**
Problem	W/T/L	W/T/L	W/T/L	W/T/L	W/T/L
1	21/29/0(+)	18/31/1(+)	35/15/0(+)	27/23/0(+)	49/1/0(+)
2	14/36/0(+)	13/37/0(+)	16/34/0(+)	12/38/0(+)	34/15/1(+)
3	9/41/0(+)	13/37/0(+)	17/33/0(+)	21/29/0(+)	23/17/10(+)
4	16/34/0(+)	15/35/0(+)	9/41/0(+)	11/39/0(+)	13/25/12(=)
5	17/33/0(+)	22/28/0(+)	22/28/0(+)	21/29/0(+)	13/28/9(=)
# of +/=/−	5/0/0	5/0/0	5/0/0	5/0/0	3/2/0

Based on the above discussions, we formulate five robotic arm control MaTOPs whose target positions of the tasks are shown in Fig. 7(a)–(e). The beginning position P_0 (red circle) of the arm is fixed at (0, 0). The target positions are distributed on the circle with same/different radii. We study these MaTOPs to investigate two important questions: Q1) can the proposed TRADE algorithm distinguish which tasks are similar (i.e., shift invariant) in these real-world MaTOPs? and Q2) can the proposed TRADE algorithm make use of the shift invariance between tasks to improve the overall search performance to efficiently solve these real-world MaTOPs?

In our experimental setting, each MaTOP contains 50 tasks with $D = 10$. We compare the TRADE algorithm with AEMTO, EBSDE, EMaTO, MaTDE, and MTDE-AD. They all adopt the population size of 50 for each task. We set $MAXGEN = 200$. G_1 is set as 20 for the TRADE algorithm. The grouped results of the tasks after the first evolution stage of the TRADE is plotted in Fig. 7(f)–(j) on MaTOP1–MaTOP5, respectively. The TRADE can approximately group similar tasks whose target positions having the same [i.e., Fig. 7(f)–(g)] or similar distance [i.e., Fig. 7(h)–(j)] to the beginning position into the same group. These results indicate that the proposed algorithm can distinguish which tasks are similar as an answer to Q1. The comparative results after 20 independent runs are shown in Table VI. The TRADE algorithm generally outperforms the compared algorithms in terms of the final average fitness. These results indicate that the proposed algorithm can make use of the shift invariance to achieve better search performance as an answer to Q2.

V. CONCLUSION

In this article, we considered a new kind of task similarity (i.e., shift invariance) and proposed a two-stage TRADE algorithm that can solve MaTOP efficiently. The TRS and TGS in the first evolution stage were efficient to identify shift invariance between tasks and to group up similar tasks. The SEET method in the second evolution stage was efficient in transferring successful parameters among the tasks within the same groups. In this way, the similarity between the tasks within the same group was made best use of to improve the search efficiency.

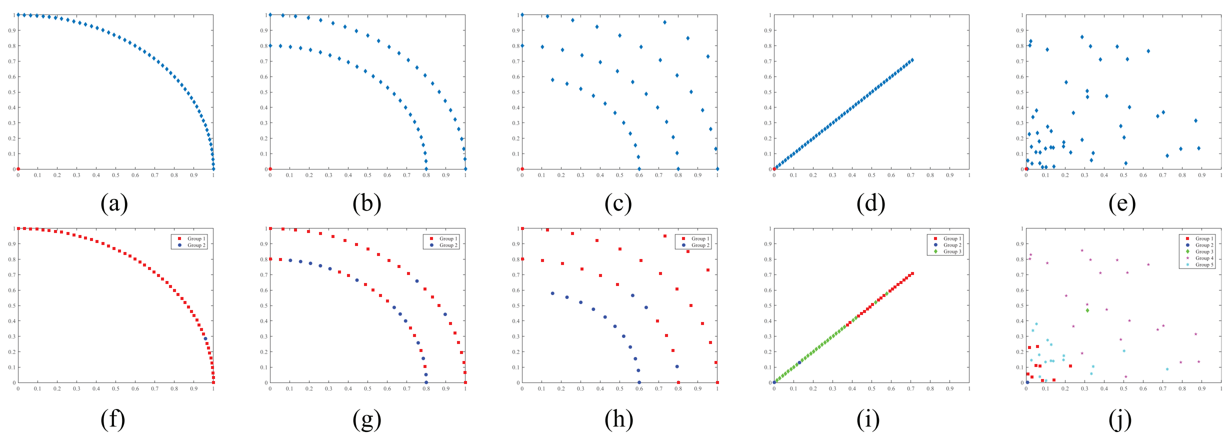


Fig. 7. (a)–(e) Target positions of multiple tasks in the robotic arm control MaTOP1–MaTOP5. (f)–(j) Corresponding grouped results on the tasks of the robotic arm control MaTOP1–MaTOP5 by TRADE.

The proposed TRADE algorithm has shown promising performance in solving MaTOP. However, there are still some issues that can be further studied in the future. For example, the TRADE algorithm consumes a certain amount of extra fitness evaluations to identify similar tasks and the TRS may not be accurate enough to distinguish similar tasks when a MaTOP contains many types of heterogeneous tasks. Therefore, for future work, researchers can consider the following two aspects: 1) discovering a more efficient and accurate TRS for identifying and capturing shift invariance between tasks and 2) extending the scope of similarity between tasks such as rotated invariance between function landscapes of the tasks and biobjective similarity (i.e., shape and domain) of the tasks [53].

References

[1] K. C. Tan, L. Feng, and M. Jiang, "Evolutionary transfer optimization—A new frontier in evolutionary computation research," *IEEE Comput. Intell. Mag.*, vol. 16, no. 1, pp. 22–33, Feb. 2021.

[2] A. Gupta, Y. Ong, and L. Feng, "Insights on transfer optimization: Because experience is the best teacher," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 51–64, Feb. 2018.

[3] X.-F. Liu et al., "Neural network-based information transfer for dynamic optimization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 5, pp. 1557–1570, May 2020.

[4] Q. Song, Y.-J. Zheng, J. Yang, Y.-J. Huang, W.-G. Sheng, and S.-Y. Chen, "Predicting demands of COVID-19 prevention and control materials via co-evolutionary transfer learning," *IEEE Trans. Cybern.*, early access, Apr. 21, 2022, doi: 10.1109/TCYB.2022.3164412.

[5] S. Yao, Q. Kang, M. Zhou, M. J. Rawa, and A. Abusorrah, "A survey of transfer learning for machinery diagnostics and prognostics," *Artif. Intell. Rev.*, to be published, doi: 10.1007/s10462-022-10230-4.

[6] Z.-H. Zhan, L. Shi, K. C. Tan, and J. Zhang, "A survey on evolutionary computation for complex continuous optimization," *Artif. Intell. Rev.*, vol. 55, no. 1, pp. 59–110, 2022.

[7] Z.-H. Zhan et al., "Matrix-based evolutionary computation," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 6, no. 2, pp. 315–328, Apr. 2022.

[8] Q. Deng, Q. Kang, L. Zhang, M. C. Zhou, and J. An, "Objective space-based population generation to accelerate evolutionary algorithms for large-scale many-objective optimization," *IEEE Trans. Evol. Comput.*, early access, Apr. 22, 2022, doi: 10.1109/TEVC.2022.3166815.

[9] Y. Liu, D. Gong, J. Sun, and Y. Jin, "A many-objective evolutionary algorithm using a one-by-one selection strategy," *IEEE Trans. Cybern.*, vol. 47, no. 9, pp. 2689–2702, Sep. 2017.

[10] R. Chai, A. Tsourdos, A. Savvaris, S. Chai, and Y. Xia, "Solving constrained trajectory planning problems using biased particle swarm optimization," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 57, no. 3, pp. 1685–1701, Jun. 2021.

[11] X. F. Liu, Z.-H. Zhan, Y. Gao, J. Zhang, S. Kwong, and J. Zhang, "Coevolutionary particle swarm optimization with bottleneck objective learning strategy for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 23, no. 4, pp. 587–602, Aug. 2019.

[12] J. Tang, G. Liu, and Q. Pan, "A review on representative swarm intelligence algorithms for solving optimization problems: Applications and trends," *IEEE/CAA J. Automatica Sinica*, vol. 8, no. 10, pp. 1627–1643, Oct. 2021.

