Evolutionary Multitasking With Dynamic Resource Allocating Strategy

Maoguo Gong, *Senior Member, IEEE*, Zedong Tang, Hao Li, and Jun Zhang, *Fellow, IEEE*

***Abstract*—Evolutionary multitasking is a recently proposed paradigm to simultaneously solve multiple tasks using a single population. Most of the existing evolutionary multitasking algorithms treat all tasks equally and then assign the same amount of resources to each task. However, when the resources are limited, it is difficult for some tasks to converge to acceptable solutions. This paper aims at investigating the resource allocation in the multitasking environment to efficiently utilize the restrictive resources. In this paper, we design a novel multitask evolutionary algorithm with an online dynamic resource allocation strategy. Specifically, the proposed dynamic resource allocation strategy allocates resources to each task adaptively according to the requirements of tasks. We also design an adaptive method to control the resources invested into cross-domain searching. The proposed algorithm is able to allocate the computational resources dynamically according to the computational complexities of tasks. The experimental results demonstrate the superiority of the proposed method in comparison with the state-of-the-art algorithms on benchmark problems of multitask optimization.**

***Index Terms*—Dynamic resource allocation, evolutionary multitasking, multifactorial optimization (MFO), multitask optimization (MTO).**

I. Introduction

EVOLUTIONARY algorithms (EAs) are a kind of population-based meta-heuristics approaches for search and optimization [1]. Each individual of population is regarded as a candidate solution to the optimization problem. The fitness function estimates the quality of an individual. The algorithms apply the reproduction and selection operation to the individuals for generating better individuals. Currently, multitask optimization (MTO), and particularly evolutionary multitasking, [2] is a newly emerging research theme in the field of EAs. MTO algorithms utilize the underlying similarity of optimization tasks to efficiently solve heterogeneous tasks. This novel concept shares many commons with the multitask learning [3], [4] in the field of machine learning. The two regimes have the similar target on using the additional sources of information to improve the accuracy and learning/optimizing speed. A multitask system considers a compromise between the underlying similarity and the distinction of tasks. Evolutionary multitasking algorithms utilize the implicit parallelism of population-based searching algorithms to simultaneously solve multiple problems. Currently, there exist some works focusing on explicit reusing of the knowledge from the similar problems to solve similar incoming problems [5]–[7].

In the aim of efficiently realizing the evolutionary multitasking paradigm, multifactorial optimization (MFO) was proposed in [8]. MFO attempts to solve several heterogeneous tasks by using a single population. Each component task in the framework of MFO is referred as a "factor" which influences the evolution of a population. There is no need to obtain any knowledge on the relationship of tasks, as the MFO algorithms can handle implicit transfer of high-quality genetic building blocks. There are two superiorities of MFO over conventional batch optimization: 1) the increasing accuracies of component tasks because of knowledge transferring and reusing and 2) the reducing total make-span by reusing the knowledge existing in the similar tasks and not searching from scratch. In [8], an MFO algorithm incorporated with genetic algorithms (GAs), called multifactorial evolutionary algorithm (MFEA), was proposed. MFEA is the first instructive work in the area of evolutionary multitasking. MFEA, inspired by the concept of multifactorial inheritance, is characterized by the transfer of the biological and cultural building blocks. By computationally simulating the multifactorial inheritance, MFEA can seek optimums of multiple tasks simultaneously via the cross-domain implicit transfer of information. It has achieved a considerable convergence speed and accuracy. MFEA has already been used to solve single-objective optimization [8] and multiobjective optimization (MOO) tasks [9] and achieved some impressive results [10]–[13].

The previous approaches for evolutionary multitasking invest equal amount of resources in each task. However, tasks in evolutionary multitasking could have different hardness. If the computational resources are equally allocated to each task, a lot of resources will be wasted on the solved tasks while the

Manuscript received January 26, 2018; revised May 7, 2018, August 8, 2018, and November 9, 2018; accepted January 3, 2019. Date of publication January 17, 2019; date of current version October 1, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61772393, in part by the National Key Research and Development Program of China under Grant 2017YFB0802200, and in part by the Key Research and Development Program of Shaanxi Province under Grant 2018ZDXM-GY-045. *(Corresponding author: Maoguo Gong.)*

M. Gong, Z. Tang, and H. Li are with the School of Electronic Engineering, Key Laboratory of Intelligent Perception and Image Understanding, Ministry of Education, Xidian University, Xi'an 710071, China (e-mail: gong@ieee.org; omegatangzd@gmail.com; omegalihao@gmail.com).

J. Zhang is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510640, China (e-mail: junzhang@ieee.org).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author. This PDF document includes tables on the specifications and experimental results of the compared algorithms on the benchmark problems. This material is 580 KB in size.

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.2019.2893614

other tasks still require resources. So investing computational resources as per the requirements of tasks is straightforward and could be a promising avenue to improve the efficiency of the evolutionary multitasking algorithms. In this principle, the resources should be allocated to the task making a progress within the previous period. Although the resource allocation is so important for MTO, there is no attempt to design a proper strategy to address this issue for MTO.

In this paper, we propose a novel evolutionary multitasking algorithm with dynamic resource allocating strategy (MTO-DRA). The proposed method is characterized by the dynamic resource allocation according to the requirements of tasks. For the sake of not bothering to design a measure of hardness, we employ an online way to calculate the measure of each task's computational requirement and allocate resources as per the measure. Specifically, improvement of each task is calculated online in the process of evolution. Then, improvement of each task is transformed into the investing probability, named index of improvement, by a *softmax* function. In the proposed allocating strategy, a polling cycle is incorporated into the proposed algorithm to determine if a task can undergo search per resource allocation cycle. Then the task with great requirement of computational resources is likely to be chosen to undergo evolution. We also take account of how to balance the resources invested in the inter-domain and inner-domain search. So, an adaptive method is proposed to control the amount of resources allocated to the interdomain and inter-domain search.

The rest of this paper is organized as follows. Section II describes related background, including the mathematical definition of MTO, the detailed description of MFEA, and terms in the resource allocation in EAs. The proposed MTO-DRA is described in Section III. In Section IV, we first compare the the proposed resource allocation strategy based on the index of improvement (IoI) with the previous work to show our work's superiority. Then the performance of the proposed algorithm is also evaluated, compared with other two MFO algorithms and single objective differential EAs (DEs) on two-task benchmark problems. Finally, we show a case of preliminary application in Section V. In Section VI, the work in this paper is concluded and the limitation of the proposed IoI measure and the future works are discussed. The supplementary materials include the experiments on the sensitivity analysis of additional parameters, the experiments on three-task benchmark problems and the analysis of the learning parameter adaption strategy.

II. Background

A. Evolutionary Multitasking

Evolutionary multitasking [9] aims at sharing problem-solving experience of different tasks to solve multiple tasks simultaneously. In a multitask setting with K tasks, the kth task, denoted by T_k, has a search space X_k with an objective function f_k. Such a multitasking instance can be formulated as

$$\begin{aligned}&\{\mathbf{x}_1^*, \mathbf{x}_2^*, \ldots, \mathbf{x}_K^*\}\\ &\quad = \{\mathrm{argmin} f_1(\mathbf{x}_1), \mathrm{argmin} f_2(\mathbf{x}_2), \ldots, \mathrm{argmin} f_K(\mathbf{x}_K)\}\end{aligned} \quad (1)$$

Algorithm 1 Basic Structure of MFEA

```
1:  Generate an initial population of particle $P$.
2:  for each $p_i$ in $P$ do
3:      Assign skill factor $\tau_i = mod(i, K) + 1$, for the case of tasks.
4:      Evaluate $p_i$ for task $\tau_i$ only.
5:  end for
6:  Calculate scalar fitness $\theta_i$ of $p_i$.
7:  repeat
8:      Apply genetic operators to $P$ to generate an offspring-pop $C$.
9:      for each $c_i$ in $C$ do
10:         Determine skill factor $\tau_i$.
11:         Evaluate the individual $c_i$ for $\tau_i$ only.
12:     end for
13:     Concatenate offspring-pop $C$ and current-pop $P$ to form $R$.
14:     Select the fittest individuals from $R$ to the next population $P$.
15: until Stopping condition is satisfied
```

where $\mathbf{x}_k^*$ is the optimal solution of T_k. Evolutionary multitasking is to search multiple decision spaces of problems via implicit parallel of population-based optimization algorithms to efficiently solve the multiple problems.

