Generalized Multitasking for Evolutionary Optimization of Expensive Problems

Jinliang Ding, *Senior Member, IEEE*, Cuie Yang, Yaochu Jin, *Fellow, IEEE*, and Tianyou Chai, *Fellow, IEEE*

***Abstract*—Conventional evolutionary algorithms (EAs) are not well suited for solving expensive optimization problems due to the fact that they often require a large number of fitness evaluations to obtain acceptable solutions. To alleviate the difficulty, this paper presents a multitasking evolutionary optimization framework for solving computationally expensive problems. In the framework, knowledge is transferred from a number of computationally cheap optimization problems to help the solution of the expensive problem on the basis of the recently proposed multifactorial EA (MFEA), leading to a faster convergence of the expensive problem. However, existing MFEAs do not work well in solving multitasking problems whose optimums do not lie in the same location or when the dimensions of the decision space are not the same. To address the above issues, the existing MFEA is generalized by proposing two strategies, one for decision variable translation and the other for decision variable shuffling, to facilitate knowledge transfer between optimization problems having different locations of the optimums and different numbers of decision variables. To assess the effectiveness of the generalized MFEA (G-MFEA), empirical studies have been conducted on eight multitasking instances and eight test problems for expensive optimization. The experimental results demonstrate that the proposed G-MFEA works more efficiently for multitasking optimization and successfully accelerates the convergence of expensive optimization problems compared to single-task optimization.**

***Index Terms*—Evolutionary algorithms (EAs), evolutionary multitasking optimization, expensive optimization, knowledge transfer.**

I. Introduction

EVOLUTIONARY algorithms (EAs) are population-based optimization methods that emulate the natural process of natural selection and genetic variations [1]. EAs distinguish themselves with the distinct advantages that do not require analytical formulations of the objective functions and are able to perform global search, among many others. Due to the above advantages, they have achieved widespread applications in real-world complex optimization problems, such as production planning [2], job-shop scheduling [3], and aircraft design [4] in the past decades. However, EAs have apparent weaknesses in solving computationally expensive optimization problems, where the evaluation of candidate solutions needs to perform time-consuming numerical simulations or expensive physical experiments [5], [6], mainly because a large number of fitness evaluations are often needed before they can find acceptable solutions. In the past decades, developing efficient EAs for expensive optimization problems has attracted increasing research interests and many surrogate-assisted EAs (SAEAs) have been developed [7]–[9], [21]–[25]. The basic idea of SAEAs is to use computational cheap surrogate models to replace in part the time-consuming fitness evaluations to reduce computational cost. So far the most commonly used surrogate models include artificial neural network [9], Kriging model [26], [27], and radial basis function networks [28], [29].

Multitasking EAs, also known as multifactorial EAs (MFEAs), were first proposed in [10] and [11] to address multifactorial optimization (MFO) problems, which is a problem containing multiple distinct optimization tasks to be solved simultaneously. A main feature of MFEAs is that they are able to transfer knowledge between tasks during the evolution via two approaches, namely, assortative mating and vertical cultural transmission. It is a kind a multitasking knowledge transfer method according to [39]. MFEAs have been shown to be effective in accelerating convergence and enhancing global search capabilities, with the help of positive knowledge transferring among tasks [20]. Inspired by the effectiveness of knowledge transfer in MFEAs, this paper presents a new multitasking optimization framework for solving evolutionary expensive computational problems, in which knowledge from computationally cheap problems is used to accelerate the convergence of computationally expensive problems.

Although MFEAs have been successfully extended to solve many types of problems, such as continuous, discrete, mix-integer, and even combinatorial optimization problems [10]–[14], it has been reported that the performance enhancement of MFEAs may become less reliable as the degree of separation between the optimums in MFO problem

Manuscript received May 11, 2017; revised August 9, 2017 and October 19, 2017; accepted December 10, 2017. Date of publication December 20, 2017; date of current version January 28, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61525302, Grant 61590922, and Grant 61621004, in part by the Projects of Liaoning Province under Grant 2014020021 and Grant LR2015021, and in part by the Fundamental Research Funds for the Central Universities under Grant N160801001 and Grant N161608001. *(Corresponding author: Yaochu Jin.)*

J. Ding, C. Yang, and T. Chai are with the State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang 110819, China (e-mail: jlding@mail.neu.edu.cn; cuieyang@outlook.com; tychai@mail.neu.edu.cn).

Y. Jin is with the State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang 110819, China, and also with the Department of Computer Science, University of Surrey, Guildford GU2 7XH, U.K. (e-mail: yaochu.jin@surrey.ac.uk).

This paper has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org provided by the authors.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2017.2785351

increases. This is due to the fact that the separation of optimums introduces too much diversity in the evolution, slowing down the convergence in the optimization [10]. Unfortunately, this phenomenon is very common as the location of the optimum in an optimization problem is often unknown *a priori* and can lie anywhere in the decision space. In addition, MFEAs do not work well for MFO problems when the dimensions of decision space of different tasks are different due to inefficient knowledge transfer. To address the above issues, a generalized MFEA (G-MFEA) consisting of two new strategies, i.e., decision variable translation strategy and a decision variable shuffling strategy, are proposed in this paper. While the proposed G-MFEA is suited for conventional multitasking problems, in this paper, we focus on applying G-MFEA to the solution of expensive optimization problems. The main contributions of this paper can be summarized as follows.

1) A decision variable translation strategy is proposed to address MFO problems in which the locations of the optimums of the tasks are different. The decision variable translation strategy adjusts the location of individuals according to the estimated optimum of each task in order to enhance knowledge transfer in the optimization process.
2) A decision variable shuffling strategy is also introduced to deal with MFO problems having different numbers of decision variables. The decision variable shuffling strategy can not only change the order of the decision variables in the chromosome to give every variable an opportunity to communicate with the other tasks, thereby improving the efficiency of knowledge transfer, it can also replace decision variables that are not in use with the corresponding useful information to ensure the quality of the transferred knowledge.
3) The generalized multitasking framework is further extended for solving expensive optimization problems, termed MCEEA. In MCEEA, a number of computationally cheap tasks (C-tasks) are used to accelerate the optimization of a computationally expensive task (E-task), thereby reducing the needed number of fitness evaluations of the E-task. It should be emphasized that, in contrast to existing multitasking optimization algorithms, the proposed MCEEA mainly focuses on improving the convergence speed of the E-task. For this reason, there are some notable discrepancies between MCEEA and G-MFEA. For example, multiple C-tasks are included in MCEEA to produce diverse knowledge in order to achieve robust performance enhancement of the E-task. Moreover, only the elite individuals in the C-tasks with useful knowledge for the E-task are selected for knowledge transfer in solving the E-task, thus achieving a faster convergence of the E-task.

The remainder of this paper is organized as follows. Section II introduces the basic ideas in MFEAs and discusses the motivation of the proposed G-MFEA. The details of the proposed G-MFEA and its performance validation are described in Section III. Section IV applies the proposed generalized multitasking EA to address expensive optimization problems, known as MCEEA. The experimental results of the proposed MCEEA is presented and discussed in Section V. Section VI concludes this paper.

II. Preliminaries

In this section, the concept of multitasking evolutionary optimization and the related definitions are first introduced. Then the MFO algorithm (MFEA) is briefly described. Finally, the motivation of our proposed G-MFEA is presented.

A. Definitions

Multitasking evolutionary optimization was first suggested by Gupta *et al*. [10], which aims to enhance global optimum search by simultaneously optimizing multiple optimization problems. In multitasking evolutionary optimization, the multiple optimization problems to be tackled and the algorithm to solve the multiple optimization problems are denoted as MFO problems, and MFEA, respectively. An MFO problem where K continuous single objective minimization problems are to be simultaneously addressed can be formulated as follows:

$$\{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_K}\} = \{\text{argmin}f_1(\mathbf{x_1}), \text{argmin}f_2(\mathbf{x_2}), \ldots \text{argmin}f_K(\mathbf{x_K})\} \quad (1)$$

where $f_k(\mathbf{x}_k)$, $k = 1, 2, \ldots, K$, represents one distinct optimization task, and $\mathbf{x}_k = (x_{k,1}, x_{k2}, \ldots, x_{k,D_k})$ is a feasible solution in the kth task solution space X_k, D_k is the dimensionality of X_k.

An MFEA optimizes an MFO problem with a single population in which each individual is associated with one optimization task. Knowledge transfer among tasks is a significant feature of MFEAs to achieve better performance in problem solving. To share knowledge among different tasks, MFEAs introduces a unified search space Y for individuals of all tasks. The dimensionality of this unified search space is defined to be $D_{\text{multitask}} = \max\{D_k\}, k = 1, 2, \ldots, K$, where D_k is the dimensionality of the kth task and the boundaries of Y are normalized to [0, 1]. Each individual is decoded into its corresponding task-specific solution space before it is evaluated. In case an individual $\mathbf{y}$ is associate with the kth task, the first D_k decision variables of $\mathbf{y}$ are decoded as a task meaningful solution x, that is, $\mathbf{x} = \mathbf{y}(1 : D_k) \times (U_k - L_k) + L_k$, where D_k is the dimension of the ith task, U_i and L_i are upper and lower boundary of the kth task.

In the following, we briefly review the definitions related to each individual p_i in an MFEA [10].

Factorial Rank: The factorial rank r_k^i of p_i for the kth task is the index of p_i in the list of individuals in the population sorted in an ascending order of f_k, where f_k is the function value of the kth task.

Skill Factor: In the multitasking population, each individual only tackles one task and the skill factor τ_i of p_i represents the task that p_i is associated with. $\tau_i = \arg\min_k\{r_k^i\}$, where $k \in \{1, 2, \ldots, K\}$.

Scalar Fitness: The scalar fitness φ_i of p_i is its normalized fitness in its associated task, $\varphi_i = 1/\min\{r_k^i\}, k = 1, 2, \ldots, K$.

Algorithm 1 Framework of an MFEA

Input: NP, the number of individuals in the parent population; K, the number of tasks;
Output: The best solutions of all tasks.
1: Randomly generate NP individuals in the unified search space Y as an initial population P
2: Assign a skill factor τ for each individual, for the *i-th* individual $\tau_i = \text{mod}(i, K) + 1$.
3: Population evaluation
4: **While** the termination criterion is not fulfilled **do**
5: Generate offspring population O by assortative mating
6: Perform vertical cultural transmission
7: Evaluate offspring
8: $\text{R} = \text{O} \bigcup \text{P}$
9: Update the scalar fitness in R
10: Select NP fittest individuals according to the scalar fitness as the parents of the next generation
11: **End while**

The scalar fitness enables to compare the performance of individuals from different tasks.

B. Multifactorial Optimization Algorithms

The main components of an MFEA is described in Algorithm 1. An MFEA begins with a population with NP candidate solutions randomly initialized in the unified search space Y. Then the skill factor of each individual is assigned and evaluated, where the individuals are evaluated only for the associated task (the assigned skill factor) and the objective values for other tasks are set to be infinite (a very large number). The offspring generation in an MFEA is based on assortative mating and vertical cultural transmission to transfer knowledge between different tasks. Specifically, assortative mating is adopted to generate offspring, where individuals from the same task have a high probability to mate for generating offspring, while those for different tasks can mate at a smaller probability. Algorithm 2 presents the pseudo code of assortative mating. After the offspring are created, an MFEA uses vertical cultural transmission to assign the skill factor for each offspring individual. In vertical cultural transmission, an offspring imitates one task randomly from its parents associated with, leading to exchanging solutions across the tasks. With vertical cultural transmission, it is easy for a complex optimization problem to find better solutions by obtaining promising solutions from other tasks. Algorithm 3 lists the pseudo code of the vertical cultural transmission. At the end of each iteration, the parents for the next generation are selected according to the scalar fitness.

C. Motivation of the Proposed Algorithm G-MFEA

According to the above descriptions, we can see that assortative mating is one important feature of MFEAs, which is able to promote population diversity by allowing individuals for different tasks to mate [20]. For an MFO problem whose optimums are located in the same position of the unified search space Y, the individuals associated with different tasks

Algorithm 2 Assortative Mating

Input: p_1, p_2, two randomly selected parents; τ_1, τ_2, the skill factor of two parents; rmp, the crossover probability of parents from different tasks;
Output: The generated offspring o_1, o_2.
1: **If** $\tau_1 == \tau_2$ or rand $< rmp$
2: $(o_1, o_2) =$ crossover (p_1, p_2)
3: **Else**
4: $o_1 = \text{mutate}(p_1)$
5: $o_2 = \text{mutate}(p_2)$
6: **End if**

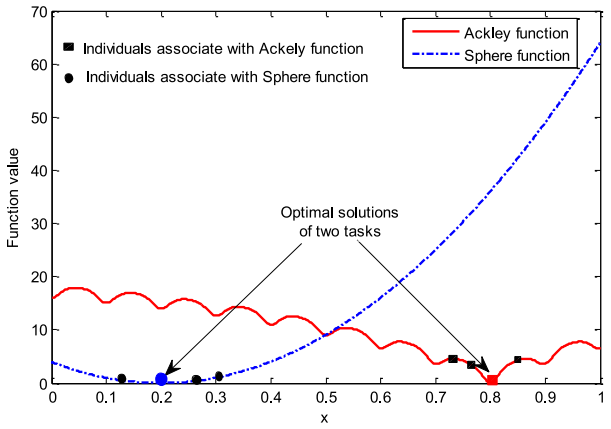

Fig. 1. Illustrative example of an MFO with separate optimal tasks in 1-D unified search space.

are likely to be distributed at different locations during the optimization process and they converge to the same location at the end of the optimization. Therefore, the individuals of each task are able to maintain a high degree of diversity at the early stage of the optimization process and this diversity decreases as the evolution proceeds, which is beneficial for improving the convergence at the later stage of the optimization process.

However, in case the optimums of the MFO problem are situated at different locations, individuals of each task will remain to be distant from each other even at the later stage of the optimization process. Fig. 1 shows an example of an MFO problem with separate optimums in a 1-D unified search space Y. In this situation, assortative mating will introduce diversity to the population during the entire optimization process according to analyzed above. Note however, that excessive diversity may seriously slow down the convergence of searching tasks' optimal solutions.

Vertical cultural transmission is another key feature of MFEAs that exchanges solutions across different tasks. It aims to transfer high quality solutions from easy tasks to complex tasks in order to enhance the convergence speed of the complex tasks [10]. According to the MFEA framework, all individuals in the population have the same search dimension in the unified search space Y, with only the first D_k variables of an individual being decoded into the kth task space before fitness evaluation [10], where D_k is the dimensionality of the kth task.

Algorithm 3 Vertical Cultural Transmission

Input: o, the generated offspring by crossover(p_1, p_2) or mutation (p) in assortative mating;
Output: The skill factor τ of o.
If c generated by crossover (p_1, p_2)
  If rand < 0.5
    τ imitates the skill factor of p_1
  Else
    τ imitates the skill factor of p_2
  End if
Else
    τ imitates the skill factor of p
End if

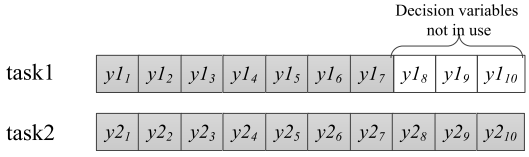

Fig. 2. Example of MFO having two tasks whose dimensions of the decision space are different: task1 has 7 decision variables and task2 has 10.

In case the dimensions of decision space of different tasks in the MFO problem are different like the example shown in Fig. 2, where the MFO problem contains two tasks, namely task1 and task2, whose dimensions are 7 and 10, respectively. In this situation, only the first part (the first 7 decision variables) of the individuals in task1 represent the solutions, while the remaining variables are not in use. As a result, the last variables of task2 cannot exchange knowledge with task1. In addition, a good solution of task1 is not necessarily good for task2, as it is a nonintegrated solution for task2. Therefore, the effectiveness of knowledge transfer in MFEAs may degrade when the dimensions of task1 and task2 are not the same.

To address the above limitations of MFEAs in solving MFO problems whose optimums are located in different positions in the unified search space Y or whose dimensions of the decision space are different, this paper presents a G-MFEA. G-MFEA is equipped with two new strategies, one is the decision variable translation strategy aiming to tackle MFO problems having separate optimums, and the other is the decision variable shuffling strategy dealing with different dimensions. More specifically, the decision variable translation strategy maps the solutions of an individual into a new space in which the optimums of all tasks are located in the same position. By applying the decision variable translation strategy, a better balance between enhancing the population diversity and accelerating the convergence of the population can be maintained. Meanwhile, the decision variable shuffling strategy will not only enhance the effectiveness of knowledge transfer by giving every decision variable an opportunity to mate but also improve the quality of the transferred solution by replacing the variables both in use with corresponding information of the task to which the solution is transferred.

Most recently, linearized domain adaptation that transforms one task to another has been proposed independently [38]. It should be pointed out that the ideas proposed in this paper differ from those in [38] in that G-MFEA transforms all tasks to a new space while maintaining the same geometric properties of solutions, thereby avoiding unfavorable influences on MFEAs' ability to diversify population and perform global search.

Algorithm 4 Framework of the Proposed G-MFEA

Input: NP, population size; K, the number of tasks;
Output: The best solution of each task.
Set: θ, the frequency of calculating translated direction; ϕ, the threshold of starting the decision variable translation strategy; $\overline{\mathrm{P}} = \emptyset$.
$P =$ Population initialization and evaluation
While the termination criterion is not fulfilled **do**
  $\overline{\mathrm{P}} =$ Decision variable translation strategy (P,θ,ϕ)
  For $i = 1{:}NP/2$
    Randomly select two parents $\overline{p}_1$ and $\overline{p}_2$ from $\overline{\mathrm{P}}$ and obtain their skill factor
    $(\overline{\overline{p}}_1, \overline{\overline{p}}_2) =$ decision variable shuffling of $(\overline{p}_1, \overline{p}_2)$
    $(\overline{\overline{o}}_1, \overline{\overline{o}}_2) =$ Assortative mating $(\overline{\overline{p}}_1, \overline{\overline{p}}_2)$
    $(\overline{o}_1, \overline{o}_2) =$ Re-transfer the offspring $(\overline{\overline{o}}_1, \overline{\overline{o}}_2)$
    Vertical cultural transmission
    $(o_1, o_2) =$ Re-change the order of decision variables of individuals $(\overline{o}_1, \overline{o}_2)$
  End for
  Evaluate offspring O
  $\mathrm{R} = \mathrm{O} \bigcup \mathrm{P}$
  Update the scalar fitness in R
  Select NP fittest individuals according to scalar fitness as the next generation
End while

III. Proposed Algorithm G-MFEA

In this section, we first present the main framework of the proposed G-MFEA, followed by a detailed description of the two strategies, the decision variable translation strategy and the decision variable shuffling strategy. After that, the efficiency of G-MFEA is examined on eight MFO problems and the experimental results are analyzed.

A. Main Framework of G-MFEA

The main framework of G-MFEA is summarized in Algorithm 4. G-MFEA differs from the MFEA mainly in the process of offspring generation. At each generation, the location of the individuals in the original population is first translated to a new location by the proposed decision variable translation strategy, with the population in the new location being denoted by $\overline{P}$. Algorithm 5 details the decision variable translation strategy. Once the translated population $\overline{P}$ is formed, offspring will be generated from the parents in $\overline{P}$. Given two randomly selected parents, the order of the decision variables are further perturbed and the variables not in use are replaced by the decision variable shuffling strategy before they undergo assortative mating. The decision variable shuffling strategy is described in Algorithm 6.

Algorithm 5 Decision Variable Translation Strategy

Input: *gen*, the current iterations; *Maxgen*, the maximum iterations; P, the current population; *NP*, population size; θ, the frequency to change translated direction; ϕ,the threshold value to start the decision variable translation strategy;
Output: The translated population $\overline{\mathrm{P}}$;
Set: $\overline{\mathrm{P}} = \emptyset$.
1: **If** $gen > \phi$
2: **If** mod(gen, θ) $= 0$
3: Calculate the translated direction of each task according to Equation (4)
4: **End if**
5: **For** $i = 1{:}NP$
Calculate $\overline{p}_i$ according to Equation (6)
6: $\overline{\mathrm{P}} = \overline{\mathrm{P}} \cup \overline{p}_i$
7: **End for**
8: **Else**
9: $\overline{\mathrm{P}} = P$
10: **End if**

Algorithm 6 Decision Variable Shuffling Strategy

Input: $\overline{p}_1$, $\overline{p}_2$, two randomly selected parents from $\overline{\mathrm{P}}$
Output: The two parents after shuffling $\overline{\overline{p}}_1$, $\overline{\overline{p}}_2$, the order list l_1 , l_2
Set: $D_{max} = \max(D_1, D_2)$; $l_1 = 1, 2, \ldots, D_{max}$; $l_2 = 1, 2, \ldots, D_{max}$;
1: Obtain the skill factor of $\overline{p}_1$ and $\overline{p}_2$
2: **If** $D_1 < D_{max}$
3: $\overline{p} =$ randomly select one individual from $\overline{\mathrm{P}}$ that has the same skill factor as $\overline{p}_2$
4: Randomly perturb the order of l_1
5: $\overline{p}(l_1(1{:}D_1)) = \overline{p}_1(1{:}D_1)$
6: $(\overline{\overline{p}}_1, \overline{\overline{p}}_2) = (\overline{p}, \overline{p}_2)$
7: **Else if** $D_2 < D_{max}$
8: $\overline{p} =$ randomly select one individual form $\overline{\mathrm{P}}$ that has the same skill factor to $\overline{p}_1$
9: Randomly perturb the order of l_2
10: $\overline{p}(l_2(1{:}D_2)) = \overline{p}_2(1{:}D_2)$
11: $(\overline{\overline{p}}_1, \overline{\overline{p}}_2) = (\overline{p}_1, \overline{p})$
12: **Else**
13: $(\overline{\overline{p}}_1, \overline{\overline{p}}_2) = (\overline{p}_1, \overline{p}_2)$
14: **End if**

It should be noted that the generated offspring are also in the translated solution space. Therefore, these offspring must be translated back to the original solution space (step 8). An offspring is translated back to the space of the parent having a closer inheritance relationship with it. Assume an offspring $\overline{\overline{o}}_1$ is generated by one of the following two cases using assortative mating, it will be mapped back to the solution space of the task that $\overline{\overline{p}}_1$ is associated with according to (2), as it inherits more information from $\overline{\overline{p}}_1$

$$\overline{o}_1 = \overline{\overline{o}}_1 - d_{\tau_1}. \tag{2}$$

First perform a crossover operation for parents $\overline{\overline{p}}_1$ and $\overline{\overline{p}}_2$ as follows:

$$\begin{aligned}\overline{\overline{o}}_1 &= \text{Crossover}\left(\overline{\overline{p}}_1, \overline{\overline{p}}_2\right) = 0.5\left((1+cf)\overline{\overline{p}}_1 + (1-cf)\overline{\overline{p}}_2\right) \\ &= \overline{\overline{p}}_1 + 0.5(cf-1)\left(\overline{\overline{p}}_1 - \overline{\overline{p}}_2\right)\end{aligned} \tag{3}$$

where cf is a random number generated using a distribution designed in [15]. Then, perform a mutation operation from $\overline{\overline{p}}_1$. Otherwise, if $\overline{\overline{o}}_1$ is generated by Crossover($\overline{\overline{p}}_1, \overline{\overline{p}}_2$) or mutation of $\overline{\overline{p}}_2$, it will be mapped back to the solution space of $\overline{\overline{p}}_2$.

After the offspring are assigned a skill factor using vertical cultural transmission, the decision variables of each offspring are also changed back to their original order (step 10). The details for translating solutions will be presented in Section III-B, while the decision variable shuffling strategy and the order rechange are detailed in Section III-C.

B. Decision Variable Translation Strategy

The main purpose of the decision variable translation strategy is to map individuals to a new location so that the optimums of all tasks are located at the same position in the unified search space Y to facilitate knowledge transfer and strike a good balance between the population diversity and convergence during evolutionary optimization. The details of the decision variable translation strategy are summarized in Algorithm 5.

The center point of the unified search space Y, that is $\text{cp} = (0.5, 0.5, \ldots, 0.5)$, is the designated location to which all optimums are transferred to. To achieve this, it is important to estimate the distance from its current location of each task to the center point, called translated direction d_k, $k = 1, 2, \ldots, K$, where K is the number of tasks. The translated directions are estimated based on the promising solutions of each task at the current generation as follows:

$$d_k = \text{sf} \times \alpha(\text{cp} - m_k), k = 1, 2, \ldots, K \tag{4}$$

where m_k is the estimated optimum by calculating the mean value of the μ percent best solutions of the kth task. Since the estimated optimum m_k may be different from the true one, d_k may be smaller than the truth translated directions, which may result in a situation where the tasks cannot be translated to the same space even until at the end of the evolutionary optimization. To make sure that the locations of the two tasks are able to overlap at certain time during the optimization, we slightly speed up the translated distance at each step using a scale factor (sf) in (4). Sensitivity analysis of the parameter will be discussed in Section III-D.

To keep a stable translation, the distance d_k is calculated at every θ generations as shown in steps 2–4 in Algorithm 5. Considering the inaccurate estimation of optimums at the early optimization process, α is initialized to zero and increases gradually as the evolution proceeds from the ϕth generation

$$\alpha = \left(\frac{\text{gen}}{\text{Maxgen}}\right)^2 \tag{5}$$

where Maxgen is the maximum number of generations and gen is the current number of generations.

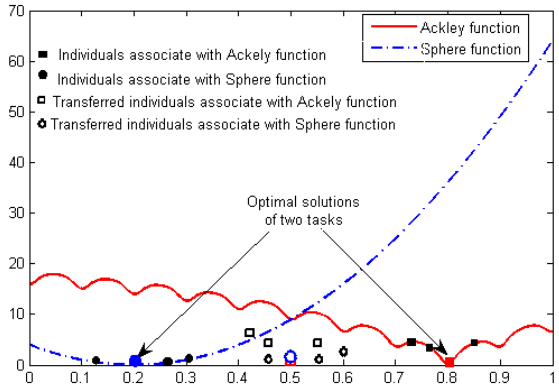

Fig. 3. Illustration of distribution of individuals in the 1-D new locations in the unified search space.

Once the translated direction is estimated, each individual is moved to a new location according to the following equation:

$$\overline{p}_i = p_i + d_{\tau_i}, i = 1, 2, \ldots, \mathrm{NP} \tag{6}$$

where τ_i is the skill factor of ith individual, and NP is the population size. The population in the new location is shown in Fig. 3.

C. Decision Variable Shuffling Strategy

The proposed decision variable shuffling strategy in G-MFEA aims to enhance the effectiveness of knowledge transfer between tasks of an MFO whose dimensions are different. To be specific, the decision variable shuffling strategy first randomly changes the order of the variables in the solutions with a lower dimension to give each variable an opportunity of knowledge transfer between two tasks. Then, the decision variables of the individual for the lower-dimensional task that are not in use are replaced with those of the individual for the higher-dimensional task. For this reason, the decision variable shuffling strategy can be applied only on the parents that have a lower dimension.

Fig. 4 presents an example to illustrate the main idea of the decision variable shuffling strategy, where two parents, $\overline{p}_1 = (y1_1, y1_2, \ldots, y1_{10})$ and $\overline{p}_2 = (y2_1, y2_2, \ldots, y2_{10})$, are randomly selected from the transferred population $\overline{P}$ and are associated with task1 and task2, respectively. Suppose the dimension of task1 is lower than task2, as shown in Fig. 4(a). Therefore, only $\overline{p}_1$ will be subject to the decision variable shuffling strategy. We first record the order of the decision variables of these two individuals as $l_1 = 1, 2, \ldots, D_{\max}$, $l_2 = 1, 2, \ldots, D_{\max}$, where $D_{\max} = \max(D_1, D_2)$. Then an extra individual $\overline{p} = (y_1, y_2, \ldots, y_{10})$ that is associated with task2 (the task that $\overline{p}_2$ is associated with), which is used to replace the variables that are not in use in $\overline{p}_1$, is randomly selected from $\overline{P}$. After that, the alternated $\overline{\overline{p}}_1$ will be obtained by merging $\overline{p}$ and $\overline{p}_1$. Specifically, the list l_1 is randomly changed and the variables whose index are in the first D_1 elements of list l_1 are from $\overline{p}_1$, while the remaining variables are from $\overline{p}$. Fig. 4(b) illustrates the results of the decision variable shuffling strategy applied, where the disturbed list $l_1 = (1, 2, 4, 5, 6, 9, 10, 7, 3, 8)$, the index in the first D_1 elements of l_1 in $\overline{\overline{p}}_1$ are from the first D_1 variables of $\overline{p}_1$, while the remaining variables are from $\overline{p}$. Once decision variable shuffling has been performed, the initial offspring will be generated by $\overline{\overline{p}}_1$ and $\overline{\overline{p}}_2$ via assortative mating. Fig. 4(c) illustrates the offspring $\overline{o}_1$ and $\overline{o}_2$ that are moved back to their original space from the initial offspring (step 8 in Algorithm 4). Before an offspring is passed to the population, the order of the decision variable needs to be rechanged if it is assigned a task whose dimension is lower. Fig. 4(d) shows an example, where $\overline{o}_1$ is assigned to task1 and $\overline{o}_2$ assigned to task2. Then o_1, which is obtained by $o_1(1 : D_1) = \overline{o}_1(l_1(1 : D_1))$, is set to be the offspring of task1, while $\overline{o}_2$ itself will be the offspring o_2 of task2.

D. Multifactorial Problem Experiments

In this section, the efficiency of G-MFEA is empirically investigated by comparing it with the original MFEA [10] and the base single task EA, denoted as SOEA, on eight MFO problems.

1) Test Instances: To assess the performance of G-MFEA, eight MFO test instances are applied, as listed in Table A in the supplementary materials, in which F1, F2, F4, and F7 belong to complete intersection and high similarity (CI+HS), complete intersection and medium similarity (CI+MS), no intersection and low similarity (NI+LS), and no intersection and high similarity (NI+HS), respectively, as categorized in [33]. The rest test functions are proposed in this paper. The value of Spearman's rank correlation coefficient [37] (*Rs*) is also calculated and listed in Table A in the supplementary material for each test instance using 1 000 000 randomly generated points in the unified search space. These test instances are constructed from seven commonly used single objective optimization problems [16], namely, Sphere, Schwefel, Rosenbrock, Rastrigin, Griewank, Ackley, and Weierstrass. They can be categorized into the following four groups according to the dimensions of the decision space and the locations of the optimums.

1) F1 and F2, which have the same number of decision variables and the optimums are located in the same position in the unified search space *Y*.
2) F3 and F4, whose decision spaces are of the same dimension but the optimums are located in different positions in the unified search space *Y*.
3) F5 and F6, which have different numbers of decision variables but the optimums are located in the same position.
4) F7 and F8, which have different number of decision variables and their optimums are located in different places.

2) Parameter Settings: Parameters in experiments are as follows.

a) Public parameter settings: For fair comparisons, all compared algorithms adopt the same genetic operators, i.e., the simulated binary crossover [15] and Gaussian mutation [17]. The distribution indices of crossover is set to be 20. The crossover probability is set to pc = 1.0, mutation probability in MFO algorithm is pm $= 1/D_{\mathrm{multitask}}$, where $D_{\mathrm{multitask}}$

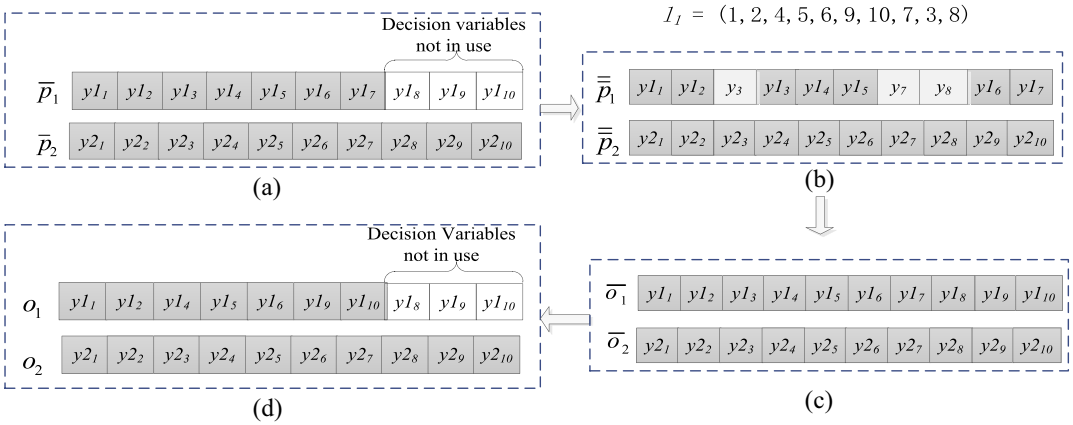

Fig. 4. Example illustrating the decision variable shuffling strategy. (a) Two randomly selected individuals $\overline{p}_1$ and $\overline{p}_2$ from task1 and task2, where the search dimension of task1 and task2 is 7 and 10, respectively. (b) $\overline{\overline{p}}_1$, which is obtained by perturbing the order of the decision variables of $\overline{p}_1$, $\overline{\overline{p}}_2$ is equal to $\overline{p}_2$. (c) $\overline{o}_1, \overline{o}_2$, two offspring generated by $\overline{\overline{p}}_1$ and $\overline{\overline{p}}_2$. (d) If $\overline{o}_1$ is assigned to task1 and $\overline{o}_2$ assigned to task2, the decision variables of $\overline{o}_1$ which index are in the first 7 l_1 are extracted as the useful variables of o_1, while the order list of $\overline{o}_2$ remained unchanged.

