Differential Evolutionary Multi-task Optimization

Xiaolong Zheng*†, Yu Lei*†, A. K. Qin‡, Deyun Zhou*, Jiao Shi*†, Maoguo Gong§
*School of Electronics and Information, Northwestern Polytechnical University, Xi'an 710072, China
†Research & Development Institute of Northwestern Polytechnical University in Shenzhen
Northwestern Polytechnical University in Shenzhen, Shenzhen 518057, China
‡Department of Computer Science and Software Engineering, Swinburne University of Technology, VIC 3122, Australia
§Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education of China
Xidian University, Xi'an 710071, China
Email: xlzheng@mail.nwpu.edu.cn, leiy@nwpu.edu.cn, kqin@swin.edu.au,
dyzhounpu@nwpu.edu.cn, jiaoshi@nwpu.edu.cn, gong@ieee.org

***Abstract*—Evolutionary multi-task optimization (EMTO) studies on how to simultaneously solve multiple optimization problems, so-called component problems, via evolutionary algorithms, which has drawn much attention in the field of evolutionary computation. Knowledge transfer across multiple optimization problems (being solved) is the key to make EMTO to outperform traditional optimization paradigms. In this work, we propose a simple and effective knowledge transfer strategy which utilizes the best solution found so far for one problem to assist in solving the other problems during the optimization process. This strategy is based on random replacement. It does not introduce extra computational cost in terms of objective function evaluations for solving each component problem. However, it helps to improve optimization effectiveness and efficiency, compared to solving each component problem in a standalone way. This light-weight knowledge transfer strategy is implemented via differential evolution within a multi-population based EMTO paradigm, leading to a differential evolutionary multi-task optimization (DEMTO) algorithm. Experiments are conducted on the CEC'2017 competition test bed to compare the proposed DEMTO algorithm with five state-of-the-art EMTO algorithms, which demonstrate the superiority of DEMTO.**

I. Introduction

Evolutionary multi-task optimization (EMTO) [1] is a newly emerging research area in the field of optimization, which studies on how to effectively and efficiently solve multiple optimization problems, so-called component problems, at the same time by using evolutionary algorithms [2] as the optimiser. It is not uncommon that there exist many real-world problems which possess the high degree of similarity, i.e., they are highly relevant. EMTO works by assuming that relevant optimization problems possess certain commonality in their optimal solutions which can be exploited and utilised via transfer to help their optimization processes. Specifically, by transferring the promising candidate solutions found in the process of solving one optimisation problem (so-called knowledge) into the optimization processes of relevant problems, the performance of solving the relevant (optimization) problems may get improved. For example, finding the global optimum of one optimization problem makes the other optimization problems to immediately get resolved if these problems have the same global optimum (even though they may have different fitness landscapes).

Ever since the pioneer work done by Gupta *et al.* [1], EMTO has attracted the interest of many researchers in the field of optimization. Some of the researchers devote their efforts into designing and developing more efficient EMTO solvers, such as the works on adapting knowledge transfer to task relatedness [3]–[6] and the works on building more general frameworks [7], [8]. Many of the researchers studied the feasibility and effectiveness of applying EMTO to solve real-world problems, such as the applications to power systems [9], software testing [10], cloud service composition [11] and ensemble classification [12]. The results of these works have demonstrated the promising future of EMTO as the next-generation mainstream optimization paradigm.

Most of the existing EMTO works employed a single population for solving multiple optimization problems. However, by doing so, the optimization processes for solving different problems may overly influence each other, where the impacts of undesirable negative knowledge transfer might cancel out the benefits of useful positive knowledge transfer, leading to performance degradation. In this work, we explore an EMTO paradigm which employs a unique population for solving each component problem of a multi-task optimization (MTO) problem, where across-task knowledge transfer is enabled via a simple yet effective random replacement based strategy. We implement this paradigm by using differential evolution as the optimizer, and propose a differential evolutionary multi-task optimization (DEMTO) algorithm. It is worth noting that utilization of multiple population is not a new idea in EMTO research [8], [13], [14]. The key difference of this work from the existing ones is the incorporation of a light-weight knowledge transfer strategy into the multi-population based EMTO paradigm.

In DEMTO, each component problem is solved by differential evolution that employs a unique population. Knowledge transfer occurs in a probabilistic manner, controlled by a manually specified parameter rmp, across the populations for different problems in each generation of the optimization process. When knowledge transfer happens in a certain generation, some of the newly generated mutant vectors in one problem are replaced by the best solution found so far till that generation in each of the other problems. Then, the crossover operation is applied to the updated set of mutant vectors to generate the trial vectors for evaluation. In this way, no extra computation cost in terms of objective function evaluations is introduced by this knowledge transfer strategy. Meanwhile, the problem-solving process with respect to each component problem maintains independence while benefiting from the probabilistic knowledge transfer in between. Experimental results on CEC'2017 competition test bed have demonstrated the superiority of the proposed DEMTO algorithm in comparison to five state-of-the-art EMTO algorithms.

The remaining paper is organized as follows. Section II will describe the background of this work. The proposed DEMTO algorithm will be detailed in Section III. Section IV will report and discuss experimental results. Finally, we will conclude the paper and mention some future work in Section V.

II. Background

A. *Evolutionary Multi-task Optimization*

Evolutionary algorithms are metaheuristic optimization methods based on the natural selection principle [2], [15]. This kind of algorithms starts from a group of individuals, and simulates the sexual reproduction and variation of the creatures, making the population more and more adaptable to the environment. Once the population stops evolving, the optimal solution of a optimization problem is obtained. Over the past years, evolutionary algorithms have been successfully applied to solve many scientific and practical engineering problems.

Evolutionary multi-task optimization (EMTO) makes full use of the population-based characteristic of evolutionary algorithms to simultaneously optimize multiple optimization problems. In particular, if there is k optimization problems (each is assumed to be minimization problem and treated as a task) to be optimized, the goal of EMTO is to simultaneously optimize these k optimization tasks and find the optimal solution for each of the k tasks, i.e., $\{\boldsymbol{x}_1^o, \boldsymbol{x}_2^o, ..., \boldsymbol{x}_k^o\} = argmin\{f_1(\boldsymbol{x}_1), f_2(\boldsymbol{x}_2), ..., f_k(\boldsymbol{x}_k)\}$, where $\boldsymbol{x}_j^o, \boldsymbol{x}_j \in \boldsymbol{X}_j, j = 1, 2, ..., k$ and $\boldsymbol{X}_j$ is a d_j-dimensional search space for the j^{th} task T_j as well as its objective function $f_j : \boldsymbol{X_j} \rightarrow \Re$.

To efficiently realize the simultaneous optimization of multiple tasks, multifactorial evolutionary algorithm (MFEA) was proposed by Gupta *et al.* [1]. In MFEA, a unified search space is built by employing a random key scheme to encode each dimension of all tasks' search spaces into the range of [0,1] [16], which can then be decoded via a linear mapping to tasks' original search spaces, especially for continue optimization tasks. To deal with multiple tasks, MFEA set up a set of properties as given as below:

Definition 1. (*Factorial Cost*): An individual p's factorial cost on task T_j is given by $\Psi_j = \lambda \cdot \delta_j + f_j$, where parameter λ is a penalizing multiplier, f_j and δ_j are the corresponding objective value and the total constraint violation on task T_j.

Definition 2. (*Factorial Rank*): An individual p's factorial rank on task T_j, denoted as r_j, is defined as the index of p in the list of population sorted in ascending order according to the factorial cost on task T_j.

Definition 3. (*Scalar Fitness*): An individual p's scalar fitness is given by $\varphi = 1/\min_{j\in\{1,..,k\}}\{r_j\}$.

Definition 4. (*Skill Factor*): An individual p's skill factor $\upsilon_p = argmin_j\{r_p^j\}, j = 1, 2, ..., k$, namely, the one task on which individual p possesses the strongest competence amongst all other tasks.

A skill factor indicates the task on which the individual demonstrates the strongest competence. Therefore, individuals with a same skill factor form a skill group for the corresponding task. As a result, each of the k tasks possesses a unique skill group. During the evolving process of population, MFEA tasks inspiration from the bio-cultural models of multifactorial inheritance which claims that the interactions of genetic and cultural factors are usually what results in the complex developmental features of offspring [17]. With this skill factor, different cultures can be represented as different skill groups. And this cultural aspect then works via the two features of multifactorial inheritance, i.e., assortative mating and vertical cultural transmission. The assortative mating allows the individuals with different skill factor to mate in MFEA. And the vertical cultural transmission allows their generated offspring to inherit one of their parents' skill factor. Ultimately, these cultural factors together with traditional genetic operations make MFEA to be a powerful EMTO solver as presented in [1].

Based on the work of MFEA, researchers in the optimization field have developed many EMTO algorithms. The multifactorial PSO and multifactorial DE have been developed by Feng *et al.* [18] based on the framework of MFEA. Different from these works, Cheng *et al.* [13] employed a coevolutionary multitasking framework, which puts that each optimization task should be resolved by a unique subpopulation and if a particle's personal best performance within a specific subpopulation stagnates over a user-defined number of generations, then information or knowledge that comes from the other subpopulations will be transferred across. Li *et al.* [8] directly developed a multipopulation evolutionary framework (MPEF), which optimized each of its tasks using one independent population. And knowledge transfer across tasks randomly occurs in a probability of parameter rmp which is self-adapting via some complex strategy. If knowledge transfer does occur, an individual within a particular population will generate its mutant vector through the defined mutation formulas which contains the information (or individual) from the other tasks populations. Feng *et al.* [14] solved each of tasks via a separated evolutionary solver after employing independent solution representations for these tasks and introducing a denoising autoencoder for conducting knowledge transfer.

B. *Differential Evolution*

Differential evolution (DE) [19]–[22] is one of the most well-known population-based optimization algorithms in the field of continuous optimization for its efficient and effective performance. Different from traditional evolutionary algorithms, DE generates new candidate solutions through the weighted difference of the current population. More specifically, DE generates mutant vectors for each population individual through the weighted difference of several randomly selected population individuals, which is called mutation operation. After mutation, crossover operation is performed on the original population and its corresponding mutant vectors. Next, greedy selection strategy kicks in to form a new population.

For mutation operation, there are five popular mutation strategies [20] in DE field as listed as below:

1) DE/rand/1:

$$\boldsymbol{v}_i^g = \boldsymbol{x}_{p1}^g + F \times (\boldsymbol{x}_{p2}^g - \boldsymbol{x}_{p3}^g) \tag{1}$$

2) DE/best/1:

$$\boldsymbol{v}_i^g = \boldsymbol{x}_{best}^g + F \times (\boldsymbol{x}_{p1}^g - \boldsymbol{x}_{p2}^g) \tag{2}$$

3) DE/rand-to-best/1:

$$\boldsymbol{v}_i^g = \boldsymbol{x}_i^g + F \times (\boldsymbol{x}_{best}^g - \boldsymbol{x}_i^g + \boldsymbol{x}_{p1}^g - \boldsymbol{x}_{p2}^g) \tag{3}$$

4) DE/best/2:

$$\boldsymbol{v}_i^g = \boldsymbol{x}_{best}^g + F \times (\boldsymbol{x}_{p1}^g - \boldsymbol{x}_{p2}^g + \boldsymbol{x}_{p3}^g - \boldsymbol{x}_{p4}^g) \tag{4}$$

5) DE/rand/2:

$$\boldsymbol{v}_i^g = \boldsymbol{x}_{p1}^g + F \times (\boldsymbol{x}_{p2}^g - \boldsymbol{x}_{p3}^g + \boldsymbol{x}_{p4}^g - \boldsymbol{x}_{p5}^g) \tag{5}$$

$\boldsymbol{v}_i^g$ is the mutant vector generated for the i-th population individual $\boldsymbol{x}_i^g$ at the g-th generation. $p1, p2, p3, p4, p5$ are the randomly generated indices within the range [1,n], where n is the size of population. Note that, these indices are mutually exclusive. And $\boldsymbol{x}_{best}^g$ is the best found solution for optimization problem at generation g. F is a positive scaling factor to scale the difference vector.

Crossover operation is performed on each population individual $\boldsymbol{x}_i^g$ and its corresponding mutant vector $\boldsymbol{v}_i^g$ to generate a trial vector:

$$\boldsymbol{u}_i^g = \begin{cases} \boldsymbol{v}_i^g & if(rand \leq CR) or(j = j_{rand}) \\ \boldsymbol{x}_i^g & otherwise \end{cases} \tag{6}$$

where the crossover rate CR controls the portion of offspring solutions that are copied from their corresponding mutant vectors and takes the values within the range [0,1). j_{rand} is randomly chosen integer from range [1,d] to make sure that crossover operation occurs in at least one dimension of trial vector, where d is the search space dimension of optimization problem.

$$\boldsymbol{x}_i^{g+1} = \begin{cases} \boldsymbol{u}_i^g & if \quad f(\boldsymbol{u}_i^g) \leq f(\boldsymbol{x}_i^g) \\ \boldsymbol{x}_i^g & otherwise \end{cases} \tag{7}$$

Before selection operation, all of the above generated trial vectors are evaluated on the fitness function corresponding to optimization problem. Let $f(\boldsymbol{x}_i^g)$, $f(\boldsymbol{u}_i^g)$ be the fitness of individual $\boldsymbol{x}_i^g$ and its corresponding trial vector $\boldsymbol{u}_i^g$. Subsequently DE selects the best one between $\boldsymbol{x}_i^g$ and $\boldsymbol{u}_i^g$ to enter the new formed population, which can be shown as equation (7).

III. The Proposed Method

In this section, we propose a differential evolutionary multitask optimization (DEMTO) algorithm characterized by its multiple populations structure and its ingenious knowledge transfer strategy. In DEMTO, differential evolution algorithm is employed to evolve each of these populations. During the period of generating offspring, knowledge transfer strategy kicks in to finish the information transmission across optimization tasks. The pseudo-code of DEMTO is described in **Algorithm 1**.

Algorithm 1 DEMTO

```
Input:
    $rmp$ (the probability of knowledge transfer)
    $k$ (the number of tasks)
    $n$ (DE's population size for each task)
Output:
    $\{\boldsymbol{x}_1^*, \boldsymbol{x}_2^*, ..., \boldsymbol{x}_k^*\}$ (the best found solution for each of the $k$ tasks)
1:  Initialize population $pop_j$ with $n$ individuals for task $T_j, j = 1, 2, ..., k$.
2:  Evaluate each individual in $pop_j$ on task $T_j$.
3:  while (stopping criterion is not satisfied) do
4:    for task $j = 1$ to $k$ do
5:      Generate $n$ mutant vectors w.r.t. $T_j$ by applying the mutation operation (Eq. 1) to $pop_j$.
6:      if $rand < rmp$ then
7:        Randomly replace $k - 1$ newly generated mutant vectors with the best solution found so far in each of the other $k - 1$ tasks.
8:      end if
9:      Apply the crossover operation (Eq. 6) to the updated set of $n$ mutant vectors to generate $n$ trail vectors.
10:     Evaluate the generated $n$ trail vectors on task $T_j$.
11:     Update $pop_j$ via selection between the old population and the newly generated (and evaluated) set of $n$ trail vectors as per Eq. 7.
12:     Update the best solution found so far for task $T_j$.
13:   end for
14: end while
```

Considering the optimization of k tasks, DEMTO optimizes each of these tasks via an independent population with size of n. The random key scheme used in MFEA is employed here to encode the search spaces of all tasks into a unified search space. Accordingly, each of these k population is randomly initialized within the unified search space with n individuals at the beginning of DEMTO. Subsequently, these populations will independently undergo mutation, crossover, evaluation and selection operations with respect to their corresponding tasks. Note that, traditional differential evolution algorithm is employed in DEMTO and mutation is performed using strategy in equation (1). With this multiple populations structure, each of the tasks can be independently optimized without the disturbances from the optimization process of the rest tasks, as each of the populations can independently maintain the diversity and elites of its own.

Knowledge transfer in DEMTO contains two steps, i.e., the moment to transfer knowledge across tasks and the specific form of transferring knowledge. For simplicity, DEMTO allows each of the k tasks to receive the information or knowledge from the rest $k-1$ tasks in a probability of rmp, where rmp is a manually-set constant. Once the knowledge transfer does happen, the so far best found solutions on these $k-1$ tasks will be transfer to the task and randomly replace its $k-1$ individuals as described in step 7 in **Algorithm 1**. By randomly replacing some unevaluated offspring, DEMTO can effectively and simply achieve the goal of transferring knowledge across tasks without adding additional computational costs. After the subsequent crossover and evaluation operation, if the transferred individuals (or some kind of knowledge) is not suitable to one task, the selection operation in step 11 in **Algorithm 1** will soon eliminate them from the task's population (due to the employed greedy selection strategy in DE), thereby reducing the negative effect of knowledge transfer on this task. Note that, the transferred individuals is suitable to one task or not can be hard to measure during an optimization process, therefore we simply compare these transferred individuals with corresponding parent individuals as did in the employed greedy selection strategy to decide whether the transferred individuals should be discarded.

IV. Experiments

In this section, a set of experiments are conduced to demonstrate the performance of the proposed DEMTO algorithm in comparison with the original MFEA [1], the MFEARR [5], the MFPSO [18], the MFDE [18] and the MPEF-SHADE [8]. Meanwhile, an in-depth analysis is made to verify the effectiveness of knowledge transfer in DEMTO via the comparison with its single task version DEMTO_STO, following with a parameter study for DEMTO. Before the experiments, an introduction to test problems and experiment environments is presented.

A. Test Problems

The test problems considered in this paper come from the CEC'2017 Evolutionary Multi-Task Optimization Competition, which are summarized in Table I. In this problem set, there are 9 test problems in total, each contains two component tasks (denoted as T_1 and T_2). In each of these test problems, the two distinct single-objective optimization tasks may possess different dimension size, global optima and search space range as shown in the table. Meanwhile, these 9 problems can be classified into 3 categories according to the overlap degree of optima in the unified space, i.e., complete intersection (CI), partial intersection(PI), no intersection(NI). CI contains the problem 1-3, whose components possess the same global optima as shown in Table I. PI contains the problem 4-6 and NI contains the problem 7-9. Besides, the 3 problems in each of these categories differ from each other in the inter-task similarity which measures the landscape similarity among the component optimization tasks of each problem via the Spearman's rank correlation coefficient. The details can be seen in [23].

B. Experimental Setup

To show the superiority of the proposed algorithm, we conduct experiments to (1) compare the proposed DEMTO with the original MFEA, the MFEARR, the MFPSO, the MFDE and the MPEF-SHADE in terms of the performance on the 9 problems from Table I, (2) analyze the effectiveness of knowledge transfer in DEMTO via the performance on the 9 problems in Table I and (3) study the impact of parameter settings in DEMTO.

The parameter configurations for the proposed DEMTO algorithm and the other algorithms considered in this study are listed in the following. Note that, the employed parameter settings for all algorithms except the DEMTO are as suggested in the original literature for fair comparison. And the crossover rate and the scaling factor in DEMTO are set according to Zhang *et al.* [21].

1) Parameter settings in DEMTO:
    - population size of each task n: 50
    - random mating probability rmp: 0.5
    - crossover rate CR: randomly generated from a normal distribution with mean 0.4 and sigma 0.1
    - scaling factor F: randomly generated from a cauchy distribution with location parameter 0.3 and scale parameter 0.1
2) Parameter settings in DEMTO_STO: same as the parameter settings in DEMTO.
3) Parameter settings in MFEA:
    - population size n: 100
    - random mating probability rmp: 0.3
    - distribution index of SBX: 2
    - distribution index of PM: 5
4) Parameter settings in MFEARR:
    - population size n: 100
    - random mating probability rmp: 0.3
    - the small number ϵ: 0.01
    - distribution index of SBX: 2
    - distribution index of PM: 5
5) Parameter settings in MFPSO:
    - population size n: 100
    - random mating probability rmp: 0.3
    - ω : linearly decreases from 0.9 to 0.4
    - $c_1 = c_2 = c_3 = 0.2$
6) Parameter settings in MFDE:
    - population size n: 100
    - random mating probability rmp: 0.3
    - scaling factor F: 0.5
    - crossover rate CR: 0.9
7) Parameter settings in MPEF-SHADE:
    - population size n: 100
    - learning parameter c: 0.3

TABLE I
DESCRIPTION OF THE 9 TEST PROBLEMS IN CEC'2017 EVOLUTIONARY MULTI-TASK OPTIMIZATION COMPETITION.

Problem	Component Tasks	Problem Dimension D	Global Optima $\boldsymbol{x^o}$	Optimal Values $f(\boldsymbol{x^o})$	Search Ranges	Inter-task Similarity R_s
1	T_1:Griewank T_2:Rastrigin	50	(0,0,...,0) (0,0,...,0)	0 0	[-100,100] [-50,50]	1.00
2	T_1:Ackley T_2:Rastrigin	50	(0,0,...,0) (0,0,...,0)	0 0	[-50,50] [-50,50]	0.22
3	T_1:Ackley T_2:Schwefel	50	[42.096,42.096,...,42.096] [420.968,420.968,...,420.968]	0 0	[-50,50] [-500,500]	0.00
4	T_1:Rastrigin T_2:Sphere	50	(0,0,...,0) (0,0,...,20)	0 0	[-50,50] [-100,100]	0.86
5	T_1:Ackley T_2:Rosenbrock	50	(0,0,...,1) (1,1,...,1)	0 0	[-50,50] [-50,50]	0.21
6	T_1:Ackley T_2:Weierstrass	50 (T_2:25)	(0,0,...,0) (0,0,...,0)	0 0	[-50,50] [-0.5,0.5]	0.07
7	T_1:Rosenbrock T_2:Rastrigin	50	(1,1,...,1) (0,0,...,0)	0 0	[-50,50] [-50,50]	0.94
8	T_1:Griewank T_2:Weierstrass	50	(10,10,...,10) (0,0,...,0)	0 0	[-100,100] [-50,50]	0.36
9	T_1:Rastrigin T_2:Schwefel	50	(0,0,...,0) [420.968,420.968,...,420.968]	0 0	[-50,50] [-500,500]	0.00

Each of the considered algorithms will be run for 20 times on each of the 9 test problems. The stopping criterion in both the algorithm is the maximum number of function evaluations (maxFEs), set to 100000 as did in [23]. To fairly measure and compare the performance of different algorithms, the following measurement are employed: the mean (standard deviation) of the maxFEs achieved when the algorithm terminates over 20 runs. All experiments are executed on a windows PC with a 3.6GHz Intel Core i7 CPU and 8GB RAM.

C. Experimental Results and Discussions

1) Comparison of DEMTO with the original MFEA and four other considered algorithms: To demonstrate the superiority of the proposed DEMTO, the aforementioned 9 benchmarking test problems from CEC'2017 Evolutionary Multi-Task Optimization Competition are employed to conduct experiment on DEMTO, MFEA, MFEARR, MFPSO, MFDE and MPEF-SHADE for performance comparison. In Table II, the achieved averaged fitness values over 20 runs by all the considered algorithms are presented.

From these achieved results, we can observe that DEMTO has shown its advantages over the other algorithms. In 9 out of 18 tasks, the proposed DEMTO achieves the best results, while the MPEF-SHADE perform the best in 5 out of 18 tasks. And MFDE only performs better than all the other algorithms in 3 out of 18 tasks. However, MPEF-SHADE incorporates the popular SHADE algorithm into its structure and uses some complex adaption strategies while DEMTO only employs the conventional differential evolution algorithm. Hence, the proposed DEMTO algorithm is promising in the future.

2) The effectiveness of knowledge transfer in DEMTO: To validate the effectiveness of knowledge transfer strategy used in DEMTO, we conduct this experiment on the 9 test problems in Table I to compare the performance of DEMTO with its single task version DEMTO_STO. DEMTO_STO employs the same evolutionary operations and parameters settings as DEMTO, to independently optimize each of the two component tasks in every test problem. Table III reports the means and bracketed standard deviations of the best achieved fitness values over 20 runs.

From table III, we can observe that DEMTO achieves better result than DEMTO_STO on 15 out of 18 tasks, which indicates that the knowledge transfer strategy employed in the proposed algorithm works effectively and efficiently. In particular, this knowledge transfer strategy has significantly improved the optimization quality of T_2 of problem 1-2, and of the both component tasks for problem 3 and 6. On the task T_1 of problem 4 and 8, the task T_2 of problem 9, DEMTO's knowledge transfer strategy has shown negative effects as shown in the table. But these negative effects do not greatly reduce the solution qualities of DEMTO on these tasks, such as the achieved fitness values $3.9446e+02$ comparing to the $3.9435e+02$ by DEMTO_STO, the achieved $2.47022-05$ comparing to the $1.8217e-05$ on DEMTO_STO and the $6.5142e+01$ comparing to the $5.3298e+01$.

In Fig. 1, we show the averaged convergence curves of DEMTO versus the DEMTO_STO over 20 runs during above optimization process. In each of the figures, the fitness value f for every task is depicted in Y-axis taking the base 10 logarithm, and the number of generations is used to denote X-axis. From these curves, we can further verify that the positive effects of knowledge transfer in DEMTO are significant on most of the tasks as the convergence curves from DEMTO are significantly located below those from DEMTO_STO. And on those tasks where DEMTO has shown negative effects, such as the task T_1 of problem 4 and 8, the task T_2 of problem 9,

TABLE II
AVERAGED ACHIEVED FITNESS VALUES OF THE DEMTO AND THE OTHER CONSIDERED ALGORITHMS. THE BEST RESULTS ARE SHOWN IN BOLD.

Problem	Task	MFEA	MFEARR	MFPSO	MFDE	MPEF-SHADE	DEMTO
1	T_1	3.6310e-01	2.3671e-01	2.1000e-01	1.0000e-03	1.3300e-09	**1.6421e-11**
	T_2	1.9127e+02	2.1350e+02	8.1100e+00	2.6080e+00	3.3600e-06	**2.0627e-08**
2	T_1	4.5495e+00	4.3575e+00	6.0000e-02	1.0000e-03	3.2100e-06	**9.9636e-08**
	T_2	2.1251e+02	2.0620e+02	6.2600e+00	3.0000e-03	3.1900e-08	**7.1642e-12**
3	T_1	2.0208e+01	2.0204e+01	5.6000e+00	2.1200e+01	2.1144e+01	**4.8239e-05**
	T_2	3.7174e+03	3.8056e+03	2.2263e+03	1.1843e+04	5.6537e+03	**6.3658e-04**
4	T_1	5.7420e+02	5.1221e+02	2.0573e+02	**7.8260e+01**	2.5663e+02	3.9446e+02
	T_2	9.3383e+00	4.5776e+00	3.8418e+03	2.2000e-05	**1.1800e-13**	2.3345e-12
5	T_1	3.5657e+00	3.3212e+00	3.5800e+00	1.0000e-03	3.6300e-05	**1.4149e-06**
	T_2	6.4618e+02	3.7320e+02	1.2367e+02	6.0340e+01	**4.8602e+01**	8.6367e+01
6	T_1	2.0004e+01	1.6074e+01	1.0000e-02	4.6000e-01	**5.9200e-06**	2.4288e-04
	T_2	2.0304e+01	1.6230e+01	5.0000e-02	2.2000e-01	**5.1700e-04**	8.5442e-03
7	T_1	1.0178e+03	5.9358e+02	4.3920e+01	8.9280e+01	4.2731e+01	5.3449e+01
	T_2	2.7368e+02	2.5139e+02	3.9580e+01	**2.0540e+01**	3.7729e+01	1.0177e+02
8	T_1	4.2640e-01	3.0401e-01	4.8000e-01	2.0000e-03	**1.6800e-09**	2.4702e-05
	T_2	2.7824e+01	2.6851e+01	1.2070e+01	2.9700e+00	1.6382e+00	**1.0135e+00**
9	T_1	6.2190e+02	5.4934e+02	3.3264e+02	**9.6150e+01**	2.6593e+02	3.9765e+02
	T_2	3.5739e+03	3.7490e+03	9.2562e+03	3.9382e+03	5.9824e+03	**6.5142e+01**

TABLE III
ACHIEVED BEST FITNESS VALUES (MEAN, BRACKETED STANDARD DEVIATION) OF THE DEMTO AND DEMTO_STO. THE BETTER RESULTS ARE SHOWN IN BOLD.

Problem	Task	DEMTO	DEMTO_STO
1	T_1	**1.6421e-11**(1.7077e-11)	1.2659e-05(5.6193e-06)
	T_2	**2.0627e-08**(2.0718e-08)	3.9395e+02(1.4570e+01)
2	T_1	**9.9636e-08**(4.8121e-08)	1.4319e-03(2.7869e-03)
	T_2	**7.1642e-12**(8.7229e-12)	3.9378e+02(1.4134e+01)
3	T_1	**4.8239e-05**(6.5338e-05)	2.1216e+01(2.6150e-02)
	T_2	**6.3658e-04**(4.9475e-07)	7.1064e+01(5.9530e+01)
4	T_1	3.9446e+02(1.8846e+01)	**3.9435e+02**(1.7952e+01)
	T_2	**2.3345e-12**(9.3836e-13)	1.8260e-06(8.1662e-06)
5	T_1	**1.4149e-06**(1.0710e-06)	5.6102e-01(2.4734e+00)
	T_2	**8.6367e+01**(5.1003e-01)	1.2464e+02(1.1702e+02)
6	T_1	**2.4288e-04**(4.6903e-04)	1.3014e-01(5.6978e-01)
	T_2	**8.5442e-03**(9.9890e-03)	3.5186e-01(7.5072e-01)
7	T_1	**5.3449e+01**(2.1076e+01)	8.9141e+01(4.5442e+01)
	T_2	**1.0177e+02**(1.5889e+02)	3.9652e+02(1.3016e+01)
8	T_1	2.4702e-05(2.3900e-05)	**1.8217e-05**(1.1751e-05)
	T_2	**1.0135e+00**(8.2462e-01)	2.2101e+01(2.2605e+01)
9	T_1	**3.9765e+02**(1.6452e+01)	4.0177e+02(1.4149e+01)
	T_2	6.5142e+01(8.1288e+01)	**5.3298e+01**(7.1632e+01)

their convergence curves is closely overlapped with the ones from DEMTO_STO over the entire iteration process, which indicates that the employed knowledge transfer strategy would not bring too much negative impact into the proposed DEMTO algorithm. Therefore, the employed knowledge transfer strategy in DEMTO is effective and efficient.

3) Parameter Study: The parameter rmp directly impact the knowledge transfer rate in DEMTO. Therefore, we here study the performance influence of different parameter settings of rmp on the 9 test problems in Table I. The other parameter settings of DEMTO are kept the same as the settings in the previous experiments. The means and bracketed standard deviations of the best achieved fitness values over 20 runs by the DEMTO on these settings are presented in Table IV .

The parameter rmp directly controls the knowledge transfer rate across the component tasks of a multi-task problem in DEMTO. A larger rmp will facilitate the transferring of knowledge across different tasks. On the contrary, a small rmp will reduce the knowledge transfer rate between tasks. From Table IV, we can observe that the increased rmp can further improved the optimization quality of the component tasks of problem 1-3, as the means of the best achieved fitness values are getting smaller. But these improvements have slowed down after rmp is set a value greater than 0.3. On the problem 4-9, the adjustment of rmp has no significant influence on the optimization quality of their component tasks. Comparing to the problem 1-3, the most reasonable explanation is that the component tasks of problem 4-9 have not the same global optima as presented in Table I. Therefore, there is mainly negative knowledge transfer occurred in the last period of their optimization process, and the increase of rmp can not bring to each of these tasks with more useful knowledge at the time. Even though the slight performance improvement in problem 1-3, the setting of parameter rmp is less sensitives to the achieved results on these 9 test problems in overall, as the achieved results of different parameter settings do not greatly differ from each other, especially in problem 4-9.

V. CONCLUSIONS AND FUTURE WORK

We proposed a differential evolutionary multi-task optimization (DEMTO) algorithm which employs different populations for solving different component problems (tasks), respectively, where a simple but effective knowledge transfer strategy

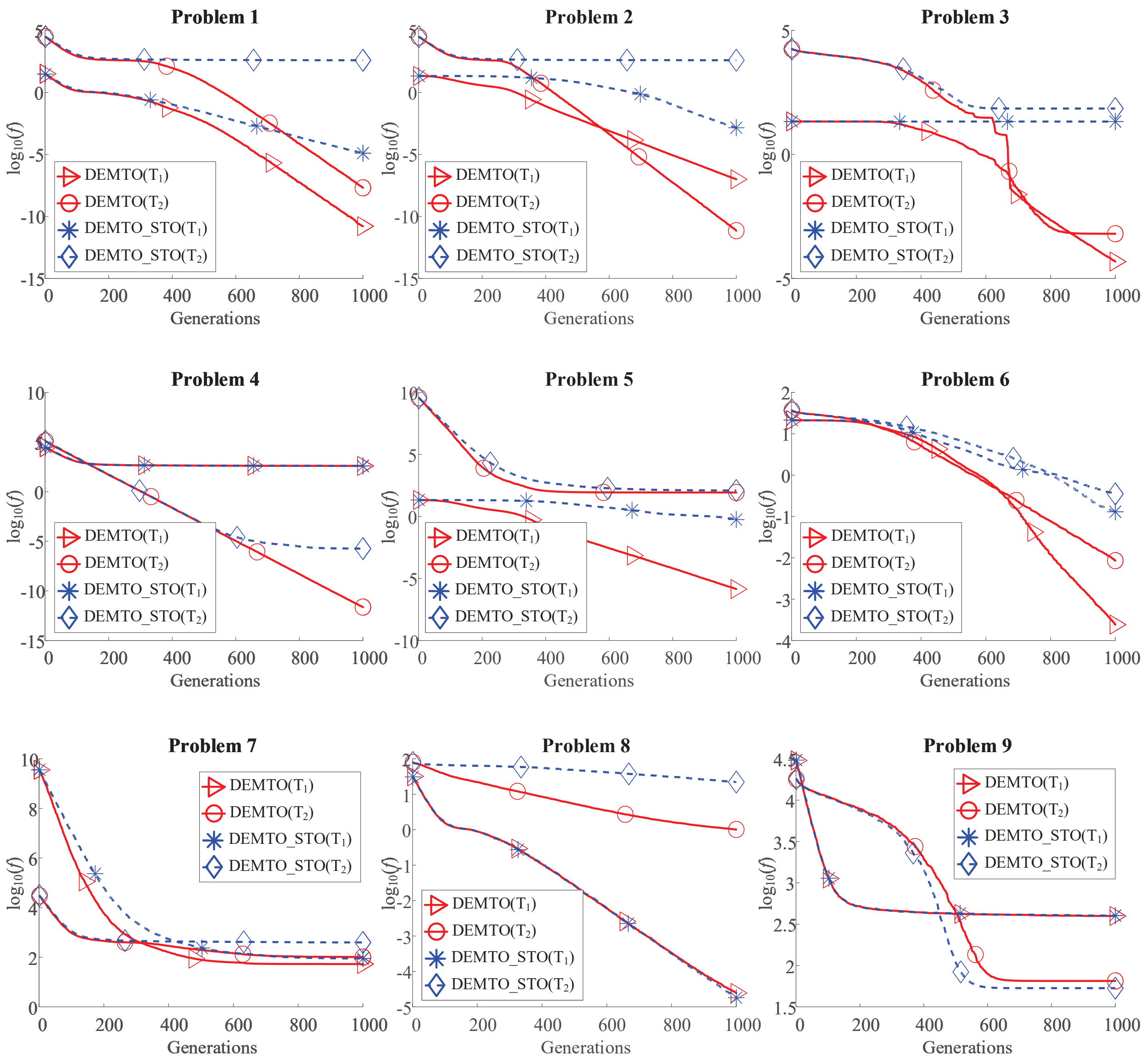

Fig. 1. Averaged convergence curves of DEMTO versus the DEMTO_STO on the 9 test problems from CEC'2017 Evolutionary Multi-Task Optimization Competition.

based on random replacement was proposed. This knowledge transfer strategy, implemented via differential evolution, does not introduce extra computational cost in terms of objective function evaluations for solving each component problem but can much help to improve optimization effectiveness and efficiency in comparison to solving each component problem in a standalone way. Experiments were conducted on the CEC'2017 competition test bed to compare DEMTO with five state-of-the-art EMTO algorithms, analyze the effectiveness of the proposed knowledge transfer strategy, and study the sensitivity of parameter settings in DEMTO. The results have demonstrated the superiority of the proposed algorithm.

In future, we plan to investigate the self-adaptation of parameter rmp to task relatedness, study MTO problems that contain many (far more than two) component tasks [6], explore the potential of the proposed algorithm for solving large-scale optimization problems [24], and apply the proposed algorithm to solve challenging real-world problems, particularly machine learning problems which involves complicated non-convex optimization tasks [12], [25], [26].

ACKNOWLEDGMENT

This work was supported by the Shenzhen Research and Development Foundation (Grant No. JCYJ20170306153943097), the National Natural Science Foundation of China (Grant Nos. 61772393, 61603299 and 61602385), and Australian Research Council (Grant Nos. DP160103595 and LP180100114).

TABLE IV
ACHIEVED BEST FITNESS VALUES (MEAN, BRACKETED STANDARD DEVIATION) OF THE DEMTO WITH DIFFERENT PARAMETER SETTINGS.

Problem	Task	rmp=0.1	rmp=0.3	rmp=0.5	rmp=0.7	rmp=0.9
1	T_1	1.5551e-06(2.6587e-06)	4.9092e-10(6.3310e-10)	1.2573e-11(1.5325e-11)	2.1138e-11(7.6146e-11)	1.9656e-12(4.8301e-12)
	T_2	1.8051e-03(3.8630e-03)	5.1020e-07(6.9961e-07)	1.7854e-08(2.3222e-08)	2.5309e-08(8.5760e-08)	2.7432e-09(7.2021e-09)
2	T_1	1.4246e-06(4.6922e-07)	3.9900e-07(1.3343e-07)	1.3250e-07(8.2692e-08)	4.1544e-08(2.8453e-08)	1.8667e-08(6.0741e-09)
	T_2	1.9628e-09(1.4847e-09)	1.1396e-10(7.5006e-11)	1.4685e-11(2.5048e-11)	1.3689e-12(3.1792e-12)	9.6634e-14(1.4859e-13)
3	T_1	2.4586e-01(9.6345e-01)	9.5387e-05(7.8788e-05)	3.8529e-05(2.2634e-05)	3.4121e-05(5.6943e-05)	3.8250e-05(6.9989e-05)
	T_2	1.7913e+01(7.9419e+01)	6.3663e-04(3.1001e-07)	6.3645e-04(8.0684e-08)	6.3651e-04(3.5584e-07)	6.3698e-04(2.5172e-06)
4	T_1	3.9536e+02(1.0409e+01)	3.9244e+02(1.5950e+01)	3.9661e+02(1.3631e+01)	3.9170e+02(1.2126e+01)	3.9611e+02(1.2529e+01)
	T_2	2.1902e-12(1.0328e-12)	2.5692e-12(1.9099e-12)	2.4447e-12(1.2928e-12)	3.0446e-12(1.2705e-12)	2.9716e-12(1.6197e-12)
5	T_1	2.5807e-06(7.0917e-07)	1.4815e-06(4.2603e-07)	1.3178e-06(5.6922e-07)	1.0541e-06(3.3671e-07)	1.2142e-06(1.2172e-06)
	T_2	7.6234e+01(1.9805e+01)	8.4850e+01(7.2803e+00)	8.4660e+01(8.1887e+00)	8.6375e+01(4.5155e-01)	8.6520e+01(5.1133e-01)
6	T_1	5.5403e-04(1.4325e-03)	8.6542e-04(3.1327e-03)	5.3076e-04(8.7754e-04)	6.6647e-04(1.7172e-03)	7.7074e-04(2.4609e-03)
	T_2	1.2187e-02(8.5667e-03)	1.5837e-02(3.2751e-02)	1.4113e-02(1.6016e-02)	1.5606e-02(2.8950e-02)	1.2406e-02(3.1441e-02)
7	T_1	7.1636e+01(3.0313e+01)	7.6276e+01(3.4523e+01)	6.2746e+01(2.4718e+01)	5.8721e+01(3.6644e+01)	8.5557e+01(3.8757e+01)
	T_2	1.4922e+02(1.5519e+02)	1.3188e+02(1.5570e+02)	7.3664e+01(1.2034e+02)	3.3117e+01(9.4672e+01)	1.5866e+02(1.5705e+02)
8	T_1	1.7502e-05(1.3136e-05)	2.6163e-05(1.6474e-05)	2.0382e-05(1.3991e-05)	9.8320e-04(4.3083e-03)	1.5287e-05(1.0567e-05)
	T_2	7.9496e-01(7.3358e-01)	9.2339e-01(7.3487e-01)	1.1424e+00(8.7927e-01)	9.5524e-01(7.3912e-01)	1.1332e+00(8.1367e-01)
9	T_1	3.9800e+02(1.3210e+01)	3.9134e+02(1.8208e+01)	3.9205e+02(1.5982e+01)	3.9152e+02(1.5485e+01)	3.9976e+02(1.0150e+01)
	T_2	1.0067e+02(9.6258e+01)	8.2907e+01(1.0936e+02)	7.6986e+01(8.8255e+01)	7.6986e+01(1.0364e+02)	7.1064e+01(8.0604e+01)

REFERENCES

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

[2] T. Back, U. Hammel, and H. P. Schwefel, “Evolutionary computation: Comments on the history and current state,” *IEEE transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 3–17, 1997.

[3] K. K. Bali, A. Gupta, L. Feng, Y. S. Ong, and T. P. Siew, “Linearized domain adaptation in evolutionary multitasking,” in *2017 IEEE Congress on Evolutionary Computation (CEC)*, June 2017, pp. 1295–1302.

[4] Q. Chen, X. Ma, Y. Sun, and Z. Zhu, “Adaptive memetic algorithm based evolutionary multi-tasking single-objective optimization,” in *Asia-Pacific Conference on Simulated Evolution and Learning*, 2017, pp. 462–472.

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

[6] J. Tang, Y. Chen, Z. Deng, Y. Xiang, and C. P. Joy, “A group-based approach to improve multifactorial evolutionary algorithm.” in *IJCAI*, 2018, pp. 3870–3876.

[7] R. T. Liaw and C. K. Ting, “Evolutionary many-tasking based on biocoenosis through symbiosis: A framework and benchmark problems,” in *Evolutionary Computation*, 2017, pp. 2266–2273.

[8] G. Li, Q. Zhang, and W. Gao, “Multipopulation evolution framework for multifactorial optimization,” in *2018 Genetic and Evolutionary Computation Conference*, 2018, pp. 215–216.

[9] L. Sampath, A. Gupta, Y.-S. Ong, and H. Gooi, “Evolutionary multitasking to support optimal power flow under rapid load variations,” *Southern Power System Technology, China*, vol. 11, no. 10, 2017.

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

[11] L. Bao, Y. Qi, M. Shen, X. Bu, J. Yu, Q. Li, and P. Chen, “An evolutionary multitasking algorithm for cloud computing service composition,” in *World Congress on Services*, 2018, pp. 130–144.

[12] B. Zhang, A. K. Qin, and T. Sellis, “Evolutionary feature subspaces generation for ensemble classification,” in *2018 Genetic and Evolutionary Computation Conference*, 2018, pp. 577–584.

[13] M.-Y. Cheng, A. Gupta, Y.-S. Ong, and Z.-W. Ni, “Coevolutionary multitasking for concurrent global optimization: With case studies in complex engineering design,” *Engineering Applications of Artificial Intelligence*, vol. 64, pp. 13–24, 2017.

[14] L. Feng, L. Zhou, J. Zhong, A. Gupta, Y.-S. Ong, K.-C. Tan, and A. K. Qin, “Evolutionary multitasking via explicit autoencoding,” *IEEE Transactions on Cybernetics*, 2018.

[15] J. J. Liang, S. Baskar, P. N. Suganthan, and A. K. Qin, “Performance evaluation of multiagent genetic algorithm,” *Natural Computing*, vol. 5, no. 1, pp. 83–96, 2006.

[16] J. C. Bean, “Genetic algorithms and random keys for sequencing and optimization,” *ORSA Journal on Computing*, vol. 6, no. 2, pp. 154–160, 1994.

[17] C. R. Cloninger, J. Rice, and T. Reich, “Multifactorial inheritance with cultural transmission and assortative mating. II. A general model of combined polygenic and cultural inheritance.” *American Journal of Human Genetics*, vol. 31, pp. 176–198, 1979.

[18] L. Feng, W. Zhou, L. Zhou, S. Jiang, J. Zhong, B. Da, Z. Zhu, and Y. Wang, “An empirical study of multifactorial PSO and multifactorial de,” in *Evolutionary Computation (CEC), 2017 IEEE Congress on*, 2017, pp. 921–928.

[19] R. Storn and K. Price, “Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces,” *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[20] A. K. Qin, V. L. Huang, and P. N. Suganthan, “Differential evolution algorithm with strategy adaptation for global numerical optimization,” *IEEE transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.

[21] J. Zhang and A. C. Sanderson, “JADE: adaptive differential evolution with optional external archive,” *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.

[22] B. Kazimipour, X. Li, and A. K. Qin, “Effects of population initialization on differential evolution for large scale optimization,” in *2014 IEEE Congress on Evolutionary Computation (CEC)*, 2014, pp. 2404–2411.

[23] B. Da, Y.-S. Ong, L. Feng, A. K. Qin, A. Gupta, Z. Zhu, C.-K. Ting, K. Tang, and X. Yao, “Evolutionary multitasking for single-objective continuous optimization: Benchmark problems, performance metric, and baseline results,” 2017. [Online]. Available: http://arxiv.org/abs/1706.03470

[24] B. Kazimipour, X. Li, and A. K. Qin, “Initialization methods for large scale global optimization,” in *2013 IEEE Congress on Evolutionary Computation*, 2013, pp. 2750–2757.

[25] B. Y. Qu, B. F. Lang, J. J. Liang, A. K. Qin, and O. D. Crisalle, “Two-hidden-layer extreme learning machine for regression and classification,” *Neurocomputing*, vol. 175, pp. 826–834, 2016.

[26] A. K. Qin, P. N. Suganthan, and M. Loog, “Uncorrelated heteroscedastic LDA based on the weighted pairwise Chernoff criterion,” *Pattern Recognition*, vol. 38, no. 4, pp. 613–616, 2005.