[13] S.-H. Wu, Z.-H. Zhan, and J. Zhang, "SAFE: Scale-adaptive fitness evaluation method for expensive optimization problems," *IEEE Trans. Evol. Comput.*, vol. 25, no. 3, pp. 478–491, Jun. 2021.

[14] Y. Wang, S. Gao, M. Zhou, and Y. Yu, "A multi-layered gravitational search algorithm for function optimization and real-world problems," *IEEE/CAA J. Automatica Sinica*, vol. 8, no. 1, pp. 94–109, Jan. 2021.

[15] L. Zhou et al., "Toward adaptive knowledge transfer in multifactorial evolutionary computation," *IEEE Trans. Cybern.*, vol. 51, no. 5, pp. 2563–2576, May 2021.

[16] S.-H. Wu, Z.-H. Zhan, K. C. Tan, and J. Zhang, "Orthogonal transfer for multitask optimization," *IEEE Trans. Evol. Comput.*, early access, Mar. 17, 2022, doi: 10.1109/TEVC.2022.3160196.

[17] A. Gupta, Y. Ong, and L. Feng, "Multifactorial evolution: Toward evolutionary multitasking," *IEEE Trans. Evol. Comput.*, vol. 20, no. 3, pp. 343–357, Jun. 2016.

[18] J.-Y. Li, Z.-H. Zhan, K. C. Tan, and J. Zhang, "A meta-knowledge transfer-based differential evolution for multitask optimization," *IEEE Trans. Evol. Comput.*, vol. 26, no. 4, pp. 719–734, Aug. 2022.

[19] K. K. Bali, Y. S. Ong, A. Gupta, and P. S. Tan, "Multifactorial evolutionary algorithm with online transfer parameter estimation: MFEA-II," *IEEE Trans. Evol. Comput.*, vol. 24, no. 1, pp. 69–83, Feb. 2020.

[20] L. Feng et al., "Solving generalized vehicle routing problem with occasional drivers via evolutionary multitasking," *IEEE Trans. Cybern.*, vol. 51, no. 6, pp. 3171–3184, Jun. 2021.

[21] A. D. Martinez, J. Del Ser, E. Osaba, and F. Herrera, "Adaptive multi-factorial evolutionary optimization for multi-task reinforcement learning," *IEEE Trans. Evol. Comput.*, vol. 26, no. 2, pp. 233–247, Apr. 2022, doi: 10.1109/TEVC.2021.3083362.

[22] L. Zhou et al., "Towards effective mutation for knowledge transfer in multifactorial differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, 2019, pp. 1541–1547.

[23] Z. Liang, H. Dong, C. Liu, W. Liang, and Z. Zhu, "Evolutionary multitasking for multiobjective optimization with subspace alignment and adaptive differential evolution," *IEEE Trans. Cybern.*, vol. 52, no. 4, pp. 2096–2109, Apr. 2022.

[24] R. Liaw and C. Ting, "Evolutionary many-tasking based on biocoenosis through symbiosis: A framework and benchmark problems," in *Proc. IEEE Congr. Evol. Comput.*, 2017, pp. 2266–2273.

[25] Y. Chen, J. Zhong, L. Feng, and J. Zhang, "An adaptive archive-based evolutionary framework for many-task optimization," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 4, no. 3, pp. 369–384, Jun. 2020.

[26] J.-B. Mouret and G. Maguire, "Quality diversity for multi-task optimization," in *Proc. Genet. Evol. Comput. Conf.*, 2020, pp. 121–129.

[27] C. Wang, J. Liu, K. Wu, and C. Ying, "Learning large-scale fuzzy cognitive maps using an evolutionary many-task algorithm," *Appl. Soft Comput.*, vol. 108, Sep. 2021, At. no. 107441.

[28] C. Wang, J. Liu, K. Wu, and Z. Wu, "Solving multi-task optimization problems with adaptive knowledge transfer via anomaly detection," *IEEE Trans. Evol. Comput.*, vol. 26, no. 2, pp. 304–318, Apr. 2022, doi: 10.1109/TEVC.2021.3068157.

[29] H. Xu, A. K. Qin, and S. Xia, "Evolutionary multi-task optimization with adaptive knowledge transfer," *IEEE Trans. Evol. Comput.*, vol. 26, no. 2, pp. 290–303, Apr. 2022, doi: 10.1109/TEVC.2021.3107435.