TABLE I
BEST FITNESS OF TWO TASKS AND THE PERFORMANCE SCORE RESULTS OF THE THREE ALGORITHMS ON EIGHT TEST INSTANCES, WHERE THE BEST RESULTS ARE HIGHLIGHTED

Name	Task1				Task2				Performance Score		
	G-MFEA Mean(Std)	MFEA Mean(Std)		SOEA Mean(Std)	G-MFEA Mean(Std)	MFEA Mean(Std)		SOEA Mean(Std)	G-MFEA	MFEA	SOEA
F1	**0 (0)**	**0 (0)**	=	16.8 (9.66e-12) –	**0 (0)**	**0 (0)**	=	31.23 (14.99) –	**-12.54**	**-12.54**	25.08
F2	**0 (0)**	**0 (0)**	=	19.82 (0.064) –	**0 (0)**	**0 (0)**	=	107.91 (19.49) –	**-13.76**	**-13.76**	27.52
F3	**5.37 (0.92)**	20.72 (0.9358)	–	31.1 (1.45) –	**7.06 (1.56)**	11.44 (1.89)	–	68.43 (12.35) –	**-20.08**	-4.44	24.56
F4	137.75 (18.31)	146.26 (18.48)	–	**114.14** (18.84) +	**4.31e+03 (340.36)**	**4.66e+03 (539.27)**	–	5.67e+03 (234.84) –	**-5.96**	2.86	3.10
F5	**0 (0)**	105.56 (14.06)	–	143.34 (23.28) –	**0 (0)**	**0 (0)**	=	**0 (0)** =	**-24.4278**	12.05	12.37
F6	**3.07e-14 (1.94e-15)**	19.84 (0.0441)	–	19.94 (0.03) –	**2.41e-05 (5.4e-05)**	23.9 (3.33)	–	30.71 (0.75) –	**-13.45**	5.46	7.99
F7	**1.08e-11 (2.2e-12)**	1.02e-11 (0.58)	–	1.72e-11 (5.8e-12) –	**26.68 (1.6792)**	**27.13 (1.72)**	=	58.05 (5.41) –	**-12.06**	-10.43	22.49
F8	**105.13 12.10**	122.78 19.79	–	128.63 18.76 –	2.46e+03 **328.40**	2.69e+03 257.26	–	**2.34e+03 284.93** +	**-7.76**	7.54	0.21

is the maximum dimensionality of all tasks. For single task optimization, the pm $= 1/D$, where D denotes the number of decision variables.

The population size is set to be 100 and a maximum of 50 000 fitness evaluations are allowed. As suggested in [10], each individual in G-MFEA undergo a learning step via BFGS quasi-Newton method and the principle of Lamarckian [18], [19] in order to obtain high-quality solutions.

b) Parameters in G-MFEA: The percentage of the best solutions to estimate current optimums $\mu = 40\%$, as discussed and recommend in [36]. The threshold of triggering the variable translation strategy: $\varphi = 0.1$gen, where gen is the maximum number of generations. The frequency of changing the translate direction $\theta = 0.02$gen, where gen is the maximum number of generations. The scale factor sf in (4) is set to be 1.25.

3) Performance Indicators: In multitasking optimization, a performance score [33] evaluating an algorithm on all tasks is proposed, in addition to the obtained best fitness of each single task. Consequently, we adopt the same performance indicator to compare the three algorithms, MFEA, G-MFEA, and SOEA in this paper.

Let K be the number of tasks in a test problem, denoted by $T_1, T_2, \ldots, T_K$, and the best fitness obtained by an algorithm in a run for T_k be denoted as $I_k, k = 1, 2, \ldots, K$. If each algorithm is executed for L times, the performance score of the ith algorithm is defined as follows:

$$\text{score}_i = \sum_{k=1}^{K} \sum_{l=1}^{L} \frac{I_l - \mu_k}{\delta_k} \tag{7}$$

where μ_k and δ_k are the mean and standard value of I_l obtained by all compared algorithms. The smaller the performance score, the better an algorithm is.

4) Results and Analysis: The mean and standard deviations of the best fitness of each task obtained by each algorithm and the performance score of three algorithms G-MFEA, MFEA, SOEA on the eight test instances are presented in Table I and Fig. A in the supplementary materials, respectively, where the results are obtained over 20 independent runs. In Table I, the best result of each test instance is highlighted. The Wilcoxon

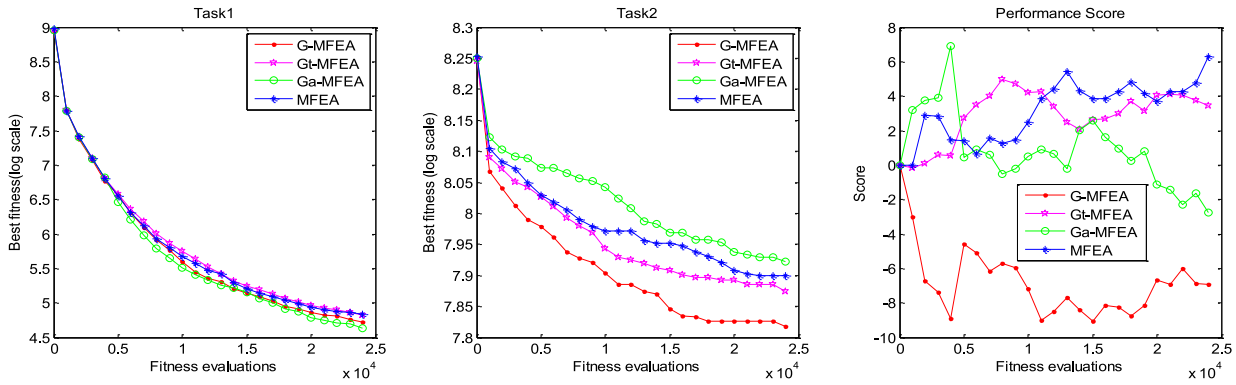

Fig. 5. Comparison of G-MFEA, the decision variable translation strategy: Gt-MFEA, the decision variable shuffling strategy: Ga-MFEA, MFEA, on F8 over ten independent runs.

rank sum test is also performed at a significance level of 0.05. In the table, symbols "+," "−," "=" denote that the result is significantly better, significantly worse, or comparable with that of G-MFEA, respectively. From these results, the following observations can be made.

1) The overall performance of both the G-MFEA and MFEA are better than SOEA. These results confirm that implicit knowledge transfer across tasks in multitasking optimization is capable of accelerating convergence and finding better solutions.
2) G-MFEA and MFEA perform similarly on F1 and F2. This can be attributed to fact that these two instances have the same optimum and search dimension, thus both G-MFEA and MFEA can work well on this type of MFO.
3) For F3 and F4, whose locations of the optimum are different, the final performance of G-MFEA is better than that of MFEA. Fig. A in the supplementary materials also shows that G-MFEA continues to converge quickly at the later optimization stage, whereas MFEA nearly stagnates. This is due to the fact that offspring generated in the translated solution space enables G-MFEA to transfer knowledge between tasks more effectively, thereby striking a better balance between diversity and convergence and enhancing the convergence.
4) G-MFEA achieves much better performance than MFEA on F5 and F6. Taking a closer look at the results on F5, it can be seen that SOEA is able to quickly converge to the global optimum of task2, as shown in Table I and Fig. A in the supplementary materials. By contrast, G-MFEA has achieved a better solution for task1 than MFEA. The results on F6 show that G-MFEA converges rapidly on both tasks even though MFEA and SOEA have not improved much. This may be attributed to the decision variable shuffling strategy in G-MFEA that keeps the better quality of solutions transferred from task1 to task2 by replacing the decision variables not in use for task1 with corresponding information of task2.
5) On F7 and F8, the two most challenging problems, G-MFEA outperforms MFEA and SOEA, owing to the decision variable shuffling strategy and the decision variable translation strategy in G-MFEA.

To further demonstrate the contribution of the two strategies, two variants of G-MFEA, one with the decision variable translation strategy only, termed Gt-MFEA, and the other with the decision variable shuffling strategy only, termed Ga-MFEA, are compared with G-MFEA, MFEA, and G-MFEA on F8. The results from the compared algorithms over ten independent runs are presented in Fig. 5, showing that Ga-MFEA and G-MFEA perform better than Gt-MFEA and MFEA on task1. This indicates that the decision variable shuffling strategy is able to maintain the quality of the transferred knowledge between tasks. In addition, G-MFEA and Gt-MFEA outperform Ga-MFEA and MFEA on task2 and the performance score of Ga-MFEA and Gt-MFEA are not better than G-MFEA, confirming that search operator has benefited from the decision variable translation strategy.

5) Sensitivity Analysis of the Control Parameters: We first examine the sensitivity of the performance to the threshold for starting the variable translation strategy. We set $\varphi = 0.05\text{gen}, 0.1\text{gen}, 0.3\text{gen}, 0.5\text{gen}$ in the experiments. The test instances and other parameters remain the same as those in Section V-C2. Fig. 6 shows the mean best fitness values obtained by G-MFEA with different thresholds φ on F3 and F8. Fig. 6 shows that G-MFEA with different φ has different performance on each instance. Moreover, a small threshold φ is appropriate for G-MFEA. When $\varphi = 0.1\text{gen}$, G-MFEA has the fastest convergence speed on F3 and F8.

Second, we test the sensitivity of the algorithm to the frequency of changing the translate direction. To this end, $\theta = 1, 0.02\text{gen}, 0.05\text{gen}, 0.1\text{gen}$ are separately tested on F3 and F8. The remaining parameters are the same as those in Section V-C2. Fig. 7 shows the mean best fitness values obtained by G-MFEA with different thresholds θ on each test instance. From Fig. 7, we can find that on F3 and F8, the final results obtained by G-MFEA with different θ for each instance are different. Moreover, it can also be discovered that when $\theta = 0.02\text{gen}$, G-MFEA has a relative better performance on all the instances. Fig. 7 reveals that the appropriate setting of θ is problem-dependent.

Third, we examine the influence of the scale factor sf in (4) on the performance. To this end, we compare the

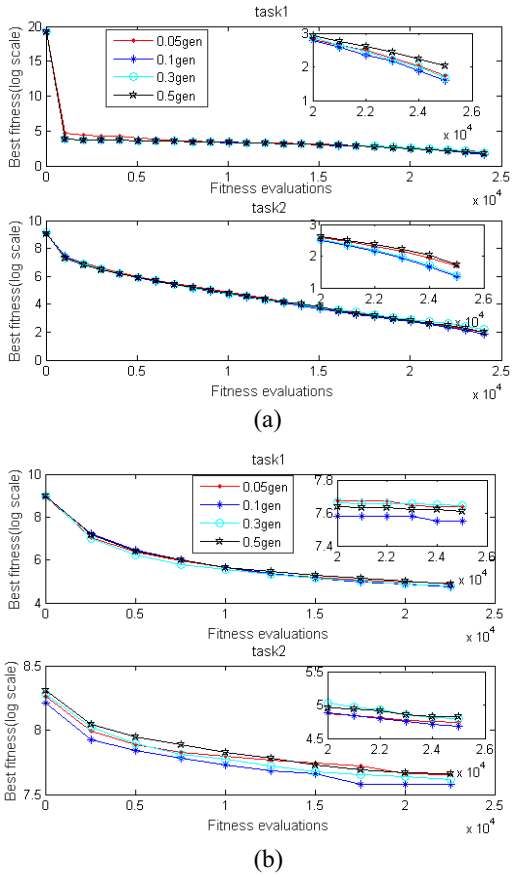

Fig. 6. Mean best fitness obtained by G-MFEA with different φ over 30 runs on (a) F3 and (b) F8.

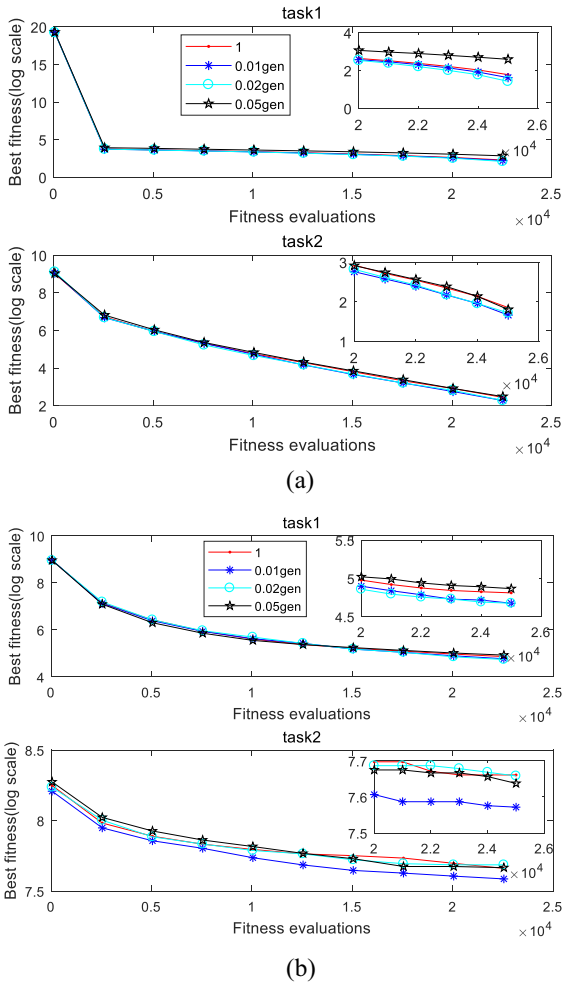

Fig. 7. Mean best fitness obtained by G-MFEA with different θ over 30 runs on (a) F3 and (b) F8.

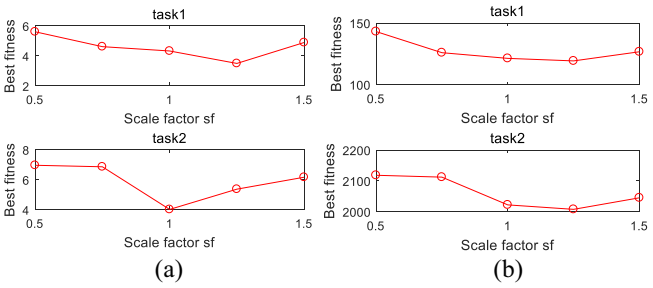

Fig. 8. Mean best fitness obtained by G-MFEA with different sf over 30 runs on (a) F3 and (b) F8.

performance of G-MFEA using five different scale factors sf $= 0.5, 0.75, 1, 1.25, 1.5$ on F3 and F8. The other parameter settings of G-MFEA are the same as those described in Section V-C2. Fig. 8 summarizes the results of average best fitness values obtained by G-MFEA over 30 independent runs. According to the results shown in Fig. 8, G-MFEA is able to achieve the best performance when the scale factor is around 1.25, which is adopted in all empirical studies.

IV. Multitasking Optimization for Expensive Optimization Problems

Computationally expensive optimization problems have been attracted increasing attention in the past decades as they commonly exist in the real world [2]–[4]. Most conventional EAs cannot be directly employed for solving such computationally expensive problems as they often need a large number of fitness evaluations. One popular way to address this issue is to replace part of the computationally expensive fitness evaluations with surrogate models to reduce the computation time. These algorithms are usually referred to as surrogate assisted EAs (SAEAs). In the literature of SAEAs, the research effort has been dedicated to enhancing the quality of surrogate models and designing efficient model management strategies [6]–[8], [31], [32].

According to multitasking optimization, the convergence of optimization can be significantly enhanced by simultaneously optimizing multiple tasks thanks to knowledge transfer between the tasks. Inspired by this idea, this section proposes to speed up the optimization of expensive optimization by transferring knowledge from computationally cheap problems (C-tasks) to computationally expensive problem (E-task), thereby accelerating the convergence of the E-tasks. Consequently, the required number of fitness evaluations of the expensive problem to achieve acceptable solutions can be reduced.

Knowledge transfer in multitasking is realized by exchanging solutions of different tasks. Therefore, the more and better

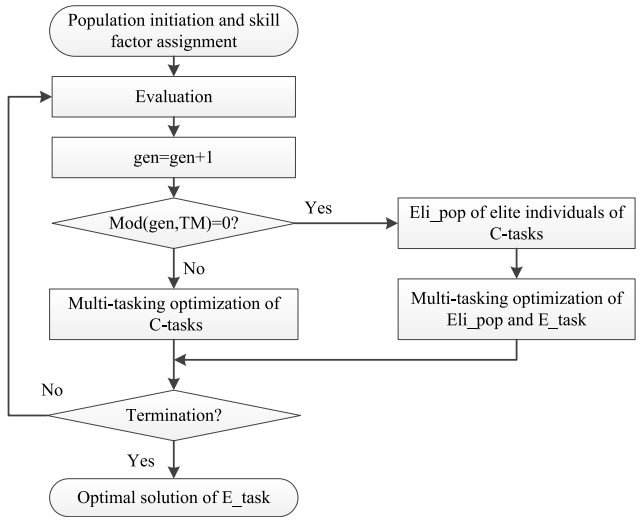

Fig. 9. Flowchart of MCEEA.

solutions of one task can obtain, the more the solutions of other tasks will benefit from those of this task. To accelerate the convergence in solving the E-task with the help of C-tasks, high quality knowledge transfer from C-tasks to the E-task becomes essential. To this end, MCEEA incorporates multiple computationally cheap problems to ensure the diversity of the transferred knowledge. Note that multifidelity optimization algorithms [34], [35] are also able to take advantage of multiple low-fidelity models to speed up convergence in solving computationally expensive problems, they work very differently from the proposed MCEEA.

In addition, in order to ensure positive knowledge transfer, the results of C-tasks after a certain number of generations (TM generations) of optimization is stored as the useful knowledge. As knowledge is acquired from the individuals of the C-tasks, MCEEA selects the best individuals of the C-tasks at every TM generations and transfers it to the individuals solving the E-task. The flowchart of the proposed MCEEA is presented in Fig. 9.

Below is a description of the main components of MCEEA.

Step 1: Initialize the population and assign the skill factors. Denote population size of each task to be *Sub*_NP.

Step 2: Evaluate the individuals.

Step 3: Once the C_tasks have been optimized for TM generations, go to steps 4 and 5, otherwise go to step 6.

Step 4: *Sub*_NP elite individuals are selected from C-task individuals according to the scalar fitness to make up the Eli_pop.

Step 5: The E-task and the Eli_pop are optimized by G-MFEA.

Step 6: All the C-tasks are optimized by G-MFEA.

Step 7: If the termination condition is fulfilled, terminate the optimization and output the optimal solution of the E-task, otherwise return to step 3.

V. Experimental Results and Discussion

In this section, the performance of MCEEA is examined and compared with the single task algorithm SOEA.

A. Experimental Settings

1) Test Instances: The proposed MCEEA framework for solving expensive problem is examined on eight test instances we construct for evaluating the performance of MCEEA. In each instance, there exists one computationally expensive problem, denoted as the E-task, and three computationally cheap problems, termed C-task1, C-task2, and C-task3, as presented in Table II. In addition, we also constructed two test instances using the artificial test problems designed in [35], denoted as Real1 and Real2. The function fidelity 6 is considered to be a computationally expensive problem, while Fidelity 1, 4, and 5 are cheap problems in Real1. In Real2, Fidelity 5 is the expensive problem and Fidelity 2 and Fidelity 3 are considered to be the computationally cheap problems.

2) Parameter Settings: To reduce the needed number of fitness evaluations of computationally expensive problems, a small population is adopted for MCEEA in the experiments. The subpopulation size of each task is set to 16, resulting in a total number of 64 individuals in the whole population for one E-task and three C-tasks. The maximum number of fitness evaluations for the E-task is set to 3200, and the optimization is terminated once this computational budget is exhausted. Knowledge transfer from the C-tasks to the E-task is carried out at every $\mathrm{TM} = 50$ generations.

3) Performance Indicators: In this paper, the obtained best fitness of the E-task is applied for evaluating the performance the compared algorithms.

B. Results and Analysis of MCEEA

The results of the E-task obtained by the two algorithms over 20 independent runs on the eight test instances are summarized in Table III, the convergence profiles of the two algorithms are plotted in Fig. 10. In Table III, the best result of each test instance is highlighted, the Wilcoxon rank sum test at a significance level of 0.05 is also adopted, where symbols $+$, $-$, $=$ denote that the result is significantly better, significantly worse or comparable with that of MCEEA. Results in Table III and Fig. 10 demonstrate that MCEEA performs much better than SOEA on all test instances in terms of both convergence speed and the final results. This indicates that the knowledge transferred from the C-tasks can significantly enhance the convergence of the E-task in MCEEA. According to the different characteristics of the test instances, the following three observations can be made.

1) MCE1 and MCE2 are test instances, where all C-tasks and the E-task have the same number of decision variables and the same optimums. For MCE3 to MCE5, the optimum of the C-tasks are the same as that of the E-task, while the dimensions of the C-tasks are different from that of the E-task. For MCE8, the optimum of the C-tasks deviates slightly from that of the E-task. On these six test instances, the results show that MCEEA achieves significantly better performance than SOEA. The promising performance of MCEEA may be attributed to the high-quality knowledge transferred from the C-tasks to the E-task during the optimization process.

TABLE II
EIGHT TEST INSTANCES OF MCEEA, WHERE **M** IS THE RANDOMLY ROTATED MATRIX, **o** IS THE OPTIMAL SOLUTION

Name	Function	Name	Function
MCE1	$E\_task = \text{Ackley}(\mathbf{z})$ $D = 50, \mathbf{x} \in [-50,50]^D, \mathbf{z} = \mathbf{M}(\mathbf{x}-\mathbf{o}), \mathbf{o} = 0^D$	**MCE5**	$E\_task = Rosenbrock(\mathbf{z})$ $D = 50, \mathbf{x} \in [-50,50]^D, \mathbf{z} = \mathbf{x}-\mathbf{o}, \mathbf{o} = 0^D$
	$C\_task1 = Rastrigin(\mathbf{z})$ $D = 50, \mathbf{x} \in [-50,50]^D, \mathbf{z} = \mathbf{M}(\mathbf{x}-\mathbf{o}), \mathbf{o} = 0^D$		$C\_task1 = Griweank(\mathbf{z})$ $D = 35, \mathbf{x} \in [-50,50]^D, \mathbf{z} = \mathbf{M}(\mathbf{x}-\mathbf{o}), \mathbf{o} = 0^D$
	$C\_task2 = Griweank(\mathbf{z})$ $D = 50, \mathbf{x} \in [-100,100]^D, \mathbf{z} = \mathbf{M}(\mathbf{x}-\mathbf{o}), \mathbf{o} = 0^D$		$C\_task2 = \text{Ackley}(\mathbf{z})$ $D = 25, \mathbf{x} \in [-500,500]^D, \mathbf{z} = \mathbf{M}(\mathbf{x}-\mathbf{o}), \mathbf{o} = 0^D$
	$C\_task3 = Sphere(\mathbf{z})$ $D = 50, \mathbf{x} \in [-100,100]^D, \mathbf{z} = \mathbf{M}(\mathbf{x}-\mathbf{o}), \mathbf{o} = 0^D$		$C\_task3 = Sphere(\mathbf{z})$ $D = 50, \mathbf{x} \in [-100,100]^D, \mathbf{z} = \mathbf{M}(\mathbf{x}-\mathbf{o}), \mathbf{o} = 5^D$
MCE2	$E\_task = Schwefel(\mathbf{z})$ $D = 50,\ \mathbf{x} \in [-500,500]^D, \mathbf{z} = \mathbf{x}-\mathbf{o}, \mathbf{o} = 0^D$	**MCE6**	$E\_task = Griweank(\mathbf{z})$ $D = 50, \mathbf{x} \in [-100,100]^D, \mathbf{z} = \mathbf{M}(\mathbf{x}-\mathbf{o})$ $\mathbf{o} = (o_1, o_2, ..., o_D)$ randomly distributed in $[-100,100]$
	$C\_task1 = Griweank(\mathbf{z})$ $D = 50, \mathbf{x} \in [-5,5]^D, \mathbf{z} = \mathbf{M}(\mathbf{x}-\mathbf{o}), \mathbf{o} = 4.25^D$		$C\_task1 = \text{Ackley}(\mathbf{z})$ $D = 50, \mathbf{x} \in [-10,10]^D, \mathbf{z} = \mathbf{M}(\mathbf{x}-\mathbf{o})$ $\mathbf{o} = (o_1, o_2, ..., o_D)$ randomly distributed in $[-10,10]$
	$C\_task2 = \text{Ackley}(\mathbf{z})$ $D = 50, \mathbf{x} \in [-5,5]^D, \mathbf{z} = \mathbf{M}(\mathbf{x}-\mathbf{o}), \mathbf{o} = 4.25^D$		$C\_task2 = Sphere(\mathbf{z})$ $D = 50, \mathbf{x} \in [-100,100]^D, \mathbf{z} = \mathbf{M}(\mathbf{x}-\mathbf{o})$ $\mathbf{o} = (o_1, o_2, ..., o_D)$ randomly distributed in $[-100,100]$
	$C\_task3 = Sphere(\mathbf{z})$ $D = 50, \mathbf{x} \in [-100,100]^D, \mathbf{z} = \mathbf{M}(\mathbf{x}-\mathbf{o}), \mathbf{o} = 88^D$		$C\_task3 = Rastrigin(\mathbf{z})$ $D = 50,\ \mathbf{x} \in [-5,5]^D,\ \mathbf{z} = \mathbf{x}-\mathbf{o}$ $\mathbf{o} = (o_1, o_2, ..., o_D)$ randomly distributed in $[-5,5]$
MCE3	$E\_task = Weierstrass(\mathbf{z})$ $D = 50, \mathbf{x} \in [-0.5,0.5]^D, \mathbf{z} = \mathbf{M}(\mathbf{x}-\mathbf{o}), \mathbf{o} = 0^D$	**MCE7**	$E\_task = Weierstrass(\mathbf{z})$ $D = 50, \mathbf{x} \in [-0.5,0.5]^D, \mathbf{z} = \mathbf{M}(\mathbf{x}-\mathbf{o})$ $\mathbf{o} = (o_1, o_2, ..., o_D)$ randomly distributed in $[-0.5,0.5]$
	$C\_task1 = Griewank(\mathbf{z})$ $D = 25, \mathbf{x} \in [-50,50]^D, \mathbf{z} = \mathbf{M}(\mathbf{x}-\mathbf{o}), \mathbf{o} = 0^D$		$C\_task1 = \text{Ackley}(\mathbf{z})$ $D = 50, \mathbf{x} \in [-5,5]^D, \mathbf{z} = \mathbf{M}(\mathbf{x}-\mathbf{o}), \mathbf{o} = -2.5^D$
	$C\_task2 = Ackley(\mathbf{z})$ $D = 25, \mathbf{x} \in [-50,50]^D, \mathbf{z} = \mathbf{M}(\mathbf{x}-\mathbf{o}), \mathbf{o} = 0^D$		$C\_task2 = Sphere(\mathbf{z})$ $D = 50, \mathbf{x} \in [-50,50]^D, \mathbf{z} = \mathbf{M}(\mathbf{x}-\mathbf{o}), \mathbf{o} = 25^D$
	$C\_task3 = Rastrigin(\mathbf{z})$ $D = 50, \mathbf{x} \in [-5,5]^D, \mathbf{z} = \mathbf{M}(\mathbf{x}-\mathbf{o}), \mathbf{o} = 0^D$		$S\_task3 = Rastrigin(\mathbf{z})$ $D = 50, \mathbf{x} \in [-5,5]^D, \mathbf{z} = \mathbf{x}-\mathbf{o}, \mathbf{o} = 0^D$
MCE4	$E\_task = Rastrigin(\mathbf{z})$ $D = 50, \mathbf{x} \in [-50,50]^D, \mathbf{z} = \mathbf{M}(\mathbf{x}-\mathbf{o}), \mathbf{o} = 0^D$	**MCE8**	$E\_task = Rastrigin(\mathbf{z}(1:25)) + \text{Rosenbrock}(\mathbf{z}(26:50))$ $D = 50, \mathbf{x} \in [-50,50]^D, \mathbf{z} = \mathbf{x}-\mathbf{o}$ $\mathbf{o}(1:25) = 10^{D/2}, \mathbf{o}(25:50) = 0^{D/2}$
	$C\_task1 = Schwefel(\mathbf{z})$ $D = 50,\ \mathbf{x} \in [-500,500]^D, \mathbf{z} = \mathbf{x}-\mathbf{o}, \mathbf{o} = 0^D$		$C\_task1 = Griewank(\mathbf{z})$ $D = 35, \mathbf{x} \in [-50,50]^D, \mathbf{z} = \mathbf{M}(\mathbf{x}-\mathbf{o}), \mathbf{o} = 10^D$
	$C\_task2 = Griewank(\mathbf{z})$ $D = 50, \mathbf{x} \in [-100,100]^D, \mathbf{z} = \mathbf{M}(\mathbf{x}-\mathbf{o}), \mathbf{o} = 0^D$		$C\_task2 = Ackley(\mathbf{z})$ $D = 25, \mathbf{x} \in [-5,5]^D, \mathbf{z} = \mathbf{M}(\mathbf{x}-\mathbf{o}), \mathbf{o} = 0^D$
	$C\_task3 = Weierstrass(\mathbf{z})$ $D = 25, \mathbf{x} \in [-0.5,0.5]^D, \mathbf{z} = \mathbf{M}(\mathbf{x}-\mathbf{o}), \mathbf{o} = 0^D$		$C\_task3 = Sphere(\mathbf{z})$ $D = 50, \mathbf{x} \in [-50,50]^D, \mathbf{z} = \mathbf{M}(\mathbf{x}-\mathbf{o})$ $\mathbf{o} = (o_1, o_2, ..., o_D)$ randomly distributed in $[0,10]$

2) The locations of the optimums of the E-tasks of MCE7 and MCE7 are randomly distributed in their search spaces. The locations of the optimums of all C-tasks in MCE6 are also randomly distributed in the search space. The optimums of C-tasks of MCE7 are located at $(0.5, 0.5, \ldots, 0.5)$, $(-0.5, -0.5, \ldots, -0.5)$, and $(0, 0, \ldots, 0)$, respectively, in the unified search space. The comparative results of MCEEA and SOEA on these two instances show that MCEEA outperforms SOEA. This indicates that although the optimums of the C-tasks and the E-task are located in different positions, the E-task can still benefit from the C-tasks during the search process. The above results confirm that MCEEA can enhance the convergence performance of the computationally expensive problems by transferring knowledge from the computationally cheap problems.

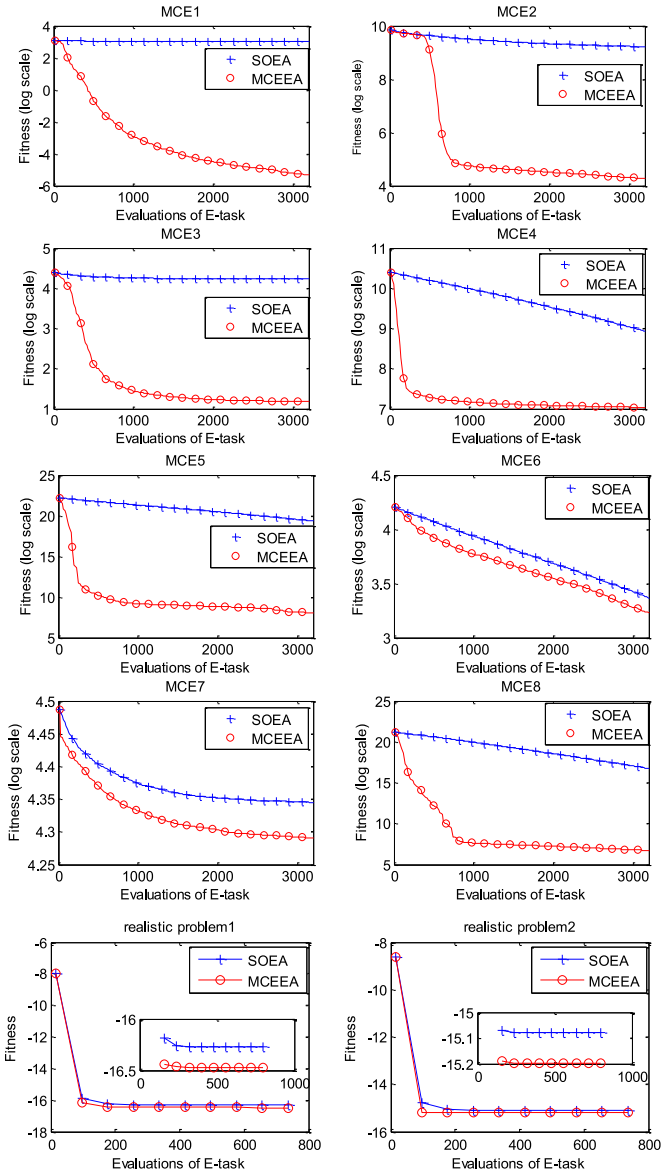

Fig. 10. Statistical results of E-task on the eight test instances. In each subplot, the x-axis represents the evaluations of E-task and the y-axis is the log mean value of best fitness obtained via 20 independent runs.

3) In the last two test instances Real1 and Real2, all C-tasks as well as the E-task have the same optimum. Note that the C-tasks are considered to be the low-fidelity models of the E-task. The results from these test instances are presented in Table III and Fig. 10, revealing that the MCEEA is able to converge faster and find better solutions for the E-task.

C. Parameters Sensitivity Analysis

1) Influence of the Frequency in Knowledge Transfer (TM): In MCEEA, the C-task is optimized for once at every TM generations. The purpose of this is to obtain sufficiently good knowledge from C-tasks to be transferred to the E-task. Therefore, TM has a great impact on the performance of MCEEA as it can influence the quality of knowledge transferred from the C-tasks. Fig. 11 plots the best fitness of the E-task obtained on the three instances when TM is set to 1, 20, 50, 75, and 100, averaged over 30 runs. The three instances are: 1) the E-task and the third C-task of test instance MCE1; 2) the E-task and the second C-task of test instance MCE4; and 3) the E-task and the first C-task of test instance MCE6.

From the results shown in Fig. 11, we can see that the performance of the E-task enhances rapidly as TM increases. However, when TM reaches 50, the performance no longer improves significantly. This implies that knowledge transferred from the C-tasks is good enough after 50 generations to be benefitted by the E-task.

2) Influence of the Number of C-Tasks: In this paper, multiple C-tasks (three tasks in this experiment) are integrated to assist the optimization of the E-task in MCEEA. To further demonstrate the benefit of using multiple C-tasks, the results on two test cases are compared when multiple C-tasks are used and one single C-task is adopted. Fig. 12 plots the statistical results on instances MCE1 and MCE6.

From the results shown in Fig. 12(a), it can be seen that although each C-task has a positive influence on the E-task throughout the optimization process, the performance of the algorithm using three C-tasks is much better than using any of the single C-task. The results on MCE6 in Fig. 12(b) can be discussed in two different stages in the optimization process, the early stage before 1000 fitness evaluations and the later stage after 1000 evaluations. In the early stage, C-task1 does not have much influence while C-task3 has relatively larger positive influence on the E-task. The performance of using all C-tasks (C-task1, C-task2, C-task3) is much better than C-task1 but a little worse than C-task3. The reason may lie in the fact that the E-task has not benefited much from C-task1. From the comparative results of the later stage, it can be seen that the overall performance using three C-tasks is better than using any single C-task, implying that knowledge from the three C-tasks has positive influence on the optimization of the E-task.

The results of MCE1 and MCE6 in Fig. 12 also demonstrate that the contribution to the E-task of each C-task may be different, where the contribution of C-task2 and C-task3 to E-tasks are more than C-task1 in MCE1 and MCE6. To take a closer look at these results, Fig. 13 plots the convergence trend of each C-task in MCE1 and MCE6 averaged over 20 runs, which are scaled between [0,100] for fair comparisons. From Fig. 13, we can see that the convergence speed on C-task2

TABLE III
BEST FITNESS OF E-TASK OF THE TWO ALGORITHMS ON EIGHT TEST INSTANCES, WHERE THE BEST RESULTS ARE HIGHLIGHTED

Name	MCEEA Mean (Std)	SOEA Mean (Std)	
MCE1	**0.0048 (8.9e-04)**	20.59 (0.08)	–
MCE2	**71.56 (10.21)**	1.01e+04 (797.84)	–
MCE3	**3.27 (1.19)**	68.49 (4.43)	–
MCE4	**1.14e+03 (207)**	7.58e+03 (812)	–
MCE5	**3.15e+03 (928)**	2.55e+08 (6e+07)	–
MCE6	**25.11 (1.16)**	28.95 (3.24)	–
MCE7	**73.02 (0.97)**	77.10 (2.99)	–
MCE8	**839.62 (138)**	1.9e+07 (1.3e+07)	–
Real1	**-16.47 (9.0e-08)**	-16.27 (0.3999)	–
Real2	**-15.20(3.2e-08)**	-15.14(0.0720)	–

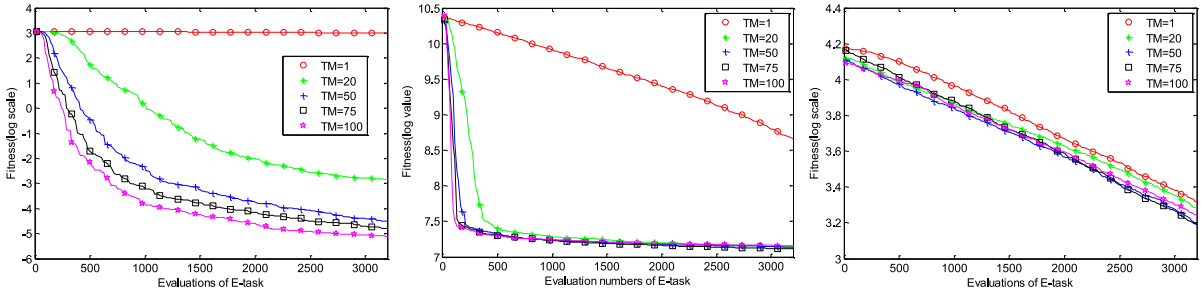

Fig. 11. Performance of MCEEA over the changing of multitasking of T-task TM. (a) E-task and the third C-task on test instance MCE1. (b) E-task and the second C-task on test instance MCE4. (c) E-task and the first C-task on test instance MCE6.

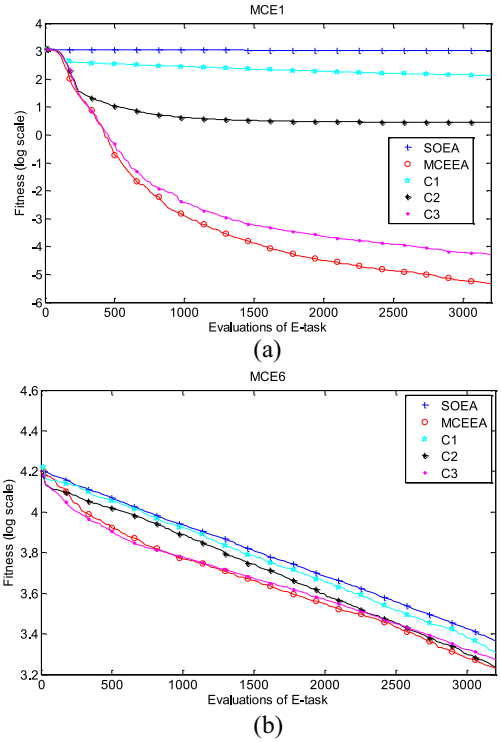

Fig. 12. Comparative results of the multiple integrated C-tasks averaged over 20 runs on instances MCE1(a) and MCE6(b). MCEEA, C1, C2, and C3 denotes the results of the integrated C-tasks, the first C-task (C-task1), the second C-task (C-task2) and the third C-task (C-task3), respectively.

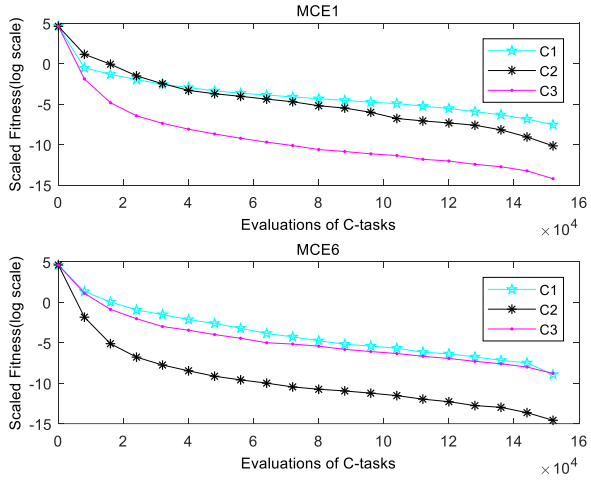

Fig. 13. Scaled convergence trend of each C-task averaged over 20 runs in MCE1 and MCE6. C1, C2, and C3 denote the results of C-task1, C-task2, and C-task3, respectively.

and C-task3 are much faster than on C-task1, which implies that C-task2 and C-task3 are easier to solve among the three C-tasks in both MCE1 and MCE6. From above observations we may conclude that the easier the C-tasks are, the more effectively they can speed up the convergence of the E-task.

From the above analysis, we conclude that MCEEA may benefit from using multiple computationally cheap problems in two different cases.

1) In case every single C-task is able to have positive influence on the E-task, multiple C-tasks can more significantly speed up the convergence of the E-task.
2) In case some C-tasks are not able to positively influence the convergence of the E-task, the E-task can still obtain useful knowledge from a combination of the C-tasks.

This means that use of multiple C-tasks can ensure a more robust performance enhancement of the E-task.

D. Remarks on C-Task Selection

Selecting C-tasks is critical for the success of multitasking optimization for assisting solving expensive problems. Actually, some ideas for quantifying the synergies between different optimization tasks have been discussed in [33]. The selection of C-tasks in the context of this paper may be slightly different from other multitasking optimization problems for two reasons. First, we are mainly concerned with the convergence of the E-task and the slow-down in the convergence of the C-tasks is less problematic. Second, since the computational budget for the E-task is very limited, it might be not affordable to test the similarity between the E-task and C-tasks based on a large number of fitness evaluations as done in [33]. Based on our empirical results in Sections V-B and V-C, we can conclude that the C-tasks should have a similar optimum to that of the E-task, but since the optimum of the E-task is unknown, it is recommended that a large number of C-tasks having slightly different fitness landscapes should be used. In addition, the C-tasks to be used for speeding up the E-task should be easy tasks, i.e., the EA should be able

to converge quickly on the C-tasks. Although it is nontrivial in general to select appropriate C-tasks since the optimum of the E-task is unknown, low-fidelity tasks of the original optimization problem may serve very well as the C-tasks in expensive optimization.

VI. Conclusion

In this paper, a multitasking optimization approach for reducing the computational costs in optimization of expensive computational problems, termed MCEEA, has been suggested. MCEEA allows knowledge to be transferred from computationally cheap optimization problems to expensive optimization problem, leading to a faster convergence of the expensive problem. In MCEEA, multiple computationally cheap problems are optimized together with the expensive problem to generate diverse knowledge for the optimization of the expensive problem, which is made possible by a generalized multitasking EA, termed G-MFEA. The main differences between the G-MFEA and existing multitasking EAs lie in newly introduced decision variable translation strategy and decision variable shuffling strategy. The decision variable translation strategy shifts the population to a new location so that the optimums of all optimization tasks have the same location in the translated solution space, which ensures positive knowledge transfer even if the optimums of the tasks in MFO are located in different positions. In addition, the decision variable shuffling strategy changes the order of the decision variables and replaces the variables not in use with corresponding useful information from other tasks, enhancing the quality of knowledge transfer between the tasks. The effectiveness of the G-MFEA and MCEEA are empirically verified and compared with existing algorithms on eight test instances.

Although MCEEA shows promises in solving computationally expensive problems on all test instances studied in this paper, it only considers the use of the elite individuals of the computationally cheap problems in assisting the expensive computational problem. In the future, we are going to develop strategies that are able to identify which computationally cheap problems are able to speed up the convergence of the expensive problem during the optimization process. The proposed framework also remains to be compared with conventional SAEAs on both single and multiobjective optimization problems. Indeed, surrogates can be seen as related tasks of an expensive optimization problem and thus it would be of great interest to explore the synergy between multitasking optimization and surrogate-assisted optimization.

References

[1] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford, U.K.: Oxford Univ. Press, 1996.

[2] G. Yu, T. Chai, and X. Luo, "Multiobjective production planning optimization using hybrid evolutionary algorithms for mineral processing," *IEEE Trans. Evol. Comput.*, vol. 15, no. 4, pp. 487–514, Aug. 2011.

[3] S. M. K. Hasan, R. Sarker, D. Essam, and D. Cornforth, "Memetic algorithms for solving job-shop scheduling problems," *Memetic Comput.*, vol. 1, no. 1, pp. 69–83, 2009.

[4] D. Simon, "Biogeography-based optimization," *IEEE Trans. Evol. Comput.*, vol. 12, no. 6, pp. 702–713, Dec. 2008.

[5] D. Lim, Y. Jin, Y.-S. Ong, and B. Sendhoff, "Generalizing surrogate-assisted evolutionary computation," *IEEE Trans. Evol. Comput.*, vol. 14, no. 3, pp. 329–355, Jun. 2010.

[6] J. Knowles, "ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 1, pp. 50–66, Feb. 2006.

[7] H. Wang, Y. Jin, and J. O. Jansen, "Data-driven surrogate-assisted multiobjective evolutionary optimization of a trauma system," *IEEE Trans. Evol. Comput.*, vol. 20, no. 6, pp. 939–952, Dec. 2016.

[8] R. G. Regis, "Evolutionary programming for high-dimensional constrained expensive black-box optimization using radial basis functions," *IEEE Trans. Evol. Comput.*, vol. 18, no. 3, pp. 326–347, Jun. 2014.

[9] Y. Jin, M. Olhofer, and B. Sendhoff, "A framework for evolutionary optimization with approximate fitness functions," *IEEE Trans. Evol. Comput.*, vol. 6, no. 5, pp. 481–494, Oct. 2002.

[10] 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.

[11] 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, doi: 10.1109/TCYB.2016.2554622.

[12] L. Zhou *et al.*, "Evolutionary multitasking in combinatorial search spaces: A case study in capacitated vehicle routing problem," in *Proc. IEEE Symp. Series Comput. Intell. (SSCI)*, Athens, Greece, 2016, pp. 1–8.

[13] 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. (TENCON)*, 2016, pp. 3157–3164.

[14] A. Gupta, J. Mañdziuk, and Y.-S. Ong, "Evolutionary multitasking in bi-level optimization," *Complex Intell. Syst.*, vol. 1, nos. 1–4, pp. 83–95, 2015.

[15] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Syst.*, vol. 9, no. 2, pp. 115–148, 1995.

[16] N. Hansen, S. Finck, R. Ros, and A. Auger, "Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions," INRIA, Rocquencourt, France, Rep. RR-6829, 2009.

[17] R. Hinterding, "Gaussian mutation and self-adaption for numeric genetic algorithms," in *Proc. IEEE Int. Conf. Evol. Comput.*, Perth, WA, Australia, Nov. 1995, pp. 384.

[18] R. Meuth, M.-H. Lim, Y.-S. Ong, and D. C. Wunsch, "A proposition on memes and meta-memes in computing for higher-order learning," *Memetic Comput.*, vol. 1, no. 2, pp. 85–100, 2009.

[19] Y. S. Ong and A. J. Keane, "Meta-Lamarckian learning in memetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 99–110, Apr. 2004.

[20] A. Gupta and Y.-S. Ong, "Genetic transfer or population diversification? Deciphering the secret ingredients of evolutionary multitask optimization," in *Proc. IEEE Symp. Comput. Intell. (SSCI)*, Athens, Greece, 2016, pp. 1–7.

[21] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Comput.*, vol. 9, no. 1, pp. 3–12, 2005.

[22] Y. S. Ong, P. B. Nair, and A. J. Keane, "Evolutionary optimization of computationally expensive problems via surrogate modeling," *AIAA J.*, vol. 41, no. 4, pp. 687–696, 2003.

[23] W. Gong, A. Zhou, and Z. Cai, "A multioperator search strategy based on cheap surrogate models for evolutionary optimization," *IEEE Trans. Evol. Comput.*, vol. 19, no. 5, pp. 746–758, Oct. 2015.

[24] J. Knowles, "ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 1, pp. 50–66, Feb. 2006.

[25] C. Sun, Y. Jin, R. Cheng, J. Ding, and J. Zeng, "Surrogate-assisted cooperative swarm optimization of high-dimensional expensive problems," *IEEE Trans. Evol. Comput.*, vol. 21, no. 4, pp. 644–660, Aug. 2017, doi: 10.1109/TEVC.2017.2675628.

[26] T. Chugh, Y. Jin, K. Miettinen, J. Hakanen, and K. Sindhya, "A surrogate-assisted reference vector guided evolutionary algorithm for computationally expensive many-objective optimization," *IEEE Trans. Evol. Comput.*, to be published, doi: 10.1109/TEVC.2016.2622301.

[27] A. Ratle, "Kriging as a surrogate fitness landscape in evolutionary optimization," *Artif. Intell. Eng. Design Anal. Manuf.*, vol. 15, no. 1, pp. 37–49, 2001.

[28] Z. Zhou, Y. S. Ong, P. B. Nair, A. J. Keane, and K. Y. Lum, "Combining global and local surrogate models to accelerate evolutionary optimization," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 37, no. 1, pp. 66–76, Jan. 2007.
[29] S. Z. Martínez and C. A. C. Coello, "MOEA/D assisted by RBF networks for expensive multi-objective optimization problems," in *Proc. 15th Annu. Conf. Genet. Evol. Comput*, Amsterdam, The Netherlands, 2013, pp. 1405–1412.
[30] K. C. Giannakoglou, "Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence," *Progr. Aerosp. Sci.*, vol. 38, no. 1, pp. 43–76, 2002.
[31] K. De Grave, J. Ramon, and L. De Raedt, "Active learning for high throughput screening," in *Proc. 12th Int. Conf. Disc. Sci*., Oct. 2008, pp. 185–196.
[32] C. Sun, Y. Jin, J. Zeng, and Y. Yu, "A two-layer surrogate-assisted particle swarm optimization algorithm," *Soft Comput.*, vol. 19, no. 6, pp. 1461–1475, 2015.
[33] B. Da *et al.*, "Evolutionary multitasking for single-objective continuous optimization: Benchmark problems, performance metric, and baseline results," Nanyang Technol. Univ., Singapore, Rep., 2016. [Online]. Available: http://www.cil.ntu.edu.sg/mfo/download.html
[34] S. J. Leary, A. Bhaskar, and A. J. Keane, "A knowledge-based approach to response surface modelling in multifidelity optimization," *J. Glob. Optim.*, vol. 26, no. 3, pp. 297–319, 2003.
[35] J. Branke, M. Asafuddoula, K. S. Bhattacharjee, and T. Ray, "Efficient use of partially converged simulations in evolutionary optimization," *IEEE Trans. Evol. Comput.*, vol. 21, no. 1, pp. 52–64, Feb. 2017.
[36] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evol. Comput.*, vol. 9, no. 2, pp. 159–195, 2001.
[37] J. H. Zar, "Significance testing of the spearman rank correlation coefficient," *J. Amer. Stat. Assoc.*, vol. 67, no. 339, pp. 578–580, 1972.
[38] K. K. Bali, A. Gupta, L. Feng, Y. S. Ong, T. P. Siew, "Linearized domain adaptation in evolutionary multitasking," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2017, pp. 1295–1302.
[39] A. Gupta, Y.-S. Ong, and L. Feng, "Insights on transfer optimization: Because experience is the best teacher," *IEEE Trans. Emerg. Topics Comput. Intell.*, to be published, doi: 10.1109/TETCI.2017.2769104.

Jinliang Ding (SM'14) received the Ph.D. degree in control theory and control engineering from Northeastern University, Shenyang, China, in 2012.

He is a Professor with the State Key Laboratory of Synthetical Automation for Process Industry, Northeastern University. He has authored or co-authored over 90 refereed journal papers and refereed papers at international conferences. He is also the Inventor or Co-Inventor of 17 patents. His current research interests include modeling, plant-wide control and optimization for the complex industrial systems, stochastic distribution control, and multiobjective evolutionary algorithms and its application.

Dr. Ding was a recipient of the Young Scholars Science and Technology Award of China in 2016, the National Science Fund for Distinguished Young Scholars in 2015, the National Technological Invention Award in 2013, two First-Prize of Science and Technology Award of the Ministry of Education in 2006 and 2012, respectively, and the Best Paper Award of 2011–2013 for Control Engineering Practice.

Cuie Yang received the B.Sc. degree from Henan Polytechnic University, Jiaozuo, China, in 2014 and the M.Sc. degree from Northeastern University, Shenyang, China, in 2016, where she is currently pursuing the Ph.D. degree with the School of Control Theory and Control Engineering.

Her current research interest is multitasking evolutionary optimization and its application.

Yaochu Jin (M'98–SM'02–F'16) received the B.Sc., M.Sc., and Ph.D. degrees from Zhejiang University, Hangzhou, China, in 1988, 1991, and 1996, respectively, and the Dr.-Ing. degree from Ruhr University Bochum, Bochum, Germany, in 2001.

He is a Professor of computational intelligence with the Department of Computer Science, University of Surrey, Guildford, U.K., where he heads the Nature Inspired Computing and Engineering Group. He is also a Finland Distinguished Professor funded by the Finnish Agency for Innovation (Tekes) and a Changjiang Distinguished Visiting Professor appointed by the Ministry of Education, Beijing, China. He has (co)-authored over 250 peer-reviewed journal and conference papers and been granted eight patents on evolutionary optimization. He has delivered 30 invited keynote speeches at international conferences. His current research interests include data-driven surrogate-assisted evolutionary optimization, evolutionary multiobjective optimization, evolutionary learning, and interpretable and secure machine learning.

Dr. Jin was a recipient of the 2018 IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION Outstanding Paper Award and the 2015 and 2017 *IEEE Computational Intelligence Magazine* Outstanding Paper Award. He is the Editor-in-Chief of the IEEE TRANSACTIONS ON COGNITIVE AND DEVELOPMENTAL SYSTEMS and the Co-Editor-in-Chief of *Complex & Intelligent Systems*. He is an IEEE Distinguished Lecturer from 2013 to 2015 and from 2017 to 2019 and past Vice President for Technical Activities of the IEEE Computational Intelligence Society from 2014 to 2015.

Tianyou Chai (M'90–M'97–F'08) received the Ph.D. degree in control theory and engineering from Northeastern University, Shenyang, China, in 1985.

He is the Founder and the Director of the Center of Automation, which became a National Engineering and Technology Research Center and a State Key Laboratory. He became a Professor with Northeastern University in 1988. He is the Director of the Department of Information Science, National Natural Science Foundation of China. He has authored 150 peer-reviewed international journal papers. He has developed control technologies with applications to various industrial processes. His current research interests include modeling, control, optimization, and integrated automation of complex industrial processes.

Dr. Chai was a recipient of one of three Best Papers for the Control Engineering Practice Paper Prize for 2011–2013 for his paper titled "Hybrid intelligent control for optimal operation of shaft furnace roasting process" and four Prestigious Awards from the National Science and Technology Progress and National Technological Innovation and the 2007 Industry Award for Excellence in Transitional Control Research from the IEEE Multiple Conference on Systems and Control for his contributions. He is a member of the Chinese Academy of Engineering and is an International Federation of Automatic Control Fellow.