MFO is a novel evolutionary multitasking paradigm where the multiple tasks are performed simultaneously. It is worthwhile to note that each task f_i in the component problem can be viewed as a factor influencing the evolution of the individuals in the K-factorial environment [8]. In MFO, all individuals are encoded into a unified representation space. A single population of individuals navigate the unified representation space with a unified performance criterion, seeking for optimums of multiple tasks. Several definitions associated with the individuals are shown as follows.

Definition 1: The factorial cost of individual p_i on task T_j is the objective value f_j of potential solution p_i noted as ψ_j^i.

Definition 2: The factorial rank of p_i on T_j is the rank index of p_i in the sorted objective value list given by r_j^i.

Definition 3: The skill factor is defined by the index of the task which an individual is assigned to. Skill factor of p_i is given by $\tau_i = \mathrm{argmin}_{j\in\{1,2,\ldots,K\}} r_j^i$.

Definition 4: The scalar fitness of p_i is the inverse of r_{ϕ_i} given by $\theta_i = 1/\min_{j\in\{1,\ldots,K\}} r_j^i$.

Herein, the skill factor is regarded as the cultural trait which can be inherited from the parents in MFO. We consider the scalar fitness as the unified performance criterion.

B. Multifactorial Evolutionary Algorithms

MFEA is a popular implementation of MFO which incorporates GAs into the MFO paradigm [8]. The basic structure of MFEA is shown in Algorithm 1. The feasible solutions of specific tasks are encoded into a unified representation space first by the uniform random key scheme [14], to efficiently tackle the cross-domain information. The scalar fitness is used to determine the individual's performance in MFEA.

MFEA features assortation mating and selective imitation to transfer the information among different tasks. Specifically, assortation mating (step 8 of Algorithm 1) is employed to generate the offspring. It guarantees that the individuals associated with the same task have high probability to mate and generate the offspring, while the individuals associated with different tasks mate at a smaller probability. This procedure is controlled by an additional parameter, denoted by rmp. Selective imitation simulates the feature that the phenotype of offspring can inherit the phenotype of parents. It helps MFEA assign the skill factor of offspring. In the selective imitation strategy, an individual is evaluated on only one task, which can reduce the computational costs.

Currently, some theoretical works and real-world applications have been investigated since the primary framework of MFO was proposed in [8]. Many works have focused on improving the efficiency of MFEA by adapting the parts of the original framework. Bali *et al.* [15] designed a linear domain adaptation and obtained a high order representative space to alleviate the negative knowledge transfer between uncorrelated tasks. Wen and Ting [16] analyzed the behavior of cross-domain crossover and proposed a novel strategy to balance the interdomain and inner-domain crossover. Feng *et al.* [7] proposed an evolutionary memetic computing paradigm which is capable of learning and leveraging the interdomain knowledge of two kinds of assignment problems. Besides, there are some works devoting to adopting the MFO paradigm in other EAs. Gupta *et al.* [9] adapted MFEA in the field of MOO by employing the nondomination rank as the performance criteria. Feng *et al.* [17] incorporated the swarm intelligence method, like particle swarm optimization and DEs, into MTO. Wen and Ting [18] incorporated the MFEA paradigm into genetic programming, leading to the MFGP algorithm for learning the ensemble of classifiers.

C. Resource Allocation in Evolutionary Algorithms

Usually, some problems involving more than one subproblems may face the problem of the computational resource allocation, like MOO and cooperative co-evolution optimization. In the previous works, the resource allocation strategies are mainly designed for MOO [19]–[21] and large-scale cooperative co-evolutionary optimization [22]–[26], where a problem is decomposed into many subproblems. Above mentioned methods could be classified into two kinds. Zhou and Zhang [21] concluded two frameworks of resource allocation strategies, offline and online strategies.

1) *Offline Resource Allocation (OFRA):* Allocate resources according to a hardness measure of tasks calculated in advance.
2) *Online Resource Allocation (ONRA):* Dynamically allocate resources according to the feedback of the performances online.

The resource allocation strategies for cooperative coevolution and MOO, proposed in current years, can be clustered into these two categories. In OFRA, proposed for MOEA/D in [21], a single-objective evolutionary solver first runs on each task. The average number of function evaluations, which the algorithm needs to reach a given accuracy, can indicate how many resources a task should be invoked. ONRA is another kind of strategies which is often used in the design of resource allocation strategies. In MOO, Zhang and Li [19] proposed a dynamic resource allocation strategy according to the values of utility functions which are updated online. Zhou and Zhang [21] proposed a generalized resource allocation strategy MOEA/D-GRA with ONRA by using a probability of improvement vector. In the field of cooperative coevolution, the resource allocation method in [27] allocates the resources as per the contribution of subpopulation to the overall task. Yang *et al.* [26] improved the dynamically resource allocation based on the contribution by accumulating the improvement of a subproblem.

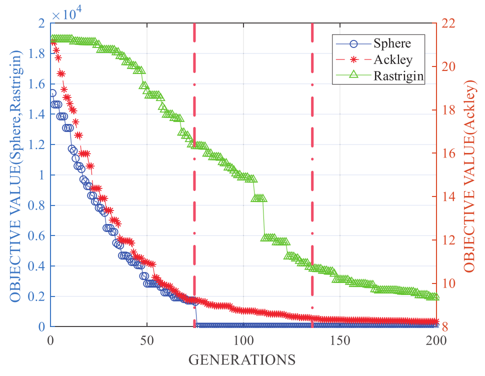


Fig. 1. Example of MFEA solving three tasks. The convergence curves of a three-task problems (Sphere, Ackley's, and Rastrigin's functions) are shown.

D. Motivation

When tackling with MTO problems, tasks can have different computational complexities for an evolutionary multitasking algorithm to obtain an acceptable optimum. So, investing computational resources according to the requirement of each task is a promising way to improve the efficiency of the evolutionary multitasking. Although, an existing work in [16] aimed at handling the resource allocation in the inter-task and inner-task crossover, the tasks are still viewed equally and the same amount of resources are allocated to each task. Thus, designing a resource allocation strategy according to the requirements of tasks is still needed to improve the efficiency of evolutionary multitasking algorithms.

We discuss how the computational resource requirements of tasks are generally parted as the search going on. A population-based optimization algorithm usually considers maintaining the balance of exploration and exploitation during searching. The individuals in exploration behavior trend to randomly search the decision space. In another hand, individuals in exploitation behavior attempt to utilize the elite individuals to guide the search. We would use these two concepts to analyze the long-term process of a population-based optimization algorithm for evolutionary multitasking. As depicted in Fig. 1, the process of an evolutionary multitasking algorithm consists of three stages. In the first stage, the algorithm mainly explores the searching space and benefits from inter-task information transfer [16]. The individuals converge to the promising region near the optimum fast. In the minimization optimization case, all corresponding objective function values decrease significantly in this period. After the first stage, the performances of the algorithm to tackle different types of objective functions become parted in the second stage. The easy task, like sphere function, achieves less improvement with the individuals getting close to the optimums. In contrast, relatively difficult tasks, like Rastrigin's function and Ackley's function, progresses significantly, because the population of individuals are still spread through the region far away from the global optimum. In the last stage, the easy task has found the optimum, thus stagnated. However, the relatively difficult tasks, like Rastrigin's function and Ackley's function, still have a larger improvement compared with the easy task. It is reasonable to allocate relative large number of resources to difficult tasks for a better overall performance.

Due to the various degrees of task hardness and the distinction of the optimization algorithms' searching capacities, the tasks have different stagnation timings. The stagnation phenomenon usually happens in the second and third stages. Actually, the aim of the resource allocation strategy is to reduce the resources invested into the stagnated tasks and invest more resources into the other tasks. Sometimes, the population may stagnate and trap in the local optimums temporarily. In order to help the individuals escape from the local optimums, the resource allocation strategy should give these tasks a chance to undergo evolution. In the previous works, tasks are treated equally in the previous evolutionary multitasking paradigms and allocated the same amount of computational efforts. There is no attempt in the literatures to address this issue in the field of evolutionary multitasking. In this paper, we try to design a proper and usable dynamic resource allocation strategy for MTO by adopting the ONRA method.

III. Evolutionary Multitasking With Dynamic Resource Allocation

The resource allocation is an important issue in the self-adaptive topic of EAs [28] and a promising venue to improve the efficiency of MTO for tackling the tasks with various computational resource requirements. However, a proper resource allocation strategy still needs further research in the field of MTO. Since the population is to solve multiple tasks simultaneously, computational resources are rather limited in an evolutionary multitasking algorithm. In the previous methods [8], [9], each task is viewed equally. All tasks are allocated the same amount of resources. However, this is not an efficient way to make full use of the limited computational resources. An efficient resource allocation strategy based on the task's hardness is needed for an efficient MTO algorithm. Thus, a novel multipopulation framework for evolutionary multitasking with dynamic resource allocation, denoted by MTO-DRA, is presented in this section. It is capable of allocating the computational resources dynamically as per the computational difficulties of tasks. Besides, an adaptive method to balance interdomain and inner-domain information transfers is also discussed herein.

A. MTO-DRA Evolutionary Process

In order to implement the efficient resource allocation strategy, we employ a multipopulation framework in the proposed algorithm. At the beginning of each iteration, subpopulations are constructed from the overall population by the subpopulation construction process. Each subpopulation is associated with one task. Then, an evolutionary multitasking optimizer is run on each subpopulation. During the evolution procedure, to measure the computational resource requirement of each task, the IoI is calculated online according to the runtime performance feedback of previous generation of each subpopulation. Then, in the resource allocation cycle, the resources are allocated to subpopulations based on IoIs of tasks. The task which gets the resources can undergo evolution. When evolutionary optimizer runs for several generations, the elite individuals replace the aged individuals of the corresponding task in the overall population. In addition, to make the cross-domain transfer adaptive, the learning parameter is altered by using a self-adaptive technique. If the terminal condition is not hold, the above process continues. The overall workflow of the proposed MTO-DRA is shown in Fig. 2. It is worthwhile to note that the mixed subpopulation consisting of the individuals from differential tasks collects the cross-domain knowledge. The proposed method combines the building blocks from the mixed subpopulation through the differential mutation and crossover to transfer the cross-domain knowledges.

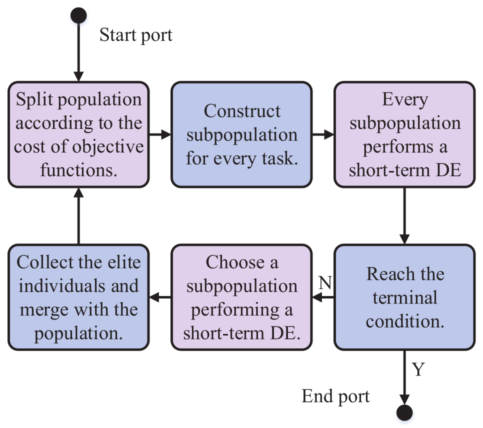

Start port
Split population according to the cost of objective functions.
Construct subpopulation for every task.
Every subpopulation performs a short-term DE
Reach the terminal condition.
N
Choose a subpopulation performing a short-term DE.
Collect the elite individuals and merge with the population.
Y
End port

Fig. 2. Overview of the workflow of MTO-DRA.

The proposed MTO-DRA algorithm actually takes a two-step approach. In the first step, an evolutionary multitasking optimizer (a DE optimizer incorporated into the proposed multipopulation framework for MTO) within a short period is executed on all tasks. Then IoIs are calculated via the function values of the best solutions. And the second step has a polling cycle, which actually is the resource allocation cycle that determines which task can undergo evolution according to the index of improvement. If a task is easy to solve, the population converges fast and could stagnated in the middle period leading to a small IoI. So, it should be assigned with less computational efforts for the sake of reducing waste of computational resources and improving overall performance of the algorithm. In contrast, more resources are allocated to tasks in progress, which have large IoIs. In the experiments, we will analyze the two-step approach by empirical experiments. Moreover, the allocation strategy proposed in this paper can be very convenient to be converted to a parallel variant. The evolutionary operations in each resource allocation cycle are performed on an isolated subpopulation constructed from the overall population in advance. Each evolutionary optimizer can run independently on each subpopulation of each task, rather

Algorithm 2 Construction of Subpopulation P_k of T_k

```
1: Compute the size of external task group $N_{ext}$ by equation (2).
2: $N'_k \leftarrow N_k + N_{ext}$.
3: Select $N'_k$ individuals from $P$ forming $P_k$.
4: Select $N_{ext}$ individuals from $P \cap \complement_P P_k$ forming $P_{ext}$.
5: $P_k \leftarrow P_k \cup P_{ext}$.
```

than relying on the overall population. Therefore, each task can be performed in parallel to dramatically shorten overall wall-clock time. Only the construction of subpopulations and the updating of the overall population need to be synchronized.

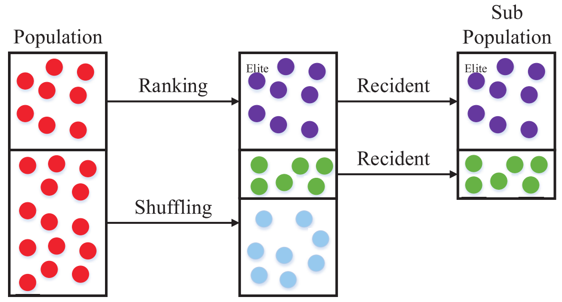


Fig. 3. Construction of subpopulation for tasks. The cycles represent the individuals.

B. Construction of Subpopulations

In the original MFO framework, the skill factor is separated the population into subgroups. To prevent the excessive diversity in the multitasking environment, MFO employs an elaborated strategy, called assortative mating. The proposed multipopulation framework is a straightforward way to implement the isolation of the genetic materials. We adopt the multipopulation framework herein to isolate the domain knowledge and enable the cross-domain transfer by migrating high-quality individuals. The efficacy of the proposed evolutionary multitasking algorithm is affected by how the subpopulation of each task is constructed. In the proposed approach, the population is split into K subpopulations for K tasks, respectively. Each subpopulation includes $N_k + N_{\text{ext}}$ individuals. Specifically, N_k individuals are selected from the original population by ranking them in the ascending order of the objective values. Herein $N_k = \lfloor N/K \rfloor$, where N is the population size and K is the number of tasks. The extra N_{ext} individuals are randomly selected from other tasks and appended to the task-related subpopulation. The N_{ext} is determined by the cross-domain learning parameter LP calculated by (2). Herein, LP plays the same role as the random mating parameter *rmp* in MFEA. They are both used to control transfer of intertask problem-solving knowledge. However, these two parameters are slightly different. LP is a deterministic parameter which controls the number of transferred individuals and rmp is the probability of cross-domain crossover. The number of extra individuals, denoted by N_{ext}, is the product of learning parameter LP and the number of individuals in the subpopulation N_k as

$$N_{\text{ext}} = \max\left\{\lfloor \text{LP} \times N_k \rfloor, N_{\text{ext,min}}\right\} \tag{2}$$

where N_{ext} is the number of the extra individuals, and N_k is the number of individuals assigned to T_k. LP is the learning parameter which controls the agree of the cross-domain implicit transfer. The $N_{\text{ext,min}}$ is the minimal number of injected individuals which guarantees the diversity of subpopulation. The process of constructing a subpopulation for T_k is listed in Algorithm 2. Then an evolutionary optimizer runs on the subpopulation. When the evolutionary optimizer runs for ΔG generations, the individuals in the subpopulation will update the aged individuals of the corresponding task in the overall population.

For clarifying the subpopulation construction process, we give an overview in Fig. 3. As shown in Fig. 3, assume that an overall population contains all individuals each of which is assigned to one task. We consider the construction of subpopulation for task T_k, for example. The proposed method first collects the inter-task information through constructing a mixed subpopulation for each task. The mixed subpopulation mainly consists of the individuals belonging to T_k and a number of extra individuals randomly selected from other tasks. The transferred individuals can be viewed as the carries of problem-solving knowledge from other tasks. Once the mixed subpopulation is constructed, we can transfer the inter-task information through applying the evolutionary operations to individuals. The building blocks are combined from different tasks to generate offspring. The proposed method has two advantages. First, the proposed method can keep the cross-domain information in subpopulation for ΔG generations to make full use of the building blocks in the transferred individuals. Second, it can be straightly adopted in the cases of multiple tasks, i.e., more than two tasks, without modifying any part of the method.

C. Proposed Dynamic Resource Allocation Strategy

We implement the proposed resource allocation strategy for MTO in two aspects: a) allocating the resources as per the hardness of task and b) controlling the computational efforts invested into inter-task search and inner-task search. Specifically, in order to implement the dynamic resource allocation as per the computational complexities of tasks, a method must address the following two issues: 1) how to determine the computational requirements of tasks among the MTO problems and 2) which task has access to the computational resources to undergo evolution in each resource allocation cycle. In the following part, we will explain how the proposed MTO-DRA deals with these two important issues 1) and 2) by implementing the a) and b) aspects.

In this section, we design a vector of index of improvement, denoted by IoI, which is calculated online, to invest the computational resources as per. If a task is easy, it needs relatively fewer generations to obtain the acceptable optimal solution. In the case of a difficult task, more iterations are required for an algorithm to obtain a good solution. The needed computational efforts for a task could be determined by running an EA on it ahead or calculated by an elaborated difficulty measure. However, this is an inefficient way to be applied to MTO in practice. First, this method, which needs to run a full

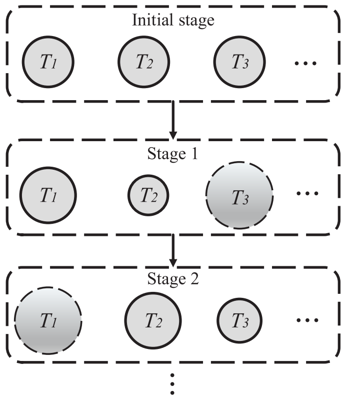


Fig. 4. Process of dynamic resource allocation. The size of each circle represents the requirement of each task which is computed by using the index of improvement. The dashed cycles are the running tasks. In each iteration, the task with high index of improvement has a greater chance to undergo evolution.

evolutionary searching process, is less computationally efficient. Second, the information in calculating measure vector of IoI could not be reused to facilitate resolving the tasks. Last not least, a proper measure is hard to establish when one is to optimize a black-box optimization problem and takes in account of the complementarity of tasks.

In this paper, an improved ONRA is proposed, where the requirement of a task is estimated from the performance feedback of the population. The main motivation of our proposed method is to detect the stagnated tasks and save the resources invested in these tasks in the middle and late stages. The algorithm tries to trace the best solutions that the tasks have found to detect if the subpopulation is stagnated. The process is shown in Fig. 4. Given a population of individuals, the improvement for task is defined as the reduction of the best objective values in minimization optimization (or the increment in maximization optimization) within fixed period ΔT. The improvement of population ω_k with respect to task k could be formulated as

$$\omega_k = \frac{|f_j(x^*(t)) - f_k(x^*(t-\Delta T))|}{\max\{f_k(x^*(t)), f_k(x^*(t-\Delta T))\} + \mu} \tag{3}$$

where $f(x^*(t))$ is the best objective value obtained in the current generation, $f(x^*(t-\Delta T))$ is the best objective value ΔT generations ago. And μ is a small positive number which avoids divide by zero issue. When the specific improvement of each subpopulation is calculated, the negative exponent of the improvement is divided by the sum of those upon all subpopulation. The formula of the index of improvement is shown as follows:

$$\text{IoI}_k = \frac{\exp(S\omega_k)}{\sum_{j\in T}\exp(S\omega_j)} \tag{4}$$

where the ω_k is the improvement of subpopulation assigned with T_k and $S = 1$. IoI is a positive number within the range [0, 1]. It could be viewed as a probability of a task being allocated computational resources. A small IoI indicates that the task is an easy one, and fewer resources would be allocated to the task. Otherwise, a large IoI suggests a relative difficult task which should be given more computational resources.

Algorithm 3 Self-Adaptive Transfer Parameter LP

```
1: if any gbest in the external repository is updated then
2:     Gather LP into the candidate learning parameter list.
3:     if Candidate learning parameter list is full then
4:         Remove the oldest learning parameter value in the list.
5:     end if
6: else
7:     Choose LP from a candidate learning parameter list randomly.
8:     Set $LP = randnorm[\overline{LP}, 0.001]$.
9: end if
```

The tasks with larger improvement have a higher probability to be chosen in the resource allocation cycle. Herein, the improvements of tasks are mapped into Boltzmann's acceptance probabilities by a *softmax* function. So, the tasks with less improvement still have a chance to be chosen.

There exists another issue in the evolutionary multitasking needed to consider if we take further account of the complementarity of tasks. During the reproduction of individuals, some offspring are generated by recombining two parents belonging to the same task. And some are generated via the recombination of those belonging to different tasks. These two offspring generation processes alternate with the search continuing. In the previous methods, the learning parameter is a very critical parameter which controls the amount of individuals investing into the cross-domain exploration. In other words, the learning parameter LP is used to maintain the balance between interdomain search and inner-domain search. Since a larger LP encourages transfer of cross-domain knowledge, the information propagates fast among the population. However, in most of cases, the excessive exchanging between the individuals belonging to different tasks could cause the phenomenon that individuals need extra generations to converge even in an easy task. Therefore, the setting of LP should be considered thoughtfully according to the cooperative relationship of tasks. In practice, the relationship of tasks in MTO is not *a priori*. Despite this, it is in the case that a large LP is suitable for the composite problems which have small divergence on optimums and search spaces. A relatively small LP is suitable for the problems having distant optimums. In the previous MFO algorithms, the learning parameter is determined by using a trail-and-error style method. However, this method must repeat this parameter setting process for newly incoming problems, which takes many efforts to achieve a good performance in practice.

We propose a self-adaptive learning parameter strategy to invest individuals in either inter-domain or inner-domain search. In other words, the proportion of cross-domain transferred individuals in the subpopulation, based on the successful parameter history list. At the beginning of the optimization process, a list of candidate learning parameters are generated by sampling the uniform random distribution in the range [0, 1]. In the beginning of the evolutionary process, choose a specific value from the candidate list. When the current parameter does not lead the population to a better solution, the parameter tuning procedure starts. The parameter tuning procedure randomly chooses an element from a candidate list as $\overline{\text{LP}}$. Then the parameter is sampled from a normal

Algorithm 4 Basic Structure of MTO-DRA

```
 1: Initialize a population of individuals in $P$.
 2: Evaluate the objective values corresponding each individual of $P$.
 3: Compute the skill factor $\tau_i$ of each individual $p_i$.
 4: Select a $LP$ from a list of $LP$ candidates.
 5: repeat
 6:     for each $T_k$ in $\{T_1, T_2, \ldots, T_K\}$ do
 7:         Construct the subpopulation from $T_k$.
            ▷ Refer to Algorithm 2.
 8:         $\{P_k, G\} \leftarrow$ optimize$[P_k, T_k, G, \Delta G, LP]$
 9:         $\{P_k\} \leftarrow$ select$[P_k, N_k]$.
10:         Replace the subpopulation of $T_k$ with $P_k$ in $P$.
11:     end for
12:     Compute the index of improvement $IoIs$ of each $P_t$.
13:     $g \leftarrow 1$.
14:     {Polling cycle.}
15:     for each $T_k$ in $\{T_1, T_2, \ldots, T_K\}$ do
16:         if $rand() < IoI_k$ then
17:             Construct the subpopulation from $P_k$ for $T_k$.
                ▷ Refer to Algorithm 2.
18:             $\{P_k, G\} \leftarrow$ optimize$[P_k, T_k, G, \Delta G, LP]$
19:             $\{P_k\} \leftarrow$ select$[P_k, N_k]$.
20:             Replace the subpopulation of $T_k$ with $P_k$ in $P$.
21:         end if
22:     end for
23:     Adapt $LP$ via Algorithm 3.
24:     $g \leftarrow g + 1$
25: until Before reaching the maximum number of evaluation calls
```

distribution with a mean $\overline{\text{LP}}$ and a standard deviation 0.001. Otherwise, when a better solution is found, the current parameter value is appended at the end of the candidate list. If the candidate list is full, the oldest one in it should be removed. Refer to the content in Section SIV of supplementary material for details on the setting of variance of the distribution when sampling LP.

D. Algorithm Framework

The main structure of the proposed dynamic resource allocation approach is summarized in Algorithm 4. In order to calculate the IoIs of all tasks, MTO-DRA needs some extra computational efforts in steps 6–11. The population of individuals is explicitly grouped into K subpopulations for K tasks in step 7. The problem-solving knowledge is transferred through the individuals migrating from other tasks. Each subpopulation embraces two kinds of individuals, the corresponding N_k individuals assigned to task k and the N_{ext} cross-domain individuals. The learning parameter LP is used to adaptively control the proportion of these two kinds of individuals. The construction of subpopulation is described in Algorithm 2. The optimizer is a canonical evolutionary algorithm in step 8, which only runs for ΔG generations, $\Delta G << \text{MAX}_{\text{gen}}$. Herein, we use an optimizer of differential evolutionary algorithm. In the proposed method, each task could transfer the searching experience through the construction of mixed subpopulations described in Algorithm 2, which is executed in step 7. A canonical EA solver is responsible for combining the genetic materials in subpopulation for individuals. Above two approaches, migration of individuals and recombination of cross-domain knowledge, are responsible for transferring information in two granularities, respectively, i.e., individuals and building blocks. Then, the overall population is updated by the elite solutions selected from subpopulation in steps 9 and 10. IoIs of tasks are calculated online according to the feedback of performance via (4) in step 12. The algorithm enters the resource allocation cycle after the IoIs of tasks have updated. In the resource allocation cycle, IoI vector is used to determine which task is to undergo evolution in steps 15–22. If a uniformly distributed random number r_1 is less than IoI_k, the optimizer will be performed on T_k. The multipopulation evolutionary multitasking method in steps 17–20 is the same process as that in steps 7–10. The final step of each iteration is to self-adaptively choose the learning parameter LP via Algorithm 3 in step 23, which controls the transfer of cross-domain knowledge.

IV. Experiments

To assess the performance of the proposed MTO-DRA, it is compared with other two MFO algorithms and one single objective differential evolutionary algorithm with the same mutation operation. First, the proposed method's performance in the limited computational resources is investigated. Then, the proposed MTO-DRA method is compared with two MFO methods, MFEA and MFDE on comprehensive test problems in the two-task setting. Finally, an application of the proposed method on capacitated vehicle routing problems [29]. Due to the limitation of space, some experimental results are shown in the supplementary material. The additional experimental results in the supplementary material includes the comparison of MTO-DRA and the compared MFO methods on three-task benchmark problems in Section SI in the supplementary material, and the efficiency of the proposed dynamic resource allocation strategy is evaluated in Section SII in the supplementary material. In Section SII, in the supplementary material, we will analyze the behavior of the proposed DRA compared with the ONRA. In addition, the sensitivity analysis of additional parameters in Section SIII in the supplementary material. Some tables on the detailed experimental data referred above are also shown in the supplementary material. Table SIV in the supplementary material displays the detailed descriptions on the two-task benchmark problems. Table SX in the supplementary material reports the complete results about the comparison of the compared algorithms on two-task benchmark problems. Table SV in the supplementary material contains the details of the two-task combinational problems. Table SVI in the supplementary material contains the details of the three-task combinational problems. Table SVII in the supplementary material shows the detailed reported data of the comparison of MTO-DRA and MTO-ONRA. Tables SVIII and SIX in the supplementary material exhibit the reported data under the situations of the limited resources.

In order to compare the performances of different algorithms, we run the experiments 30 times and terminate the algorithms at the maximum number of evaluation calls. The mean and standard deviation of best objective function values are employed to measure an algorithm's global searching ability and stability. The best objective function values are also recorded to compare the best performances of algorithms.

A. Test Problems

We choose a set of functions including eight continuous objective functions. These objective functions are classified into two categories [24]: 1) unimodal and simple multimodal functions and 2) multimodal functions. Unimodal and simple multimodal problems, like Sphere modal and Rosenbrock's valley, and multimodal problems, like Ackley's path function, Griewank's function, Rastrigin's function, Schwefel's function 1.2, Schwefel's function 3.4, and Weierstrass function.

In the experiments, we design the benchmark problems based on the computational complexity of tasks. The detailed description is in Table SIV(a) in the supplementary material. Sets 1–3 have the same dimension and optimum. Sets 4–6 have different dimensions. Sets 7–9 have different optimums. These problems are also the combinations of functions with different complexity degrees. Sets 1, 3, and 7 are the combinations of unimodal functions. Sets 2, 4, and 6 are the mixed problems of unimodal and multimodal functions. Sets 3, 5, and 9 consist of multimodal functions. In [30], the CEC2017 benchmark problems are combined based on the Spearman's rank correlation and the degree of intersection between the search spaces. Totally nine sets are proposed with different degrees of intertask synergy and degrees of intersection between the unified search spaces. The details of the above problems are shown in Table SIV(b) in the supplementary material.

B. Parameter Settings

Although the proposed resource allocation strategy is automatic, this process would not work without any cost. The proposed method has two additional parameters, ΔT and ΔG. In this paper, we set $\Delta T = 20$ and $\Delta G = 20$. The investigation of the influences of ΔT and ΔG is given in Section VI of the supplementary material. Parameter settings for the proposed method and other algorithms are listed as follows.

1) *MTO-DRA:* The parameter settings of MTO-DRA are shown as follows.
   a) Population size, $N_p = 200$.
   b) Maximum NFEs, $\text{MAX}_{\text{NFE}} = 5 \times 10^5$.
   c) Crossover probability, $C_r = 0.9$.
   d) The scale factor, $F = 0.5$.
   e) The improvement detecting period, $\Delta T = 20$.
   f) The improvement detecting period, $\Delta G = 20$.
   g) The initial history list of LP as $[0.2, 0.4, 0.6, 0.8]$.
   h) DE mutation strategy, DE/rand/1 [31].
2) *MFEA:* The parameter settings of MFEA are shown as follows.
   a) Population size, $N_p = 200$.
   b) Maximum NFEs, $\text{MAX}_{\text{NFE}} = 5 \times 10^5$.
   c) Crossover probability, $C_r = 0.9$ [8].
   d) Mutation probability, $M_r = 0.1$ [8].
   e) Random mating probability, $\text{rmp} = 0.3$ [8].
3) *MFDE:* The parameter settings of MFDE are shown as follows.
   a) Population size, $N_p = 200$.
   b) Maximum NFEs, $\text{MAX}_{\text{NFE}} = 5 \times 10^5$.
   c) Crossover probability, $C_r = 0.9$ [17].
   d) The scale factor, $F = 0.5$ [17].
   e) Random mating probability, $\text{rmp} = 0.3$ [17].
   f) DE mutation strategy, DE/rand/1 [31].
4) *DE:* The parameter settings of single objective DE are shown as follows.
   a) Population size, $N_p = 200$.
   b) Maximum NFEs, $\text{MAX}_{\text{NFE}} = 2.5 \times 10^5$ for each task.
   c) Crossover probability, $C_r = 0.9$ [32], [33].
   d) The scale factor, $F = 0.5$ [32], [33].
   e) DE mutation strategy, DE/rand/1 [31].

It is noted that all algorithms perform without local search for investigating the computational resource usage, i.e., the number of objective function evaluations. In the following experiments, the above parameters keep the same for all experiments unless the changes are emphasized for some experiments. For all conducted experiments, reported values are averaged over 30 independent runs.

C. Performance With Limited Resources

In order to investigate the efficiency of resource allocation, this experiment tests MFEA and MTO-DRA with limited computational resources. It is assumed that the full computational budget is FE_{max} function evaluations. The algorithms are given 10%, 30%, 50%, and 70% of FE_{max} function evaluations. In each case, the best function values are recorded. The averaged function values and standard deviations are reported in Tables SVIII and SIX in the supplementary material. The best and similar results are marked out in the tables.

As shown in Table SVIII in the supplementary material, the proposed method is superior in term of the accuracy. For the first tasks of Sets 1, 6, and 7, the obtained solutions are close to the optimal solutions at the error less than 10^{-6}. By comparing the best results found at the different situations, we can see that the proposed algorithm can find a good solution of Sets 5 and 6 efficiently when the algorithm spends about 30% maximum FEs.

In Table SIX in the supplementary material, it is observed that MTO-DRA achieves better solutions on 11 out of the 12 benchmark sets on the second task. We can see that the MTO-DRA algorithm has made great progress in performance compared with the MFEA. In Sets 1–8, the proposed method achieves the best results. In Sets 12 and 18, the second task of each problem is a difficult task because its optimal solution is close to the bounds of the feasible domain and a lot of local optimums are around the global optimum. The dynamic resource allocating strategy can allocate frequently computational resources to these tasks for improving the overall performance. Therefore, the experimental results show the superiority of the proposed dynamic resource allocating strategy, which can make efficient use of the computational resources.

The average normalized objective function errors of two methods are plotted in Fig. 5. We only report some representative results, such as Sets 2, 7, 8, 10, and 12 in these plots. We use the notation S*k*-*t* to represent the *t*-th task of Set *k*, for example, S10-1 represents the first task of Set 10.

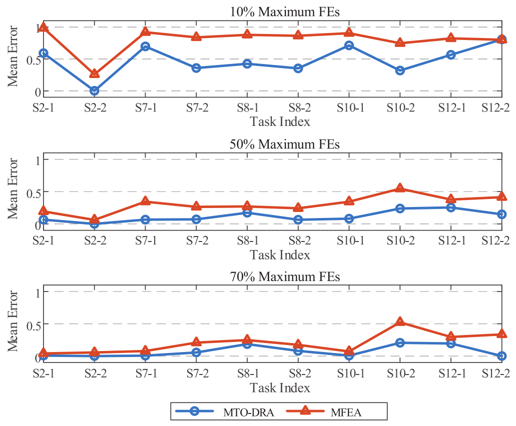


Fig. 5. Comparison results with MFEA with limited resources of 10% maximum FEs; 50% maximum FEs; and 70% maximum FEs.

As shown in Fig. 5, MTO-DRA is able to utilize the computational resources more efficiently than MFEA. In the timing of 10% FEs, the curve of MTO-DRA is under the one of MFEA on almost all points. Especially, MTO-DRA pays attention to the S7-2, S8-1, S8-2, and S10-2. As the search undergoing, the values of mean errors reduce gradually in both two methods MFEA and MTO-DRA. In the case of 50% FEs, MFEA still has not achieved a significant improvement compared with the results of 10% FEs in some tasks like task S2-2, S10-2, and S12-2. It suggests that some resources have been unnecessarily invested into a converged subpopulation because the resources are divided equally among all tasks. This is not an efficient way to allocate too many resources to these tasks. Conversely, MTO-DRA would allocate many resources to difficult tasks, like S2-1, S7-1, and S10-1. It is worth noting that the proposed method achieves significant improvements for selected benchmark problems even in the early period when allowable FEs increases from 10% to 50%. In contrast, MFEA needs more computational costs to obtain the same improvements than MTO-DRA. It achieves the same improvements on S2-1, S7-1, S8-2, and S10-1 after the checkpoint of 50% to 70%. From the above analysis, the efficiency of the MTO-DRA attributes to the capacity of investing more resources in the relative difficult tasks. To meet the all requirements of tasks, MFEA must increase totally resources, since the resources are equally allocated to each task.

D. Comparison on Two-Task Problems

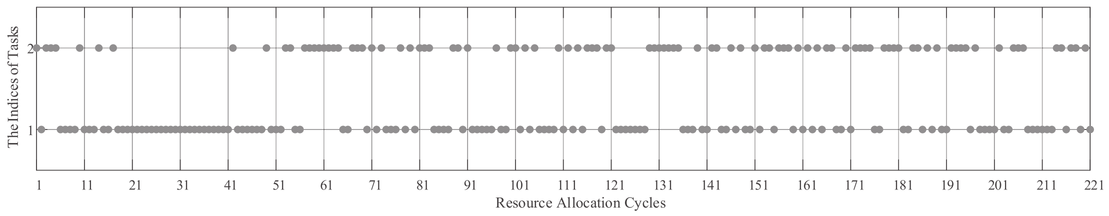
In order to evaluate the performance of MTO-DRA, we compare the proposed method with two MFO algorithms and the single-objective DE. Herein, the algorithms are given enough computational resources (i.e., $\mathrm{FE}_{\max} = 10^4 \times D$ where D is the dimensions of a problem). The parameter settings are the same as what is mentioned above. All these algorithms perform 30 independent runs.

The detailed results of the comparison of MTODRA, MFEA, MFDE, and DE are shown in Table SX in the supplementary material. The experiment results show that the proposed resource allocation strategy makes MTO algorithms efficient. As shown in Table SX in the supplementary material, the proposed algorithm shows its superiority over other algorithms. On Sets 1, 4, and 7, every algorithm obtains a good performance. Especially, MTO-DRA can obtain the optimums stably compared with the other algorithms. It achieves the optimums close to 0 which are the best results among other methods. Sets 2, 5, and 8 consist of tasks with different degrees of hardness. MTO-DRA has a superior performance on these tasks compared with other MFO. Dynamic resource allocating strategy can invest enough computational resources into hardness tasks, which can improve the accuracy of hardness task. As shown in the results, on the easy tasks of above problems, the performance and stability of MTO-DRA are improved compared with the other MFOs and DE. Multipopulation framework and the adaptive learning parameter help the method enhance the ability of fine search to make the performance stable in easy task. Consider the combination of difficult tasks in Sets 3, 6, and 9. On Sets 3 and 6 consisting of two relevant tasks, MTO-DRA is superior the other algorithms. But when tackling with the irrelevant tasks, like Set 9, where the optimums of two tasks are distinct each other, MTO-DRA is not the best, while DE ranks the top on this problem. This is caused by the unexpected failure of dynamic allocating strategy, which invests most resources into the first task of Set 9. That is the reason why MTO-DRA only obtains the best solution compared with DE on Set 9. On Sets 10–18, MTO-DRA achieves the top results. Though, those problems are with different similarities, their optimums locate in positions not distinct from each other. So, MTO-DRA with the aids of dynamic resource allocating strategy and adaptive learning parameter can easily handle with these problems.

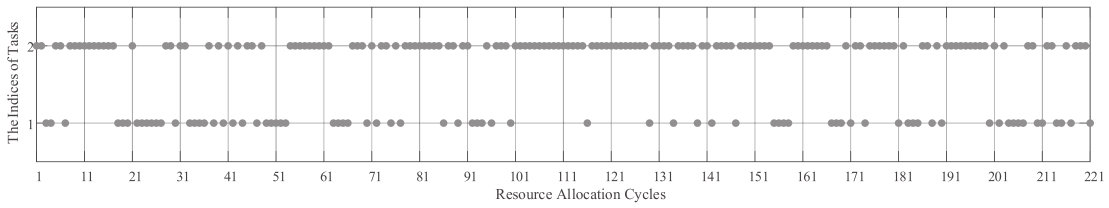
Fig. 6 shows the behavior of the proposed method on two representative two-task problems Sets 2 and 3. Before every resource allocation cycle, the indices of improvement of all tasks are calculated. At the beginning of optimization process, each task has large improvement in every resource allocation cycle. It can be seen that in the first ten cycles, each task has an almost equal chance to undergo search. The improvement of each task becomes parted in the middle period. Fig. 6(a) shows that the first task of Set 2 is easy for the algorithm so that it is allocated less computational resources than the second task. The same behavior is observed in Set 3 as shown in Fig. 6(b). At the later stage, when the global optimum is found or the algorithm is stagnated, each task does not obtain any novel good solutions. So that the rest of computational resources are equally allocated to each task in the last dozen cycles as shown in Fig. 6(a).

V. Preliminary Application

In this section, we investigate the scalability and efficiency of the proposed method on some practical problems. Herein we only prove a preliminary and simply demonstration of utilizing the proposed MTO-DRA to address two kinds of well-known problems. In this experiment, we choose 6 two-task and 2 three-task benchmark problems from two different domains, CVRP [29] and QAP [34], [35].



(a)



(b)

Fig. 6. Activation patterns of tasks for continuous optimization in one run on two benchmark problems. (a) Set 2 and (b) Set 3. The filled cycles represent the tasks undergo evolution in the correspondent resource allocation cycles.

TABLE I
RESULTS OF THE COMPARISON ON TWO-TASK COMBINATIONAL PROBLEMS. THE MAXIMUM, MINIMUM, MEAN, AND STD VALUES OF OBJECTIVE VALUES AFTER MAXIMUM FEs OVER 30 RUNS. THE SUPERIOR PERFORMANCE IS HIGHLIGHTED IN **BOLD**. AND THE SECOND BEST ONE IS TYPED IN *Underline and Italic* TYPE

Problem		MTO-DRA		MFEA		MFDE		DE	
		T_1	T_2	T_1	T_2	T_1	T_2	T_1	T_2
S21	Min	**375**	**1316**	*375*	1819	375	1408	375	*1391*
	Mean	**379.50**	1608.50	414.27	1942.30	*389.73*	*1534.57*	392.27	**1507.83**
	Std	7.39	222.48	23.32	55.03	13.31	66.61	13.03	60.01
S22	Min	**375**	**6924**	384	6998	*375*	*6932*	375	6952
	Mean	**376.50**	**7002.53**	439.57	7110.87	*393.27*	7071.47	398.43	*7062.60*
	Std	3.88	52.20	28.14	73.57	17.37	46.20	15.74	59.11
S23	Min	*1371*	**159590**	1806	167892	**1347**	162214	1379	*161392*
	Mean	1614.03	**163101.01**	1937.00	169081.60	*1517.03*	163591.93	**1495.53**	*163578.27*
	Std	202.56	3812.96	75.55	789.51	57.12	893.33	57.29	856.87
S24	Min	**566**	991	663	1021	604	**970**	*594*	*982*
	Mean	**636.67**	**1024.20**	735.90	1136.13	*661.63*	*1026.20*	664.57	1043.47
	Std	36.68	24.69	29.98	54.06	32.65	31.22	31.59	49.90
S25	Min	**1196**	**797**	1329	1051	1253	*843*	*1215*	852
	Mean	**1264.13**	**910.24**	1416.07	1125.33	1320.77	928.10	*1295.20*	*924.57*
	Std	76.10	109.30	40.74	32.57	41.18	49.57	45.10	37.20
S26	Min	**155134**	**157220**	163152	164252	*158060*	159796	158430	*159302*
	Mean	**159824.20**	**160042.15**	165435.27	167002.73	160051.00	161248.53	*159872.13*	*160813.87*
	Std	3387.58	3697.42	893.28	1058.05	1042.53	801.38	879.63	751.07

When optimizing CVRP, the random keys are decoded in the form of a traveling salesman route which is a permutation of all customer, which then is split into individual vehicle paths by a partition procedure in [36]. In QAP, the permutation of factories is represented by the rank of elements of random keys. We consider solving them simultaneously by utilizing the proposed method. Totally eight benchmark problems, including 6 two-task problems and 2 three-task problems, are presented here, which are summarized in Tables SV and SVI in the supplementary material. Specifically, Sets 21, 24, 25, and 26 are the intradomain test problems. And Sets 22 and 23 are the cross-domain problems. We include some large scale problems herein. Sets 21, 23, and 26 include large scale combination problems with the dimensions up to 100. Sets 21, 24, and 25 have the tasks with different scales for testing the efficiency in utilizing the resources.

As shown in Table I, MTO-DRA achieves the better solutions compared with MFEA and is also stable. It is first observed that in some large scale test problems, such as Sets 21 and 23, DE obtains better solutions compared with other MFO. Specifically, DE finds a better solution of Sets 21 and 23. For example, in Set 23, it finds the best solutions of 1495.53 and 163 578.27. The excessive diverse of cross-domain transfer added to tasks has negative influence on the solution of large-scale problems. In spite of this, the proposed method shows the superiority compared with DE, especially other multifactorial algorithms. In the problems including small scale tasks, such as Sets 21 and 22, the performances of all algorithms are similar, but the proposed method is more stable. In the large-scale problems Sets 24 and 25, the proposed method is even outperformance the DE and other MFO algorithms. Like in Set 24, MTO-DRA achieves best solutions of all algorithms. Compared with MFEA and MFDE, MTO-DRA has superior performance in large-scale problems, because it employs a multipopulation framework to isolate the building blocks and control the transfer of the interdomain knowledge. Meanwhile the dynamic resource allocating strategy helps MTO-DRA efficiently utilize the computational resources to promote the fine search. We plot the box plot of the amount of resources allocated to T1 and T2 of Set 21. As what is seen, observably, MTO-DRA invests more resources into T2 which is hard, and

TABLE II
RESULTS OF THE COMPARISON ON THREE-TASK COMBINATIONAL PROBLEMS. THE MINIMUM, MEAN, AND STD VALUES OF OBJECTIVE VALUES AFTER MAXIMUM FEs OVER 30 RUNS. THE SUPERIOR PERFORMANCE IS HIGHLIGHTED IN **BOLD**. AND THE SECOND BEST ONE IS TYPED IN *Underline and Italic* TYPE

Problem		MTO-DRA			MFEA			MFDE			DE		
		T_1	T_2	T_3	T_1	T_2	T_3	T_1	T_2	T_3	T_1	T_2	T_3
S27	Min	**794**	**160182**	**162268**	1036	167446	*169586*	*871*	163288	175856	888	*162232*	175268
	Mean	**849.12**	**162033.30**	**164150.50**	1111.23	169521.33	*171724.40*	*957.50*	164490.07	177012.53	958.10	*163890.87*	176888.40
	Std	42.13	1640.06	2018.26	37.50	1077.15	840.10	50.69	879.55	531.13	44.14	908.65	523.01
S28	Min	**560**	**778**	*259333224*	653	913	**252098700**	*616*	*779*	513465266	617	864	544391206
	Mean	**607.07**	**816.34**	**273274922.15**	736.03	1014.53	*278110340.80*	684.23	*926.33*	578255109.73	*682.80*	945.50	592602248.93
	Std	23.23	37.67	10352940.72	37.41	53.94	12814405.36	33.52	70.15	30911353.91	26.61	54.30	20385056.12

reduces the resources invested into T1. So, MTO-DRA obtains better solution in T2.

Next, we validate MTO-DRA in three-task problems. Sets 27 and 28 are two large-scale cross-domain problems. As what is seen in Table II, DE outperforms two MFO algorithms in some tasks. The negative transfer can influence the performances of MFO algorithms in large-scale. However, because DE has an ability of rapid convergence compared with GA which helps DE tackle with the excessive diverse caused by cross-domain, MTO-DRA and MFDE obtain better results than MFEA. In addition, MTO-DRA achieves the best results on almost all problems. This is attributed to the multipopulation framework, which can isolate the building blocks and control the cross-domain transfer, and the dynamic resource allocating strategy, which can make full use of the computational resources.

VI. CONCLUSION

In this paper, we present an online dynamic resource allocation strategy for evolutionary multitasking. An IoI using softmax function has been proposed to measure the hardness of a task. Then, a resource allocation strategy based on the IoI measure is incorporated into the multipopulation framework to solve MTO. A detailed analysis of the IoIs has been conducted on a popular benchmark, which demonstrates the superiority of the proposed dynamic resource allocation strategy. In order to take the complementarity of tasks into consideration, we propose an adaptive method to control the computational resources invested in interdomain searching. The proposed method is also tested on a preliminary realistic case of capacitated vehicle routing problems and the three-task benchmark problems. The experimental results suggest the superiority of MTO-DRA over other methods no matter two MFOs and the single-objective DE.

This paper focuses on a proof of preliminary concept research for this strategy. There are many researches needed to undergo. First, the IoI-based measure traces the short-term property of an evolutionary algorithm. So, the IoI-based measure may produce an unstable output when the improvement of a task changes greatly, which causes excessively unbalance resource allocations. Second, the proposed IoI-based measure indicates the hardness of a task by measuring the improvements of tasks. Thus, it may not suitable for the excessive complicated and large-scale problems where it is hard to count the improvement of a task.

In the future, we hope to develop an efficient way to transfer the shared knowledge which can deal with the underlying similarity and distinction, especially in the case of many tasks. Because the proposed measure is calculated based on the feedback of the algorithms within a short period, it actually ignores the potential property of task and the long term information. We hope to design a stable and effective measure to trace the long-term behavior of the algorithms and mining the properties of tasks. It seems that the refinement learning is a promising method to address this issue.

REFERENCES

[1] X. Yao, "Evolutionary computation," in *Evolutionary Optimization*. Boston, MA, USA: Springer, 2002, pp. 27–53.

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

[3] R. Caruana, "Multitask learning," in *Learning to Learn*. Boston, MA, USA: Springer, 1998, pp. 95–133.

[4] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[5] M. Iqbal, W. N. Browne, and M. Zhang, "Reusing building blocks of extracted knowledge to solve complex, large-scale Boolean problems," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 465–480, Aug. 2014.

[6] R. Mills, T. Jansen, and R. A. Watson, "Transforming evolutionary search into higher-level evolutionary search by capturing problem structure," *IEEE Trans. Evol. Comput.*, vol. 18, no. 5, pp. 628–642, Oct. 2014.

[7] L. Feng, Y.-S. Ong, M.-H. Lim, and I. W. Tsang, "Memetic search with interdomain learning: A realization between CVRP and CARP," *IEEE Trans. Evol. Comput.*, vol. 19, no. 5, pp. 644–658, Oct. 2015.

[8] 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.

[9] 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.

[10] 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)*, Singapore, 2016, pp. 3157–3164.

[11] 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.

[12] R. Sagarna and Y.-S. Ong, "Concurrently searching branches in software tests generation through multitask evolution," in *Proc. IEEE Symp. Series Comput. Intell. (SSCI)*, Athens, Greece, 2016, pp. 1–8.

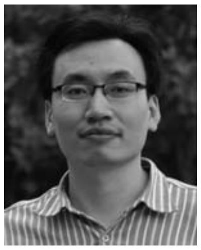
[13] R. Chandra, A. Gupta, Y.-S. Ong, and C.-K. Goh, "Evolutionary multitask learning for modular training of feedforward neural networks," in *Proc. Int. Conf. Neural Inf. Process. (ICONIP)*, Kyoto, Japan, 2016, pp. 37–46.

[14] Y.-S. Ong, "Towards evolutionary multitasking: A new paradigm in evolutionary computation," in *Proc. Comput. Intell. Cyber Security Comput. Models (ICC3)*, Coimbatore, India, 2015, pp. 25–26.

[15] K. K. Bali, A. Gupta, L. Feng, Y. S. Ong, and T. P. Siew, "Linearized domain adaptation in evolutionary multitasking," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, San Sebastián, Spain, 2017, pp. 1295–1302.

[16] Y.-W. Wen and C.-K. Ting, "Parting ways and reallocating resources in evolutionary multitasking," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, San Sebastián, Spain, 2017, pp. 2404–2411.

[17] L. Feng *et al.*, "An empirical study of multifactorial PSO and multifactorial DE," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, San Sebastián, Spain, 2017, pp. 921–928.
[18] Y.-W. Wen and C.-K. Ting, "Learning ensemble of decision trees through multifactorial genetic programming," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Vancouver, BC, Canada, 2016, pp. 5293–5300.
[19] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.
[20] Q. Zhang, W. Liu, and H. Li, "The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Trondheim, Norway, 2009, pp. 203–208.
[21] A. Zhou and Q. Zhang, "Are all the subproblems equally important? Resource allocation in decomposition-based multiobjective evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 20, no. 1, pp. 52–64, Feb. 2016.
[22] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 225–239, Jun. 2004.
[23] Y.-J. Shi, H.-F. Teng, and Z.-Q. Li, "Cooperative co-evolutionary differential evolution for function optimization," in *Advances in Natural Computation*. Heidelberg, Germany: Springer, 2005, pp. 1080–1088.
[24] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. Evol. Comput.*, vol. 10, no. 3, pp. 281–295, Jun. 2006.
[25] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Inf. Sci.*, vol. 178, no. 15, pp. 2985–2999, Aug. 2008.
[26] M. Yang *et al.*, "Efficient resource allocation in cooperative co-evolution for large-scale global optimization," *IEEE Trans. Evol. Comput.*, vol. 21, no. 4, pp. 493–505, Aug. 2017.
[27] M. N. Omidvar, X. Li, and X. Yao, "Smart use of computational resources based on contribution for cooperative co-evolutionary algorithms," in *Proc. ACM Genet. Evol. Comput. Conf. (GECCO)*, 2011, pp. 1115–1122.
[28] R. Hinterding, Z. Michalewicz, and A. E. Eiben, "Adaptation in evolutionary computation: A survey," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 124–141, Apr. 1997.
[29] P. Augerat *et al.*, "Computational results with a branch and cut code for the capacitated vehicle routing problem," Univ. Joseph Fourier, Grenoble, France, Rep. 1 RR949-M, 1995.
[30] B. Da *et al.* (2017). *Evolutionary Multitasking for Single-Objective Continuous Optimization: Benchmark Problems, Performance Metric, and Baseline Results*. [Online]. Available: https://arxiv.org/abs/1706.03470
[31] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, Feb. 2011.
[32] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2005, pp. 1785–1791.
[33] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, Apr. 2009.
[34] J. Skorin-Kapov, "Tabu search applied to the quadratic assignment problem," *ORSA. J. Comput.*, vol. 2, no. 1, pp. 33–45, Feb. 1990.
[35] S. W. Hadley, F. Rendl, and H. Wolkowicz, "A new lower bound via projection for the quadratic assignment problem," *Math. Oper. Res.*, vol. 17, no. 3, pp. 727–739, Aug. 1992.
[36] C. Prins, "A simple and effective evolutionary algorithm for the vehicle routing problem," *Comput. Oper. Res.*, vol. 31, no. 12, pp. 1985–2002, Oct. 2004.



Maoguo Gong (M'07–SM'14) received the B.S. degree in electronic engineering (First Class Hons.) and the Ph.D. degree in electronic science and technology from Xidian University, Xi'an, China, in 2003 and 2009, respectively.

Since 2006, he has been a Teacher with Xidian University. In 2008 and 2010, he was promoted as an Associate Professor and as a Full Professor, respectively, both with exceptive admission. He received the Prestigious National Program for the support of Top-Notch Young Professionals from the Central Organization Department of China, the Excellent Young Scientist Foundation from the National Natural Science Foundation of China, and the New Century Excellent Talent in University from the Ministry of Education of China. His current research interests include computational intelligence with applications to optimization, learning, data mining, and image understanding.

Dr. Gong is an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS.

Zedong Tang received the B.S. degree from Xidian University, Xi'an, China, in 2014, where he is currently pursuing the Ph.D. degree.

His current research interests include computational intelligence and machine learning.

Hao Li received the B.S. degree in electronic engineering and the Ph.D. degree in pattern recognition and intelligent systems from Xidian University, Xi'an, China, in 2013 and 2018, respectively.

He is currently a Lecturer with the School of Electronic Engineering, Xidian University. His current research interests include computational intelligence and machine learning.

Jun Zhang (M'02–SM'08–F'17) received the Ph.D. degree from the City University of Hong Kong, Hong Kong, in 2002.

He is currently a Changjiang Chair Professor with the Key Laboratory of Machine Intelligence and Advanced Computing, Ministry of Education, Guangzhou, China. His current research interests include computational intelligence, cloud computing, data mining, and power electronic circuits. He has published over 200 technical papers in the above areas.

Dr. Zhang was a recipient of the China National Funds for Distinguished Young Scientists from the National Natural Science Foundation of China in 2011, and the First-Grade Award in Natural Science Research from the Ministry of Education, in 2009. He is currently an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTION ON INDUSTRIAL ELECTRONICS, and the IEEE TRANSACTIONS ON CYBERNETICS.