[30] Z. Liang, X. Xu, L. Liu, Y. Tu, and Z. Zhu, "Evolutionary many-task optimization based on multisource knowledge transfer," *IEEE Trans. Evol. Comput.*, vol. 26, no. 2, pp. 319–333, Apr. 2022, doi: 10.1109/TEVC.2021.3101697.

[31] S. Huang, J. Zhong, and W.-J. Yu, "Surrogate-assisted evolutionary framework with adaptive knowledge transfer for multi-task optimization," *IEEE Trans. Emerg. Topics Comput.*, vol. 9, no. 4, pp. 1930–1944, Oct.–Dec. 2021, doi: 10.1109/TETC.2019.2945775.

[32] Z.-H. Zhan, Z.-J. Wang, H. Jin, and J. Zhang, "Adaptive distributed differential evolution," *IEEE Trans. Cybern.*, vol. 50, no. 11, pp. 4633–4647, Nov. 2020.

[33] X. Qiu, K. C. Tan, and J.-X. Xu, "Multiple exponential recombination for differential evolution," *IEEE Trans. Cybern.*, vol. 47, no. 4, pp. 995–1006, Apr. 2017.

[34] J.-Y. Li, Z.-H. Zhan, K. C. Tan, and J. Zhang, "Dual differential grouping: A more general decomposition method for large-scale optimization," *IEEE Trans. Cybern.*, early access, Mar. 25, 2022, doi: 10.1109/TCYB.2022.3158391.

[35] Z.-J. Wang, J.-R. Jian, Z.-H. Zhan, Y. Li, S. Kwong, and J. Zhang, "Gene targeting differential evolution: A simple and efficient method for large scale optimization," *IEEE Trans. Evol. Comput.*, early access, Jun. 23, 2022, doi: 10.1109/TEVC.2022.3185665.

[36] Z.-G. Chen, Z.-H. Zhan, H. Wang, and J. Zhang, "Distributed individuals for multiple peaks: A novel differential evolution for multimodal optimization problems," *IEEE Trans. Evol. Comput.*, vol. 24, no. 4, pp. 708–719, Aug. 2020.

[37] Z.-J. Wang et al., "Dual-strategy differential evolution with affinity propagation clustering for multimodal optimization problems," *IEEE Trans. Evol. Comput.*, vol. 22, no. 6, pp. 894–908, Dec. 2018.

[38] S.-C. Liu, Z.-H. Zhan, K. C. Tan, and J. Zhang, "A multiobjective framework for many-objective optimization," *IEEE Trans. Cybern.*, vol. 52, no. 12, pp. 13654–13668, Dec. 2022.

[39] Y. Yu, Z. Lei, Y. Wang, T. Zhang, C. Peng, and S. Gao, "Improving dendritic neuron model with dynamic scale-free network-based differential evolution," *IEEE/CAA J. Automatica Sinica*, vol. 9, no. 1, pp. 99–110, Jan. 2022.

[40] Z. G. Chen, Z.-H. Zhan, S. Kwong, and J. Zhang, "Evolutionary computation for intelligent transportation in smart cities: A survey [review article]," *IEEE Comput. Intell. Mag.*, vol. 17, no. 2, pp. 83–102, May 2022.

[41] G. Karafotias, M. Hoogendoorn, and A. E. Eiben, "Parameter control in evolutionary algorithms: Trends and challenges," *IEEE Trans. Evol. Comput.*, vol. 19, no. 2, pp. 167–187, Apr. 2015.

[42] Z.-H. Zhan et al., "Cloudde: A heterogeneous differential evolution algorithm and its distributed cloud version," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 3, pp. 704–716, Mar. 2017.

[43] J. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, Oct. 2009.

[44] R. Zhang, "Making convolutional networks shift-invariant again," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7324–7334.

[45] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient *k*-means clustering algorithm: Analysis and implementation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 881–892, Jul. 2002.

[46] D. M. Blei and P. I. Frazier, "Distance dependent Chinese restaurant processes," *J. Mach. Learn. Res.*, vol. 12, no. 74, pp. 2461–2488, 2011.

