An Adaptive Archive-Based Evolutionary Framework for Many-Task Optimization

Yongliang Chen, Jinghui Zhong, Liang Feng, and Jun Zhang, *Fellow, IEEE*

***Abstract*—Multi-task optimization is an emerging research topic in computational intelligence community. In this paper, we propose a novel evolutionary framework, many-task evolutionary algorithm (MaTEA), for many-task optimization. In the proposed MaTEA, an adaptive selection mechanism is proposed to select suitable "assisted" task for a given task by considering the similarity between tasks and the accumulated rewards of knowledge transfer during the evolution. Besides, a knowledge transfer schema via crossover is adopted to exchange information among tasks to improve the search efficiency. In addition, to facilitate measuring similarity between tasks and transferring knowledge among tasks that arrive at different time instances, multiple archives are integrated with the proposed MaTEA. Experiments on both single-objective and multi-objective optimization problems have demonstrated that the proposed MaTEA can outperform the state-of-the-art multi-task evolutionary algorithms, in terms of search efficiency and solution accuracy. Besides, the proposed MaTEA is also capable of solving dynamic many-task optimization where tasks arrive at different time instances.**

***Index Terms*—Evolutionary algorithm, multi-task optimization, many-task optimization, dynamic control, adaptive strategy.**

I. Introduction

EVOLUTIONARY algorithms (EAs) are population-based stochastic optimization methods inspired by biological evolution [1]–[4]. EAs generally start with a population of randomly generated individuals, with each individual representing a candidate solution to the problem of interest. Then, new offsprings are generated iteratively by performing evolutionary operators such as mutation and crossover. In each iteration, fitter offsprings will survive to the next generation, while those with poor quality will be eliminated. In this way, the population of solutions would gradually converge to the global (or near global) optimal solutions. Owing to their simple implementation and high search efficiency, EAs have been applied to a wide range of applications, including multi-objective optimization [5]–[9], bi-level optimization [10], combinational optimization [11], dynamic optimization [12], and analog circuit optimization problems [13].

Despite their great successes, EAs still contain some drawbacks which limit their practicability. One of the major drawbacks is that EAs start to solve a problem from scratch and focus on solving one problem at a time. However, it is well established that problems in our daily life are often related to each other. Problem-specific knowledge or history knowledge of solving related similar problems are often helpful to solve the new encountered problem. Although the concept of knowledge transfer has been prominent in machine learning field, it has received far less attention in the EA community.

Recently, a multi-task optimization (MTO) paradigm has been introduced and a new evolutionary algorithm named multifactorial evolutionary algorithm (MFEA) [14] has been proposed. In MFEA, the common knowledge between similar related tasks is utilized to improve the search efficiency. Empirical studies on numerical optimization problems have shown that the MFEA can significantly improve the search efficiency in comparison to the single-task EAs. MTO has now become an emerging research topic that has attracted increasing attention in the field of computational intelligence. In the last three years, a number of multi-task EAs have been proposed and applied to a number of applications such as bi-level optimization problems and symbolic regression problems [15]–[19]. However, existing MFEAs contain some drawbacks. First, they are only suitable for solving a small number of tasks (e.g., two) at the same time. If there are multiple related problems at hand, existing methods would become ineffective. This is because the knowledge transfer strategy in MFEA is aimless with no feedback to adjust itself in a complicated multi-task environment. When there is a mixture of positive and negative tasks, the evolution of a task might be slowed down because of the influences from the negative tasks. Second, existing MFEA variants require all tasks to be solved at the same time. Nevertheless, in practical applications, problems often arrive at different time instances (e.g., tasks submitted to cloud computing platform). Few works have been proposed to handle the situation when tasks are in incoordinate conditions (i.e., at different evolution phases). Thus, designing an effective multi-task EA framework to solve multiple tasks arriving dynamically is still a promising research topic remained to be explored.

To address the above issues, this paper proposes a new many-task optimization (MaTO) framework named many-task evolutionary algorithm (MaTEA) framework for solving many

Manuscript received November 7, 2018; revised April 3, 2019; accepted May 4, 2019. Date of publication June 3, 2019; date of current version May 23, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 61602181 and Grant 61876025, in part by the Program for Guangdong Introducing Innovative and Enterpreneurial Teams under Grant 2017ZT07X183, and in part by the National Training Program of Innovation and Entrepreneurship for Undergraduates under Grant 201810561127. *(Corresponding author: Jinghui Zhong.)*

Y. Chen and J. Zhong are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510330, China (e-mail: chanwl0629@gmail.com; jinghuizhong@gmail.com).

L. Feng is with the College of Computer Science, Chongqing University, Chongqing 400044, China (e-mail: liangf@cqu.edu.cn).

J. Zhang is with Victoria University, Melbourne, VIC 8001, Australia (e-mail: junzhang@ieee.org).

Digital Object Identifier 10.1109/TETCI.2019.2916051

tasks that arrive at different time instances. MaTEA uses multiple subpopulations to solve multiple tasks simultaneously, with each subpopulation focusing on solving the assigned task using an independent EA solver. The core idea of MaTEA is to timely perceive the relation among tasks in the system. When knowledge transfer is triggered, an adaptive strategy is proposed to choose the most suitable task to be paired with the present task for knowledge transfer. Every task has its own "interest" in finding knowledge transfer target, since a task is only helpful to some tasks. In MaTEA, the selection probabilities of knowledge transfer targets for each task are updated dynamically. A similarity measurement is proposed based on the KullbackLeibler divergence (KLD) [20] of distribution, which is utilized together with the accumulated rewards during the evolution process to update the selection probabilities. In this way, positive knowledge transfer could be encouraged and negative transfer is then reduced. Furthermore, archives are adopted in the proposed framework to record related information of different tasks during the evolution process, which is used to measure the similarity among tasks. By using the archive-assisted knowledge transfer strategy, the MaTEA is capable of efficiently solving many tasks which arrive at different time instances. To evaluate the effectiveness and efficiency of the proposed MaTEA, both single-objective and multi-objective many-task benchmark problems are considered for investigation [21], [22]. Several state-of-the-art MFEAs, together with a recently published many-task EA (EBS) [23] are used for comparison. The experiment results show that the proposed MaTEA can offer superior performance in terms of solution quality and convergence speed.

The remainder of this paper is organized as follows. Section II introduces some preliminary background knowledge and related work. The details of MaTEA are described in Section III and the experimental study is presented in Section IV and V. Lastly, Section VI draws the conclusions.

II. Preliminaries

In this section, the definition of multi-task optimization problem is given at first. Then, the well-known MFEA is introduced. Finally, related works on multi-task optimization are presented.

A. Multi-Task Optimization Problem

Generally, the goal of multi-task optimization (MTO) is to find the optimal solutions for multiple tasks in a single run of the solver. Specifically, denote T_j as the jth task to be solved, which has a search space X_j, and an objective function $f_j : X_j \rightarrow \mathbb{R}$. Suppose there are K tasks to be optimized simultaneously and they are all *minimization* problems. Then, the aim of MTO is to find a set of solutions $\{x_1^*, x_2^*, \ldots, x_K^*\}$ which satisfies

$$\mathbf{x}_i^* = \underset{\mathbf{x}}{\operatorname{argmin}} f_i(\mathbf{x}), i = 1, 2, \ldots, K \tag{1}$$

Most existing algorithms for MTO focus on problems with a small number of tasks (e.g., two). In this paper, we aim to solve MTO problems with more than three tasks. We call this kind of MTO many-task as optimization(MaTO).

Algorithm 1: The Pseudocode of MFEA.

Input: NP, the population size; K, the number of the tasks;
Output: The best solutions of all tasks.

1: Randomly generate NP individuals to form the initial population (P).
2: Evaluate all individuals of P on all tasks, and calculate their *skill factor*.
3: **repeat**
4: Generate an offsprings population (O) by means of *assortative mating* based on P.
5: Evaluate individuals in O on the selected tasks assigned by *vertical cultural transmission*.
6: Merge P and O to generate an $intermediate - pop$.
7: Update the *scalar fitness* and *skill factor* of each individual in $intermediate - pop$.
8: Select the fittest NP individuals from $intermediate - pop$ to form P for the next generation.
9: **until** termination criterion is met.

B. Multifactorial Evolutionary Algorithm (MFEA)

In MFEA, all tasks are evolved by a population simultaneously and every individual can have the probability to take charge of every task. In this manner, all individuals are required to be encoded in a unified search space Y. Commonly, Y is normalized to $[0, 1]^D$, where D is the dimension number of the unified space, so that tasks of different domains can evolve collectively. Generally, if there are K tasks to be optimized, the dimension of the tasks are $D_1, D_2, \ldots, D_K$. Then, the dimension number (i.e., chromosome length) of the unified search space Y should be $D = \max\{D_j\}$, where $j = 1, 2, \ldots, K$. It should be noticed that, solutions encoded in space Y need to be decoded to the problem-dependent domain by $x = L_j + (U_j - L_j) \cdot y$ to evaluate their fitness values, where L_j and U_j are respectively the upper bound and lower bound of the jth task and x is a feasible solution of the corresponding task in the problem-specific space X_j.

In MFEA, there are some definitions for p_i in the population as follows.

1) Factorial Cost: For a given task T_j, the *factorial cost* of individual p_i is denoted as Ψ_j^i. If p_i is a feasible solution to T_j and not violating the constraints of T_j, Ψ_j^i will be the fitness value of p_i on T_j. Otherwise, it will be a very large number meaning that p_i should not be considering as a candidate solution to T_j.

2) Factorial Rank: The *factorial rank* r_j^i of p_i on T_j is the index of p_i that gained when competing with all other individuals with respect to Ψ_j in an ascending order.

3) Scalar Fitness: Values in the list of *factorial rank* of p_i are then reduced to the *scalar fitness*, which is computed as by $\psi_i = \frac{1}{\min\{r_j^i\}}$, where $j = 1, 2, \ldots, K$.

4) Skill Factor: The *skill factor* τ_i refers to the most effective task that p_i is capable of, which is defined as $\tau_i = \operatorname{argmin}_{\mathbf{j}}\{r_j^i\}$.

The whole procedure of MFEA is presented in Alg. 1. It starts with a randomly generated population, which are then evaluated

on all tasks before getting their *scalar fitness* and *skill factor*. Then, MFEA adopts the following two strategies to evolve the population:

Assortative Mating: For the two randomly chosen parents p_a and p_b, if they focus on the same task ($\tau_a = \tau_b$) or a threshold named *random mating probability (rmp)* is satisfied, p_a and p_b will perform crossover to generate two offsprings. Otherwise, p_a and p_b will perform mutation to generate two offsprings independently.

Vertical Cultural Transmission: For a given offspring c, it is not efficient to evaluate it on all tasks. Therefore, the *skill factor* of c inherits from one of its parents randomly. If c has two parents, there is a equal probability of inheriting each of their *skill factor*. Otherwise, if c has only one parent, the *skill factor* will be assigned as the same as its parent.

The MFEA uses the above two strategies to generate offsprings and evaluate their fitness according to their *skill factor*. Lastly, MFEA adopts an elitist EA selection method to update the population.

C. Existing Work on Multitask Optimization

The term evolutionary multi-task is first introduced by Gupta *et al.* [14] as a new paradigm in evolutionary optimization field. It is classified into one of the three distinct methodological perspectives that have been developed for transfer optimization [24]. In [14], a new algorithm named multi-factorial evolutionary algorithm (MFEA) is proposed, which has now become a mainstreaming algorithm framework for MTO. Two core ingredients of MFEA are the *assortative mating* and *vertical cultural transmission*, which conduct knowledge transfer between different tasks. As MFEA has achieved promising results, Ong and Gupta [25] made a profound discussion from the view of computer science and applied it to many different types of optimization problems. The results have shown the great potential of MTO. Then, based on MFEA, Gupta *et al.* [16] further extended it to solve multi-objective optimization problems. It has been shown that on both single-objective and multi-objective problems, MFEA can outperform the single-task evolutionary algorithm.

Since the first establishment of MFEA, a number of MFEA variants have been proposed over the past few years. Feng *et al.* [17] made an empirical study on embedding particle swarm optimization (PSO) [26] and differential evolution (DE) [27] in MFEA, proposing two evolutionary algorithms named multifactorial PSO and multifactorial DE. Zhong *et al.* [18] integrated genetic programming (GP) with MFEA to form multifactorial GP to solve regression problems. Compared with single-task evolutionary optimizers, the multifactorial solvers achieve evident superiority. Wen *et al.* [28] made investigation on how knowledge transfer contribute to solving multi-task problems, suggesting that due to the differences among the fitness landscape of the problems, knowledge transfer may cause the impairment of exploitation capability consumption of extra resources in the late phase. In [28], the authors proposed a new method to detect the *parting ways*, which is the threshold of stopping knowledge transfer, and a way of resource reallocation. The results show that avoiding negative transfer can effectively enhance the performance of MTO.

As MFEA has obtained prominent performance in optimization field, it has been applied to many different applications. Gupta *et al.* [15] applied MFEA to the bi-level optimization field, extending the basic nested bi-level evolutionary algorithm (NBLEA) in a multi-task manner and proposing a new algorithm named multi-task BLEA (M-BLEA). In M-BLEA, lower level optimization is a multi-task problem, which can efficiently accelerate the convergence to solutions of high quality and better promote solving the upper level problem. Moreover, Ding *et al.* [19] proposed a general framework based on MFEA to solve expensive problem. Two strategies named *decision variable translation strategy* and *decision variable shuffling strategy*, are proposed to overcome the problems of differences in optimums locations in the unified space and mismatches in chromosomes length, respectively. Zhou *et al.* [29] conducted a case study in capacitated vehicle routing problem, which shows MFEA can have practical usage in real-world problems.

MTO associated with transfer learning is another hot topic. The key of transfer learning is how to manage the information of solved tasks to assist and accelerate the optimization of new tasks. Feng *et al.* [30], [31] proposed a method, which transfer knowledge explicitly between two populations linearly by generating an autoencoder. In that way, a task being optimized can benefit from tasks solved by reconstructing genetic chromosomes using past experiences. Min *et al.* [32] used transfer evolutionary multi-objective optimization to solve computationally expensive problems with the help of surrogates, by sharing parameters information in the models. It is justified that sharing and transferring are important approaches to enhance capability of optimization algorithms.

Different from the basic MFEA which evolves in one population, some researchers suggest using multi-population method for MTO. Chen *et al.* [33] proposed a multi-population MTO framework for multi-objective problems. The results on benchmark problems had shown that the multi-population framework can have distinct advantages over MO-MFEA. Similarly, Liu *et al.* [34] made study on using surrogate model to assist the evolution procedure, proposing a multi-population-based algorithm named SaM-MA for single-objective problems. Inspired by cooperative co-evolutionary genetic algorithm [35], Chen *et al.* [36] came up with EMTSO-CCMA using *quasi-Newton* method as learning strategy to solve single-objective problems.

However, existing MTO methods mainly focus on solving problems with a few number of tasks (e.g., two). Few work have been proposed on solving many-task optimization problems (MaTO). Liaw and Ting [23] proposed a preliminary algorithm named Evolution of biocoenosis through symbiosis (EBS) for MaTO. EBS allows adaptive control of the knowledge transfer probability and transfers knowledge by sharing offsprings of the corporate populations. This strategy can prevent negative transfer by reducing the knowledge transfer rate adaptively, which is different from the fixed parameter *rmp* in MFEA. Results indicate that, compared with MFEA, EBS can have much better performance when faced with situations of multiple tasks. Nevertheless, drawbacks still exist in EBS. Because of

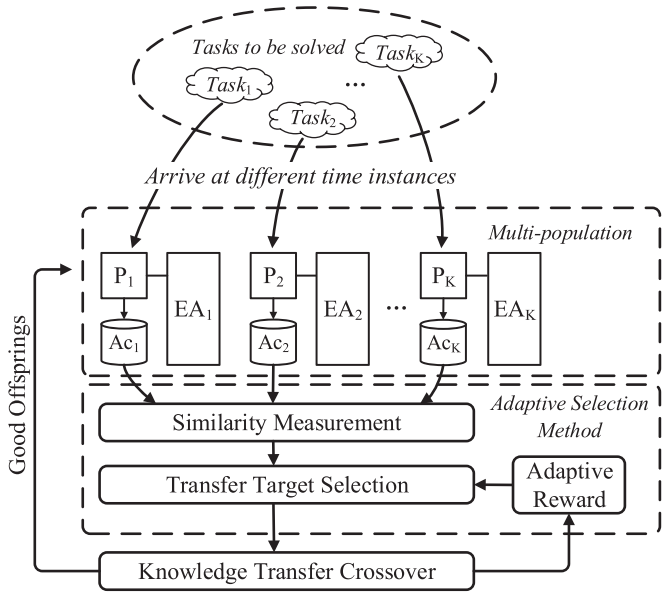


Fig. 1. The framework of the proposed MaTEA.

gathering information from all the other tasks, the knowledge transfer is "aimless", which is also a problem in most of the existing MTO algorithms. The "aimless" transfer can result in the waste of computational resources when only few tasks are helpful. On the other hand, transferring knowledge by roughly removing individuals from one task to another works only in the condition that two tasks share the similar optima. Situations in reality are often so complicated that tasks are only similar partially, in other words, an individual performs badly in one task cannot declare that it contains no constructive information for the task.

III. The Proposed Method

In this section,the proposed many-task evolutionary algorithm (MaTEA) is introduced. Based on the proposed framework, a particular realization based on differential evolution (DE), named MaTDE is presented.

A. The Framework: MaTEA

The proposed MaTEA uses multiple subpopulations to solve multiple tasks, with each subpopulation focusing on one specific task. During the evolution, a knowledge transfer rate (α) is utilized to control the probability of information exchange among tasks. When a knowledge transfer happens, an $assisted$ task (T_a) will be assigned to the $target$ task (T_t) by using an adaptive selection method which consists of two parts: *similarity measurement mechanism* and *adaptive reward strategy*. Once the $assisted$ task is determined, a *knowledge transfer crossover* is performed to deliver information from T_a to T_t.

Figure 1 presents the basic structure of MaTEA. In the proposed framework, each EA solver focuses on a single task, and the tasks can come or leave arbitrarily. A task can start to evolve or be terminated when some criteria are met (e.g., the maximum generation is reached). There are two modes for a solver to evolve a task: knowledge transfer or independent evolution. When the former way is chosen, an adaptive selection method is proposed to select a suitable transfer target for the target task. The adaptive selection method consists of two components, namely, *similarity measurement mechanism* and *adaptive reward strategy*. To better measure the similarity between tasks, every subpopulation is associated with an archive. Once the target is determined, *knowledge transfer crossover* is used to generate offsprings.

1) Similarity Measurement Mechanism: In MaTEA, the similarity between two tasks is considered as a factor to select *assisted* task for knowledge transfer. Generally, knowledge transfer between similar tasks should be encouraged. This is because similar tasks are more likely to contain common knowledge (e.g., similar optimal solutions) that can help the search . In our work, the similarity is measured by the KullbackLeibler divergence (KLD) [20]. A concise comprehension of KLD is how much information is lost when one probability distribution is replaced by another probability distribution. In another word, it indicates the differences of two distinct distributions numerically. A small KLD means that little information will be lost when replacing one distribution with another, indicating that the two distributions are similar.

When it comes to a population with NP individuals of D-dimensional in EAs, KLD can be regarded as probability distribution model that has NP samples in a D-dimensional space. As the distribution of a population can be arbitrary and unpredictable during the evolution process, it is difficult to define a highly accurate distribution model to fit the data. However, in many applications, most individuals in the population would usually converge to the best-so-far individuals. Bearing this characteristic in mind, we adopt multivariate Gaussian Distribution to approximately capture the distribution characteristics of the individuals. It should be noticed that for problems that contain multiple scattered optimal solutions, the multivariate Gaussian Distribution may not be effective enough to accurately capture the distributions of individuals. How to measure these problems remains a future research direction.

In MaTEA, rather than using the current populations directly, we develop archives to store more individuals for the tasks. For a given task T_i, an archive (Ac_i) is used to record not only the latest individuals of P_i, but also individuals in previous generations. At each generation, each individual of P_i has an *update rate* (UR) probability to be added to Ac_i. If the number of individuals reaches $archivesize$ (AcS), then a randomly-chosen individual in Ac_i will be replaced by the new individual. AcS is set to be larger than NP, so that the archive can contain sufficient information for similarity measurement. Based on the archives, the KLD between two tasks (two approximated multivariate Gaussian Distributions) can be calculated by:

$$KLD(Ac_0||Ac_1) = \frac{1}{2}\left(tr(\Sigma_1^{-1}\Sigma_0) + (\mu_1-\mu_0)^T\Sigma_1^{-1}(\mu_1-\mu_0) - D + ln\left(\frac{det\Sigma_1}{det\Sigma_0}\right)\right) \quad (2)$$

where Ac_0 and Ac_1 represents the archives of two arbitrary tasks, Σ represents the covariance matrix, μ represents the mean vector. D denotes the dimension of the covariance matrix, which is

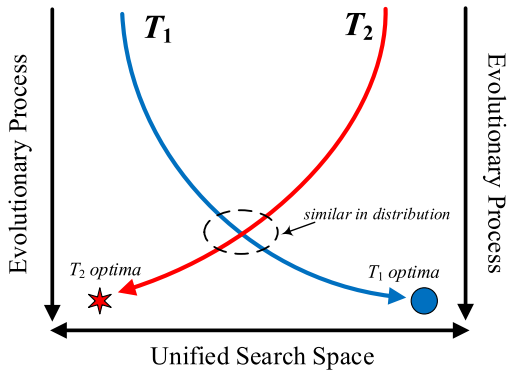


Fig. 2. Illustration of tasks with temporary similar distributions can have different evolutionary directions.

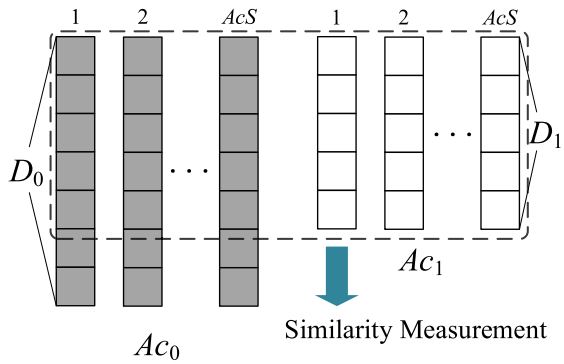


Fig. 3. Similarity measurement of two tasks with different dimensions.

calculated as $D = \min\{D_0, D_1\}$. If $D_0 > D_1$, then the first D_1 dimension of Ac_0 will be separated to measure the similarity with Ac_1, which is shown at Fig. 3. The calculation of KLD is asymmetric, which means $KLD(Ac_0||Ac_1) \neq KLD(Ac_1||Ac_0)$. To make the $similarity$ symmetric, we define the $similarity$ between two tasks by:

$$Sim(T_0, T_1) = \frac{1}{2}(KLD(Ac_0||Ac_1) + KLD(Ac_1||Ac_0)) \tag{3}$$

However, it should be noted that only considering similarity is not enough, because two tasks evolving in different directions may have similar population distributions. In this case, negative knowledge transfer between the two tasks may happen (see Fig. 2). Thus, we introduce the following adaptive reward strategy to address this issue.

2) Adaptive Reward Strategy: In MaTEA, the *adaptive reward strategy* is another factor to select the *assisted* task for knowledge transfer. Inspired by the idea of reinforcement learning [37] that an agent can make better choice in the environment guided by the accumulated reward gained through different trials, we develop a reward system in the proposed framework. For a given task T_i, there are reward values between itself and all other tasks, which can be denoted as R_i^j ($j = 1, 2, \ldots, K$ and $j \neq i$). R_i^j is initialized as 1 at the beginning and only updated when T_i is the $target$ task (T_t) and T_j is the $assisted$ task (T_a). If the best individual p_{best} of T_t is worse than an offspring (p') generated through knowledge transfer crossover, then R_t^a will be enlarged, otherwise, R_t^a will be reduced:

$$R_t^a = \begin{cases} R_t^a/\lambda & \text{if } p_{best} \text{ is worse than } p' \\ R_t^a \cdot \lambda & \text{otherwise} \end{cases} \tag{4}$$

where λ is the shrink rate within (0, 1).

A task is chosen to be T_a of T_t by considering both the task similarity and reward. When T_t performs knowledge transfer, all the other started or finished tasks can become candidate of T_a, the selection score of the jth task is calculated by:

$$score_t^j = \rho \cdot score_t^j + \frac{R_t^j}{log(Sim(T_t, T_j))} \tag{5}$$

where ρ is an attenuation coefficient and the $log(\cdot)$ operator is used to circumscribe the magnitudes of reward and similarity. At last, the T_a will be chosen by the *roulette wheel selection*, in which the jth task has the probability of:

$$probability_t^j = \frac{score_t^j}{\sum_{\text{for all i} \neq \text{t}} score_t^i} \tag{6}$$

3) Knowledge Transfer Crossover: When knowledge transfer is performed between T_t and the chosen T_a, information will be transferred through a crossover operator. As the crossover is used to transfer knowledge, therefore, it is called *knowledge transfer crossover*. Different forms of crossover can be designed to combine information of two individuals. In this paper, we adopt a simple uniform crossover. Specifically, for every individual p_t^i of T_t, a randomly selected individual p_a of T_a is used to deliver knowledge, and a *crossover rate* (CR_{KTC}) is used to control the probability of crossover in each dimension. The procedure can be expressed by:

$$v^i(j) = \begin{cases} p_a(j) & \text{if } rand(0,1) < CR_{KTC} \text{ or } j = k \\ p_t^i(j) & \text{otherwise} \end{cases} \tag{7}$$

where v^i represents the offspring for next generation generated through crossover from p_t^i, $j = 1, 2, \ldots, D$ denotes the dimension index and D is the dimension number of the unified search space, k is a random integer in $[1, D_t]$ to ensure that at least one dimension of p_t^i is replaced and D_t is the problem-specific dimension number for T_t.

It should be noticed that the *knowledge transfer crossover* is performed on the unified search space. Though there are cases that dimension number of T_t (D_t) is less than the dimension number of Y (D), the extra dimensions in T_t should perform the crossover. This is because though these genes contribute nothing to the $phenotype$ of T_t, they exist to carry $implicit$ information of T_t. In the MaTO manner, individuals do not serve their own tasks only, but are also associated with other tasks through information exchange. When proceeding EA in one population, the extra genes are affected by those available, so that they also contain information of their tasks. Therefore, a crossover on the unified search space is necessary for MaTO.

B. The Proposed MaTDE Algorithm

In this section, we integrate MaTEA with DE [27] to form a many-task algorithm named MaTDE. The procedure of MaTDE

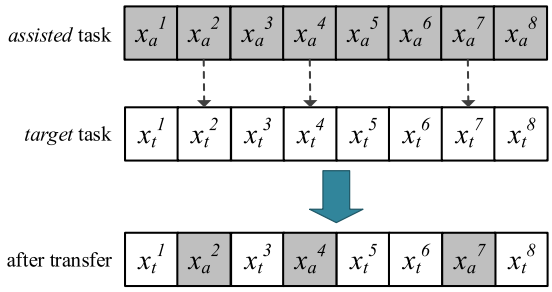


Fig. 4. A case of *knowledge transfer crossover*.

is presented in Alg. 2. In the following parts, the procedure of MaTDE is described in a step-by-step manner.

Step 1-Initialization: Tasks can arrive and start to evolve at different time instances. Suppose there are K' new tasks joining at the moment. In this step, K' new subpopulations are generated randomly with each containing NP individuals. The K' tasks are then merged with the old started tasks to evolve simultaneously. The length of the chromosomes is set to be a predefined large-enough number, so that it is capable of handling tasks of different dimension. For the K' new tasks, the values of the genes are generated randomly within the unified space $[0, 1]^D$. Together with the K' new subpopulations, K' new archives are generated. To guarantee that the archives is no smaller than the subpopulations, the archives are initialized to be a copy of the corresponding subpopulation:

$$Ac_i \leftarrow P_i, \quad i = 1, 2, \ldots, K' \tag{8}$$

Last, we denote that after adding K' new tasks, the total task number is K.

Step 2-New Individual Creation: In this step, NP new offsprings are generated. For each of the K subpopulations, if its termination criteria are not met , it will create new offsprings for the next generation. There are two ways for the creation, which are determined by the knowledge transfer rate (α). If a randomly generated number within [0, 1] is less than α, the task will perform knowledge transfer with other tasks by the adaptive method in MaTEA, which is presented previously. Otherwise, the subpopulation will generate offsprings independently through DE.

When a task chooses to perform knowledge transfer, *similarity measurement mechanism* and *adaptive reward strategy* are integrated to choose suitable transfer target. Specifically, MaTDE maintains two tables. The first table stores the accumulated reward information. Denote that R_i^j represents the reward of choosing T_j to assist T_i. It is updated through Eq. 4, only when T_j is chosen as the transfer target for T_i. Another table stores the scores of choosing transfer target. We use the tables to calculate probability, denoting the probability for T_i to choose T_j as transfer target. Each time when T_i is ready to choose a transfer target, similarity between the ith archive and the other archives are calculated. Then, by comprehensively combining the reward table and the similarity information, the score table is updated through Eq. 5. Based on the new probabilities, MaTDE uses *roulette wheel selection* to choose transfer target. It should be noticed that, an active task can choose an inactive task (tasks whose termination criteria are met) as transfer target. In other words, an inactive task's information (i.e., subpopulation and archive) can be stored in the system until all the tasks are finished. Last, each individual in the target task will generate an offspring through *knowledge transfer crossover*, so that NP

Algorithm 2: The Pseudocode of MaTDE.

Input: NP, the subpopulation size; P_i, the current population of the ith task; α, the *knowledge transfer rate*; Ac_i, the archive for T_i; λ, the *shrink rate* of reward; $probability_i^j$ and R_i^j, the probability and reward for T_j to be the $assisted$ task of T_i.
Output: The best solutions of all tasks.

```
Initialize the parameter settings.
repeat
    Integrate started tasks to form executing-task ({ET}).
    for every newly started tasks T_i do
        Generate NP individuals randomly to form P_i.
        Initialize the corresponding archive (Eq. 8).
        Evaluate the origin subpopulation P_i.
    end for
    for T_i ∈ {ET} do
        if ending criterion of T_i is not met then
            Generate a random number r within [0, 1].
            if r > α or only one task in {ET} then
                /*Evolve T_i independently through DE. */
                Generate NP mutant vectors (Eq. 9).
                Crossover to create NP offsprings (Eq.10).
                Update all individuals in P_i through Eq. 11.
            else
                /*Evolve T_i through knowledge transfer. */
                for T_j ∈ {ET} − {T_i} do
                    Update probability_i^j using Eq. 5.
                end for
                p_best ← the best individual in P_i.
                T_a ← roulette-selection ({ET} − {T_i}).
                Perform knowledge transfer crossover to generate NP offsprings.
                Update all individuals in P_i through Eq. 11.
                if p_best is worse than an offspring then
                    R_i^a ← R_i^a / λ.
                else
                    R_i^a ← R_i^a * λ.
                end if
            end if
            Update Ac_i.
        end if
    end for
until termination criterion is met.
```

new offsprings are constructed by combining information of the target task and the chosen $assisted$ task.

On the other hand, when a task chooses to evolve independently, it adopts DE as solver. In MaTDE, for the given ith individual of a subpopulation, DE will first perform mutation operator. A modified DE mutation schema of DE/rand/1 is utilized to generate a mutant vector, which adopts the target vector as base and the difference between target vector and a randomly chosen vector as differential variation:

$$\vec{w^i} = \vec{p^i} + F * (\vec{p^r} - \vec{p^i}) \tag{9}$$

where $\vec{w^i}$ represents the ith mutant vector, F is the scalar factor, r is an index generated randomly within $[1, NP]$. The differential variation $(\vec{p^r} - \vec{p^i})$ reflects the differences between the two individuals. Then, F is used to further control the amplification of the variation. Next, a new offspring is generated according to a crossover operator by combining both the mutation vector and the origin parent:

$$v^i(j) = \begin{cases} w^i(j) & \text{if } rand(0,1) < CR \text{ or } j = k \\ p^i(j) & \text{otherwise} \end{cases} \tag{10}$$

where j is the index of gene in the chromosome and k is a randomly generated index to ensure at least one gene is replaced in v^i (the offspring is different from its parent) and CR is the crossover rate. In this study, the settings of F and CR are set randomly within the predefined ranges.

Step 3-Selection: When all individuals finish generating their offsprings, the offspring-parent comparison is used to select the proper individuals for the next generation. The better one between offspring and its parent will be selected to form the population for the next generation, and the other one will be eliminated:

$$p^i_{G+1} = \begin{cases} v^i & \text{if } f(v^i) < f(p^i_G) \\ p^i_G & \text{otherwise} \end{cases} \tag{11}$$

where G and $G+1$ denote the current and the next generation, respectively.

If the offsprings are generated through knowledge transfer, an extra procedure is executed. The reward table is updated according to the quality of the new offsprings by Eq. 4.

Step 4-Archive Update: After generating the new subpopulations, archives are updated. The update rate (UR) controls the probability that an individual is added to its corresponding archive. If the archive is full, then a randomly-chosen old individual in the archive will be replaced. Specifically, for every individual p^j_i from P_i, Ac_i can be updated by:

$$Ac_i = \begin{cases} Ac_i \cup \{p^j_i\} & \text{if } |Ac_i| < ACS \\ & \text{and } rand(0,1) < UR \\ (Ac_i - \{a^r_i\}) \cup \{p^j_i\} & \text{if } |Ac_i| = ACS \\ & \text{and } rand(0,1) < UR \end{cases} \quad j = 1, 2, \ldots, NP \tag{12}$$

where a^r_i denotes a randomly chosen individual from Ac_i. The archive size (ACS) is set to be larger than NP, so that the archive can store sufficient information for a task to perform similarity measurement.

There is a loop from *Step 1* to *Step 4* (*Step 1* is performed only when new tasks arrive), until all the tasks are terminated.

IV. Experiment on Single-Objective Problems

In this section, the efficiency of MaTEA is empirically investigated by comparing the performance of MaTDE with MFEA, EBS-DE (an implementation of EBS with a DE). The component analysis of MaTEA is also conducted to evaluate the effectiveness of each component in MaTDE.

A. Experimental Settings

The test suite for MaTO is generated by the following benchmark functions for single-objective continuous optimization.

a) *Sphere:*

$$F(\boldsymbol{z}) = \sum_{i=1}^{D} z_i^2 \tag{13}$$

b) *Weierstrass:*

$$\begin{aligned} F(\boldsymbol{z}) = & \sum_{i=1}^{D} \left(\sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k (z_i + 0.5))] \right) \\ & - D \sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k * 0.5)]; \\ & a = 0.5, b = 3, k_{\max} = 20 \end{aligned} \tag{14}$$

c) *Rosenbrock:*

$$F(\boldsymbol{z}) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2); \tag{15}$$

d) *Ackley:*

$$\begin{aligned} F(\boldsymbol{z}) = & -20exp\left(-0.2\sqrt{\frac{1}{D}\sum\nolimits_{i=1}^{D} z_i^2}\right) \\ & - exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i)\right) + 20 + e \end{aligned} \tag{16}$$

e) *Schwefel:*

$$F(\boldsymbol{z}) = 418.9829 \times D - \sum_{i=1}^{D} z_i \sin(|z_i|^{\frac{1}{2}}) \tag{17}$$

f) *Griewank:*

$$F(\boldsymbol{z}) = 1 + \frac{1}{4000}\sum_{i=1}^{D} z_i^2 - \prod_{i=1}^{D} \cos\left(\frac{z_i}{\sqrt{i}}\right) \tag{18}$$

g) *Rastrgin:*

$$F(\boldsymbol{z}) = \sum_{i=1}^{D} (z_i^2 - 10\cos(2\pi z_i) + 10) \tag{19}$$

These problems can be classified into two categories, namely easy task (E-task) and complicated task (C-task). For the C-tasks, an EA may quickly get trapped into local optima and

TABLE I
DETAILS OF THE TEST SUITE FOR SINGLE-OBJECTIVE OPTIMIZATION

Task	Function	Shift Vector(O)	Category	Ideal Assisted Task
T_2	$Sphere_1$	$(o_1, o_2, ..., o_D) = (0, 0, ..., 0)$ $\mathbf{x} \in [-100, 100]^D, D = 50$	E-task	None
T_2	$Sphere_2$	$(o_1, o_2, ..., o_D) = (80, 80, ..., 80)$ $\mathbf{x} \in [-100, 100]^D, D = 50$	E-task	None
T_3	$Sphere_3$	$(o_1, o_2, ..., o_D) = (-80, -80, ..., -80)$ $\mathbf{x} \in [-100, 100]^D, D = 50$	E-task	None
T_4	$Weierstrass_{25D}$	$(o_1, o_2, ..., o_D) = (-0.4, -0.4, ..., -0.4)$ $\mathbf{x} \in [-0.5, 0.5]^D, D = 25$	E-task	None
T_5	$Rosenbrock$	$(o_1, o_2, ..., o_D) = (0, 0, ..., 0)$ $\mathbf{x} \in [-50, 50]^D, D = 50$	C-task	T_1
T_6	$Ackley$	$(o_1, o_2, ..., o_D) = (40, 40, ..., 40)$ $\mathbf{x} \in [-50, 50]^D, D = 50$	C-task	T_2
T_7	$Weierstrass_{50D}$	$(o_1, o_2, ..., o_D) = (-0.4, -0.4, ..., -0.4)$ $\mathbf{x} \in [-0.5, 0.5]^D, D = 50$	C-task	T_3, T_4
T_8	$Schwefel$	$(o_1, o_2, ..., o_D) = (420.9687, 420.9687, ..., 420.9687)$ $\mathbf{x} \in [-500, 500]^D, D = 50$	C-task	None
T_9	$Griewank$	$(o_1, o_2, ..., o_{\frac{D}{2}}) = (-80, -80, ..., -80)$ $(o_{\frac{D}{2}+1}, o_2, ..., o_D) = (80, 80, ..., 80)$ $\mathbf{x} \in [-100, 100]^D, D = 50$	C-task	T_4
T_{10}	$Rastrgin$	$(o_1, o_2, ..., o_{\frac{D}{2}}) = (40, 40, ..., 40)$ $(o_{\frac{D}{2}+1}, o_2, ..., o_D) = (-40, -40, ..., -40)$ $\mathbf{x} \in [-50, 50]^D, D = 50$	C-task	None

commonly requires a large number of iterations to find a satisfying solution. On the other hand, for the E-tasks, an EA usually can locate the global optima rapidly. In our experiment, some C-tasks are assigned with ideal $assisted$ E-tasks. In this case, the optima of these C-tasks and E-tasks are similar in the unified space, so that C-tasks can obtain positive knowledge transfer to accelerate their search process if the corresponding E-tasks are chosen. More details of the test suite are shown in Table I. Specifically, four E-tasks, a $25D$ $Weierstrass$ function and three $50D$ $Sphere$ functions which are shifted to three different optima, are assumed as ideal *assisted* tasks of different C-tasks. For instance, T_1 is an E-task, whose optima is similar with T_5's ($(0.5, 0.5, \ldots, 0.5)^{50}$ in the unified space), implying that T_5 can obtain helpful improvement of the performance when choosing T_1 as its $assisted$ task, which makes T_1 to be T_5's ideal *assisted* task. Notice that, as T_4 is a $25D$ problem, we label it as the ideal *assisted* task of the C-tasks whose front $25D$ of the optima are similar with T_4 (T_7 and T_9). In the test suite, for a given C-task, most tasks are negative tasks whose optima are dissimilar, and only a few tasks can help to guide the evolution. Also, there are C-tasks with no ideal *assisted* task (T_8 and T_{10}), which means there are no E-tasks similar to those C-tasks in optima distribution. A shifting operation is performed on the solutions, which means $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{O}$, where $\boldsymbol{x}$ is the solution in the search space and $\boldsymbol{O}$ is the shifting vector.

The experiment simulates the complicated situation where various kinds of tasks exist. Our objective is to investigate whether the proposed algorithm can select the ideal tasks and perform positive knowledge transfer to improve the search efficiency. To better ascertain the effectiveness of the MaTO algorithms, a single-task DE (STDE) is used as the baseline algorithm.

TABLE II
PARAMETER SETTINGS FOR MATDE

Parameter	Value
population size (NP)	100
reward shrink rate(λ)	0.8
attenuation coefficient (ρ)	0.8
knowledge transfer rate (α)	0.1
archive update rate (UR)	0.2
archive size (AcS)	300
crossover rate (CR_{KTC}) and (CR)	$rand(0.1, 0.9)$
scalar factor (F)	$rand(0.1, 2)$

The parameter settings for MaTDE are listed in Table II. To guarantee the validity, the DE settings and the population size of EBS-DE and STDE are the same as MaTDE, with the offspring size the same as population size at each generation. The population size of MFEA is set to be $NP * K = 1,000$ and other settings follows the literature [14]. In the experiment, all the ten tasks are started simultaneously. Each algorithm can evolve for 1, 000 generations, with the same total population size and number of newly generated offsprings at each generation. The experiment is conducted for 30 independent runs with different random seeds.

TABLE III
THE PERFORMANCE OF MATDE, MFEA, EBS-DE AND SINGLE-TASK DE (STDE) ON TEN TASKS FOR 30 INDEPENDENT RUNS ($-$, $\approx$ AND $+$ DENOTE THE CORRESPONDING ALGORITHM IS SIGNIFICANTLY WORSE THAN, SIMILAR AND BETTER THAN MATDE ON WILCOXON SIGNED-RANK TEST AT $\alpha = 0.05$)

Task	MaTDE	MFEA	EBS-DE	STDE
T_1	**0**	1.32E+00 $(-)$	2.20E-05 $(-)$	**0** $(\approx)$
T_2	**0**	1.27E+00 $(-)$	5.10E-05 $(\approx)$	1.00E-06 $(\approx)$
T_3	**0**	1.17E+00 $(-)$	7.64E-04 $(-)$	**0** $(\approx)$
T_4	**0**	3.37E+00 $(-)$	7.55E-04 $(-)$	3.27E-02 $(-)$
T_5	**2.50E-05**	8.15E+02 $(-)$	5.80E-04 $(-)$	1.81E+02 $(-)$
T_6	**2.70E-04**	1.99E+01 $(-)$	5.70E-04 $(\approx)$	1.99E+01 $(-)$
T_7	**4.65E-04**	1.05E+01 $(-)$	1.65E-02 $(-)$	6.46E-01 $(-)$
T_8	**1.39E-03**	2.34E+03 $(-)$	1.30E+03 $(-)$	2.85E+01 $(-)$
T_9	**6.62E-03**	4.11E-02 $(-)$	6.73E-02 $(-)$	7.32E-03 $(-)$
T_{10}	8.82E+01	**1.55E+01** $(+)$	4.80E+02 $(-)$	1.28E+02 $(-)$

B. Results and Discussion

The mean value of the final optimized results for MaTDE, MFEA, EBS-DE and STDE in 30 independent runs are listed in Table III, where symbols $-$, $\approx$ and $+$ denote the corresponding algorithm is significantly worse than, similar and better than MaTDE on Wilcoxon signed-rank test at $\alpha = 0.05$. The best results are highlighted in bold font.

It can be observed from the two tables that MaTDE has distinct advantage over the other algorithms. Generally, EBS-DE performs better than the multi-task algorithm MFEA in solving MaTO problems, but is only competitive to MaTDE on two tasks (T_2 and T_6). The main reason is that the many-task environment with ten tasks is much more complicated than multi-task with only two or three tasks. It can be discovered from the settings of the test suite that only a small number of tasks can provide useful information to help solve a specific C-task. If the knowledge transfer is aimless, it is likely negative. Negative transfer slows the evolution process, which violates the initial idea of multi-task optimization that using information from other tasks to help solve current task.

MFEA exposes its weakness when faced with complicated many-task situation, for its aimless and negative knowledge transfer among individuals. From the results, we can find out that MFEA has bad performance when even solving E-tasks. As for the C-tasks with ideal *assisted* tasks, compared with MaTDE, MFEA fails to excavate useful knowledge from other tasks to locate the optima. MFEA is only competitive to MaTDE in C-tasks with no ideal *assisted* tasks, where the adaptive strategy makes no sense for the lack of suitable transfer target.

Thus a conclusion can be drawn that MFEA is not suitable for MaTO problems. When it comes to EBS-DE, its performance is generally better than that of MFEA and STDE, which indicates that it is a more desired MaTO algorithm. This is because EBS-DE uses the adaptive strategy based on the feedback from the transfer. If a task can benefit from knowledge transfer, it can have larger probability to perform knowledge transfer, otherwise, the probability may be reduced. However, EBS-DE takes all tasks equally, which means it cannot select the most suitable ideal *assisted* task. When the number of tasks is large, this method is inefficient.

It is shown that MaTDE is more suitable for MaTO problems, compared with MFEA and EBS-DE. Compared with EBS-DE, MaTDE is able to maintain the prominent performance of STDE in solving E-tasks. MaTDE adopts an adaptive strategy based on distribution similarity and accumulated reward to choose transfer target for assistance. Contrary to the other two algorithms, it avoids the aimless information exchange by encouraging positive transfer and preventing negative transfer. What's more, using multiple subpopulations to evolve many-task problems can improve the exploitation capability of a task, which can enhance the accuracy of the solution. To further explain the results, Fig. 5 shows the frequency of choosing *assisted* tasks for the six C-tasks. It can be observed that the MaTDE can always select the best E-tasks to assist C-tasks (i.e., the correct E-tasks are assigned with much higher selection frequency). When there is no proper ideal *assisted* task for a C-task, all tasks will have the same opportunities to perform knowledge transfer, like T_8 and T_{10}. No task can effectively promote the optimization of T_8 and T_{10} through knowledge transfer, because the optima of the two tasks are unique to others. Thus, the adaptive strategy does not have a preference on a certain $assisted$ task. Furthermore, for the situation that E-tasks can only "partially" help C-tasks (the dimension of E-tasks is less than the dimension of C-task), like T_9 and T_7, MaTDE can also choose the correct ideal *assisted* tasks.

Apart from the transfer frequencies of different tasks, the average successful transfer times are summed. A successful transfer is defined as the frequency of the reward increase, implying the best individual of a task is updated through knowledge transfer, which is represented in Eq. 4. The results are shown in Table IV. Generally, from Table IV, we can observe that it coincides with the results depicted in Fig. 5. C-tasks choose and benefit from their corresponding ideal *assisted* E-tasks most. However, results also show that the $25D$ E-task T_4 improves the C-tasks the least, compared with other three E-tasks. This is because the *knowledge transfer crossover* in MaTDE uses T_4's rear $25D$ elements, which are not optimized in the objective function, to generate new offsprings. In this case, the performance of the new individual may be affected. This phenomenon could result in that the frequency of choosing T_4 to be *assisted* task is not so prominent as the other E-tasks in Fig 5. In spite of this, the front $25D$ optima of T_4 and T_9 are similar, which makes T_4 to be $T9$'s *assisted* task more frequently compared with other tasks, due to the similarity measurement. What's more, not only E-tasks can boost C-tasks, E-tasks can benefit from its corresponding C-tasks through knowledge transfer. MaTDE encourage knowledge transfer to happen between populations similar in distribution, from which both populations have higher probability to improve the accuracy of the solutions.

Besides the overview, a scrutiny into T_6 as an example is presented in Fig. 7. The 1, 000 generations are divided into ten intervals, each point plotted denotes the accumulated successful transfer times up to the corresponding generations. It is obvious that with the help of the adaptive strategy, T_6 chooses T_2 as *assisted* task the most frequently. The increasing trend of the red curve grows as the process of evolution due to the convergence of T_2, which may provide more and more valuable knowledge. Successful transfers incurred by other tasks only

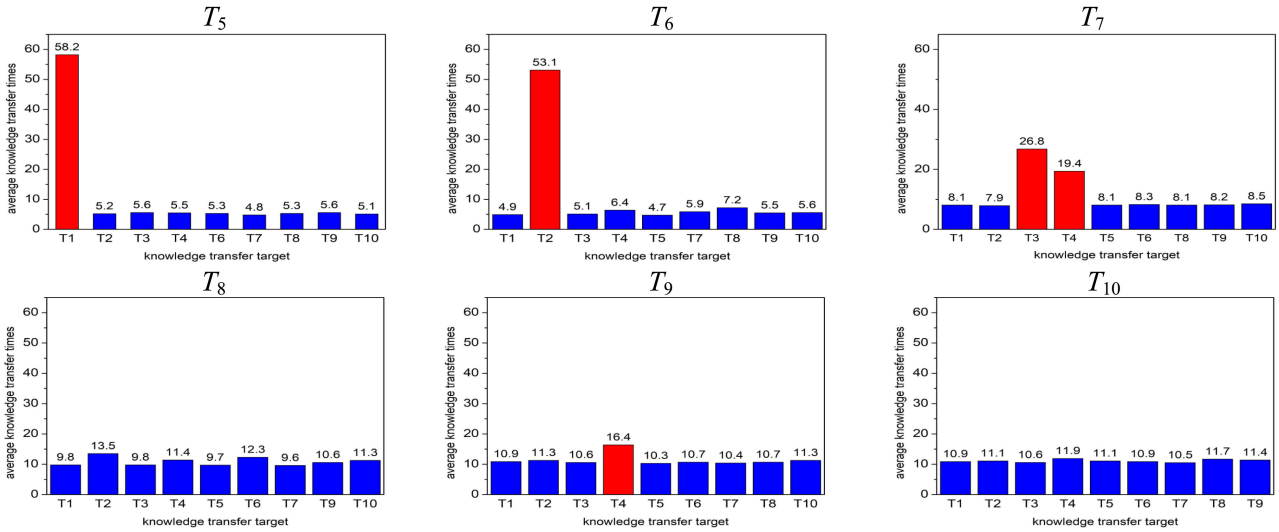


Fig. 5. The average times of choosing knowledge transfer target for C-tasks, the $assisted$ task is highlighted in red.

TABLE IV
SUCCESSFUL TRANSFER FREQUENCIES OF EACH TASK

		Knowledge Transfer Target									
		T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}
Target Task	T_1	*	0.0000	0.0000	0.1000	2.1667	0.0000	0.0000	0.0333	0.0000	0.0000
	T_2	0.0333	*	0.0000	0.0000	0.0000	18.9000	0.0333	0.1333	0.0667	0.0000
	T_3	0.0333	0.0000	*	0.2667	0.0000	0.0000	50.2333	0.0333	0.0000	0.0667
	T_4	0.0333	0.0000	0.0667	*	0.0000	0.1000	0.3000	0.0000	0.0000	0.0000
	T_5	37.5667	0.0333	0.0000	0.0000	*	0.0000	0.0000	0.0000	0.0667	0.0333
	T_6	0.2333	35.3667	0.1333	0.5000	0.1000	*	0.2667	0.6667	0.3333	0.0667
	T_7	0.0000	0.0333	7.2000	1.3333	0.0333	0.0333	*	0.0000	0.0000	0.0333
	T_8	0.0333	1.6667	0.2000	0.1000	0.0333	0.4667	0.0000	*	0.0000	0.0336
	T_9	0.0333	0.3333	0.0667	0.4000	0.1000	0.0999	0.3000	0.1333	*	0.0000
	T_{10}	0.0000	0.2999	0.4333	0.0000	0.0667	0.1999	0.0999	0.0000	0.0333	*

occur at the early stage of the evolution, where distributions of the solutions are arbitrary. The above results demonstrate that the proposed adaptive strategy is effective to improve the search performance.

Lastly, Fig. 6 presents the convergence trends of the three algorithms (i.e., MaTDE, EBS-DE and MFEA) on the ten tasks. MFEA performs the worst among the three algorithms. As for EBS-DE, the way it transfers knowledge results in some problems. The core idea of EBS-DE is to share offsprings from different populations. This exchanging method needs a perfectly-match of the optima of two tasks. If solutions from the E-task can help to solve the C-task, taking them as new individuals for the C-task can promote the search very rapidly in a short time, like T_6. However, exchanging information in a non-crossover way suffers from a problem of high similarity among individuals in one population. The high similarity results in the prematurity of a population because of the reduction of population diversity, which limits the global search capability of the algorithm, like T_8 and T_{10}. Therefore, from the trends, we can find that using crossover to conduct knowledge transfer is an efficient way to exchange information.

By adopting an adaptive method to choose transfer target and an efficient crossover to perform knowledge transfer, MaTDE can guarantee the accuracy and efficiency when solving MaTO problems. The experiment results demonstrate that MaTEA is a promising framework for many-task optimization.

C. Component Analysis

To verify the effectiveness of components in MaTEA, experiments are conducted on three simplified algorithms of MaTDE, which are: MaTDE/AS (replace the adaptive strategy by a random choice strategy), MaTDE/S (MaTDE without similarity measurement) and MaTDE/R (MaTDE without adaptive reward strategy). The experiments use the same test suite in Table I, and the components that removed from Eq. (5) are replaced by constant 1. The results are listed in Table V.

It can be observed from Table V that in all the cases, MaTDE is significantly better or similar with the other three simplified algorithms. Specifically, from T_5 to T_7, these three tasks who have explicit ideal $assisted$ tasks in the test suite, MaTDE can have better performance. This indicates that every component

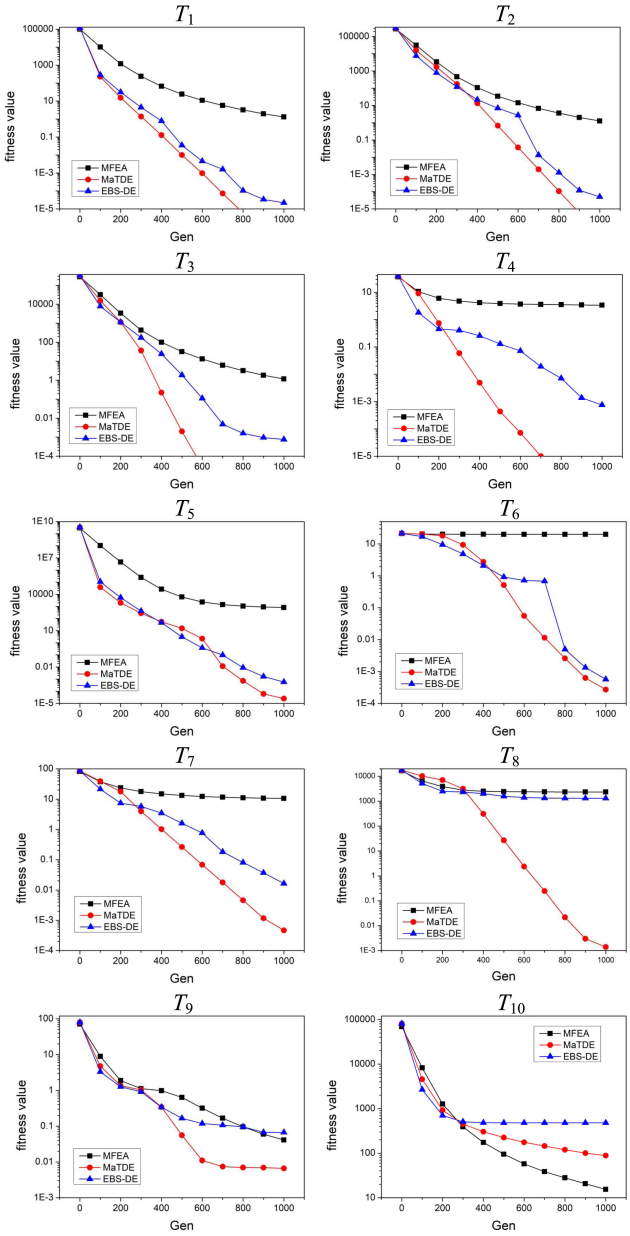


Fig. 6. The convergence trends of the three algorithms on single-objective optimization problems.

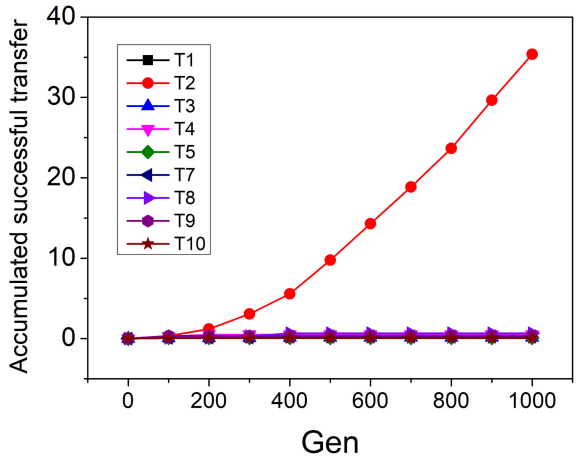


Fig. 7. The accumulated successful transfer of T_6.

TABLE V
THE PERFORMANCE OF THE COMPONENT-VERIFYING EXPERIMENTS ON TEN TASKS FOR 30 INDEPENDENT RUNS ($-$, $\approx$ AND $+$ DENOTE THE CORRESPONDING ALGORITHM IS SIGNIFICANTLY WORSE THAN, SIMILAR AND BETTER THAN MATDE ON WILCOXON SIGNED-RANK TEST AT $\alpha = 0.05$)

Task	MaTDE	MaTDE/AS	MaTDE/S	MaTDE/R
T_1	0	0 ($\approx$)	0 ($\approx$)	0($\approx$)
T_2	**0**	3.00E-06 ($-$)	1.00E-06 ($-$)	1.00E-06($-$)
T_3	**0**	**0** ($\approx$)	**0** ($\approx$)	**0**($\approx$)
T_4	**0**	**0** ($\approx$)	1.00E-06 ($\approx$)	**0**($\approx$)
T_5	**2.50E-05**	4.95E-02 ($-$)	**2.50E-05** ($\approx$)	4.45E($-$)
T_6	**2.70E-04**	6.47E-04 ($-$)	4.01E-04 ($-$)	3.75E-04($-$)
T_7	**4.65E-04**	1.88E-03 ($-$)	5.70E-04 ($-$)	5.84E-04($-$)
T_8	1.39E-03	4.13E-03 ($\approx$)	1.33E-03 ($\approx$)	**8.24E-04**($\approx$)
T_9	6.62E-03	**3.03E-03** ($\approx$)	5.66E-03 ($\approx$)	6.01E-03($\approx$)
T_{10}	8.82E+01	8.74E+01 ($\approx$)	**8.61E+01** ($\approx$)	9.18E+01($\approx$)

TABLE VI
THE α ANALYSIS EXPERIMENTS ON TEN TASKS FOR 30 INDEPENDENT RUNS (THE BEST RESULTS ESTIMATED UNDER WILCOXON SIGNED-RANK TEST AT $\alpha = 0.05$ ARE HIGHLIGHTED IN BOLD FORMAT)

Task	$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$	$\alpha = 0.7$
T_1	**0**	5.00E-06	2.88E-04	1.26E-02
T_2	**0**	**0**	**0**	7.14E-01
T_3	**0**	**0**	**0**	**0**
T_4	**0**	**0**	5.00E-06	**0**
T_5	**2.50E-05**	4.34E-04	1.93E-05	5.90E-01
T_6	2.70E-04	1.31E-04	**4.40E-05**	7.06E-01
T_7	4.65E-04	3.31E-04	4.90E-05	**0**
T_8	**1.39E-03**	6.51E-02	7.90E+00	3.92E+02
T_9	**6.62E-03**	4.66E-03	1.77E-02	7.39E-01
T_{10}	**8.82E+01**	1.11E+02	1.91E+02	3.65E+02

of the adaptive strategy contributes to selecting transfer target with high accuracy. Although in some cases, the mean values of MaTDE are worse, they are similar in a statistic analysis perspective. To this end, we can conclude that with the help of the adaptive strategy, MaTEA is an efficient framework for many-task optimization.

D. Parametric Analysis

As many parameters are listed in Table II, we typically analyse the knowledge transfer rate α and the reward shrink rate λ, which have significant influence on the proposed framework.

The first parameter is the knowledge transfer rate α, which controls the frequency of knowledge transfer. We set α as 0.1, 0.3, 0.5, 0.7 in MaTDE algorithm, with other settings kept the same as the previous experiments. The general results are presented in Table VI. From the table, we can observe that with the increase of α, the optimization performance of the algorithm decreases, especially on E-tasks and C-tasks without ideal *assisted* tasks. However, if an E-task can finally locate the optima, its corresponding C-tasks can be promoted effectively. For instance, T_3 can locate its optima in the four cases, therefore, with

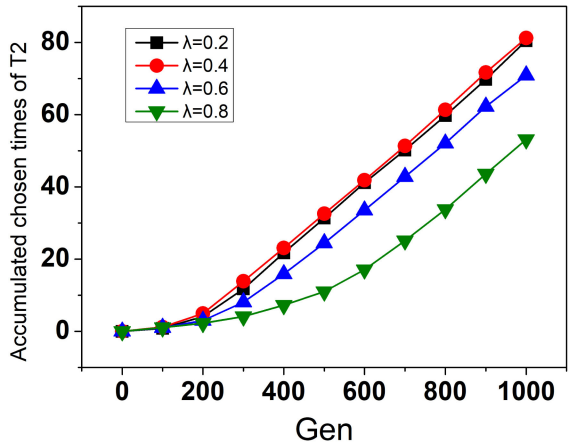


Fig. 8. Accumulated times of T_2 being chosen to be T_6's *assisted* task.

a higher possibility to perform knowledge transfer, there is an ascending trend of the performance of the algorithm. Nevertheless, the performance of the algorithm is highly dependent on the performance of E-tasks, which cannot be guaranteed when a large α is used. Thus, in accordance to the experiment results, a small α, representing a more balanced performance in both searching and transferring properties, is recommended.

Another important parameter is λ, the shrink rate of the reward. Generally, a smaller λ means that the framework is sensitive to the reward update. Fig. 8 depicts the average accumulated times of T_2 being chosen to be T_6's *assisted* task when various λ s are used. It can be observed that as λ decreases, the frequency of T_6 to choose the ideal *assisted* task T_2 increases. This is because the update of the reward is more significant. If there is a successful transfer, the same transfer target is much likely to be chosen again when next knowledge transfer happens. In the case of $T6$, when the α are 0.2 and 0.4, due to the high sensitive reward, T_2 becomes the only choice of T_6's *assisted* task. Using a small λ can be sensible, if there are ideal *assisted* tasks that can provide useful transfer all the time. However, the condition is difficult to be assured as the unknown evolutionary process. Therefore, a small λ may result in the situation of losing the opportunity to discover more potential *assisted* tasks, since the reward of a certain task is too large, compared with the others. Based on the discussion above, a relatively large λ is recommended, for both being capable of choosing the ideal *assisted* task and maintaining the probability of choosing other potential contributive tasks.

E. Time Complexity

Besides the capability of finding solutions of high quality, time complexity is a significant issue for an optimization algorithm. In MaTEA, the time complexity of calculating the similarity between tasks is $O(D^3)$. To investigate how dimensionality affects the executive time of MaTEA, we compare MaTDE with MFEA and EBS-DE under the problems of the same scalar. Specifically, we use the ten tasks in Table I, setting their dimension number as 50, 100, 150 and 200. The experiments are conducted under the same experimental environment, and the average consumption time to finish the optimization tasks in a run of each algorithm is recorded.

TABLE VII
TIME CONSUMPTION OF DIFFERENT ALGORITHMS UNDER DIFFERENT DIMENSION NUMBERS (UNIT: S)

Dimension number	MaTDE	MFEA	EBS-DE
50	97.98	427.72	53.58
100	437.34	448.21	83.36
150	1266.66	464.78	111.55
200	2720.28	531.04	140.79

The results are presented in Table VII. It can be observed from the results that among the three algorithms, EBS-DE runs the fast. When comparing with MFEA, MaTDE costs less time than it when the dimension number is 50, and similar to it when the dimension number increases to 100. With the dimension number keeps growing, it takes more time for MaTDE to complete the tasks, compared the other algorithms.

It should be noticed that the experiments are conducted based on tasks of cheap evaluation cost which only needs simple numerical computation. In other words, the $O(D^3)$ extra numerical computational cost due to the similarity measurement in MaTDE weighs high in contrast to the time that evaluation process costs. However, the time complexity for calculating similarity is not dependent on the fitness evaluation. Thus, for complicated applications where the fitness evaluation is expensive such as those involve simulation, the extra time cost for similarity calculation will not significantly influence the total executive time. As the adaptive strategy used in MaTEA framework can promote positive transfers to find high-quality solution using less evaluation times, it could reduce the total executive time in the long run.

F. Another Implementation: MaTGA

To investigate the generality of the MaTEA framework, we further integrate it with Genetic Algorithm (GA). Comparisons are conducted among the other two previous methods-MFEA and EBS framework. For the demand of fairness, SBX crossover and Gaussian mutation operators in MFEA are employed to generate new offsprings in MaTGA, EBS-GA, as well as a baseline single-task GA (STGA) using the same parameters. The three algorithms are run independently for 30 times, with a total generations of 1, 000 each time adopting different random seeds.

The results have been shown in Table VIII. Generally, the two frameworks designed for MaTO problem have better performance than MFEA in nine of the ten tasks in the test suite. When GA is utilized as baseline method, compared with STGA, the performance on T_{10} has a deterioration in the three multi-task algorithms, which indicates that the optimization of C-task without ideal *assisted* tasks will be impaired by inescapable negative transfer. However, as for T_8, who has no ideal *assisted* task either, the performances of MaTGA and EBS-GA are improved. Thus, whether a C-task without ideal *assisted* task will be better optimized in the multi-task environment is task-dependent, nevertheless, a well designed transfer strategy takes an important role.

TABLE VIII
THE PERFORMANCE OF THE GA-BASED EXPERIMENTS ON TEN TASKS FOR 30 INDEPENDENT RUNS ($-$, $\approx$ AND $+$ DENOTE THE CORRESPONDING ALGORITHM IS SIGNIFICANTLY WORSE THAN, SIMILAR AND BETTER THAN MATGA ON WILCOXON SIGNED-RANK TEST AT $\alpha = 0.05$)

Task	MaTGA	MFEA	EBS-GA	STGA
T_1	**2.45E-04**	1.31E+00 ($-$)	5.58E-04 ($-$)	6.26E-04 ($\approx$)
T_2	4.75E-04	1.27E+00 ($-$)	**6.10E-05** ($+$)	2.32E-03 ($-$)
T_3	**0**	1.17E+00 ($-$)	2.00E-06 ($-$)	8.90E-05 ($-$)
T_4	**1.00E-06**	3.37E+00 ($-$)	1.60E-05 ($\approx$)	2.86E-04 ($-$)
T_5	**2.16E-02**	8.15E+02 ($-$)	3.29E-02 ($-$)	9.16E+01 ($-$)
T_6	3.61E-03	1.99E+01 ($-$)	**8.59E-04** ($+$)	1.93E+01 ($-$)
T_7	**5.57E-04**	1.05E+01 ($-$)	1.60E-03 ($-$)	1.62E+00 ($-$)
T_8	**3.40E-02**	2.34E+03 ($-$)	4.73E-02 ($\approx$)	2.55E+03 ($-$)
T_9	**4.52E-03**	4.11E-02 ($-$)	4.58E-03 ($\approx$)	9.04E-03 ($\approx$)
T_{10}	1.57E+01	1.54E+01 ($\approx$)	3.41E+01 ($-$)	9.04E+00($+$)

In general, when compared with EBS-GA, MaTGA has better performance. Both frameworks use adaptive strategies to control knowledge transfer, which are verified useful contrasting to MFEA. Different from EBS-GA, MaTGA excavates the relationships among tasks during the evolutionary process, which makes the knowledge transfer more concentrative, helping avoid negative transfer more effectively. The transfer with distinct target makes MaTGA outperforms EBS-GA in most of the C-task optimizations.

V. EXPERIMENT ON MULTI-OBJECTIVE AND DYNAMIC CONTROL PROBLEMS

To validate the generality of the proposed framework, in this section, further experiment is conducted on MaTO for multi-objective optimization problems (MOOPs). Furthermore, the dynamic control of MaTO is utilized and its influence of MaTO is discussed.

A. Experimental Settings

We slightly extended the MaTDE for multi-objective optimization by using a selection operator based on *nondominated relation* and *crowding distance* from the non-dominated sorting genetic algorithm II (NSGA-II). Specifically, if an offspring $dominates$ its parent, it will replace its parent directly, otherwise, if they are in a $non-dominated$ relation, the offspring will be added to a temporary list and will perform survival selection with the whole population eventually based on NSGA-II. To verify the performance, comparisons with a multi-objective EBS-DE [23] with NSGA-II selection strategy and the popular multi-objective factorial EA (MO-MFEA) [16] are conducted. The parameter settings for MaTDE and EBS-DE are the same as mentioned previously (Table II) and the settings of MO-MFEA are configured according to [16].

A four-task problem is used as test instance. Two multi-objective problems integrated with Ackley and Griewank function based on $ZDT4$ [22] are adopted, which are named $ZDT4-A$ and $ZDT4-G$, respectively. Moreover, rotation and shift operations are imposed on the two problems to form the test instances. The details of the tasks are listed in Table IX.

TABLE IX
DETAILS OF THE TEST SUITE FOR MULTI-OBJECTIVE OPTIMIZATION

Task	Extent of Design Space	Properties
ZDT4-G	$x_1 \in [0,1], D = 50$ $x_i \in [-100,100](i = 2,...,D)$ $o_i = 0(i = 1,...,D-1)$	Multimodal
ZDT4-GR	$x_1 \in [0,1], D = 50$ $x_i \in [-100,100](i = 2,...,D)$ $o_i = 0(i = 1,...,D-1)$	Multimodal + Rotated
ZDT4-GRS	$x_1 \in [0,1], D = 50$ $x_i \in [-100,100](i = 2,...,D)$ $o_i = 50(i = 1,...,D-1)$	Multimodal + Rotated + Shifted
ZDT4-AR	$x_1 \in [0,1], D = 50$ $x_i \in [-100,100](i = 2,...,D)$ $o_i = 0(i = 1,...,D-1)$	Multimodal + Rotated

TABLE X
THE AVERAGE IGD OF MATDE, MFEA AND EBS-DE ON FOUR MULTI-OBJECTIVE TASKS FOR 30 INDEPENDENT RUNS ($-$, $\approx$ AND $+$ DENOTE THE CORRESPONDING ALGORITHM IS SIGNIFICANTLY WORSE THAN, SIMILAR AND BETTER THAN MATDE ON WILCOXON SIGNED-RANK TEST AT $\alpha = 0.05$)

Task	MaTDE	MO-MFEA	EBS-DE
ZDT4-G	**1.78E-04**	3.93E-03 ($-$)	2.28E-03 ($-$)
ZDT4-GR	**2.87E-03**	1.29E-02 ($-$)	7.37E-03 ($-$)
ZDT4-GRS	**8.40E-02**	1.45E-01 ($-$)	9.45E-02 ($\approx$)
ZDT4-AR	**3.45E-02**	1.11E-01 ($-$)	6.97E-02 ($-$)

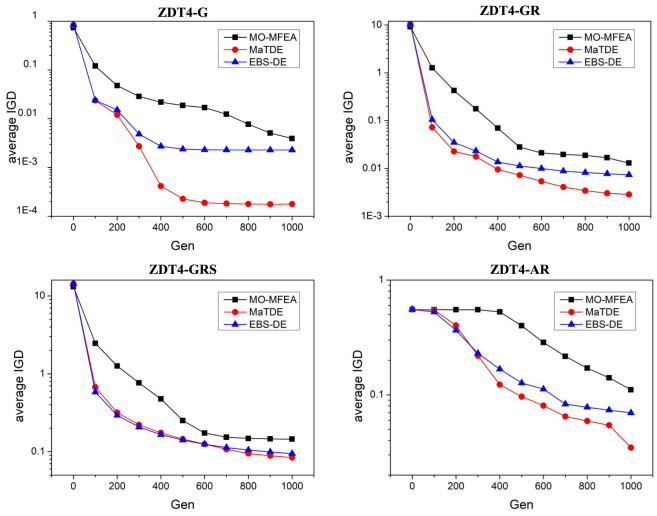


Fig. 9. The IGD convergence trends of the three algorithms.

The variable vector $\mathbf{z}$ is generated by:

$$z_1 = x_1$$
$$(z_2, z_3, \ldots, z_D) = \mathbf{M} \cdot ((x_2, x_3, \ldots, x_D) - (o_1, o_2, \ldots, o_{D-1}))^T \quad (20)$$

where $\mathbf{M}$ is a $(D-1) \times (D-1)$ randomly generated rotation metrix and $\mathbf{O}$ is a shift vector. When there are no rotation or shift operations to the design space, then $\mathbf{M}$ will become an identity matrix and $\mathbf{O}$ will be a zero vector.

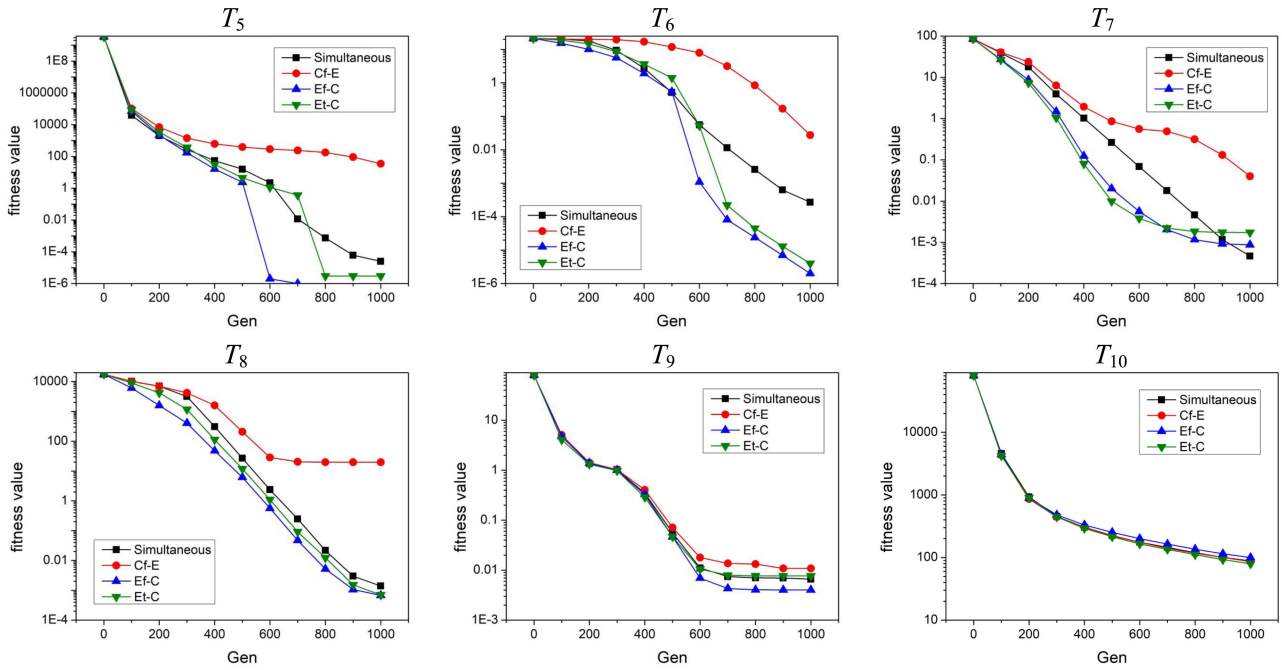


Fig. 10. The fitness convergence trends of different control methods.

The problems are run for 30 independent times with 1, 000 generations each time. As the true Pareto-front (PF) is known beforehand, we use *inverted generational distance* (IGD) [38] to estimate the performance of the algorithms. The IGD calculate the Euclidean distance between the true PF and the approximate PF obtained by the algorithm. A smaller IGD indicates a better approximate PF in both diversity and convergence.

Furthermore, dynamic control of task order can be utilized in the MaTEA framework, and we intend to figure out how the order of solving different tasks influences the optimization performance. We conduct the experiment on the test suite in last section (Table III). Instead of running the tasks at the same time, we start the tasks at different time instances of the evolution according to their categories (C-task and E-task). All the tasks can evolve for 1, 000 generations. If a task is terminated while there are still tasks evolving, it will then remain as an $assisted$ task. The experiment simulates three different situations, C-tasks start first and E-tasks join halfway (E-tasks start to evolve at the 500th generation of C-task) ($Cf-E$), E-tasks start first and C-tasks join halfway ($Ef-C$) and C-tasks join when E-tasks are terminated ($Et-C$). As E-tasks can always converge to global optimum independently, in this experiment, we focus on the performance of the C-tasks. Our objective is to examine whether the proposed algorithm can handle MaTO problems effectively when tasks arrive at different time instances and explore how the sequence influence the performance.

B. Results and Discussion

1) Many-Task for MOOPs: The results of the MaTO experiment for MOOP are listed in Table X. MO-MFEA, the algorithm for multi-task MOOPs obtains the worst performance among the three algorithms, which again demonstrates that it is not suitable for MaTO problems. The EBS-DE performs better than MO-MFEA on all tasks, but it performs significantly worse than MaTDE on three of the four tasks. MaTDE can search for ideal PF for MOOPs, for it balance the weight between individual evolution and co-evolution, together with adaptive strategy to perform implicit knowledge transfer.

The convergence trends of IGD for the four tasks are shown in Fig. 9. From the results, we can observe that MaTDE outperforms the other two algorithms in both convergence speed and solution accuracy. These results indicates the strong search capability of the proposed algorithm for MaTO problems.

2) MaTO With Dynamic Control: To examine how different dynamic control methods influence the algorithm performance, three methods ($Cf-E$, $Ef-C$ and $Et-C$) as well as the results of $simultaneous$ method (all tasks start simultaneously), are used for comparison. In dynamic control schema, every task has its own counter to judge whether the task should be started or terminated. We draw the convergence trends of these four methods in Fig. 10. Notice that, all tasks are evolved for 1, 000 generations, while the same generations of different tasks do not mean the same evolutionary phases (i.e., the first generation of C-tasks begin at the 500th generation of E-tasks in $Ef-C$).

Some interesting phenomena can be observed from the curves. First, for C-tasks with ideal $assisted$ tasks, most of them need an advancing start of E-tasks. Among the tasks, $Cf-E$ performs the worst significantly, especially in the cases that C-task has a complete ideal $assisted$ E-task (i.e., T_5, T_6 and T_7). This is because the advancing evolution of E-tasks can convey information of high quality at the early phase of C-tasks. Second, $Ef-C$ has better performance in most cases, rather than $Et-C$. Two reasons may account: a) Knowledge transfer is performed in

the form of crossover. In some cases, a solution generated by combining individuals from a terminated task and an active task may not better than those generated from two active tasks. It is easy to realize that the variable values of this kind of solutions differ very much, because some of them are from a task being optimized and others are from an optimized task. Solutions of this kind cannot survive in some cases, even their parents have similar optima (problems that variables of different dimensions are related, i.e., $Rosenbrock$ function); b) The differences of distributions between optimizing tasks and optimized tasks are usually greater than two optimizing tasks. This may result in the enlargement of KLD values and the reduction of probability of choosing correct $assisted$ E-task. The two reasons demonstrate that $Ef - C$ dynamic control method may be a wise strategy. Third, the results of C-tasks with no ideal $assisted$ E-task (T_8 and T_{10}) are contrary. The results of T_{10} are in high consistency no matter what control strategy is adopted, however, T_8 can somehow benefit from E-tasks though it has no ideal $assisted$ E-tasks. The latter result shows that knowledge transfer can help solve a task jump out of local optima, even the global optima of the two tasks are different.

VI. Conclusions

In this paper, we proposed a general evolutionary framework (named MaTEA) for many-task optimization. The core idea of MaTEA is to use subpopulations to solve different tasks, together with adaptive strategy to choose target task to perform knowledge transfer. The adaptive control of choosing $assisted$ tasks is based on a *similarity measurement mechanism* and an *adaptive reward strategy*. The former is used to dynamically measure the similarity between different tasks, while the latter concentrates on analyzing the effectiveness of knowledge transfer between tasks. Archives are developed to store sufficient information of tasks, so that the similarities are more reliable. Crossover is used to transfer knowledge in the framework, which is named as knowledge transfer crossover. The proposed framework is implemented based on DE, forming an MaTDE algorithm to test the effectiveness of the proposed framework. Experimental results have shown that the proposed MaTDE can outperform the state-of-the-art multi-task EAs on both single-objective and multi-objective optimization problems. In addition, the proposed MaTEA allows dynamic control of tasks, which make it applicable to applications where tasks are arriving at different time instances.

There are several future research directions. One direction is to improve the accuracy and efficiency of the similarity measurement. In addition, as our work focuses on numerical optimization problems, applying the proposed framework to other types of problems such as regression and classification problems can be another promising research topic.

References

[1] T. Back, U. Hammel, and H.-P. Schwefel, "Evolutionary computation: Comments on the history and current state," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 3–17, Apr. 1997.

[2] D. B. Fogel, "An introduction to simulated evolutionary optimization," *IEEE Trans. Neural Netw.*, vol. 5, no. 1, pp. 3–14, Jan. 1994.

[3] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments —A survey," *IEEE Trans. Evol. Comput.*, vol. 9, no. 3, pp. 303–317, Jun. 2005.

[4] T. T. Nguyen, S. Yang, and J. Branke, "Evolutionary dynamic optimization: A survey of the state of the art," *Swarm Evol. Comput.*, vol. 6, pp. 1–24, 2012.

[5] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," Comput. Eng. Netw. Lab. (TIK), Swiss Federal Inst. Technol., Zurich, Switzerland, Tech. Rep. 103, 2001.

[6] H. A. Abbass, R. Sarker, and C. Newton, "PDE: A Pareto-frontier differential evolution approach for multi-objective optimization problems," in *Proc. IEEE Congr. Evol. Comput.*, vol. 2, 2001, pp. 971–978.

[7] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[8] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.

[9] K. Deb, "Multi-objective optimization," in *Search Methodologies*. New York, NY, USA: Springer, 2014, pp. 403–449.

[10] J. S. Angelo, E. Krempser, and H. J. Barbosa, "Differential evolution for bilevel programming," in *Proc. IEEE Congr. Evol. Comput.*, 2013, pp. 470–477.

[11] Y. Yuan, Y.-S. Ong, A. Gupta, P. S. Tan, and H. Xu, "Evolutionary multitasking in permutation-based combinatorial optimization problems: Realization with TSP, QAP, LOP, and JSP," in *Proc. IEEE Region 10 Conf.*, 2016, pp. 3157–3164.

[12] R. Chen, K. Li, and X. Yao, "Dynamic multiobjectives optimization with a changing number of objectives," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 157–171, Feb. 2018.

[13] B. Liu *et al.*, "Analog circuit optimization system based on hybrid evolutionary algorithms," *Integration, VLSI J.*, vol. 42, no. 2, pp. 137–148, 2009.

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

[15] A. Gupta, J. Mańdziuk, and Y.-S. Ong, "Evolutionary multitasking in bi-level optimization," *Complex Intell. Syst.*, vol. 1, no. 1–4, pp. 83–95, 2015.

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

[17] L. Feng *et al.*, "An empirical study of multifactorial PSO and multifactorial DE," in *Proc. IEEE Congr. Evol. Comput.*, 2017, pp. 921–928.

[18] J. Zhong, L. Feng, W. Cai, and Y.-S. Ong, "Multifactorial genetic programming for symbolic regression problems," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published.

[19] J. Ding, C. Yang, Y. Jin, and T. Chai, "Generalized multi-tasking for evolutionary optimization of expensive problems," *IEEE Trans. Evol. Comput.*, vol. 23, no. 1, pp. 44–58, Feb. 2019.

[20] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Statist.*, vol. 22, no. 1, pp. 79–86, 1951.

[21] B. Da *et al.*, "Evolutionary multitasking for single-objective continuous optimization: Benchmark problems, performance metric, and baseline results," 2017, *arXiv preprint arXiv:1706.03470*.

[22] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evol. Comput.*, vol. 8, no. 2, pp. 173–195, 2000.

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

[24] A. Gupta, Y.-S. 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.

[25] Y.-S. Ong and A. Gupta, "Evolutionary multitasking: A computer science view of cognitive multitasking," *Cogn. Comput.*, vol. 8, no. 2, pp. 125–142, 2016.

[26] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Mach. Human Sci.*, 1995, pp. 39–43.

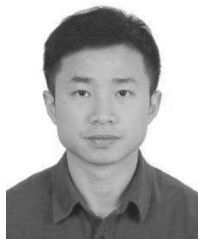
[27] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

[28] Y.-W. Wen and C.-K. Ting, "Parting ways and reallocating resources in evolutionary multitasking," in *Proc. IEEE Congr. Evol. Comput.*, 2017, pp. 2404–2411.

[29] L. Zhou, L. Feng, J. Zhong, Y.-S. Ong, Z. Zhu, and E. Sha, "Evolutionary multitasking in combinatorial search spaces: A case study in capacitated vehicle routing problem," in *Proc. IEEE Symp. Series Comput. Intell.*, 2016, pp. 1–8.
[30] L. Feng, Y.-S. Ong, S. Jiang, and A. Gupta, "Autoencoding evolutionary search with learning across heterogeneous problems," *IEEE Trans. Evol. Comput.*, vol. 21, no. 5, pp. 760–772, Oct. 2017.
[31] L. Feng *et al.*, "Evolutionary multitasking via explicit autoencoding," *IEEE Trans. Cybern.*, to be published.
[32] A. T. W. Min, Y.-S. Ong, A. Gupta, and C.-K. Goh, "Multi-problem surrogates: Transfer evolutionary multiobjective optimization of computationally expensive problems," *IEEE Trans. Evol. Comput.*, vol. 23, no. 1, pp. 15–28, Feb. 2019.
[33] Y. Chen, J. Zhong, and M. Tan, "A fast memetic multi-objective differential evolution for multi-tasking optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2018, pp. 1–8.
[34] D. Liu, S. Huang, and J. Zhong, "Surrogate-assisted multi-tasking memetic algorithm," in *Proc. IEEE Congr. Evol. Comput.*, 2018, pp. 1–8.
[35] M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," in *Proc. Int. Conf. Parallel Problem Solving Nature*, Springer, 1994, pp. 249–257.
[36] Q. Chen, X. Ma, Z. Zhu, and Y. Sun, "Evolutionary multi-tasking single-objective optimization based on cooperative co-evolutionary memetic algorithm," in *Proc. 13th Int. Conf. Comput. Intell. Secur.*, 2017, pp. 197–201.
[37] R. S. Sutton *et al.*, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.
[38] S. Jiang, Y.-S. Ong, J. Zhang, and L. Feng, "Consistencies and contradictions of performance metrics in multiobjective optimization," *IEEE Trans. Cybern.*, vol. 44, no. 12, pp. 2391–2404, Dec. 2014.

Yongliang Chen is currently working toward the bachelor's degree with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China. His current research interests include evolutionary computation, swarm intelligence, and multi-task optimization.

Jinghui Zhong received the Ph.D. degree from the School of Information Science and Technology, Sun Yat-sen University, Guangzhou, China, in 2012. He is currently an Associate Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. From 2013 to 2016, he was a Postdoctoral Research Fellow with the School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include evolutionary computation (e.g., genetic programming and differential evolution), machine learning, and agent-based modeling.



Liang Feng received the Ph.D. degree from the School of Computer Engineering, Nanyang Technological University, Singapore, in 2014. He was a Postdoctoral Research Fellow with the Computational Intelligence Graduate Laboratory, Nanyang Technological University, Singapore. He is currently an Assistant Professor with the College of Computer Science, Chongqing University, Chongqing, China. His research interests include computational and artificial intelligence, memetic computing, big data optimization and learning, as well as transfer learning.

Jun Zhang (F'17) received the Ph.D. degree from the City University of Hong Kong, Hong Kong, in 2002. He is currently a Visiting Scholar with Victoria University, Melbourne, VIC, Australia. His current research interests include computational intelligence, cloud computing, high performance computing, data mining, wireless sensor networks, operations research, and power electronic circuits. He has authored or coauthored more than 150 technical papers (90 IEEE Transactions papers) in the above areas. He was a recipient of the China National Funds for Distinguished Young Scientists from the National Natural Science Foundation 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 for the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON CYBERNETICS, and the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS.