Multitasking Genetic Algorithm (MTGA) for Fuzzy System Optimization

Dongrui Wu, *Senior Member, IEEE*, and Xianfeng Tan, *Student Member, IEEE*

***Abstract*—Multitask learning uses auxiliary data or knowledge from relevant tasks to facilitate the learning in a new task. Multitask optimization applies multitask learning an optimization to study how effectively and efficiently tackle the multiple optimization problems, simultaneously. Evolutionary multitasking, or multifactorial optimization, is an emerging subfield of multitask optimization, which integrates evolutionary computation and multitask learning. This article proposes a novel and easy-to-implement multitasking genetic algorithm (MTGA), which copes well with significantly different optimization tasks by estimating and using the bias among them. Comparative studies with eight state-of-the-art single-task and multitask approaches in the literature on nine benchmarks demonstrated that, on average, the MTGA outperformed all of them and had lower computational cost than six of them. Based on the MTGA, a *simultaneous optimization strategy* for fuzzy system design is also proposed. Experiments on simultaneous optimization of type-1 and interval type-2 fuzzy logic controllers for couple-tank water level control demonstrated that the MTGA can find better fuzzy logic controllers than other approaches.**

***Index Terms*—Evolutionary multitasking, fuzzy logic controller (FLC), genetic algorithm, multifactorial optimization (MFO), multitask learning.**

I. Introduction

MULTITASK learning [1], [2] is a subfield of machine learning, particularly, transfer learning [3]–[6], which uses auxiliary data or knowledge from related/similar tasks to facilitate the learning in a new task. As a result, a learning model for the new task can be built with much less task-specific training data. Or, in other words, with the same amount of task-specific data, a much better model could be trained. In multitask learning, multiple related learning tasks are performed simultaneously using a (partially) shared model representation. As a result, the common information contained in these related tasks can be exploited to improve the learning efficiency and generalization performance of each task-specific model.

Manuscript received May 2, 2019; revised September 9, 2019; accepted January 17, 2020. Date of publication January 23, 2020; date of current version June 3, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 61873321 and in part by the Technology Innovation Project of Hubei Province of China under Grant 2019AEA171. *(Corresponding author: Dongrui Wu.)*

The authors are with the Ministry of Education Key Laboratory of Image Processing and Intelligent Control, School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: drwu@hust.edu.cn; xianfeng_tan@hust.edu.cn).

This article has supplementary downloadable material available at https://ieeexplore.ieee.org, provided by the authors.

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

Digital Object Identifier 10.1109/TFUZZ.2020.2968863

Multitask optimization (MTO) [7]–[10] applies multitask learning an optimization to study how to effectively and efficiently tackle the multiple optimization problems, simultaneously. Evolutionary multitasking [9], or multifactorial optimization (MFO) [8], is an emerging subfield of MTO, which integrates evolutionary computation and multitask learning. It assumes that each constitutive task has some (positive) influence on the evolutionary process of a single population of individuals, and hence, evolving multiple populations from different tasks together simultaneously could be more efficient than evolving each individual task separately. A multifactorial evolutionary algorithm (MFEA) has recently been proposed in [8] and demonstrated promising performance in synthetic and real-world MTO problems. More details about MFEA can be found in the Supplementary Material.

In this article, we consider single-objective multitasking optimization, where every point in the search space maps to a scalar objective value. There have been multiple such approaches in the literature, which mainly focus on the following five aspects.

1) *How to effectively transfer relevant information between tasks:* Yuan *et al.* [11] introduced two new improvements—a new unified representation and a new survivor selection procedure—to the MFEA and demonstrated their effectiveness. Bali *et al.* [12] proposed a linearized domain adaptation strategy to improve the MFEA. It transforms the search space of a simple task to the search space similar to its constitutive complex task. This high-order representative space resembles high correlation with its constitutive task and provides a platform for efficient knowledge transfer via crossover. The proposed linearized domain adaptation MFEA (LDA-MFEA) demonstrated competitive performances against the MFEA. Liew and Ting [13] propose a general framework, the evolution of biocoenosis through symbiosis, for evolutionary algorithms to deal with many-tasking problems, and showed that its performance may be better than the MFEA. Ding *et al.* [14] proposed decision variable translation and shuffling strategies to facilitate knowledge transfer between optimization problems having different locations of the optimums and different numbers of decision variables, and verified their effectiveness in multitasking optimization. Feng *et al.* [15] used autoencoding to explicitly transfer knowledge across tasks in evolutionary multitasking, and demonstrated its performance in both single-objective and multiobjective MTO problems. Hashimoto *et al.* [16] pointed out that the MFEA can be viewed as a special

island model, and proposed a simple implementation of evolutionary multitasking using the standard island model, which achieved promising performance.

2) *How to allocate computing resources to different tasks:* Gong *et al.* [17] proposed an evolutionary multitasking algorithm using a dynamic resource allocation strategy, which can dynamically allocate more resources to the more difficult tasks.
3) *How to dynamically adjust the amount of information passed between the tasks:* Wen and Ting [18] proposed two improvements to the MFEA (parting ways detection and resource reallocation), and showed that they can often result in better solutions, especially when the tasks share low similarity of landscapes.
4) *How to combine with other optimization algorithms:* Cheng *et al.* [19] developed a particle swarm optimization-based coevolutionary multitasking approach for concurrent global optimization, and demonstrated its performance on synthetic functions and in real-world complex engineering design. Chen *et al.* [20] proposed an evolutionary multitasking single-objective optimization approach based on the cooperative coevolutionary memetic algorithm. Local search based on the quasi-Newton approach was used to accelerate its convergence.
5) *How to apply to new applications:* Sagarna and Ong [21] applied multitask evolutionary computation to concurrently searching branches in software test generation. Tang *et al.* [22] used evolutionary multitasking to evolve the modular topologies of extreme learning machine classifiers. Li *et al.* [23] employed evolutionary multitasking to optimize multiple sparse reconstruction tasks simultaneously.

To our knowledge, except [11], all other approaches adopt the *unified representation* used in the MFEA, which transforms the solutions of different tasks into a common *unified search space*, and then, performs information transfer in this space. A careful examination of the principle of unified representation reveals that the location of the optimum in the unified search space is related to the ranges of the variables. That is, if two tasks have different variable ranges, even if their fitness landscapes are identical in the original space, their optima will be different in the unified search space. In the absence of prior knowledge, correctly setting the variable ranges is very challenging, but is also critical to the success of existing multitasking evolutionary algorithms. Furthermore, even if we can correctly specify the variable ranges, the locations of the optima of different tasks in the unified search space may also be different. Hence, it is very important to consider the bias between different tasks.

To mitigate the negative effects of bias, we should estimate and use it during chromosome transfer. Based on this motivation, we propose a novel multitasking genetic algorithm (MTGA), which on average, outperforms eight state-of-the-art approaches on nine multitasking benchmarks [10]. It also demonstrates outstanding performance in simultaneous optimization of type-1 (T1) and interval type-2 (IT2) fuzzy logic controllers (FLCs).

The main contributions of this article are as follows.

1) We propose a novel MTGA approach for MTO, which has superior performance and low computational cost.
2) We propose a novel MTGA-based simultaneous optimization strategy for fuzzy system design, and demonstrate its effectiveness in FLC optimization.

The remainder of the article is organized as follows. Section II proposes the MTGA approach. Section III validates the performance of the MTGA on nine benchmarks. Section IV proposes an MTGA-based *simultaneous optimization strategy* for fuzzy system design, and demonstrates its performance on simultaneous optimization of T1 and IT2 FLCs for coupled-tank water level control. Section V concludes this article.

II. Multitasking Genetic Algorithm

This section introduces our proposed MTGA, whose pseudocode is given in Algorithm 1.

We use the same problem setting as the one in [8]. Without loss of generality, assume that there are M tasks, all of which are minimization problems. The mth task T_m, has an objective function $f_m : \mathbf{X}_m \to \mathbb{R}$ on a search space $\mathbf{X}_m$. Each task may also be constrained by several equalities and/or inequalities that must be satisfied by a feasible solution. The goal of the MTGA is to find

$$\{\mathbf{x}_1^*, \ldots, \mathbf{x}_M^*\} = \left\{ \arg\min_{\mathbf{x}_1} f_1(\mathbf{x}_1), \ldots, \arg\min_{\mathbf{x}_M} f_M(\mathbf{x}_M) \right\} \tag{1}$$

where $\mathbf{x}_m$ is a feasible solution in $\mathbf{X}_m$, $m = 1, \ldots, M$.

For the ease of illustration, we only consider two tasks, i.e., $M = 2$. The extension to more than two tasks is straightforward.

A. Motivation of the Proposed MTGA

The MFEA may offer no advantage over optimizing each single task separately, if in the unified search space, the optimal solutions of different tasks or their fitness landscapes are significantly different. Unfortunately, in practice, often we do not know *a priori* how similar the tasks are, and it is desirable to have a MTO algorithm that can achieve good performance even in the worst case scenario that the tasks are significantly different.

The MTGA is proposed to cope with the abovementioned problem. Particularly, it addresses two important questions: 1) how to estimate the difference between the optimal solutions of the two tasks and 2) how to effectively transfer the fittest chromosomes between the two populations (tasks). Its main idea is to estimate the bias between the two tasks and then, remove it in chromosome transfer, so that the optimal solutions of the two tasks are close to each other. In this way, a promising chromosome from one task can also be transformed into a promising solution for the other task, expediting the convergence.

The MTGA's approach to estimate the bias can be explained using the example in Fig. 1, where the two tasks are one-dimensional (1-D) Ackley and Sphere functions [10]. Ackley maintains a population P_1, and Sphere a population P_2, each of which has ten chromosomes. Clearly, there is a large bias

Algorithm 1: Pseudocode of the proposed MTGA.

Input: Two tasks T_1 and T_2;
N, the population size;
K, the maximum number of generations;
n_t, the number of transferred chromosomes.
Output: $\mathbf{x}_m^*$, the best chromosome for each T_m, $m = 1, 2$.
for $m = 1, 2$ **do**
  Randomly generate N chromosomes $\{\mathbf{x}_{m,n}\}_{n=1}^{N}$ to initialize the population P_m for Task T_m;
  Compute the fitness for each $\mathbf{x}_{m,n}$ in P_m;
  Sort all chromosomes in P_m in descending order according to their fitness;
end
Set $k = 1$;
while $k \leq K$ **do**
  Compute $\mathbf{m}_1$ and $\mathbf{m}_2$ in (2);
  for $m = 1, 2$ **do**
    Initialize a temporary population $P_t = \emptyset$;
    Construct the first n_t chromosomes of P_t from the n_t best chromosomes in P_{3-m}, using (4);
    Construct the remaining $N - n_t$ chromosomes of P_t as the $N - n_t$ best chromosomes in P_m;
    Construct an index vector $\mathbf{s}$ as a random permutation of $[1, ..., N]$;
    Initialize the offspring population $O = \emptyset$;
    for $n = 1, ..., N/2$ **do**
      Pick two parents $\mathbf{x}_{m,\mathbf{s}(n)}$ and $\mathbf{x}_{m,\mathbf{s}(N/2+n)}$;
      Crossover to generate two offsprings $\mathbf{x}_e$ and $\mathbf{x}_f$ according to (6) and (7);
      Mutate each of $\mathbf{x}_e$ and $\mathbf{x}_f$;
      Add $\mathbf{x}_e$ and $\mathbf{x}_f$ to O;
    end
    Evaluate the fitness of each chromosome in O;
    Set $P = P_m \cup O$;
    Sort the chromosomes in P in descending order according to their fitness;
    Form the new population P_m using the N fittest chromosomes in P;
  end
  $k = k + 1$;
end
Return The fittest chromosome for each T_m, $m = 1, 2$.

between their optimal solutions, and hence, blindly transferring a promising solution from P_1 to P_2 (or the opposite) may not benefit the search. The MTGA first computes the mean of a few fittest chromosomes (four were used in our example) in each population, and then, estimates the bias between their optimal solutions as the difference between these two means. When transferring a promising chromosome from one population to the other, it adds (or subtracts, depending on the direction) this bias to make it more compatible with the new task. In the first few generations, bias estimation may not be very accurate, because the fittest chromosomes in each population may be far away from its global optimum. However, as the evolution goes on, the fittest chromosomes will move toward their corresponding global optima, and hence, the bias estimate will be more accurate, which benefits the transfer more.

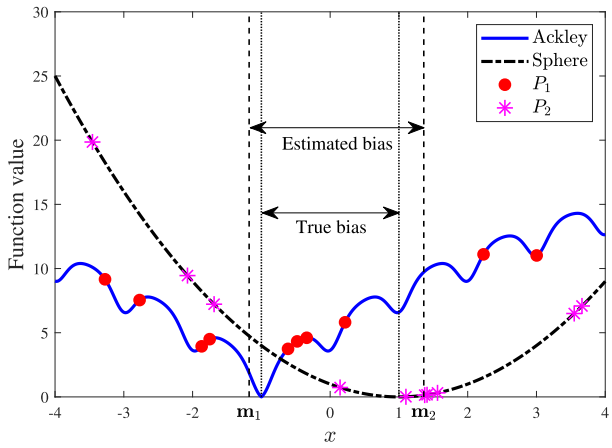


Fig. 1. Illustration of how the MTGA estimates the bias between two tasks. P_1 (P_2) is the population of 1-D Ackley (Sphere). $\mathbf{m}_1$ ($\mathbf{m}_2$) is the mean of the fittest four chromosomes in P_1 (P_2). $|\mathbf{m}_2 - \mathbf{m}_1|$ is the estimated bias.

To effectively transfer the fittest chromosomes between the two populations, the MTGA uses sequential transfer in each generation, i.e., P_1 first transfers its fittest chromosomes to P_2 (after considering the bias), then P_2 performs crossover, mutation, fitness evaluation, and reproduction to generate a new P_2. The fittest chromosomes in this new P_2 are then transferred to P_1 (after considering the bias), which next goes through crossover, mutation, fitness evaluation, and reproduction to generate a new P_1. In this way, the updated fittest chromosomes in one population are immediately used by the other in the same generation, expediting the convergence. On the contrary, the MFEA uses simultaneous chromosome transfer, i.e., the two tasks transfer their fittest chromosomes to the other simultaneously (without considering the bias), and then perform crossover, mutation, fitness evaluation, and reproduction separately. As a result, the updated fittest chromosomes for one task cannot be shared by the other until the next generation, which is a waste of information. The difference is illustrated in Fig. 2.

The details of the MTGA are presented next.

B. Population Initialization

Unlike the MFEA, which keeps a single population and uses a skill factor to identity which chromosome belongs to which task, the MTGA keeps a separate population for each individual task.

Let N be the population size of each task, and P_m the population for T_m ($m = 1, 2$). P_m is randomly initialized in the first generation of the MTGA.

C. Chromosome Transfer

Chromosome transfer is considered in each subsequent iteration. We first compute the means of the best n_t chromosomes in P_1 and P_2, respectively, and denote them as $\mathbf{m}_1$ and $\mathbf{m}_2$. The

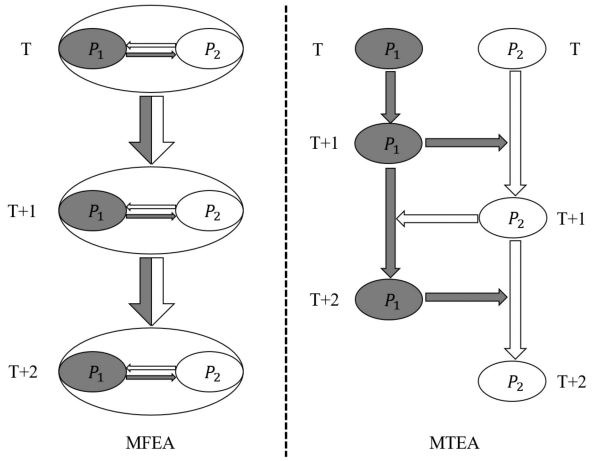


Fig. 2. Illustration of the workflow difference between the MFEA and the MTGA. T is the index of the evolutionary generation.

difference between $\mathbf{m}_1$ and $\mathbf{m}_2$ represents the bias between the two tasks, which can be used to make the transferred chromosomes more consistent with the new population.

Let us focus on T_1 as an example. We transfer n_t best chromosomes from P_2 to P_1 to replace the n_t worst chromosomes in P_1. Assume the chromosomes in P_1 and P_2 have been sorted from the best to the worst according to their fitness, respectively. Denote the sorted chromosomes as $\{\mathbf{x}_{1,n}\}_{n=1}^{N}$ and $\{\mathbf{x}_{2,n}\}_{n=1}^{N}$, respectively. Then

$$\mathbf{m}_1 = \frac{1}{n_t}\sum_{n=1}^{n_t}\mathbf{x}_{1,n} \quad \mathbf{m}_2 = \frac{1}{n_t}\sum_{n=1}^{n_t}\mathbf{x}_{2,n}. \tag{2}$$

We then construct a temporary population P_t as

$$P_t = \left\{\mathbf{x}'_{1,1}, \ldots, \mathbf{x}'_{1,n_t}, \mathbf{x}_{1,1}, \ldots, \mathbf{x}_{1,N-n_t}\right\} \tag{3}$$

where $\mathbf{x}'_{1,n}$ $(n = 1, \ldots, n_t)$ is a transferred chromosome from P_2, computed as follows.

Let $d_m = |\mathbf{X}_m|$ be the dimensionality of the search space of Task T_m, or equivalently, the number of genes in a chromosome in P_m, $m = 1, 2$. When $d_1 \geq d_2$, we construct an index set I by randomly sampling *without* replacement d_2 locations from the d_1 locations. When $d_1 < d_2$, we construct an index set I by randomly sampling *with* replacement d_2 locations from the d_1 locations. Then

$$\mathbf{x}'_{1,n}(i) = \mathbf{x}_{2,n}(I(i)) - \mathbf{m}_2(I(i)) + \mathbf{m}_1(i) \tag{4}$$

where $\mathbf{x}'_{1,n}(i)$ is the ith gene in $\mathbf{x}'_{1,n}$, $\mathbf{x}_{2,n}(I(i))$ is the $I(i)$th gene in $\mathbf{x}_{2,n}$, $\mathbf{m}_2(I(i))$ is the $I(i)$th gene in $\mathbf{m}_2$, $\mathbf{m}_1(i)$ is the ith gene in $\mathbf{m}_1$, $n = 1, \ldots, n_t$, and $i = 1, \ldots, d_1$.

Note that I is usually randomly constructed for each transferred chromosome, i.e., the ith gene in $\mathbf{x}'_{1,n}$ is randomly matched to a gene (not necessarily the ith gene) in $\mathbf{x}_{2,n}$, and the matching is also different for different n. We use a random matching instead of a fixed matching because usually, in practice, we do not know which gene in T_2 can best benefit a gene in T_1, and randomizing the matching for different genes and different n offers more diversity and higher likelihood to find a good matching than a blind fixed matching. However, if we know there is a correspondence between a pair of genes, then a fixed matching may also be used.

Once a matching between $\mathbf{x}'_{1,n}(i)$ and $\mathbf{x}_{2,n}(I(i))$ is established, $\mathbf{m}_1(i) - \mathbf{m}_2(I(i))$ represents the bias between the two genes in the two tasks, and subtracting this bias from $\mathbf{x}_{2,n}(I(i))$ makes the genes in the two tasks more consistent, and hence, facilitates the knowledge transfer.

Also, there could be optimization problems in which some gene values are essentially categorical, and hence, (2) should not be applied to them. For example, for the fuzzy system optimization problem introduced in the next section, we may also want to optimize the rulebase so that good control performance may be obtained by fewer than nine rules, instead of always using the nine rules in Table III. One approach is to add nine new genes to the chromosome, each indicating whether a particular rule in Table III should be used or not. These genes may take values of 0 (the corresponding rule is not used) or 1 (the corresponding rule is used), but we cannot simply compute their average, e.g., the average of 0 and 1 is 0.5, which is not meaningful. In this case, we should exclude these genes from chromosome transfer, and consider only the genes which are truly numerical.

D. Crossover and Mutation

Next, we perform crossover, and make sure the n_t transferred chromosomes from P_2 are all used in the crossover.

We define an index vector $\mathbf{s}$ as a random permutation of $[1, \ldots, N]$. Each time, we pick two parents $\mathbf{x}_{\mathbf{s}(n)}$ and $\mathbf{x}_{\mathbf{s}(n+N/2)}$ $(n = 1, \ldots, N/2)$, and use the simulated binary crossover (SBX) [24] and polynomial mutation [25] operators, the same as those in [10].

For notation simplicity, let the two parents be $\mathbf{x}_{\mathbf{s}(n)} = [\mathbf{x}_a(1), \ldots, \mathbf{x}_a(d)]$ and $\mathbf{x}_{\mathbf{s}(n+N/2)} = [\mathbf{x}_b(1), \ldots, \mathbf{x}_b(d)]$, where d is the dimensionality of the search space. Then, in SBX, we first compute

$$\mathbf{c}(j) = \begin{cases} (2r)^{1/(\beta+1)}, & r \leq 0.5 \\ [2(1-r)]^{-1/(\beta+1)}, & r > 0.5 \end{cases}, \quad j = 1, \ldots, d \tag{5}$$

where β is a user-specified parameter and r is a random number in [0,1], which is regenerated for each j. The two offsprings, $\mathbf{x}_e = [\mathbf{x}_e(1), \ldots, \mathbf{x}_e(d)]$ and $\mathbf{x}_f = [\mathbf{x}_f(1), \ldots, \mathbf{x}_f(d)]$, obtained from the SBX are $(j = 1, \ldots, d)$

$$\mathbf{x}_e(j) = \left[(1 + \mathbf{c}(j))\,\mathbf{x}_a(j) + (1 - \mathbf{c}(j))\,\mathbf{x}_b(j)\right]/2 \tag{6}$$

$$\mathbf{x}_f(j) = \left[(1 + \mathbf{c}(j))\,\mathbf{x}_b(j) + (1 - \mathbf{c}(j))\,\mathbf{x}_a(j)\right]/2. \tag{7}$$

Clearly, $\mathbf{x}_e + \mathbf{x}_f = \mathbf{x}_{\mathbf{s}(n)} + \mathbf{x}_{\mathbf{s}(n+N/2)}$. Additionally, it is easy to observe that $\mathbf{x}_e$ is closer to $\mathbf{x}_{\mathbf{s}(n)}$ than to $\mathbf{x}_{\mathbf{s}(n+N/2)}$, and $\mathbf{x}_f$ is closer to $\mathbf{x}_{\mathbf{s}(n+N/2)}$ than to $\mathbf{x}_{\mathbf{s}(n)}$, because $\mathbf{c}(j) > 0$.

Let η be a user-specified parameter and r be a random number in [0,1]. Then, the polynomial mutation of a gene $\mathbf{x}(j)$ with range $[l, u]$ is computed as [25]

$$\mathbf{x}'(j) = \begin{cases} \mathbf{x}(j) + \left[(2r)^{\frac{1}{1+\eta}} - 1\right](\mathbf{x}(j) - l), & r \leq 0.5 \\ \mathbf{x}(j) + \left[1 - (2(1-r))^{\frac{1}{1+\eta}}\right](u - \mathbf{x}(j)), & r > 0.5. \end{cases} \tag{8}$$

After mutation, $\mathbf{x}_e$ and $\mathbf{x}_f$ are added to the offspring population O. The abovementioned crossover and mutation operations are repeated $N/2$ times so that O has N chromosomes.

E. Reproduction

We then evaluate the fitness of each chromosome in O, combine the chromosomes in O with those in P_m, and sort the $2\tilde{}N$ chromosomes from the best to the worse according to their fitness. We use an elitist selection mechanism to prorogate the first N best chromosomes to the next generation for T_m.

III. Experiments on Benchmarks

This section compares the MTGA with eight state-of-the-art single-task and multitask evolutionary algorithms on the nine benchmarks introduced in [10].

A. Performance Measures

In addition to the error (the difference from the known minimum) in each task, the simple performance metric proposed in [10] is also used to quantify the performance of different algorithms.

Assume that there are K algorithms, $A_1, \ldots, A_K$ for a problem with M minimization tasks $T_1, \ldots, T_M$, and each algorithm has been run for L repetitions. Let $B(k,m)_l$ denote the best obtained result on the lth repetition by Algorithm A_k on Task T_m, and μ_m and σ_m be the mean and standard deviation (std) of $B(k,m)_l$, $k = 1, \ldots, K$, $l = 1, \ldots, L$. Then, the normalized performance $B'(k,m)_l$ is computed as

$$B'(k,m)_l = \frac{B(k,m)_l - \mu_m}{\sigma_m} \tag{9}$$

and the *performance score* of Algorithm A_k is

$$s_k = \sum_{m=1}^{M} \sum_{l=1}^{L} B'(k,m)_l. \tag{10}$$

A smaller performance score indicates a better overall performance.

B. Algorithms

We compare the performance of our proposed MTGA with a classic single-task evolutionary algorithm, and seven state-of-the-art multitask algorithms given as follows.

1) The single-objective evolutionary algorithm (SOEA), which considers each task independently. We used the SOEA code (in MATLAB) provided in the WCCI2018 competition on evolutionary MTO. Essentially, each SOEA implements a genetic algorithm [26], [27], with SBX crossover and polynomial mutation.
2) The MFEA [8]. We also used the MFEA code (in MATLAB) provided in the WCCI2018 competition on evolutionary MTO.
3) The evolution of biocoenosis through symbiosis (EBS) algorithm [13], which can deal with many-tasking problems. The basic evolutionary algorithm used in the EBS was identical to the genetic algorithm in the SOEA and the MFEA.
4) The MFEA-LBS [11], which employs a permutation-based unified representation and level-based selection (LBS) to enhance the original MFEA.
5) The MFEA with resource reallocation (MFEARR) [18], which adds a resource allocation mechanism to facilitate the discovery and utilization of synergy among tasks.
6) The LDA-MFEA [12], which uses LDA to transform the search space of a simple task to a new one, similar to its constitutive complex task for efficient problem solving.
7) The generalized MFEA (G-MFEA) [14], which uses decision variable translation and shuffling strategies to facilitate knowledge transfer between optimization problems.
8) The evolutionary multitasking via explicit autoencoding (EMEA) algorithm [15], which uses autoencoding to explicitly transfer knowledge across tasks in evolutionary multitasking.

We implemented the last six algorithms in MATLAB according to the corresponding publications, and tried our best to optimize them.

We used a population size of 200 in the MFEA and its variants. For the SOEA, the G-MTGA, the EMEA, and the MTGA, each task had a population size of 100. The maximum number of function evaluations was 100 000 for all algorithms, i.e., all algorithms terminated after 500 iterations. rmp $= 0.3$, $\beta = 2$ in (5) of the SBX, and $\eta = 5$ in (8) of polynomial mutation, were used in all algorithms. Additionally, $n_t = 10$ was used in the EMEA (as in [15]), and $n_t = 40$ in the MTGA.

To cope with the randomness, each algorithm was run 20 times, each time with a randomly initialized population. Then, several statistics, such as the mean and std of the objectives in the two tasks, were computed.

C. Experimental Results

Individual experimental results on the nine benchmarks are shown in Fig. 3, and the average performance scores of the nine algorithms across the nine benchmarks are shown in Fig. 4. The following observations are made.

1) On average, all multitask algorithms outperformed the SOEA, which suggests that the transfer of information between the tasks can indeed improve the overall optimization performance.
2) On average, the EBS and the MFEARR performed worse than the MFEA, and the MFEA-LBS had comparable performance as the MFEA.
3) The LDA-MFEA achieved the best performance when the number of function evaluations was small, but gradually degraded when the number of function evaluations increased.
4) The G-MFEA slightly outperformed the MFEA when the number of function evaluations was large.
5) Among the eight existing algorithms, on average, the EMEA achieved the second best performance (worse only

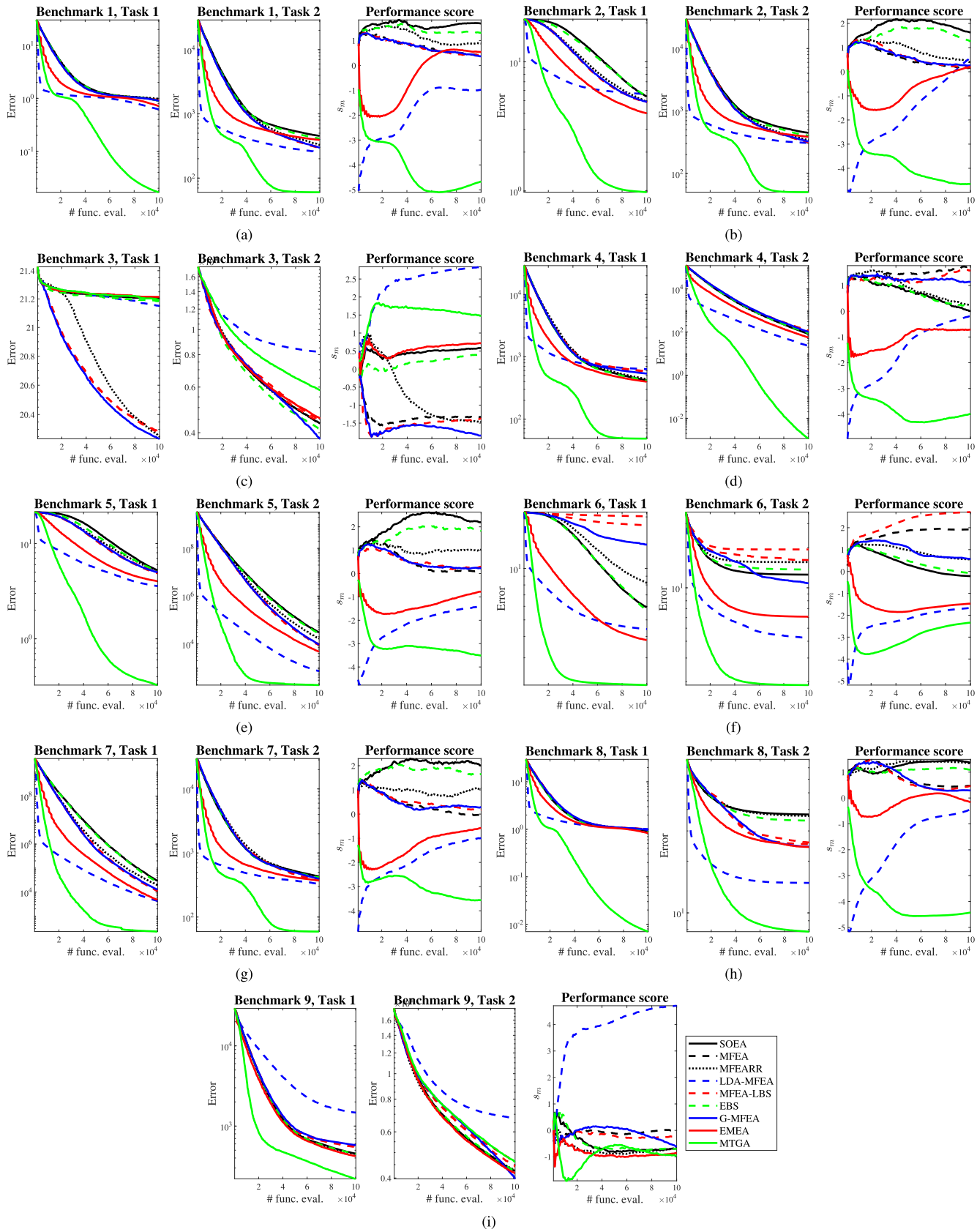


Fig. 3. Experimental results on the nine benchmarks.

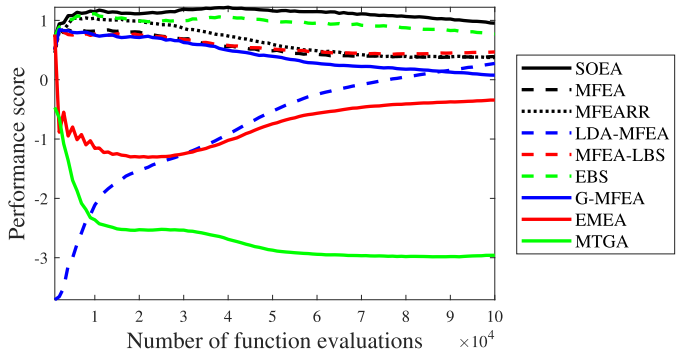


Fig. 4. Average performance scores across the nine benchmarks.

than the LDA-MFEA) when the number of function evaluations was small, and the best performance when the number of function evaluations was large.

6) Among all nine algorithms, on average, our proposed MTGA achieved the second best performance (worse only than the LDA-MFEA) when the number of function evaluations was small, and the best performance when the number of function evaluations was large.

The average errors (mean and std) of different algorithms on different tasks, after 100 000 function evaluations, are given in Table I. Observe that the MTGA achieved the best performance score in eight out of the nine benchmarks. Overall, it dominated the other eight algorithms. More experimental results on the parameter sensitivity of the MTGA, and its performance on nine more challenging and practical benchmarks, can be found in the Supplementary Material.

Fig. 3 and Table I also show that MTGA did not achieve the best performance on Benchmarks 3 and 9. The intertask similarity of the two tasks in each benchmark is given in [10, Table III]. The two tasks in Benchmarks 3 and 9 have the lowest intertask similarity (0.0002 and 0.0016, respectively) among the nine benchmarks. This fact may explain why MTGA did not perform well on them—though MTGA explicitly considers the bias between two tasks, it is still based on the assumption that the two tasks are similar. When the two tasks are almost completely different, MTGA has difficulty transferring useful information between them. This is a problem deserving future research.

The normalized mean and std of different algorithms after 100 000 function evaluations, w.r.t. the SOEA, are shown in Fig. 5. Clearly, the MTGA achieved, on average, the smallest mean and std, suggesting that the MTGA consistently outperformed other approaches.

It is also interesting to compare the computational cost of different algorithms. For each benchmark, we normalized the actual computation time (recorded in MATLAB) of the eight multitask algorithms w.r.t. the SOEA, and plot the results in Fig. 6. On average, the SOEA, the EMEA, and the MTGA had comparable computational cost (with values 1.0000, 0.9891, and 1.0368, respectively, in the last group of Fig. 6), and they were the fastest among the nine. The MFEA-LBS and the EBS also had comparable computational cost, all of which were much smaller than the MFEA, the G-MFEA, and the MFEARR. The computational cost of the LDA-MFEA was almost three times higher than the MTGA.

In summary, we can conclude that the proposed MTGA is effective and efficient.

IV. MTGA-Based Fuzzy System Optimization

This section applies the MTGA to fuzzy system optimization.

IT2 fuzzy systems have become very popular in the last two decades, because they have demonstrated outstanding performances in numerous applications [27]–[30]. Evolutionary algorithms, such as genetic algorithms, are frequently used to optimize IT2 fuzzy systems. There are generally, the following two strategies for evolutionary IT2 fuzzy system optimization in the literature.

1) *Partially dependent strategy:* An optimal T1 fuzzy system is designed first, and then, its membership functions are blurred to obtain an IT2 fuzzy system.
2) *Totally independent strategy:* An IT2 fuzzy system is optimized from scratch, without using a baseline T1 fuzzy system.

Based on the MTGA, this section proposes a novel *simultaneous optimization strategy* for fuzzy system design, which simultaneously optimizes a T1 fuzzy system and an IT2 fuzzy system. This strategy has the following advantages.

1) Optimizing two fuzzy systems simultaneously by the MTGA may result in better fuzzy systems than optimizing each separately.
2) Obtaining a T1 fuzzy system and an IT2 fuzzy system simultaneously enables one to compare their performances and better determine which one to use in practice—if their performances are similar, then the T1 fuzzy system may be preferred for its simplicity; otherwise, the better performing one (which is usually the IT2 fuzzy system) is chosen.

All MTO algorithms compared in the previous section can be used in the *simultaneous optimization strategy*. As an example, we compare MTGA with SOEA and MFEA in this section, on simultaneously optimizing T1 and IT2 FLCs for coupled-tank water level control [27], [28]. Their parameters were identical to those used in the previous section. Each algorithm was again run 20 times.

A. Coupled-Tank Water Level Control System

T1 and IT2 FLCs were optimized to control the water level of the coupled-tank system shown in Fig. 7. The plant has two small tower-type tanks mounted above a reservoir. Water can be pumped into the top of each tank by two independent pumps. The water levels are measured by two capacitive-type probe sensors. Each tank has an outlet, which allows water to flow out. Raising the baffle between the two tanks allows water to flow between them. The amount of water that returns to the reservoir is approximately proportional to the square root of the height of water in the tank, and hence, the system is nonlinear.

TABLE I
AVERAGE ERRORS (MEAN AND PARENTHESIZED STD) OF DIFFERENT ALGORITHMS ON THE NINE BENCHMARKS, AFTER 100 000 FUNCTION EVALUATIONS

	Benchmark		1	2	3	4	5	6	7	8	9
SOEA	T1	mean	0.8996	5.3695	21.2084	417.8177	5.2545	5.9086	29341.5108	0.9121	452.8721
		std	(0.0685)	(1.0510)	(0.0397)	(50.3642)	(0.6350)	(3.5919)	(13428.4702)	(0.0530)	(63.0397)
	T2	mean	453.9618	446.0697	4350.4544	81.9612	30552.1236	12.5301	431.0700	37.4077	4261.6635
		std	(54.2633)	(50.7552)	(567.0568)	(23.4086)	(12233.1876)	(2.5087)	(51.2238)	(4.2753)	(438.0050)
	Score		1.6829	1.6159	0.5881	0.0030	2.1983	−0.2145	2.0058	1.3676	−0.6717
MFEA	T1	mean	0.9312	4.8785	20.2849	587.5578	4.8739	17.8681	10833.5256	0.9763	568.3542
		std	(0.0521)	(0.5738)	(0.0537)	(72.4783)	(0.3930)	(5.5267)	(3463.6525)	(0.0422)	(85.7772)
	T2	mean	294.6075	316.0845	4450.4887	116.8460	8843.0128	16.1079	371.8389	25.6697	4609.5203
		std	(40.8627)	(29.1948)	(646.8284)	(22.8572)	(2198.8143)	(5.1747)	(49.9547)	(3.4795)	(495.9894)
	Score		0.4379	0.1710	−1.3312	1.7700	0.1219	1.9218	−0.0526	0.4679	0.0027
MFEARR	T1	mean	0.9779	4.8817	20.2536	439.1479	5.2223	8.2290	19331.3190	0.9241	448.3269
		std	(0.0420)	(0.3708)	(0.0752)	(74.1382)	(0.4816)	(6.1517)	(8251.8875)	(0.0865)	(49.1185)
	T2	mean	328.0556	341.4237	4343.4306	86.2004	16927.1056	15.5357	419.0696	36.7019	4286.2454
		std	(30.5211)	(44.2208)	(586.3652)	(27.3131)	(6844.0965)	(2.9138)	(33.8599)	(4.8207)	(517.4152)
	Score		0.8810	0.3928	−1.4646	0.2216	1.0270	0.5497	1.0601	1.3404	−0.6591
LDA-MFEA	T1	mean	0.6213	5.6270	21.1512	633.4187	3.5734	4.4146	4250.2864	1.0077	1472.9144
		std	(0.1302)	(0.8936)	(0.1043)	(126.1722)	(0.3997)	(0.8332)	(1591.8700)	(0.0352)	(377.9349)
	T2	mean	252.7188	307.0645	8187.2562	22.4887	723.6025	4.2403	326.0426	14.9481	6773.0865
		std	(67.8589)	(78.7763)	(2356.7227)	(9.0959)	(211.2468)	(1.2617)	(69.9058)	(1.8482)	(882.5549)
	Score		−0.9739	0.5781	2.8382	−0.1850	−1.3942	−1.6682	−0.9915	−0.4437	4.7109
MFEA-LBS	T1	mean	0.9033	4.8463	20.2795	577.3551	5.0222	20.1757	11263.0957	0.9909	545.5573
		std	(0.0602)	(0.4312)	(0.0693)	(86.9941)	(0.4057)	(0.0795)	(3922.9374)	(0.0296)	(75.9845)
	T2	mean	304.3237	319.2167	4404.1627	111.6610	9756.5485	19.3068	393.8628	25.1179	4424.1310
		std	(42.9904)	(39.1217)	(620.6942)	(22.3018)	(3668.2769)	(2.1333)	(68.9549)	(3.8481)	(550.6847)
	Score		0.4254	0.1773	−1.3714	1.5914	0.2937	2.7136	0.1679	0.4639	−0.2499
EBS	T1	mean	0.8927	5.4012	21.1841	415.1442	5.1704	5.6786	26599.1474	0.9141	450.3920
		std	(0.0730)	(1.4196)	(0.0455)	(43.3671)	(0.5054)	(2.1553)	(12441.1179)	(0.0509)	(51.0178)
	T2	mean	412.7935	401.5351	4121.1952	89.3588	26750.3202	13.6764	416.4035	34.4474	4245.4670
		std	(48.4015)	(49.4289)	(773.8833)	(20.2276)	(10428.6188)	(2.9505)	(43.3474)	(7.9360)	(569.8224)
	Score		1.3101	1.2505	0.3941	0.1614	1.8235	−0.0717	1.6515	1.0940	−0.6951
G-MFEA	T1	mean	0.9015	4.8973	**20.2362**	538.3215	4.9770	13.7719	12605.2897	0.9645	579.1136
		std	(0.0679)	(0.4284)	(0.0647)	(63.7993)	(0.4259)	(9.0053)	(5139.6811)	(0.0483)	(106.9839)
	T2	mean	298.4773	325.5007	**3790.2887**	101.2819	9254.1989	10.8155	395.5002	24.2534	**3972.5676**
		std	(32.0489)	(37.5079)	(565.2324)	(24.6364)	(3145.3736)	(7.1813)	(45.4696)	(4.2472)	(503.3188)
	Score		0.3695	0.2648	**−1.8440**	1.1298	0.2224	0.5746	0.2950	0.2950	−0.6193
EMEA	T1	mean	0.7180	3.9971	21.2144	398.5276	4.0135	3.8051	4815.9985	0.8285	422.8324
		std	(0.0665)	(0.4488)	(0.0268)	(49.3249)	(0.2851)	(0.3574)	(2148.7461)	(0.0882)	(47.2119)
	T2	mean	391.7383	386.5877	4530.9415	56.2079	4667.8277	6.0926	370.2845	24.1708	4164.9409
		std	(68.6963)	(47.1808)	(760.3681)	(11.7526)	(1854.2606)	(2.5838)	(54.6887)	(4.8453)	(608.4902)
	Score		0.5359	0.2108	0.7125	−0.7085	−0.7801	−1.4692	−0.5737	−0.1614	−0.8535
MTGA	T1	mean	**0.0168**	**0.9795**	21.1927	**48.5233**	**0.3267**	**2.0780**	**226.9160**	**0.0070**	**220.2267**
		std	(0.0174)	(0.6684)	(0.0383)	(13.0611)	(0.4127)	(0.4938)	(404.0593)	(0.0058)	(63.5012)
	T2	mean	**60.1727**	**50.1522**	5844.5964	**0.0011**	**187.5526**	**1.8856**	**59.2972**	**7.7735**	4602.3325
		std	(19.1548)	(15.1310)	(719.3296)	(0.0016)	(339.1746)	(1.8059)	(19.6508)	(7.0682)	(492.2891)
	Score		**−4.6689**	**−4.6611**	1.4783	**−3.9836**	**−3.5124**	**−2.3361**	**−3.5624**	**−4.4237**	**−0.9648**

Best performances are marked in bold.

The dynamics of the coupled-tank plant is described as

$$A_1 \frac{dH_1}{dt} = Q_1 - \alpha_1\sqrt{H_1} - \alpha_3\sqrt{H_1 - H_2} \tag{11}$$

$$A_2 \frac{dH_2}{dt} = Q_2 - \alpha_2\sqrt{H_2} + \alpha_3\sqrt{H_1 - H_2} \tag{12}$$

where A_1 and A_2 are the cross-sectional areas of Tanks #1 and #2; H_1 and H_2 are the liquid levels in Tanks #1 and #2; Q_1 and Q_2 are the volumetric flow rates ($\mathbf{cm^3/s}$) of Pumps #1 and #2; α_1, α_2, and α_3 are the proportionality constants corresponding to the $\sqrt{H_1}$, $\sqrt{H_2}$, and $\sqrt{H_1 - H_2}$ terms, respectively.

In our experiment, Outlet 1 and Pump #2 were shut off, and Pump #1 was used to control the water level in Tank #2. In this case, $H_1 \geq H_2$ always held.

B. Fitness Function

Each chromosome in the population was evaluated on four different configurations of the couple-tank plant, shown in Table II. This enables the optimized FLC to cope with a wide range of modeling uncertainties, and hence, has a better chance to perform well on the actual plant. The same strategy was also used in our previous research [27], [28].

The fitness of each chromosome was the inverse of the sum of the integral of the time-weighted absolute errors (ITAEs) on the four plants [27], [28]

$$F = \left[\sum_{p=1}^{4} w_p \left(\sum_{t=1}^{N_p} t \cdot |e_p(t)|\right)\right]^{-1} \tag{13}$$

where $e_p(t)$ is the difference between the set point and the actual water height at the tth sampling of the pth plant, w_p is the weight corresponding to the ITAE of the pth plant, and $N_p = 200$ is the number of samples. Because the ITAE of the third plant is usually several times bigger than the others, to balance the contributions from the four plants, α_3 was defined as $1/3$ while the other three weights were unity.

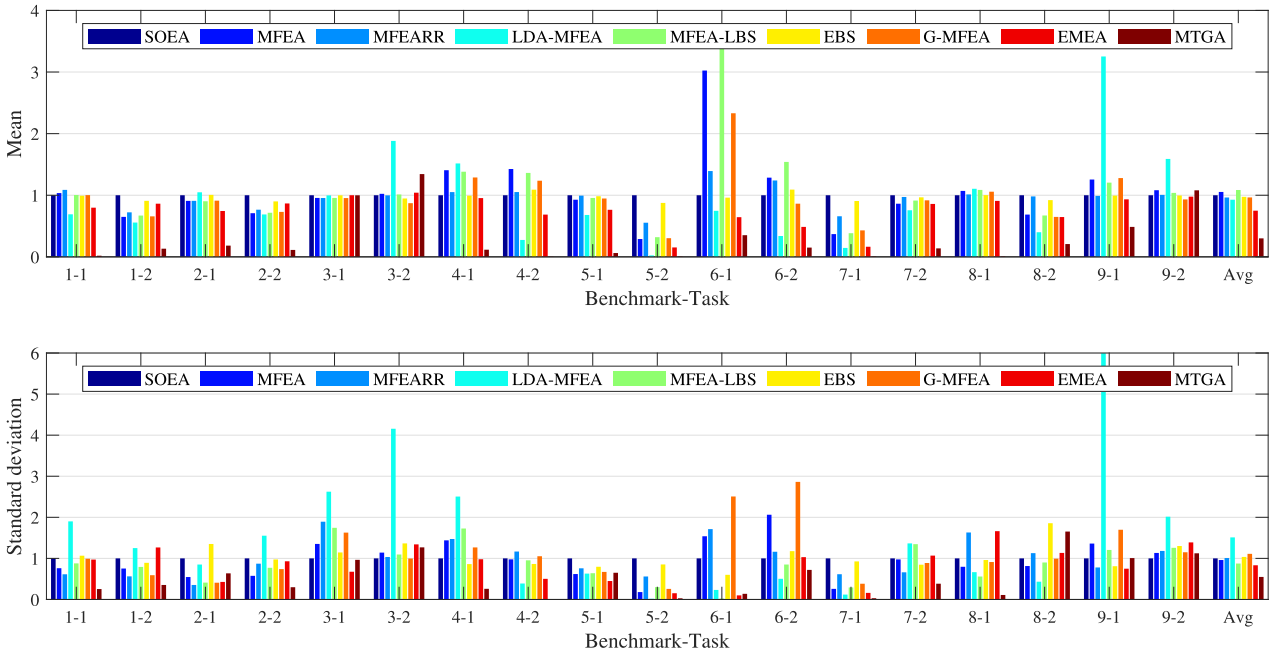


Fig. 5. Normalized mean and std of the errors of different algorithms (w.r.t. the SOEA) on the nine benchmarks after 100 000 function evaluations.

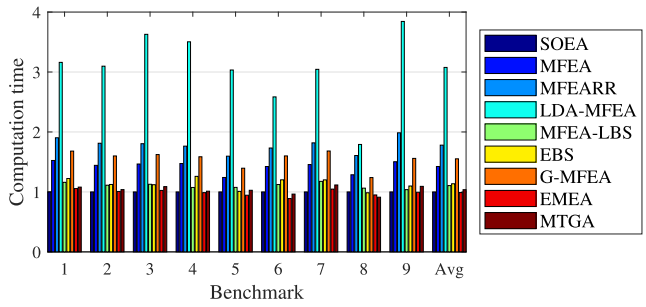


Fig. 6. Computation time of the nine algorithms.

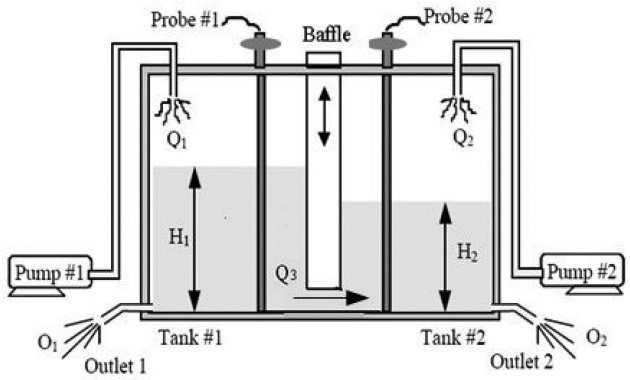


Fig. 7. Coupled-tank water-level control system.

C. FLC Structure and Parameters

Proportional-integral FLCs were used. The inputs were the feedback error e and the change of error $\dot{e}$. The output was the change of control signal $\dot{u}$. Three Gaussian membership functions were used in each input domain. The rulebase is shown in Table III, which included only five different consequents $\dot{u}_i$ ($i = 1, 2, \ldots, 5$). Each T1 Gaussian membership function had two parameters—mean (m) and std (δ). Each IT2 Gaussian membership function with a fixed mean and uncertain std had three parameters—mean (m) and the bounds of the std ($[\delta_l, \delta_r]$). So, the T1 FLC (FLC$_1$) had $2 \times 3 \times 2 + 5 = 17$ parameters, and the IT2 FLC (FLC$_2$) had $3 \times 3 \times 2 + 5 = 23$ parameters. The coding scheme is shown in Fig. 8.

There were some natural constraints on the relative values of the parameters, as shown in Fig. 8, e.g., $m_{E_1} < m_{E_2} < m_{E_3}$, where m_{E_1}, m_{E_2}, and m_{E_3} are the mean of the Gaussian membership function of E_1, E_2, and E_3, respectively. So, we need to rerank the genes to satisfy the constraints before evaluating the fitness of each chromosome.

In the previous section, MTGA used random matching in chromosome transfer. However, in this application, each parameter in the two FLCs has a specific physical meaning, which can

TABLE II
FOUR CONFIGURATIONS OF THE WATER-LEVEL CONTROL PLANT FOR FITNESS EVALUATION

	I	II	III	IV
$A_1 = A_2$ (cm^2)	36.52	36.52	36.52	36.52
$\alpha_1 = \alpha_2$	5.6186	5.6186	5.6186	5.6186
α_3	10	10	10	8
Setpoint (cm)	$0 \rightarrow 15$	$0 \rightarrow 15$	$0 \rightarrow 22.5 \rightarrow 7.5$	$0 \rightarrow 15$
Time delay (s)	0	2	0	0

TABLE III
THE FUZZY RULEBASE USED BY BOTH T1 AND IT2 FLCS

$e \backslash \dot{e}$	$\dot{E}_1$	$\dot{E}_2$	$\dot{E}_3$
E_1	$\dot{u}_1$	$\dot{u}_2$	$\dot{u}_3$
E_2	$\dot{u}_2$	$\dot{u}_3$	$\dot{u}_4$
E_3	$\dot{u}_3$	$\dot{u}_4$	$\dot{u}_5$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
FLC_1	m_{E_1}	m_{E_2}	m_{E_3}	$m_{\dot{E}_1}$	$m_{\dot{E}_2}$	$m_{\dot{E}_3}$	δ_{E_1}	δ_{E_2}	δ_{E_3}	$\delta_{\dot{E}_1}$	$\delta_{\dot{E}_2}$	$\delta_{\dot{E}_3}$	$\dot{u}_1$	$\dot{u}_2$	$\dot{u}_3$	$\dot{u}_4$	$\dot{u}_5$						
FLC_2	m_{E_1}	m_{E_2}	m_{E_3}	$m_{\dot{E}_1}$	$m_{\dot{E}_2}$	$m_{\dot{E}_3}$	$\delta_{E_{1l}}$	$\delta_{E_{1r}}$	$\delta_{E_{2l}}$	$\delta_{E_{2r}}$	$\delta_{E_{3l}}$	$\delta_{E_{3r}}$	$\delta_{\dot{E}_{1l}}$	$\delta_{\dot{E}_{1r}}$	$\delta_{\dot{E}_{2l}}$	$\delta_{\dot{e}_{2r}}$	$\delta_{\dot{E}_{3l}}$	$\delta_{\dot{E}_{3r}}$	$\dot{u}_1$	$\dot{u}_2$	$\dot{u}_3$	$\dot{u}_4$	$\dot{u}_5$

$m_{E_1} < m_{E_2} < m_{E_3}$, $m_{\dot{E}_1} < m_{\dot{E}_2} < m_{\dot{E}_3}$, $\delta_{E_{il}} < \delta_{E_{ir}}$, $\delta_{\dot{E}_{il}} < \delta_{\dot{E}_{ir}}$, $i = 1, 2, 3$, $\dot{u}_1 < \dot{u}_2 < \dot{u}_3 < \dot{u}_4 < \dot{u}_5$
$m_{E_i} \in [-20, 20]$, $m_{\dot{E}_i} \in [-3, 3]$, $\delta_{E_i} \in [1, 6]$, $\delta_{\dot{E}_i} \in [0.1, 0.8]$, $\dot{u} \in [-1, 1]$

Fig. 8. Coding scheme of the T1 and IT2 FLCs.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
FLC_1	m_{E_1}	m_{E_2}	m_{E_3}	$m_{\dot{E}_1}$	$m_{\dot{E}_2}$	$m_{\dot{E}_3}$	δ_{E_1}	δ_{E_2}	δ_{E_3}	$\delta_{\dot{E}_1}$	$\delta_{\dot{E}_2}$	$\delta_{\dot{E}_3}$	$\dot{u}_1$	$\dot{u}_2$	$\dot{u}_3$	$\dot{u}_4$	$\dot{u}_5$
	1	2	3	4	5	6	7	9	11	13	15	17	19	20	21	22	23
FLC_2	m_{E_1}	m_{E_2}	m_{E_3}	$m_{\dot{E}_1}$	$m_{\dot{E}_2}$	m_{E_3}	$\delta_{E_{1l}}$	$\delta_{E_{2l}}$	$\delta_{E_{3l}}$	$\delta_{\dot{E}_{1l}}$	$\delta_{\dot{E}_{2l}}$	$\delta_{\dot{E}_{3l}}$	$\dot{u}_1$	$\dot{u}_2$	$\dot{u}_3$	$\dot{u}_4$	$\dot{u}_5$

Fig. 9. Gene matching relationship in chromosome transfer between the T1 and IT2 FLCs.

be used to find the correspondence between parameters of the two FLCs. So, we used fixed matching in chromosome transfer, as shown in Fig. 9. The two genes in the same column were matched in chromosome transfer, and the genes in FLC_2 that do not appear in Fig. 9 were not used in chromosome transfer. More specifically, $\delta_{E_{ir}}$ and $\delta_{\dot{E}_{ir}}$ $(i = 1, 2, 3)$ in the original FLC_2 chromosome were kept unchanged. However, because $\delta_{E_{il}} \le \delta_{E_{ir}}$ and $\delta_{\dot{E}_{il}} \le \delta_{\dot{E}_{ir}}$ should always be satisfied, when the transferred genes violated these constraints, the corresponding $\delta_{E_{il}}$ and $\delta_{E_{ir}}$ (or $\delta_{\dot{E}_{il}}$ and $\delta_{\dot{E}_{ir}}$) were switched.

It is also possible to transfer the average value of the upper and lower membership functions, instead of the upper or lower membership function only. This approach is similar to the *partially dependent strategy* introduced at the beginning of this section, in which an optimal T1 fuzzy system is designed first, and then, its membership functions are blurred to obtain an IT2 fuzzy system. A frequently used approach for blurring the std of the membership function is to change the std of a T1 membership function, e.g., δ, to an interval $[\delta - \sigma, \delta + \sigma]$ centered at δ. This chromosome transfer approach will be considered in our future research.

D. Optimization Results

The optimization results of the three algorithms, averaged over 20 runs, are shown in Fig. 10. We only plot the results after 25 generations so that they can be better distinguished.

Fig. 10 shows the following:

1) When two SOEAs were used separately to optimize the T1 and IT2 FLCs, on average, the IT2 FLC had smaller ITAE than the T1 FLC, which is consistent with many results in the literature [27], [28], and also our expectation.
2) When the MFEA was used to optimize the T1 and IT2 FLCs simultaneously, FLC_2 may not outperform FLC_1, which is contradicting with our expectation, and suggests that MFEA may not be suitable for this real-world application.

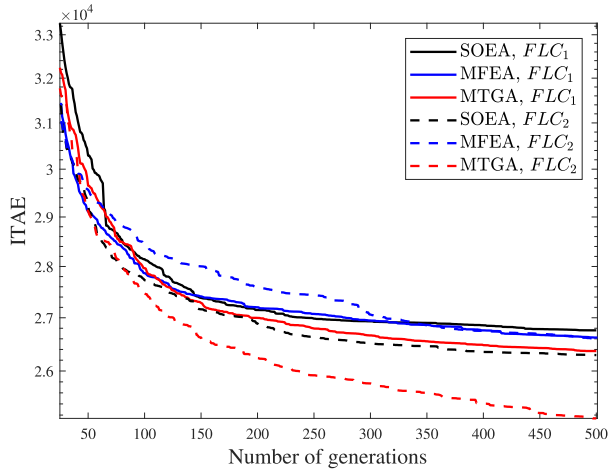


Fig. 10. ITAEs of the three optimization algorithms. To better visualize the differences, we only show the ITAEs after 25 generations.

3) When the MTGA was used to optimize the T1 and IT2 FLCs simultaneously, the IT2 FLC outperformed the T1 FLC, which coincided with the results in SOEA and also our expectation.
4) When the number of generations was large (e.g., larger than 125), the T1 (IT2) FLC obtained from the MTGA almost always gave a smaller ITAE than those obtained from the SOEA and the MFEA, suggesting that the MTGA is superior to the SOEA and the MFEA in real-world FLC optimization.

The membership functions of the best T1 and IT2 FLCs, obtained both from the MTGA, are shown in Fig. 11. The corresponding rule consequents are shown in Table IV. It is interesting that E_2 and $\dot{E}_3$ of the IT2 FLC had almost completely filled-in footprint of uncertainties [29], i.e., the lower membership functions were very narrow.

In summary, we have demonstrated that when the MTGA is used, the newly proposed *simultaneous optimization strategy*

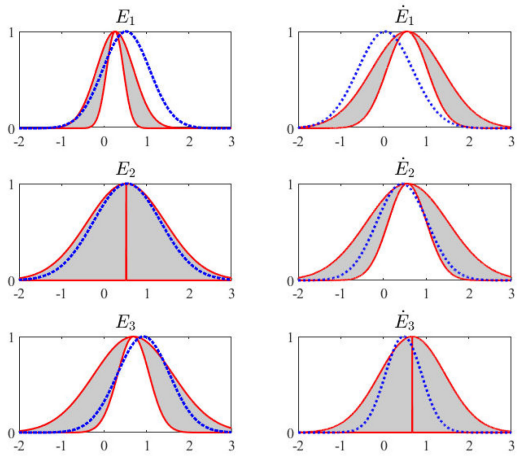


Fig. 11. Membership functions of the best T1 FLC (blue dotted curves) and IT2 FLC (red solid curves).

TABLE IV
RULE CONSEQUENTS OF THE BEST T1 FLC AND IT2 FLC (SHOWN IN PARENTHESES)

$e\backslash\dot{e}$	$\dot{E}_1$	$\dot{E}_2$	$\dot{E}_3$
E_1	0.0338 (0.0636)	0.1193 (0.3213)	0.5345 (0.6013)
E_2	0.1193 (0.3213)	0.5345 (0.6013)	0.7438 (0.7949)
E_3	0.5345 (0.6013)	0.7438 (0.7949)	0.9970 (0.8568)

can be used to effectively optimize a T1 FLC and an IT2 FLC together.

V. CONCLUSION

Evolutionary multitasking or MFO, is an emerging subfield of MTO, which integrates evolutionary computation and multitask learning to optimize multiple optimization tasks simultaneously. This article has proposed a novel and easy-to-implement MTGA, which copes well with different tasks by estimating and using the bias among them. Comparative studies with eight existing approaches in the literature on nine benchmarks demonstrated that, on average, MTGA outperformed all of them, and had lower computational cost than six of them. Moreover, MTGA also demonstrated outstanding performance in simultaneous optimization of a T1 FLC and an IT2 FLC for coupled-tank water level control, which represents a brand-new *simultaneous optimization strategy* for fuzzy system design. Although we only demonstrated the effectiveness of the MTGA-based *simultaneous optimization strategy* in FLC optimization as an example, it could also be used in other applications of fuzzy systems, e.g., classification and regression.

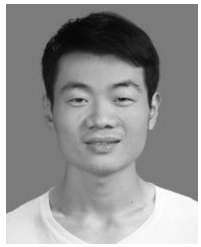
REFERENCES

[1] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, 1997.
[2] D. Wu and J. Huang, "Affect estimation in 3D space using multi-task active learning for regression," *IEEE Trans. Affect. Comput.*, to be published, doi: 10.1109/TAFFC.2019.2916040.
[3] 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.
[4] D. Wu, "Online and offline domain adaptation for reducing BCI calibration effort," *IEEE Trans. Human-Mach. Syst.*, vol. 47, no. 4, pp. 550–563, Aug. 2017.
[5] D. Wu, V. J. Lawhern, S. Gordon, B. J. Lance, and C.-T. Lin, "Driver drowsiness estimation from EEG signals using online weighted adaptation regularization for regression (OwARR)," *IEEE Trans. Fuzzy Syst.*, vol. 25, no. 6, pp. 1522–1535, Dec. 2017.
[6] D. Wu, V. J. Lawhern, W. D. Hairston, and B. J. Lance, "Switching EEG headsets made easy: Reducing offline calibration effort using active wighted adaptation regularization," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 24, no. 11, pp. 1125–1137, Nov. 2016.
[7] K. Swersky, J. Snoek, and R. P. Adams, "Multi-task Bayesian optimization," in *Proc. 26th Int. Conf. Neural Inf. Process. Syst.*, Lake Tahoe, NV, USA, Dec. 2013, pp. 2004–2012.
[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] Y. S. Ong and A. Gupta, "Evolutionary multitasking: A computer science view of cognitive multitasking," *Cogn. Comput.*, vol. 8, no. 2, pp. 125–142, 2016.
[10] B. Da *et al.*, "Evolutionary multitasking for single-objective continuous optimization: Benchmark problems, performance metric, and baseline results," 2017, *arXiv:1706.03470*.
[11] Y. Yuan, Y.-S. Ong, A. Gupta, P. S. Tan, and H. Xu, "Evolutionary multitasking in permutation-based combinatorial optimization problems: Realization with TSP QAP LOP and JSP," in *Proc. IEEE Region 10 Conf.*, 2016, pp. 3157–3164.
[12] K. K. Bali, A. Gupta, L. Feng, Y. S. Ong, and T. P. Siew, "Linearized domain adaptation in evolutionary multitasking," in *Proc. IEEE Int. Conf. Evol. Comput.*, San Sebastian, Spain, Jun. 2017, pp. 1295–1302.
[13] R.-T. Liaw and C.-K. Ting, "Evolutionary many-tasking based on biocoenosis through symbiosis: A framework and benchmark problems," in *Proc. IEEE Int. Conf. Evol. Comput.*, San Sebastian, Spain, Jun. 2017, pp. 2266–2273.
[14] J. Ding, C. Yang, Y. Jin, and T. Chai, "Generalized multi-tasking for evolutionary optimization of expensive problems," *IEEE Trans. Evol. Comput.*, vol. 23, no. 1, pp. 44–58, Feb. 2019.
[15] L. Feng *et al.*, "Evolutionary multitasking via explicit autoencoding," *IEEE Trans. Cybern.*, vol. 49, no. 9, pp. 3457–3470, Sep. 2019.
[16] R. Hashimoto, H. Ishibuchi, N. Masuyama, and Y. Nojima, "Analysis of evolutionary multi-tasking as an island model," in *Proc. Genetic Evol. Comput. Conf. Companion*, Kyoto, Japan, Jul. 2018, pp. 1894–1897.
[17] M. Gong, Z. Tang, H. Li, and J. Zhang, "Evolutionary multitasking with dynamic resource allocating strategy," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 858–869, Oct. 2019.
[18] Y.-W. Wen and C.-K. Ting, "Parting ways and reallocating resources in evolutionary multitasking," in *Proc. IEEE Int. Conf. Evol. Comput.*, San Sebastian, Spain, Jun. 2017, pp. 2404–2411.
[19] M.-Y. Cheng, A. Gupta, Y.-S. Ong, and Z.-W. Ni, "Coevolutionary multitasking for concurrent global optimization: With case studies in complex engineering design," *Eng. Appl. Artif. Intell.*, vol. 64, pp. 13–24, 2017.
[20] Q. Chen, X. Ma, Z. Zhu, and Y. Sun, "Evolutionary multi-tasking single-objective optimization based on cooperative co-evolutionary memetic algorithm," in *Proc. 13th Int. Conf. Comput. Intell. Secur.*, 2017, pp. 197–201.
[21] R. Sagarna and Y.-S. Ong, "Concurrently searching branches in software tests generation through multitask evolution," in *Proc. IEEE Symp. Ser. Comput. Intell.*, 2016, pp. 1–8.
[22] Z. Tang, M. Gong, and M. Zhang, "Evolutionary multi-task learning for modular extremal learning machine," in *Proc. IEEE Int. Conf. Evol. Comput.*, San Sebastian, Spain, Jun. 2017, pp. 474–479.
[23] H. Li, Y. Ong, M. Gong, and Z. Wang, "Evolutionary multitasking sparse reconstruction: Framework and case study," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 733–747, Oct. 2019.
[24] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Syst.*, vol. 9, no. 2, pp. 1–15, 1994.
[25] K. Deb and S. Agrawal, "A niched-penalty approach for constraint handling in genetic algorithms," in *Proc. Int. Conf. Artif. Neural Netw. Genetic Algorithms*, 1999, pp. 235–243.
[26] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA, USA: Addison-Wesley, 1989.
[27] D. Wu and W. W. Tan, "Genetic learning and performance evaluation of type-2 fuzzy logic controllers," *Eng. Appl. Artif. Intell.*, vol. 19, no. 8, pp. 829–841, 2006.
[28] D. Wu and W. W. Tan, "A simplified type-2 fuzzy controller for real-time control," *ISA Trans.*, vol. 15, no. 4, pp. 503–516, 2006.
[29] J. M. Mendel, *Uncertain Rule-Based Fuzzy Systems: Introduction and New Directions*, 2nd ed. Berlin, Germany: Springer, 2017.
[30] J. M. Mendel and D. Wu, *Perceptual Computing: Aiding People in Making Subjective Judgments*. Hoboken, NJ, USA: Wiley-IEEE Press, 2010.

Dongrui Wu (Senior Member, IEEE) received the B.E. degree in automatic control from the University of Science and Technology of China, Hefei, China, in 2003, the M.E. degree in electrical engineering from the National University of Singapore, Singapore, in 2005, and the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, CA, USA, in 2009.

He is currently a Professor with the School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan, China, and Deputy Director of the Key Laboratory of Image Processing and Intelligent Control, Ministry of Education. He has authored or coauthored more than 130 publications, including a book entitled *Perceptual Computing* (Wiley-IEEE Press, 2010). His research interests include affective computing, brain computer interfaces, computational intelligence, and machine learning.

Dr. Wu was the recipient of the IEEE Computational Intelligence Society Outstanding Ph.D. Dissertation Award in 2012, the IEEE TRANSACTIONS ON FUZZY SYSTEMS Outstanding Paper Award in 2014, NAFIPS Early Career Award in 2014, IEEE Systems, Man and Cybernetics (SMC) Society Early Career Award in 2017, and IEEE SMC Society Best Associate Editor Award in 2018. He was also a finalist for three other best paper awards. He was/is an Associate Editor for the IEEE TRANSACTIONS ON FUZZY SYSTEMS (2011–2018), IEEE TRANSACTIONS ON HUMAN-MACHINE SYSTEMS (2014–present), IEEE COMPUTATIONAL INTELLIGENCE MAGAZINE (2017–present), and IEEE TRANSACTIONS ON NEURAL SYSTEMS AND REHABILITATION ENGINEERING (2019–present).



Xianfeng Tan (Student Member, IEEE) received the B.E. degree in automation from the Wuhan University of Technology, Wuhan, China, in 2017, and the M.E. degree in automation from the Huazhong University of Science and Technology, Wuhan, China, in 2019.

He is currently with Horizon Robotics. His research interests include evolutionary computation and machine learning.