[47] L. Feng, K. Qin, A. Gupta, Y. Yuan, Y.-S. Ong, and X. Chi. "IEEE CEC 2019 competition on evolutionary multi-task optimization." 2019. [Online]. Available: http://www.bdsc.site/websites/MTO_competiton_2019/MTO_Competition_CEC_2019.html

[48] L. Feng et al. "GECCO2020 competition on evolutionary multi-task optimization," 2020. [Online]. Available: http://www.bdsc.site/websites/MTO_competition_2020/MTO_Competition_GECCO_2020.html

[49] J. Tvrdík and R. Poláková, "Competitive differential evolution applied to CEC 2013 problems," in *Proc. IEEE Congr. Evol. Comput.*, 2013, pp. 1651–1657.

[50] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, Mar. 2011.

[51] O. J. Dunn, "Multiple comparisons among means," *J. Amer. Stat. Assoc.*, vol. 56, no. 293, pp. 52–64, Apr. 2012.

[52] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 86, pp. 2579–2605, Nov. 2008.

[53] Y. Jiang, Z.-H. Zhan, K. C. Tan, and J. Zhang, "A bi-objective knowledge transfer framework for evolutionary many-task optimization," *IEEE Trans. Evol. Comput.*, early access, Sep. 29, 2022, doi: 10.1109/TEVC.2022.3210783.

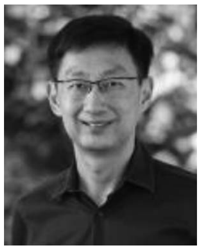
Sheng-Hao Wu (Student Member, IEEE) received the B.S. degree in computer science and technology from the South China University of Technology, Guangzhou, China, in 2019, where he is currently pursuing the Ph.D. degree in computer science and technology with the School of Computer Science and Engineering.

His research interests mainly include computational intelligence, machine learning, and their applications in real-world problems.

Zhi-Hui Zhan (Senior Member, IEEE) received the bachelor's degree and the Ph.D. degree in computer science from Sun Yat-Sen University, Guangzhou China, in 2007 and 2013, respectively.

He is currently the Changjiang Scholar Young Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. His current research interests include evolutionary computation, swarm intelligence, and their applications in real-world problems.

Dr. Zhan was a recipient of the IEEE Computational Intelligence Society (CIS) Outstanding Early Career Award in 2021. He is one of the World's Top 2% Scientists for both Career-Long Impact and Year Impact in Artificial Intelligence and one of the Highly Cited Chinese Researchers in Computer Science. He is currently the Chair of Membership Development Committee in IEEE Guangzhou Section and the Vice-Chair of the IEEE CIS Guangzhou Chapter. He is currently an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the *Neurocomputing*, the *Memetic Computing*, and the *Machine Intelligence Research*.



Kay Chen Tan (Fellow, IEEE) received the B.Eng. (First Class Hons.) and Ph.D. degrees from the University of Glasgow, Glasgow, U.K., in 1994 and 1997, respectively.

He is currently a Chair Professor of Computational Intelligence with the Department of Computing, Hong Kong Polytechnic University, Hong Kong SAR. He has published over 300 refereed articles and seven books.

Prof. Tan is currently the Vice-President (Publications) of IEEE Computational Intelligence Society, USA. He served as the Editor-in-Chief for the *IEEE Computational Intelligence Magazine* from 2010 to 2013 and for the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION from 2015 to 2020. He currently serves as the Chief Co-Editor for Springer Book Series on *Machine Learning: Foundations, Methodologies, and Applications*, and as an Editorial Board Member for more than ten journals.

Jun Zhang (Fellow, IEEE) received the Ph.D. degree from the City University of Hong Kong, Hong Kong, SAR, in 2002.

He is currently a Korea Brain Pool Fellow Professor with Hanyang University, Seoul, South Korea. His current research interests include computational intelligence, cloud computing, operations research, and power electronic circuits. He has published over more than 150 IEEE TRANSACTIONS papers in his research areas.

Dr. Zhang was a recipient of the Changjiang Chair Professor from the Ministry of Education, China, in 2013, the National Science Fund for Distinguished Young Scholars of China in 2011, and the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He is currently an Associate Editor of the IEEE TRANSACTIONS ON CYBERNETICS and IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION.