applied sciences

MDPI

Article

A Knowledge Sharing and Individually Guided Evolutionary Algorithm for Multi-Task Optimization Problems

Xiaoling Wang [1], Qi Kang [1,*], Mengchu Zhou [2,*], Zheng Fan [1] and Aiiad Albeshri [3]

[1] Department of Control Science and Engineering, Tongji University, Shanghai 201804, China
[2] ECE Department, New Jersey Institute of Technology, Newark, NJ 07102, USA
[3] Department of Computer Science, King Abdulaziz University, Jeddah 21481, Saudi Arabia
* Correspondence: qkang@tongji.edu.cn (Q.K.); zhou@njit.edu (M.Z.)

Abstract: Multi-task optimization (MTO) is a novel emerging evolutionary computation paradigm. It focuses on solving multiple optimization tasks concurrently while improving optimization performance by utilizing similarities among tasks and historical optimization knowledge. To ensure its high performance, it is important to choose proper individuals for each task. Most MTO algorithms limit each individual to one task, which weakens the effects of information exchange. To improve the efficiency of knowledge transfer and choose more suitable individuals to learn from other tasks, this work proposes a general MTO framework named individually guided multi-task optimization (IMTO). It divides evolutions into vertical and horizontal ones, and each individual is fully explored to learn experience from the execution of other tasks. By using the concept of skill membership, individuals with higher solving ability are selected. Besides, to further improve the effect of knowledge transfer, only inferior individuals are selected to learn from other tasks at each generation. The significant advantage of IMTO over the multifactorial evolutionary framework and baseline solvers is verified via a series of benchmark studies.

Keywords: evolutionary algorithm; multi-task optimization; knowledge transfer; skill membership

Citation: Wang, X.; Kang, Q.; Zhou, M.; Fan, Z.; Albeshri, A. A Knowledge Sharing and Individually Guided Evolutionary Algorithm for Multi-Task Optimization Problems. *Appl. Sci.* **2023**, *13*, 602. https://doi.org/10.3390/app13010602

Academic Editor: Roberto Saia

Received: 2 December 2022
Revised: 27 December 2022
Accepted: 28 December 2022
Published: 1 January 2023

1. Introduction

Evolutionary algorithms (EAs) are population-based stochastic optimization methods that include mechanisms of natural selection and population genetics in the field of Artificial Intelligence [1–3]. They are based on a collective learning process and can start with an arbitrarily initialized population [4,5]. Individuals evolve towards better and better solution regions by means of repetitive reproduction and mutation [6–9]. Due to their flexibility, ease of interfacing, extensibility, and high search efficiency, population-based optimization has been successfully applied to path planning, task assignment, energy-saving, and network configuration [10–16].

Even though EAs are proven to be useful in many applications, they have some disadvantages. For example, it is known that there are many similar optimization problems in the real world. Traditional EAs just focus on solving a single optimization problem and regard each optimization as an independent process while ignoring similarities among different tasks. When handling a new optimization problem, EAs have to start from scratch by assuming that no historical knowledge of solving related problems can be used [17]. Inspired by the idea of transfer learning and multi-task learning that a system can learn knowledge or skills from performing previous tasks [18–20], this work aims to solve multiple optimization problems and explore their implicit facilitations.

Traditionally, optimization problems can be divided into single-objective optimization (SOO) and multi-objective one (MOO). In order to improve the search efficiency among related tasks, a new paradigm named multi-task optimization is proposed. Instead of solving multiple problems sequentially, MTO can utilize similarities of different tasks to

improve search efficiencies [21–23]. Inspired by multifactorial inheritance, Gupta et al. proposed a novel genetic EA named multifactorial evolutionary algorithm (MFEA) to solve MTO problems [21]. In MFEA, complex developmental traits of offspring are influenced by the combination of genetic and cultural factors [24–26]. During the evolution, MFEA can utilize information transfer of similar problems to improve optimization performance, which differs from traditional single-task methods [27–30].

Many MFEA variants have been proposed [31,32]. For example, in order to explore the generality of multifactorial optimization (MFO) when using different population-based search mechanisms, Feng et al. [33] attempted to conduct MFO with particle swarm optimization (PSO) and differential evolution (DE). The derived two multi-tasking paradigms are named as multifactorial particle swarm optimization (MFPSO) and multifactorial differential evolution (MFDE). Chen et al. [17] proposed a general evolutionary framework named MaTEA for solving many-task optimization problems. It chose assisted tasks based on the KullbackLeibler divergence (KLD) mechanism that can be used to measure the similarity among different tasks.

With a strong motivation for improving optimization efficiency and choosing suitable individuals to learn from other tasks, we propose an individually guided multi-task optimization framework. It is capable of generalizing almost all existing single-objective solvers, and answers when to start transfer and who should learn. In this paper, we place four popular single-objective solvers, including genetic algorithm (GA), PSO, DE, and artificial bee colony (ABC) into our proposed framework to validate its usefulness in advancing the field of MTO optimization. We intend to make the following novel contributions:

(1) We propose a novel MTO framework including the partial population information sharing and individual learning schemes to achieve higher search efficiency than existing frameworks.
(2) In order to represent the interests of each individual for solving different tasks, we introduce a new concept of skill membership into the MTO framework.
(3) We divide an MTO search process into vertical and horizontal evolutions, and the latter includes crossovers of individuals belonging to different tasks. Knowledge transfer is guided according to the task performance to suppress the negative transfer of each optimization task.

The rest of this paper is organized as follows: Section 2 introduces the background of MTO. IMTO is described in Section 3. Experimental studies are presented in Section 4. Section 5 concludes this work.

2. Related Work

MTO aims to conduct multiple tasks simultaneously through the use of knowledge transfer among different tasks. Supposing that there are K tasks to be completed and all of them are minimization problems. The jth task can be denoted as T_j with objective function $f_j : X_j \longrightarrow \mathbb{R}$, in space X_j. MTO aims to find a set of global optimal solutions $\{x_1^*, x_2^*, \ldots, x_k^*\}$ which can satisfy

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

MTO can solve multiple optimization tasks in a unified space and utilize similarities among different tasks to accelerate an optimization process, while single-task optimization just focuses on solving one task in one search space and finding the global optimal solution through vertical evolution. MFEA can transfer knowledge among different tasks by the interactions of genetic and cultural factors. During its operation, offspring not only inherit genetic factors, but also are influenced by habits and preferences of parents belonging to different tasks. To ensure different tasks can share useful optimization knowledge, it creates a unified search space for MTO problems.

We first introduce some definitions related to individuals in MFEA [21].

***Factorial Cost*:** The factorial cost of candidate p_i for a given task T_j can be given as $\Psi_j^i = \Lambda \cdot \delta_j^i + f_j^i$, where Λ is a penalizing multiplier, f_j^i is an objective function value, and δ_j^i represents the total constraint violation.

***Factorial Rank*:** The factorial rank r_j^i of individual p_i on task T_j is its index in the list of population members sorted in ascending order with respect to Ψ_j.

Scalar Fitness: The scalar fitness is calculated via the list of factorial ranks $\{r_1^i, r_2^i, \ldots, r_K^i\}$, and $\varphi_i = 1/\min_{j\{1,2,\ldots,K\}}\{r_j^i\}$.

***Skill Factor*:** The skill factor τ_i is a task whose individual p_i performs the best, i.e., $\tau_i = \text{argmin}_j\{r_j^i\}$, where $j \ \{1, 2, \ldots, K\}$.

Since the appearance of the MFO [21], many improvements have been made. Considering that knowledge transfer of different optimization tasks is sensitive to negative inter-task interactions, Bali et al. [34] proposed a novel evolutionary computation framework named MFEA-II. It can achieve online learning and explore similarities among distinct tasks. Based on MFEA-II, they further proposed a cognizant evolutionary multitasking approach called MO-MFEA-II to solve MOO problems [35]. Zheng et al. [23] proposed a self-regulated evolutionary multi-task optimization (SREMTO) that used an ability vector to reflect an individual's solving ability for solving each task. It can automatically adapt the intensity of cross-task knowledge transfer to different and varying degrees of relatedness among different tasks. Liu et al. [36] proposed a surrogate-assisted multi-tasking memetic algorithm (SaM-MA) that used a surrogate model to assist an evolution procedure. Bali et al. proposed an enhanced MTO framework named LDA-MFEA that derived linear transformations among solution spaces of component tasks [37].

Existing MTO algorithms only focus on solving a small number of tasks at the same time. If many tasks need to be solved, they would become ineffective in such complex multi-task environment. To handle many-tasking problems, Tang et al. proposed a group-based MFEA to group similar tasks, such that the genetic information interaction could occur with the same group. To strengthen its effectiveness and efficiency, they further developed a new selection criterion and mating selection mechanism [38]. Zhang et al. proposed a framework named multisource selective transfer optimization that can choose source tasks well. Besides, the optimization instance representation method named centroid distribution is designed to measure the task relatedness of different optimization instances [39].

Due to its superior performance, MFEA has been used in many applications. For example, in order to evolve several Deep Q-Learning (DQL) models and converge to optimal policies, Martinez et al. [40] proposed a new MFO framework that blended meta-heuristic optimization, transfer learning, and DQL to automate the process of knowledge transfer and policy learning of distributed Reinforcement Learning (RL) agents. Feng et al. [41] explored the application of MTO to solve combinatorial optimization problems. They verified its efficiency by using vehicle routing as an illustrative combinatorial optimization problem. To overcome the difficulty that conventional evolutionary algorithms are not suitable for solving expensive optimization problems, Ding et al. [28] proposed a new multitasking evolutionary optimization framework named generalized MFEA (G-MFEA) to solve expensive problems. In G-MFEA, cheap optimization problems can transfer knowledge to expensive ones, thus leading to faster convergences of the latter.

Except for multifactorial-based MTO algorithms, multipopulation approaches also have been proposed to improve the efficiency of knowledge transfer. For example, Li et al. [42] proposed a multi-population framework to solve MTO problems, and each population has corresponding mating probability to exchange information. Cheng et al. [43] proposed a MultiTasking Coevolutionary Particle Swarm Optimization (MT-CPSO) algorithm. In it, the information from the assistant task can be transferred if the personal best solution of the target sub-population cannot improve the search performance.

3. Proposed Method

3.1. Motivations

In MTO, the optimization experience can be shared and each task can utilize the superior experience of other tasks. So, they are expected to evolve faster and achieve higher accuracy than traditional single-task optimization methods. As a typical MTO algorithm, MFEA demonstrates its performance for both SOO and MOO problems, as well as real-world optimization problems. MFEA uses the concept of a skill factor to divide the initial population into different task groups. However, one individual can be suitable for different tasks. This is common when these tasks have high similarities. The skill factor limits individuals to one task group and weakens the effects of information exchange among multiple tasks. As a result, the task pairs with high similarities can suffer from barely useful information exchange among each other. In order to better represent the solving ability of individuals on component tasks and choose more suitable individuals, we propose the concept of skill membership to show the solving ability of individuals on component tasks.

3.2. Proposed Framework

Our newly proposed IMTO is an individually guided multi-task optimization based on knowledge sharing as shown in Figure 1. Figure 2 shows the diagram of IMTO when solving two optimization problems. It includes the following novelties in comparison with other MTO algorithms:

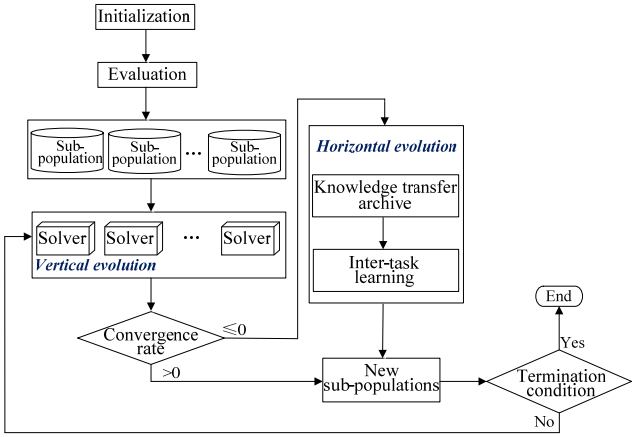

Figure 1. The flowchart of individually guided multi-task optimization (IMTO).

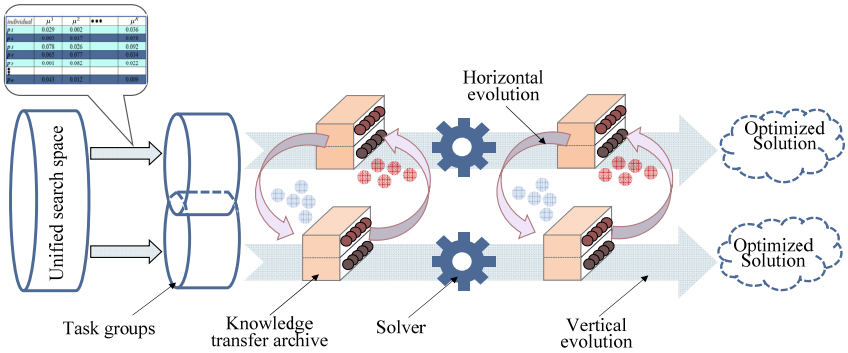

Figure 2. Individually guided multi-task optimization (IMTO) when solving two optimization problems.

(1) We divide the optimization process of MTO into vertical and horizontal evolutions. Each task has its sub-population to execute vertical evolution, and each sub-population can be called a task groups. Traditional single-task optimization methods just contain vertical evolutions that find global optimal solutions by a series of operations, e.g., selection, crossover, and mutation. The distinction between the proposed MTO framework and traditional single-task optimizers lies in horizontal evolution among

different task groups. Optimization processes of multiple tasks can be influenced by each other via the information interaction.

(2) MTO algorithms are able to perform K optimization problems simultaneously. Suppose that the dimensionality of the jth task is D_j. We define the unified search space with dimensionality $D = \max_j\{D_j\}$ and each individual is encoded with random variables lying within the fixed range [0, 1].

(3) IMTO divides the initial population into K task groups, and each task group can evolve independently. In order to represent the ability to perform each component task, we introduce the concept of skill membership. A candidate may enter multiple task groups as long as it shows high skill membership values on component tasks.

(4) In order to confirm when the optimization information should communicate, we use the convergence rate to guide knowledge transfer. When the convergence rate shows that a task may be trapped into local optima, the knowledge transfer mechanism is triggered.

3.3. Individually Guided Multi-Task Evolutionary Optimization

Putting GA, PSO, DE, and ABC into IMTO leads to the corresponding algorithms named IMGA, IMPSO, IMDE, and IMABC. To better show the designed framework, we give the IMGA implementation as an example. Its pseudocode is described in Algorithm 1, where its module performing horizontal transfer is realized in Algorithm 2.

It is worth noting that individuals in MFO evolve together in each iteration, and the random mating probability (rmp) value decides the evolution directions of each individual. Different from MFO, vertical and horizontal evolutions are independent in IMTO and each task group of IMTO has its vertical evolution. The additional horizontal evolution is triggered when the convergence rate is less than zero, which is used to improve optimization performance.

We use skill membership values to represent individuals' solving ability on different tasks, which can be used for guiding the division of different task groups before the evolution.

Definition 1 (Skill membership)**.** *For minimized optimization problems, the skill membership value of individual p_i is defined as:*

$$\mu_i^j = \min\left(\boldsymbol{f^j}\right) / f_i^j \tag{2}$$

where f_i^j is an objective function value of individual p_i on task T_j, and $\min\left(\boldsymbol{f^j}\right)$ is the obtained minimum objective function value. If $\min\left(\boldsymbol{f^j}\right)$ or f_i^j is equal to zero, it will be replaced by random values ξ_1 and ξ_2 that are close to zero. The skill membership value $\boldsymbol{\mu_i} = \mu_{ij=1}^{jk}$ reflects the solving ability of individual p_i on component task T_j. Next, we show how to divide the initial population into different task groups according to skill membership values. This paper records the rank of skill membership values in order from the highest to lowest, and the higher skill membership values mean higher solving ability.

At the beginning of IMGA, N_0 individuals are randomly initialized within the unified search space. Given K optimization tasks, IMGA uses a partition strategy to divide the initial population P_0 into K task groups that focus on solving component tasks. Since simply choosing better-performing individuals can be easily trapped into local optima, IMGA randomly chooses some individuals according to the randomly chosen ratio γ. Specifically, N_0 individuals of the initial population P_0 are ranked according to their skill membership values. P_0 is divided into P_{01}^j and P_{02}^j for task T_j. P_{01}^j contains N(1-γ) individuals with the highest skill membership values, and P_{02}^j is composed of the remaining individuals of P_0. IMGA chooses all candidates of P_{01}^j and randomly chooses $N\gamma$ candidates from P_{02}^j to form task group G_j. The setting of parameter γ with a default value 0.2 is to be discussed later.

Algorithm 1. IMGA

Input: N_0, initial population size; N, task group size; K, number of tasks; Ω: communication rate.

Output: $\{x_1^*, x_2^*, \ldots, x_k^*\}$ (the best solution found for each component task).

```
P_0 ← Random Initialization (N_0).
Evaluate each individual of the initial population on K component tasks.
Calculate the skill membership value μ_i^j of individual p_i on the jth task.
for task j = 1 to K do
    Sort (μ_i^j) for task T_j.
    Divide P_0 into P_01^j and P_02^j.
    Choose individuals from P_01^j and P_02^j to form task group G_j.
end for
while (stopping criterion is not met) do
    for task j = 1 to K do
        Mutation and crossover.
        Fitness evaluations.
        Finish vertical reproduction operations and select N candidates to update
        task group G_j.
    end for
    The knowledge transfer archive ← N_Λ better-performing individuals and N_Λ
    worse-performing individuals.
    Calculate the convergence rate.
    if (convergence rate of task T_j descends) then
        Start horizontal transfer (⊳ Algorithm 2).
    end if
    Update the best solution found so far.
end while
Output the best solution for each task, i.e., {x_1^*, x_2^*, ..., x_K^*}.
```

After a partition procedure, each task group contains N individuals, and $KN = N_0$. Individuals belonging to the same task group can evolve independently through the vertical operators. Being similar to traditional single-task optimization, the vertical offspring generation in IMGA includes crossover and mutation operators. During the vertical evolution, each task can generate vertical offspring solutions to update the task group.

To confirm when the knowledge should transfer, we use a convergence rate to guide the horizontal knowledge transfer. The convergence rate of minimization problems in generation t is defined as $\rho_t = f_{t-1}^* - f_t^*$, where f_{t-1}^* and f_t^* represent optimal fitness values in generation of $t-1$ and t. When the convergence rate of task T_j is less than zero, IMGA starts the knowledge transfer to avoid being trapped into local optima.

In this paper, the task that needs to learn knowledge from another task is called a target task, and the task used for providing knowledge is called an assistant task. To create the special environment to transfer optimization knowledge and protect specific properties of better-performing individuals, an archive-memory knowledge transfer scheme is introduced for partial population knowledge sharing. In the partial population knowledge sharing scheme, only partial information in the assistant and target tasks will be shared. The knowledge transfer archive Λ_j^t of task T_j in generation t includes superior individuals $\hat{\Lambda}_j^t$ and inferior individuals $\check{\Lambda}_j^t$. It aims to maintain excellent evolutionary information and improve behaviors of inferior individuals. This work uses communication rate (Ω)

to choose individuals of each task to form its knowledge transfer archive. By setting knowledge transfer archive size $N_\Lambda = \Omega N$, N_Λ better-performing individuals and N_Λ worse-performing individuals are added to knowledge transfer archives in each generation, and only the latter need to learn.

The communication rate Ω with a smaller value means that each task prefers maintaining its vertical evolution, and arbitrary communications among different tasks are permitted when Ω is close to 1. Cross-task communications can explore the entire search space and avoid being trapped into local optima. The setting of Ω with its default value of 0.2 is to be discussed later. IMGA applies crossover and mutation operations among different tasks to finish horizontal evolutions, as described in Algorithm 2. When there are more than two tasks, IMGA randomly chooses assisted tasks from component tasks to provide knowledge. This work adopts simulated binary crossover and polynomial mutation to realize information exchange [44]. The crossover probability of p_C is set to be 0.9. Distribution indexes for crossover and mutation operator are set as $\eta_C = 10$ and $\eta_M = 10$.

Algorithm 2. Horizontal Transfer

Input: T_j, the task that needs to learn; T_m, the task used for providing knowledge; N_Λ, knowledge transfer archive size; Λ_j^t, knowledge transfer archive of task T_j; Λ_m^t, knowledge transfer archive of task T_m.

Output: G_j, the new task group for the jth task after horizontal transfer.

```
Confirm task T_j that needs to learn from other tasks.
for inferior individuals, i = 1 to N_Λ do
    Select the ith individual (p1) of Λ̂_m^t to communicate with the ith individual of
    Λ̌_j^t (p2) for task T_j.
    if rand ≤ p_c then
        (c1, c2) = crossover (p1, p2).
        Calculate fitness values of c1 and c2 on task T_j.
        Choose the better-performing individual on task T_j.
    else
        c1 = mutation.
    end if
end for
Generate horizontal offspring solutions Λ̌_{j_H}^t.
Evaluate (Λ̌_{j_H}^t ∪ Λ̌_j^t).
Λ̌'_j^t ← Select N_Λ better-performing individuals for task T_j.
Λ̌_j^t ← Λ̌'_j^t, update G_j.
```

After the horizontal evolutin, we obtain horizontal offspring solutions $\check{\Lambda}_{j\_H}^t$ for the target task T_j. The generated horizontal offspring solutions are evaluated for the target task, which can provide additional useful evolution information. Additionally, then IMGA will choose N_Λ better-performing individuals $\check{\Lambda}'^t_j$ to replace those worse-performing individuals. The horizontal evolution continues until a termination condition is met.

3.4. Computational Complexity

IMTO includes the following two operations: (1) Initialize the initial population and divide them into different task groups; and (2) finish vertical evolutions and the knowledge transfer among different tasks.

This work sets the size of initial population to KN, and each task has a sub-population of size N. IMGA begins with skill membership values' assignment for the initial population.

It takes $O(K^2N)$ to calculate objective function values and assign task groups. In every generation, vertical evolution takes $O(KDN)$ to apply crossover and mutation operations, where D is the dimension of the decision space. For the worst case, all individuals need to perform horizontal evolution in every generation, and $O(KDN)$ is required to keep information exchange. The worst-case computational complexity in every generation is $O(KDN)$. So, IMGA has the complexity of $O\left(\hat{g}KdN + K^2N\right)$, where $\hat{g}$ is the maximal number of generations. The similar conclusions can be made with IMPSO, IMDE, and IMABC.

4. Experiments

To verify the generality and effectiveness of the MTO framework proposed in this paper, we select well-represented single-objective algorithms: GA [45], PSO [46,47], DE [48], and ABC [49] to combine with IMTO. The comparative experiments are performed on a series of MTO problems to demonstrate:

(1) IMTO can significantly outperform corresponding baseline solvers.
(2) In terms of the optimization knowledge transfer, IMTO outperforms the multifactorial optimization framework.
(3) IMTO can adapt to different task similarities and promise high transfer effectiveness.

4.1. Experimental Setup

In order to show the knowledge transfer effects of different kinds of MTO algorithms, we use two test suites of test problems in our experiments. Test suite 1 contains 9 composite two-task problems used in CEC 2017 EMTO Competition [50]. These nine problems belong to nine categories with different degrees of overlap and different degrees of inter-task similarity. It uses CI, PI, and NI to represent complete, partial, and no intersection, and HS, MS, and LS to mean high, medium, and low similarity, respectively. Details of test problems are given in Table 1. T_1 and T_2 denote two component tasks of each multi-task optimization problem. The second test suite of the CEC2021 EMTO Competition consists of much more complex objective functions.

Table 1. Summary of properties of problem pairs in test suite 1.

Task Set	Category	Task Component	Dimensionality	Search Space	Inter-Task Similarity
P_1	CI+HS	Griewank (T_1) Rastrigin (T_2)	50 50	$[-100, 100]$ $[-50, 50]$	1.00
P_2	CI+MS	Ackley (T_1) Rastrigin (T_2)	50 50	$[-50, 50]$ $[-50, 50]$	0.22
P_3	CI+LS	Ackley (T_1) Schwefel (T_2)	50 50	$[-50, 50]$ $[-500, 500]$	0.00
P_4	PI+HS	Rastrigin (T_1) Sphere (T_2)	50 50	$[-50, 50]$ $[-100, 100]$	0.86
P_5	PI+MS	Ackey (T_1) Rosenbrock (T_2)	50 50	$[-50, 50]$ $[-50, 50]$	0.21
P_6	PI+LS	Ackey (T_1) Weierstrass (T_2)	50 25	$[-50, 50]$ $[-0.5, 0.5]$	0.07
P_7	NI+HS	Rosenbrock (T_1) Rastrigin (T_2)	50 50	$[-50, 50]$ $[-50, 50]$	0.94
P_8	NI+MS	Griewank (T_1) Weierstrass (T_2)	50 50	$[-100, 100]$ $[-0.5, 0.5]$	0.36
P_9	NI+LS	Rastrigin (T_1) Schwefel (T_2)	50 50	$[-50, 50]$ $[-500, 500]$	0.00

The initial population size is set to 100, and each task is assigned with 50 individuals. In MFEA, the distribution index for crossover η_C is set as 10. Acceleration coefficients of MFPSO are set to 0.2. The weight of IMPSO and MFPSO decreases linearly from 0.9 to 0.4. In IMABC, the number of onlooker bees is set as 50. The value of random mating probability controls knowledge transfer in a multi-task paradigm, and in MFEA, MFPSO, and MFDE, we set rmp to 0.3 in all experiments by following [21,33]. We set individual learning *p_il* of MFEA, MFDE, and MFPSO to 0. The sensitivities of Ω and γ are to be discussed later.

Other parameters used in this work are summarized as: (1) Maximum number of generations: $\hat{g}$ = 1000; (2) Independent number of runs: $\hat{r}$ = 20.

We choose parallel GAs, PSOs, DEs, and ABCs as four baseline solvers, named as B-GA, B-PSO, B-DE, and B-ABC, respectively. All the baseline solvers share the same parameters and search operators of IMGA, IMPSO, IMDE, and IMABC to guarantee fair comparisons. Experiments are conducted by using the computer with a 1.00 GHz Intel Corei5 processor and 16 GB RAM under window10.

4.2. Parametric Analysis

(1) Sensitivity of Ω: Communication rate Ω is used to control knowledge transfer among different tasks. We examine performance sensitivity with respect to this parameter for IMDE. We set it to 0.2, 0.4, 0.6, 0.8, and 1. Table 2 shows the best achieved fitness values in 20 runs versus Ω in IMDE on test suite 1, and the best one is shown in bold. It is clear that IMDE performs well with Ω from 0.2 to 1 with a small difference. Nevertheless, the larger Ω can encourage more individuals to learn from other tasks, thereby consuming more computing resources. To reduce running time on various problems, we employ a relatively small Ω in our algorithms.

(2) Sensitivity of γ: Randomly chosen ratio γ is another control parameter utilized in IMTO. To discuss performance sensitivity to γ in IMTO, we test its different settings on test suite 1. Table 3 shows the best achieved fitness values in 20 runs versus γ in IMDE on test suite 1, and the best one is shown in bold. IMDE is easily trapped into local optima when only choosing better-performing individuals. However, adding some randomly selected individuals can have better convergence performance. It is clear that IMDE performs best when γ is set to 0.2. We thus set γ to 0.2 in our experiments.

4.3. Comparison with MFO

To verify the performance of IMTO, optimization results of IMTO and MFO on test suites 1 and 2 are summarized. MTO algorithms are designed based on different single-task algorithms. This work compares IMGA with MFEA, IMDE with MFDE, and IMPSO with MFPSO. The computational time results of different algorithms are shown in Table 4 and Figure 3.

It can be found that although IMGA, IMPSO, and IMDE need to finish information exchange in each generation, their running time is much shorter than MFO. For example, MFPSO needs 19.03s to optimize P_2, while IMPSO just needs 3.43s in Table 4. This can also apply to IMGA and IMDE. IMTO prefers to maintain the vertical evolution of each task, and each generation just needs to evaluate objective functions and communicate information. However, MFEA, MFPSO, and MFDE need to divide task groups at each generation, and the calculations of factorial cost, factorial rank, scalar fitness, and skill factor consume many computation resources. Hence, the knowledge transfer scheme of IMTO is much more efficient than the method of multifactorial influence used in MFEA, MFPSO, and MFDE.

Table 2. The mean and standard deviation (in brackets) of the best achieved fitness values of IMDE with different communication rates.

Task Set		IMDE (Ω = 0.2)	IMDE (Ω = 0.4)	IMDE (Ω = 0.6)	IMDE (Ω = 0.8)	IMDE (Ω = 1)
P_1	T_1	$\mathbf{1.20 \times 10^{-4}}$ (3.63×10^{-4})	2.02×10^{-3} (3.43×10^{-3})	1.92×10^{-3} (3.95×10^{-3})	6.99×10^{-4} (2.69×10^{-3})	4.44×10^{-4} (1.62×10^{-3})
	T_2	5.03×10^{1} (1.30×10^{1})	4.32×10^{1} (7.83×10^{0})	3.22×10^{1} (1.10×10^{1})	3.48×10^{1} (9.85×10^{0})	$\mathbf{2.84 \times 10^{1}}$ (6.67×10^{0})
P_2	T_1	4.74×10^{-3} (9.54×10^{-3})	8.84×10^{-2} (2.64×10^{-1})	1.11×10^{-1} (3.27×10^{-1})	$\mathbf{9.05 \times 10^{-4}}$ (3.38×10^{-3})	1.02×10^{-2} (3.95×10^{-2})
	T_2	4.43×10^{1} (1.15×10^{1})	3.98×10^{1} (1.13×10^{1})	3.78×10^{1} (1.46×10^{1})	3.04×10^{1} (7.59×10^{0})	$\mathbf{2.88 \times 10^{1}}$ (9.50×10^{0})
P_4	T_1	8.10×10^{1} (9.75×10^{0})	7.29×10^{1} (2.27×10^{1})	7.46×10^{1} (1.22×10^{1})	7.40×10^{1} (1.45×10^{1})	$\mathbf{6.77 \times 10^{1}}$ (1.66×10^{1})
	T_2	2.44×10^{-6} (7.82×10^{-6})	8.06×10^{-4} (2.91×10^{-3})	4.97×10^{-5} (2.15×10^{-4})	$\mathbf{1.46 \times 10^{-6}}$ (4.66×10^{-6})	1.23×10^{-4} (5.35×10^{-4})
P_5	T_1	$\mathbf{6.74 \times 10^{-5}}$ (9.91×10^{-5})	7.29×10^{-5} (5.01×10^{-5})	3.14×10^{-4} (5.71×10^{-4})	1.32×10^{-4} (2.92×10^{-4})	9.48×10^{-5} (1.79×10^{-4})
	T_2	$\mathbf{6.48 \times 10^{1}}$ (2.97×10^{1})	9.82×10^{1} (3.15×10^{1})	7.82×10^{1} (2.97×10^{1})	6.96×10^{1} (2.81×10^{1})	7.33×10^{1} (3.08×10^{1})
P_7	T_1	9.25×10^{1} (2.75×10^{1})	8.83×10^{1} (4.47×10^{1})	7.78×10^{1} (6.08×10^{1})	$\mathbf{6.86 \times 10^{1}}$ (3.42×10^{1})	8.29×10^{1} (4.78×10^{1})
	T_2	5.14×10^{1} (1.22×10^{1})	4.72×10^{1} (1.28×10^{1})	4.10×10^{1} (1.22×10^{1})	$\mathbf{3.53 \times 10^{1}}$ (9.74×10^{0})	3.74×10^{1} (9.87×10^{0})
P_8	T_1	$\mathbf{1.41 \times 10^{-3}}$ (3.33×10^{-3})	3.39×10^{-3} (7.11×10^{-3})	1.49×10^{-3} (3.51×10^{-3})	2.03×10^{-3} (5.95×10^{-3})	1.62×10^{-3} (3.84×10^{-3})
	T_2	5.93×10^{0} (2.04×10^{0})	3.84×10^{0} (1.34×10^{0})	4.41×10^{0} (1.28×10^{0})	4.20×10^{0} (1.30×10^{0})	$\mathbf{3.55 \times 10^{0}}$ (2.07×10^{0})

Table 3. The mean and standard deviation (in brackets) of the best achieved fitness values of IMDE with different randomly chosen ratios.

Task Set		IMDE (γ = 0)	IMDE (γ = 0.2)	IMDE (γ = 0.4)	IMDE (γ = 0.6)	IMDE (γ = 0.8)	IMDE (γ = 1.0)
P_1	T_1	2.12×10^{-3} (4.22×10^{-3})	$\mathbf{1.202 \times 10^{-4}}$ (3.632×10^{-4})	1.46×10^{-3} (4.21×10^{-3})	1.51×10^{-3} (2.95×10^{-3})	1.38×10^{-3} (3.36×10^{-3})	1.65×10^{-3} (3.22×10^{-3})
	T_2	5.55×10^{1} (1.67×10^{1})	5.032×10^{1} (1.302×10^{1})	4.95×10^{1} (1.23×10^{1})	$\mathbf{4.79 \times 10^{1}}$ (1.39×10^{1})	5.08×10^{1} (1.34×10^{1})	5.72×10^{1} (1.41×10^{1})
P_2	T_1	4.41×10^{-2} (1.92×10^{-1})	4.742×10^{-3} (9.542×10^{-3})	$\mathbf{1.33 \times 10^{-4}}$ (1.86×10^{-4})	1.79×10^{-3} (3.96×10^{-3})	1.48×10^{-1} (3.50×10^{-1})	1.44×10^{-1} (4.41×10^{-1})
	T_2	4.72×10^{1} (1.47×10^{1})	$\mathbf{4.432 \times 10^{1}}$ (1.152×10^{1})	4.80×10^{1} (1.48×10^{1})	5.24×10^{1} (1.82×10^{1})	5.11×10^{1} (1.52×10^{1})	4.95×10^{1} (1.20×10^{1})
P_4	T_1	8.57×10^{1} (2.72×10^{1})	8.102×10^{1} ($9.752 \times 10^{+00}$)	8.06×10^{1} (1.78×10^{1})	8.77×10^{1} (2.75×10^{1})	7.98×10^{1} (2.01×10^{1})	$\mathbf{7.72 \times 10^{1}}$ (2.35×10^{1})
	T_2	1.70×10^{-1} (6.46×10^{-1})	$\mathbf{2.442 \times 10^{-6}}$ (7.822×10^{-6})	1.96×10^{-4} (7.32×10^{-4})	6.10×10^{-5} (2.57×10^{-4})	6.76×10^{-5} (2.94×10^{-4})	3.07×10^{-2} (9.20×10^{-2})
P_5	T_1	3.30×10^{-4} (6.71×10^{-4})	$\mathbf{6.742 \times 10^{-5}}$ (9.912×10^{-5})	3.30×10^{-4} (8.34×10^{-4})	1.89×10^{-4} (2.95×10^{-4})	1.23×10^{-3} (4.55×10^{-3})	5.44×10^{-4} (1.38×10^{-3})
	T_2	8.93×10^{1} (2.67×10^{1})	$\mathbf{6.482 \times 10^{1}}$ (2.972×10^{1})	8.45×10^{1} (2.96×10^{1})	9.97×10^{1} (3.19×10^{1})	9.29×10^{1} (2.75×10^{1})	9.07×10^{1} (3.23×10^{1})
P_7	T_1	9.93×10^{1} (4.71×10^{1})	9.252×10^{1} (2.752×10^{1})	1.01×10^{2} (5.66×10^{1})	$\mathbf{7.44 \times 10^{1}}$ (2.56×10^{1})	8.83×10^{1} (3.74×10^{1})	7.52×10^{1} (4.35×10^{1})
	T_2	5.82×10^{1} (1.06×10^{1})	$\mathbf{5.142 \times 10^{1}}$ (1.222×10^{1})	5.76×10^{1} (9.24×10^{0})	5.53×10^{1} (1.41×10^{1})	5.37×10^{1} (1.04×10^{1})	5.60×10^{1} (1.02×10^{1})
P_8	T_1	1.72×10^{-3} (3.97×10^{-3})	$\mathbf{1.412 \times 10^{-3}}$ (3.332×10^{-3})	2.54×10^{-3} (6.20×10^{-3})	1.63×10^{-3} (2.92×10^{-3})	4.27×10^{-3} (6.69×10^{-3})	3.24×10^{-3} (6.19×10^{-3})
	T_2	6.18×10^{0} (2.55×10^{0})	5.932×10^{0} (2.042×10^{0})	5.47×10^{0} (2.95×10^{0})	6.18×10^{0} (3.71×10^{0})	$\mathbf{5.11 \times 10^{0}}$ (2.06×10^{0})	5.15×10^{0} (2.67×10^{0})

Table 4. Results of computational time(s) and the mean and standard deviation (in brackets) of the best fitness values in test suite 1.

Task Set		IMGA		MFEA		IMPSO		MFPSO		IMDE		MFDE	
		GA-Based				PSO-Based				DE-Based			
		Fitness	Time	Fitness	Time	Fitness	Time	Fitness	Time	Fitness	Time	Fitness	Time
P_1	T_1	$\mathbf{1.11 \times 10^{-1}}$ (5.27×10^{-2})	**4.02**	3.35×10^{-1} + (4.88×10^{-2})	18.57	$\mathbf{4.99 \times 10^{-3}}$ (6.57×10^{-3})	**3.49**	5.20×10^{-1} + (1.42×10^{-1})	21.79	$\mathbf{1.20 \times 10^{-4}}$ (3.63×10^{-4})	**4.87**	8.77×10^{-4} + (2.63×10^{-3})	20.29
	T_2	3.39×10^{2} (4.72×10^{1})		$\mathbf{2.27 \times 10^{2}}$ − (5.33×10^{1})		$\mathbf{3.26 \times 10^{2}}$ (6.26×10^{1})		$3.32 \times 10^{2} \approx$ (2.54×10^{1})		5.03×10^{1} (1.30×10^{1})		$\mathbf{3.69 \times 10^{0}}$ − (1.14×10^{1})	
P_2	T_1	$\mathbf{3.83 \times 10^{0}}$ (8.78×10^{-1})	**4.24**	8.00×10^{0} + (6.38×10^{-1})	19.00	$\mathbf{3.93 \times 10^{-1}}$ (4.54×10^{-1})	**3.43**	5.32×10^{0} + (6.68×10^{-1})	19.03	$\mathbf{4.74 \times 10^{-3}}$ (9.54×10^{-3})	**5.47**	1.08×10^{-1} + (3.28×10^{-1})	18.87
	T_2	$\mathbf{4.16 \times 10^{2}}$ (3.65×10^{1})		4.52×10^{2} + (5.68×10^{1})		$\mathbf{3.73 \times 10^{1}}$ (1.20×10^{1})		3.97×10^{2} + (4.24×10^{1})		4.43×10^{1} (1.15×10^{1})		$\mathbf{7.47 \times 10^{-1}}$ − (2.83×10^{0})	
P_3	T_1	$\mathbf{2.08 \times 10^{1}}$ (4.19×10^{-1})	**4.40**	$2.11 \times 10^{1} \approx$ (7.66×10^{-2})	19.06	$\mathbf{2.10 \times 10^{1}}$ (5.53×10^{-2})	**2.61**	2.13×10^{1} + (5.07×10^{2})	18.89	$\mathbf{2.12 \times 10^{1}}$ (3.86×10^{-2})	**5.79**	$\mathbf{2.12 \times 10^{1}} \approx$ (3.33×10^{-2})	18.98
	T_2	1.30×10^{4} (6.82×10^{2})		$\mathbf{9.46 \times 10^{3}}$ − (6.91×10^{2})		1.64×10^{4} (3.28×10^{2})		$\mathbf{1.54 \times 10^{4}}$ − (8.39×10^{2})		$\mathbf{9.68 \times 10^{3}}$ (2.25×10^{3})		1.15×10^{4} + (1.45×10^{3})	
P_4	T_1	$\mathbf{1.95 \times 10^{2}}$ (4.43×10^{1})	**4.25**	7.78×10^{2} + (1.00×10^{2})	19.17	$\mathbf{3.23 \times 10^{2}}$ (8.88×10^{1})	**2.60**	7.72×10^{2} + (1.09×10^{2})	21.91	$\mathbf{8.10 \times 10^{1}}$ (9.75×10^{0})	**4.79**	$8.13 \times 10^{1} \approx$ (1.71×10^{1})	20.00
	T_2	7.64×10^{3} (6.35×10^{2})		$\mathbf{2.58 \times 10^{2}}$ − (8.90×10^{1})		$\mathbf{3.38 \times 10^{-4}}$ (2.65×10^{-4})		3.53×10^{3} + (8.34×10^{2})		$\mathbf{2.44 \times 10^{-6}}$ (7.82×10^{-6})		1.64×10^{-5} + (1.30×10^{-5})	
P_5	T_1	$\mathbf{3.59 \times 10^{0}}$ (7.93×10^{-1})	**4.09**	7.21×10^{0} + (5.99×10^{-1})	18.80	$\mathbf{2.52 \times 10^{-1}}$ (3.81×10^{-1})	**2.63**	3.69×10^{0} + (6.01×10^{-1})	20.70	$\mathbf{6.74 \times 10^{-5}}$ (9.91×10^{-5})	**5.14**	2.80×10^{-3} + (5.52×10^{-3})	19.47
	T_2	$\mathbf{2.49 \times 10^{4}}$ (2.25×10^{4})		7.37×10^{4} + (4.33×10^{4})		$\mathbf{6.67 \times 10^{1}}$ (3.11×10^{1})		8.39×10^{3} + (3.85×10^{3})		$\mathbf{6.48 \times 10^{1}}$ (2.97×10^{1})		$6.52 \times 10^{1} \approx$ (2.28×10^{1})	
P_6	T_1	$\mathbf{3.74 \times 10^{0}}$ (8.99×10^{-1})	**19.46**	2.10×10^{1} + (7.61×10^{-2})	27.67	$\mathbf{3.10 \times 10^{-1}}$ (5.02×10^{-1})	**18.20**	1.02×10^{1} + (1.34×10^{0})	46.21	$\mathbf{3.06 \times 10^{-1}}$ (5.67×10^{-1})	**30.13**	7.71×10^{-1} + (1.08×10^{0})	33.32
	T_2	$\mathbf{5.52 \times 10^{0}}$ (8.98×10^{-1})		2.17×10^{1} + (2.49×10^{0})		2.19×10^{1} (3.61×10^{0})		$\mathbf{7.87 \times 10^{0}}$ − (1.47×10^{0})		1.88×10^{0} (2.37×10^{0})		$\mathbf{2.61 \times 10^{-1}} \approx$ (6.59×10^{-1})	
P_7	T_1	$\mathbf{2.25 \times 10^{3}}$ (1.74×10^{3})	**4.47**	7.75×10^{4} + (3.70×10^{4})	18.42	$\mathbf{8.56 \times 10^{1}}$ (8.06×10^{1})	**3.57**	1.05×10^{5} + (4.72×10^{4})	21.59	$\mathbf{9.25 \times 10^{1}}$ (2.75×10^{1})	**5.55**	$1.17 \times 10^{2} \approx$ (1.17×10^{2})	18.78
	T_2	5.78×10^{2} (2.49×10^{2})		$\mathbf{4.30 \times 10^{2}}$ − (6.03×10^{1})		$\mathbf{7.09 \times 10^{1}}$ (3.80×10^{1})		3.77×10^{2} + (6.90×10^{1})		5.14×10^{1} (1.22×10^{1})		$\mathbf{2.65 \times 10^{1}}$ − (1.92×10^{1})	
P_8	T_1	$\mathbf{4.21 \times 10^{-2}}$ (1.28×10^{-2})	**33.23**	1.04×10^{0} + (4.22×10^{2})	38.55	$\mathbf{5.33 \times 10^{-3}}$ (6.56×10^{-3})	**35.86**	1.06×10^{0} + (3.68×10^{-2})	70.41	$\mathbf{1.41 \times 10^{-3}}$ (3.33×10^{-3})	**51.45**	1.67×10^{-3} + (3.99×10^{-3})	63.82
	T_2	3.07×10^{1} (5.12×10^{-1})		$\mathbf{2.86 \times 10^{1}}$ − (2.66×10^{0})		5.26×10^{1} (3.80×10^{0})		$\mathbf{2.91 \times 10^{1}}$ − (2.01×10^{0})		5.93×10^{0} (2.04×10^{0})		$\mathbf{2.81 \times 10^{0}}$ − (1.20×10^{0})	
P_9	T_1	$\mathbf{3.36 \times 10^{2}}$ (8.16×10^{2})	**5.69**	7.33×10^{2} + (7.24×10^{1})	17.90	$\mathbf{3.55 \times 10^{2}}$ (6.91×10^{1})	**3.71**	2.43×10^{3} + (6.27×10^{2})	22.33	3.22×10^{2} (1.06×10^{2})	**6.23**	$\mathbf{1.01 \times 10^{2}}$ − (2.66×10^{1})	24.75
	T_2	1.75×10^{4} (5.37×10^{2})		$\mathbf{9.81 \times 10^{3}}$ − (6.92×10^{2})		1.60×10^{4} (4.90×10^{2})		$\mathbf{1.57 \times 10^{4}} \approx$ (5.88×10^{2})		5.86×10^{3} (5.84×10^{2})		$\mathbf{4.24 \times 10^{3}}$ − (8.61×10^{2})	
+/−/≈		**11/6/1**				**13/3/2**				**7/6/5**			

"+" means that IMTO significantly outperforms other algorithms. "−" means that IMTO is significantly outperformed by other algorithms. "≈" means that no significant differences between IMTO and compared algorithms.

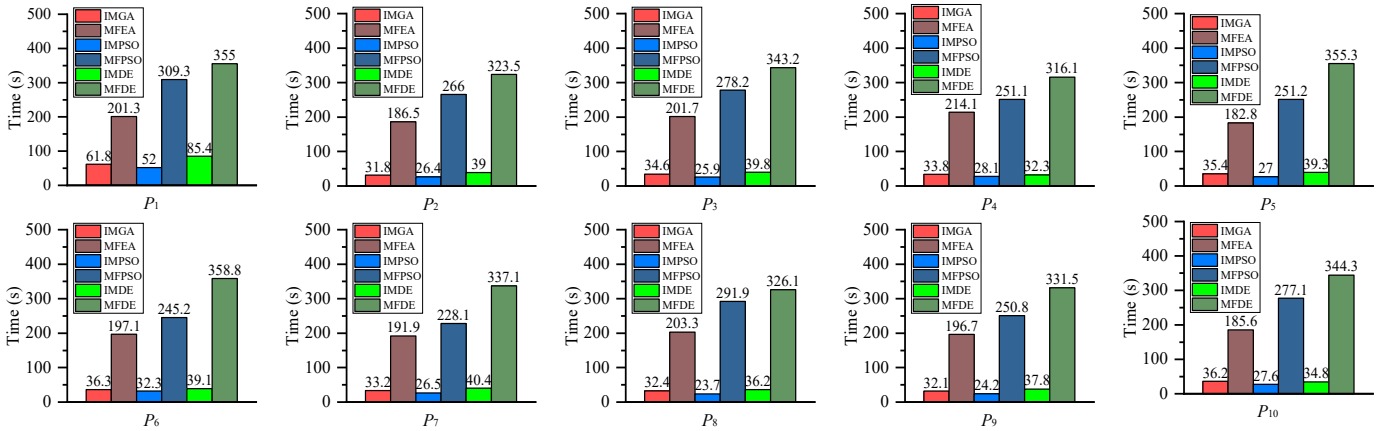

Figure 3. Comparison of six algorithms in terms of the average computational time (s) via 20 runs in test suite 2.

To better show effects of knowledge transfer on solution accuracy, the optimization results of IMTO and MFO on test suites 1 and 2 are summarized in Tables 4 and 5. The mean and standard deviation of the best achieved fitness values obtained via 20 independent runs are presented. The best performance of fitness values is shown in bold. It is obvious that the proposed IMTO exhibits a better overall performance than its peer, i.e., MFO. To judge whether there is a significant difference among IMTO and its peers, Wilcoxon's rank-sum test at the significant level of 0.05 is invoked. Symbols "$-$" and "$+$" denote that the compared MFO algorithm performs significantly better and significantly worse than IMTO, while "$\approx$" indicates there is no statistically significant difference between them.

The performance superiority of IMTO over MFO can be evidenced by the fact that IMGA significantly outperforms MFEA in 11 out of 18 tasks, IMPSO outperforms MFPSO in 13 out of 18 tasks, and IMDE outperforms MFDE in 7 out of 18 tasks on test suite 1. Table 5 shows that the optimization accuracy of IMTO and MFO are relatively similar on complex problems, but the computation time results in Figure 3 show that IMTO can greatly shorten the running time. For example, MFPSO needs 228.1 s to optimize P_7, while IMPSO just needs 26.5 s. IMTO allows individuals to belong to different tasks as long as it shows higher skill membership values, which promotes the individuals' potential in task-solving. Besides, the superior individuals of assisted tasks can be used for providing knowledge to guide the evolution of target tasks, which can improve the optimization results. It is worth noting that all the results are shown in scientific notation, which means that presented results are limited by the precision. Even though some comparative results shown in Table 4 are nearly the same because of the precision limitation, their actual average best fitness values may be very different. When executing the Wilcoxon's rank-sum test, there is a significant difference between these comparative results.

The convergence curves of IMTO and MFO in 1000 iterations are illustrated in Figure 4, and they can tell the search efficiencies of different algorithms. Each sub-figure shows the optimization processes of T_1 and T_2. Y-axis represents the fitness curves and X-axis denotes the number of generations. This work just shows the convergence trace of P_5 due to page limitation. It can be easily found that IMTO possesses faster convergence than MFO. This is because shared archives can generate similar genetic materials, and IMGA, IMPSO, and IMDE can update the evolution direction of worse-performing individuals. Obviously, the more similar tasks, the greater optimization facilitation. MFEA, MFPSO, and MFDE control knowledge transfer through the rmp value and neglect task similarities, thus easily causing negative optimization transfer. So, the convergences of MFEA and MFPSO are slower than IMTO's.

Table 5. The mean and standard deviation (in brackets) of the best achieved fitness values in test Suite 2.

Task Set		IMGA	MFEA	IMPSO	MFPSO	IMDE	MFDE
		GA-Based		PSO-Based		DE-Based	
P_1	T_1	$\mathbf{6.226 \times 10^2}$ (1.999×10^{-1})	6.241×10^2 (2.565×10^{-1}) +	$\mathbf{6.232 \times 10^2}$ (2.039×10^{-1})	6.235×10^2 (2.507×10^{-1}) +	$\mathbf{6.217 \times 10^2}$ (1.503×10^{-1})	$\mathbf{6.217 \times 10^2}$ (1.236×10^{-1}) ≈
	T_2	6.279×10^2 (1.476×10^{-1})	$\mathbf{6.272 \times 10^2}$ (1.722×10^{-1}) −	$\mathbf{6.262 \times 10^2}$ (2.346×10^{-1})	6.270×10^2 (2.411×10^{-1}) +	$\mathbf{6.246 \times 10^2}$ (1.351×10^{-1})	$\mathbf{6.246 \times 10^2}$ (1.217×10^{-1}) ≈
P_2	T_1	$\mathbf{7.112 \times 10^2}$ (2.545×10^{-3})	7.113×10^2 (1.519×10^{-2}) +	$\mathbf{7.112 \times 10^2}$ (2.068×10^{-4})	7.113×10^2 (6.677×10^{-2}) +	$\mathbf{7.112 \times 10^2}$ (7.461×10^{-10})	$\mathbf{7.112 \times 10^2}$ (6.483×10^{-8}) +
	T_2	7.194×10^2 (6.928×10^{-2})	$\mathbf{7.177 \times 10^2}$ (1.642×10^{-2}) −	$\mathbf{7.176 \times 10^2}$ (3.411×10^{-13})	7.178×10^2 (1.976×10^{-1}) +	$\mathbf{7.176 \times 10^2}$ (1.450×10^{-9})	$\mathbf{7.176 \times 10^2}$ (1.465×10^{-7}) +
P_3	T_1	$\mathbf{2.887 \times 10^6}$ (2.510×10^4)	2.974×10^6 (2.537×10^4) +	$\mathbf{2.834 \times 10^6}$ (0.000×10^0)	2.878×10^6 (6.454×10^4) +	$\mathbf{2.834 \times 10^6}$ (2.792×10^{-3})	$\mathbf{2.834 \times 10^6}$ (8.790×10^{-2}) +
	T_2	5.787×10^7 (1.480×10^6)	$\mathbf{3.597 \times 10^7}$ (2.123×10^5) −	$\mathbf{3.474 \times 10^7}$ (7.451×10^{-9})	3.637×10^7 (1.434×10^6) +	$\mathbf{3.474 \times 10^7}$ (2.889×10^{-2})	$\mathbf{3.474 \times 10^7}$ (1.096×10^0) +
P_4	T_1	$\mathbf{1.304 \times 10^3}$ (2.390×10^{-4})	3.400×10^5 (4.295×10^2) +	$\mathbf{1.304 \times 10^3}$ (2.274×10^{-13})	$\mathbf{1.304 \times 10^3}$ (2.450×10^{-3}) −	$\mathbf{1.304 \times 10^3}$ (2.119×10^{-11})	$\mathbf{1.304 \times 10^3}$ (1.346×10^{-9}) +
	T_2	$\mathbf{1.305 \times 10^3}$ (2.121×10^{-3})	8.574×10^5 (1.582×10^3) +	$\mathbf{1.305 \times 10^3}$ (4.547×10^{-13})	$\mathbf{1.305 \times 10^3}$ (3.531×10^{-3}) −	$\mathbf{1.305 \times 10^3}$ (1.993×10^{-11})	$\mathbf{1.305 \times 10^3}$ (1.320×10^{-9}) +
P_5	T_1	$\mathbf{3.374 \times 10^5}$ (4.217×10^2)	3.400×10^5 (4.230×10^2) +	$\mathbf{3.366 \times 10^5}$ (4.285×10^{-2})	3.384×10^5 (2.654×10^3) +	$\mathbf{3.366 \times 10^5}$ (3.049×10^0)	$\mathbf{3.366 \times 10^5}$ (2.084×10^{-3}) −
	T_2	9.640×10^5 (8.560×10^3)	$\mathbf{8.574 \times 10^5}$ (1.548×10^3) −	$\mathbf{8.491 \times 10^5}$ (2.755×10^{-10})	8.618×10^5 (8.758×10^3) +	$\mathbf{8.491 \times 10^5}$ (7.984×10^{-5})	$\mathbf{8.491 \times 10^5}$ (7.594×10^{-3}) +
P_6	T_1	$\mathbf{1.868 \times 10^8}$ (1.105×10^5)	1.892×10^8 (3.407×10^5) +	$\mathbf{1.867 \times 10^8}$ (5.960×10^{-8})	1.885×10^8 (1.482×10^6) +	$\mathbf{1.867 \times 10^8}$ (1.422×10^{-2})	$\mathbf{1.867 \times 10^8}$ (1.120×10^0) +
	T_2	2.815×10^9 (6.530×10^6)	$\mathbf{2.671 \times 10^9}$ (2.557×10^6) −	$\mathbf{2.653 \times 10^9}$ (4.768×10^{-7})	2.674×10^9 (1.551×10^7) +	$\mathbf{2.653 \times 10^9}$ (1.088×10^{-1})	$\mathbf{2.653 \times 10^9}$ (9.594×10^0) +
P_7	T_1	$\mathbf{6.221 \times 10^4}$ (9.702×10^1)	6.284×10^4 (1.462×10^2) +	$\mathbf{6.201 \times 10^4}$ (4.659×10^{-1})	6.323×10^4 (9.237×10^2) +	$\mathbf{6.201 \times 10^4}$ (1.970×10^0)	$\mathbf{6.201 \times 10^4}$ (5.445×10^{-4}) +
	T_2	1.724×10^4 (1.394×10^2)	$\mathbf{1.495 \times 10^4}$ (2.677×10^1) −	$\mathbf{1.478 \times 10^4}$ (7.520×10^0)	1.481×10^4 (4.567×10^1) +	$\mathbf{1.477 \times 10^4}$ (7.585×10^{-1})	$\mathbf{1.477 \times 10^4}$ (2.106×10^0) ≈
P_8	T_1	$\mathbf{5.201 \times 10^2}$ (6.791×10^{-2})	5.203×10^2 (9.244×10^{-2}) +	5.208×10^2 (1.072×10^{-1})	$\mathbf{5.205 \times 10^2}$ (1.066×10^{-1}) −	$\mathbf{5.202 \times 10^2}$ (5.078×10^{-2})	$\mathbf{5.202 \times 10^2}$ (1.028×10^{-1}) ≈
	T_2	5.214×10^2 (5.860×10^{-2})	$\mathbf{5.202 \times 10^2}$ (8.411×10^{-2}) −	5.207×10^2 (1.176×10^{-1})	$\mathbf{5.206 \times 10^2}$ (1.349×10^{-1}) ≈	$\mathbf{5.202 \times 10^2}$ (5.740×10^{-2})	$\mathbf{5.202 \times 10^2}$ (6.809×10^{-2}) ≈
P_9	T_1	$\mathbf{1.898 \times 10^4}$ (3.961×10^0)	1.902×10^4 (9.390×10^0) +	$\mathbf{1.897 \times 10^4}$ (2.119×10^0)	1.906×10^4 (1.112×10^2) +	$\mathbf{1.897 \times 10^4}$ (1.770×10^0)	$\mathbf{1.897 \times 10^4}$ (1.125×10^2) ≈
	T_2	1.624×10^3 (1.820×10^{-1})	$\mathbf{1.622 \times 10^3}$ (7.316×10^{-2}) −	$\mathbf{1.622 \times 10^3}$ (1.012×10^{-1})	$\mathbf{1.622 \times 10^3}$ (1.118×10^{-1}) ≈	$\mathbf{1.622 \times 10^3}$ (1.113×10^{-1})	$\mathbf{1.622 \times 10^3}$ (8.822×10^{-2}) ≈
P_{10}	T_1	$\mathbf{1.947 \times 10^9}$ (1.414×10^6)	1.957×10^9 (1.920×10^6) +	$\mathbf{1.945 \times 10^9}$ (2.384×10^{-7})	1.972×10^9 (1.430×10^7) +	$\mathbf{1.945 \times 10^9}$ (1.043×10^{-1})	$\mathbf{1.945 \times 10^9}$ (8.881×10^0) +
	T_2	7.516×10^8 (4.203×10^6)	$\mathbf{6.781 \times 10^8}$ (6.624×10^5) −	$\mathbf{6.728 \times 10^8}$ (1.192×10^{-7})	6.740×10^8 (2.580×10^6) +	$\mathbf{6.728 \times 10^8}$ (7.069×10^{-2})	$\mathbf{6.728 \times 10^8}$ (4.084×10^0) +
+/−/≈		11/9/0		15/3/2		12/1/7	

"+" means that IMTO significantly outperforms other algorithms. "−" means that IMTO is significantly outperformed by other algorithms. "≈" means that no significant differences between IMTO and compared algorithms.

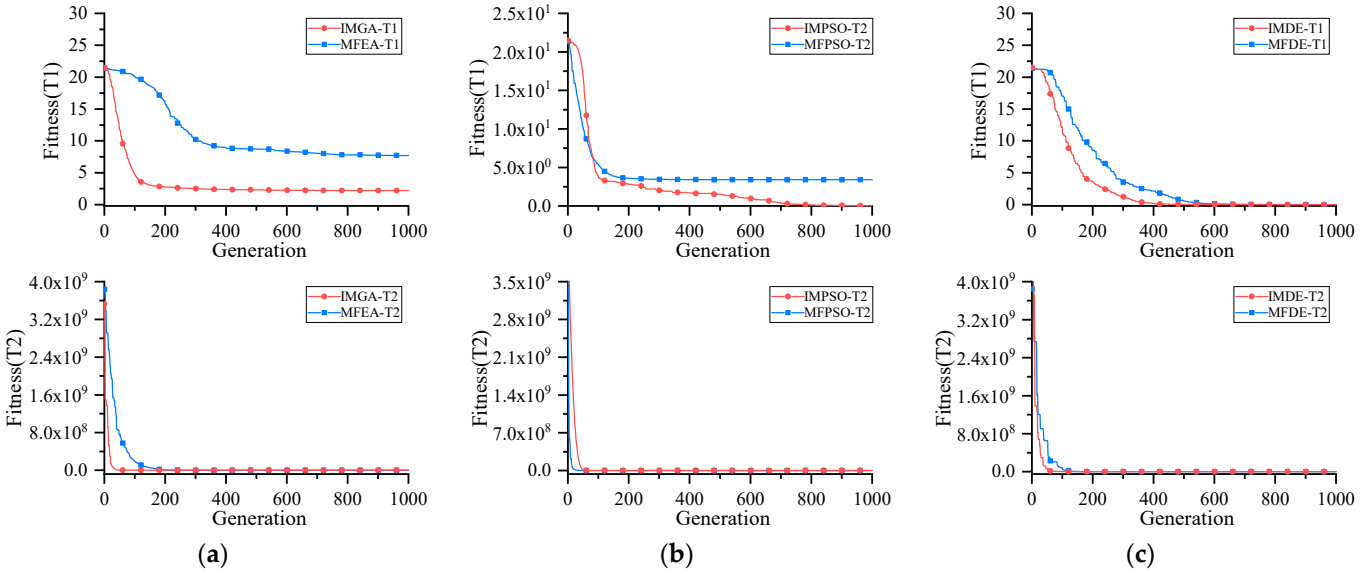

Figure 4. Convergence traces of IMTO and MFO on multitasking problem P_5 in test suite 1. (**a**) Convergence traces of IMGA and MFEA, (**b**) Convergence traces of IMPSO and MFPSO, and (**c**) Convergence traces of IMDE and MFDE.

4.4. Comparison with Baseline Solvers

This section compares IMTO with baseline solvers to further illustrate its generality. Table 6 shows the mean and standard deviation of the best achieved fitness values obtained via 20 independent runs on test suite 1. The best fitness values are shown in bold.

It can be found that even though baseline solvers have different characteristics, IMTO always has superior optimization performance over traditional baseline solvers. The performance superiority of IMTO over baseline solvers can be evidenced by the fact that IMGA significantly outperforms B-GA in 17 out of 18 tasks, IMPSO outperforms B-PSO in 14 out of 18 tasks, IMDE outperforms B-DE in 13 out of 18 tasks, and IMABC outperforms B-ABC in 15 out of 18 tasks on test suite 1. The individually guided learning and archive-memory schemes of IMTO share useful optimization experience in the unified search space, and update inferior individuals continuously through learning from better-performing individuals of other tasks.

Solving tasks with low similarities shows higher challenges for MTO. The results in Table 6 show that IMGA achieves higher better optimization accuracy than B-GA in most cases. The knowledge transfer scheme can help the algorithms find globally optimal solutions for problems possessing different properties. The P_6-T_1 optimization result shows that the baseline solver GA just finds the fitness value of 1.47e+01, but IMGA can find the value of 3.74e+00 by sharing historical experience. Similarly, IMPSO, IMDE, and IMABC perform better in most cases than PSO, DE, and ABC according to Table 6. In IMTO, the superior individuals need not learn from other tasks, which can maintain the vertical evolution of each task. Compared with a single-task solver, inferior individuals can constantly learn from other tasks to avoid being trapped into local optima, thereby suppressing negative transfer.

The convergence curve of P_5 in 1000 iterations is shown in Figure 5. It can be easily found that IMTO has faster convergence in most cases. IMTO's additional horizontal evolution allows for the exchange of information among different tasks. Different task groups are located in homogenous search spaces, and assisted tasks can provide useful knowledge, thereby speeding up the algorithms' convergence.

Table 6. The mean and standard deviation (in brackets) of the best achieved fitness values in test Suite 1.

Task Set		IMGA	GA	IMDE	DE	IMPSO	PSO	IMABC	ABC
		GA-Based		DE-Based		PSO-Based		ABC-Based	
P_1	T_1	$\mathbf{1.11 \times 10^{-1}}$ (5.27×10^{-2})	9.22×10^{-1} + (4.21×10^{-1})	$\mathbf{1.20 \times 10^{-4}}$ (3.63×10^{-4})	$2.49 \times 10^{-3} \approx$ (5.33×10^{-3})	$\mathbf{4.99 \times 10^{-3}}$ (6.57×10^{-3})	4.80×10^{-2} + (1.13×10^{-2})	$\mathbf{2.47 \times 10^{-1}}$ (1.22×10^{-1})	1.75×10^{0} + (1.32×10^{-1})
	T_2	$\mathbf{3.39 \times 10^{2}}$ (4.72×10^{1})	9.23×10^{2} + (7.73×10^{2})	$\mathbf{5.03 \times 10^{1}}$ (1.30×10^{1})	4.03×10^{2} + (1.84×10^{1})	$\mathbf{3.26 \times 10^{2}}$ (6.26×10^{1})	4.90×10^{2} + (7.21×10^{1})	$\mathbf{2.06 \times 10^{2}}$ (6.73×10^{1})	1.33×10^{3} + (1.47×10^{2})
P_2	T_1	$\mathbf{3.83 \times 10^{0}}$ (8.78×10^{-1})	1.55×10^{1} + (3.39×10^{0})	$\mathbf{4.74 \times 10^{-3}}$ (9.54×10^{-3})	2.33×10^{-1} + (4.46×10^{-1})	$\mathbf{3.93 \times 10^{-1}}$ (4.54×10^{-1})	8.16×10^{0} + (1.44×10^{0})	$\mathbf{1.28 \times 10^{0}}$ (8.18×10^{1})	2.12×10^{1} + (3.61×10^{-2})
	T_2	$\mathbf{4.16 \times 10^{2}}$ (3.65×10^{1})	7.14×10^{3} + (7.28×10^{3})	$\mathbf{4.43 \times 10^{1}}$ (1.15×10^{1})	4.04×10^{2} + (2.66×10^{1})	$\mathbf{3.73 \times 10^{1}}$ (1.20×10^{1})	5.14×10^{2} + (1.11×10^{2})	$\mathbf{1.25 \times 10^{2}}$ (8.33×10^{1})	1.31×10^{3} + (1.30×10^{2})
P_3	T_1	2.08×10^{1} (4.19×10^{-1})	$\mathbf{2.00 \times 10^{1}}$ − (4.14×10^{-2})	$\mathbf{2.12 \times 10^{1}}$ (3.86×10^{-2})	$\mathbf{2.12 \times 10^{1}}$ + (2.77×10^{-2})	2.10×10^{1} (5.53×10^{-2})	$\mathbf{2.08 \times 10^{1}}$ − (1.28×10^{-1})	$\mathbf{2.12 \times 10^{1}}$ (3.57×10^{-2})	$\mathbf{2.12 \times 10^{1}} \approx$ (3.43×10^{-2})
	T_2	$\mathbf{1.30 \times 10^{4}}$ (6.82×10^{2})	1.79×10^{4} + (6.29×10^{2})	$\mathbf{9.68 \times 10^{3}}$ (2.25×10^{3})	$9.81 \times 10^{3} \approx$ (1.72×10^{3})	$\mathbf{1.64 \times 10^{4}}$ (3.28×10^{2})	1.68×10^{4} + (5.75×10^{2})	$\mathbf{3.20 \times 10^{119}}$ (1.32×10^{120})	1.72×10^{122} + (4.00×10^{122})
P_4	T_1	$\mathbf{1.95 \times 10^{2}}$ (4.43×10^{1})	8.63×10^{2} + (2.95×10^{2})	$\mathbf{8.10 \times 10^{1}}$ (9.75×10^{0})	3.98×10^{2} + (2.01×10^{1})	$\mathbf{3.23 \times 10^{2}}$ (8.88×10^{1})	4.62×10^{2} + (7.88×10^{1})	$\mathbf{6.29 \times 10^{2}}$ (4.97×10^{1})	1.38×10^{3} + (2.19×10^{2})
	T_2	$\mathbf{7.64 \times 10^{3}}$ (6.35×10^{2})	1.02×10^{4} + (7.25×10^{2})	$\mathbf{2.44 \times 10^{-6}}$ (7.82×10^{-6})	4.64×10^{-6} + (1.81×10^{-5})	$\mathbf{3.38 \times 10^{-4}}$ (2.65×10^{-4})	7.95×10^{-1} + (2.54×10^{-1})	$\mathbf{2.50 \times 10^{2}}$ (1.06×10^{2})	2.89×10^{3} + (4.87×10^{2})
P_5	T_1	$\mathbf{3.59 \times 10^{0}}$ (7.93×10^{-1})	1.69×10^{1} + (3.94×10^{0})	$\mathbf{6.74 \times 10^{-5}}$ (9.91×10^{-5})	1.67×10^{-1} + (4.40×10^{-1})	2.52×10^{-1} (3.81×10^{-1})	$\mathbf{4.80 \times 10^{-2}}$ − (1.35×10^{0})	$\mathbf{2.64 \times 10^{-1}}$ (6.23×10^{-2})	2.12×10^{1} + (4.47×10^{-2})
	T_2	$\mathbf{2.49 \times 10^{4}}$ (2.25×10^{4})	1.02×10^{9} + (9.87×10^{8})	$\mathbf{6.48 \times 10^{1}}$ (2.97×10^{1})	4.09×10^{4} + (1.61×10^{5})	$\mathbf{6.67 \times 10^{1}}$ (3.11×10^{1})	4.90×10^{2} + (1.07×10^{2})	$\mathbf{1.07 \times 10^{2}}$ (5.57×10^{0})	8.49×10^{8} + (2.22×10^{8})
P_6	T_1	$\mathbf{3.74 \times 10^{0}}$ (8.99×10^{-1})	1.47×10^{1} + (3.81×10^{0})	3.06×10^{-1} (5.67×10^{-1})	$\mathbf{2.25 \times 10^{-1}} \approx$ (4.21×10^{-1})	$\mathbf{3.10 \times 10^{-1}}$ (5.02×10^{-1})	8.29×10^{0} + (1.31×10^{0})	$\mathbf{2.12 \times 10^{1}}$ (4.34×10^{-2})	$\mathbf{2.12 \times 10^{1}} \approx$ (3.51×10^{-2})
	T_2	$\mathbf{5.52 \times 10^{0}}$ (8.98×10^{-1})	3.58×10^{1} + (1.45×10^{0})	$\mathbf{1.88 \times 10^{0}}$ (2.37×10^{0})	$2.51 \times 10^{0} \approx$ (2.67×10^{0})	2.19×10^{1} (3.61×10^{0})	$\mathbf{2.14 \times 10^{1}} \approx$ (3.77×10^{0})	$\mathbf{2.00 \times 10^{1}}$ (4.47×10^{0})	3.15×10^{1} + (1.61×10^{0})
P_7	T_1	$\mathbf{2.25 \times 10^{3}}$ (1.74×10^{3})	4.25×10^{6} + (9.03×10^{6})	$\mathbf{9.25 \times 10^{1}}$ (2.75×10^{1})	1.21×10^{4} + (3.03×10^{4})	$\mathbf{8.56 \times 10^{1}}$ (8.06×10^{1})	3.47×10^{2} + (1.81×10^{2})	$\mathbf{5.78 \times 10^{2}}$ (1.66×10^{2})	8.29×10^{8} + (1.76×10^{8})
	T_2	$\mathbf{5.78 \times 10^{2}}$ (2.49×10^{2})	1.60×10^{3} + (1.25×10^{3})	$\mathbf{5.14 \times 10^{1}}$ (1.22×10^{1})	4.01×10^{2} + (1.82×10^{1})	$\mathbf{7.09 \times 10^{1}}$ (3.80×10^{1})	4.87×10^{2} + (9.55×10^{1})	$\mathbf{2.67 \times 10^{2}}$ (3.88×10^{1})	1.30×10^{3} + (1.25×10^{2})
P_8	T_1	$\mathbf{4.21 \times 10^{-2}}$ (1.28×10^{-2})	1.06×10^{0} + (3.91×10^{-1})	$\mathbf{1.41 \times 10^{-3}}$ (3.33×10^{-3})	9.94×10^{-3} + (2.17×10^{-2})	$\mathbf{5.33 \times 10^{-3}}$ (6.56×10^{-3})	5.57×10^{-2} + (1.68×10^{-2})	$\mathbf{1.01 \times 10^{0}}$ (3.17×10^{-2})	1.79×10^{0} + (1.32×10^{-1})
	T_2	$\mathbf{3.07 \times 10^{1}}$ (5.12×10^{-1})	7.86×10^{1} + (1.90×10^{0})	$\mathbf{5.93 \times 10^{0}}$ (2.04×10^{0})	$6.71 \times 10^{0} \approx$ (1.44×10^{0})	5.26×10^{1} (3.80×10^{0})	$\mathbf{4.41 \times 10^{1}} \approx$ (1.42×10^{1})	$\mathbf{2.37 \times 10^{1}}$ (1.15×10^{0})	7.38×10^{1} + (1.76×10^{0})
P_9	T_1	$\mathbf{3.36 \times 10^{2}}$ (8.16×10^{2})	8.94×10^{2} + (3.07×10^{2})	$\mathbf{3.22 \times 10^{2}}$ (1.06×10^{2})	3.97×10^{2} + (2.92×10^{1})	$\mathbf{3.55 \times 10^{2}}$ (6.91×10^{1})	4.64×10^{2} + (1.33×10^{2})	2.23×10^{3} (3.87×10^{2})	$\mathbf{1.28 \times 10^{3}}$ − (1.86×10^{2})
	T_2	$\mathbf{1.75 \times 10^{4}}$ (5.37×10^{2})	1.81×10^{4} + (6.28×10^{2})	$\mathbf{5.86 \times 10^{3}}$ (5.84×10^{2})	9.33×10^{3} + (1.57×10^{3})	$\mathbf{1.60 \times 10^{4}}$ (4.90×10^{2})	1.70×10^{4} + (5.97×10^{2})	$\mathbf{3.08 \times 10^{120}}$ (1.33×10^{121})	1.22×10^{122} + (4.80×10^{122})
+/−/≈		17/1/0		13/0/5		14/2/2		15/1/2	

"+" means that IMTO significantly outperforms other algorithms. "−" means that IMTO is significantly outperformed by other algorithms. "≈" means that no significant differences between IMTO and compared algorithms.

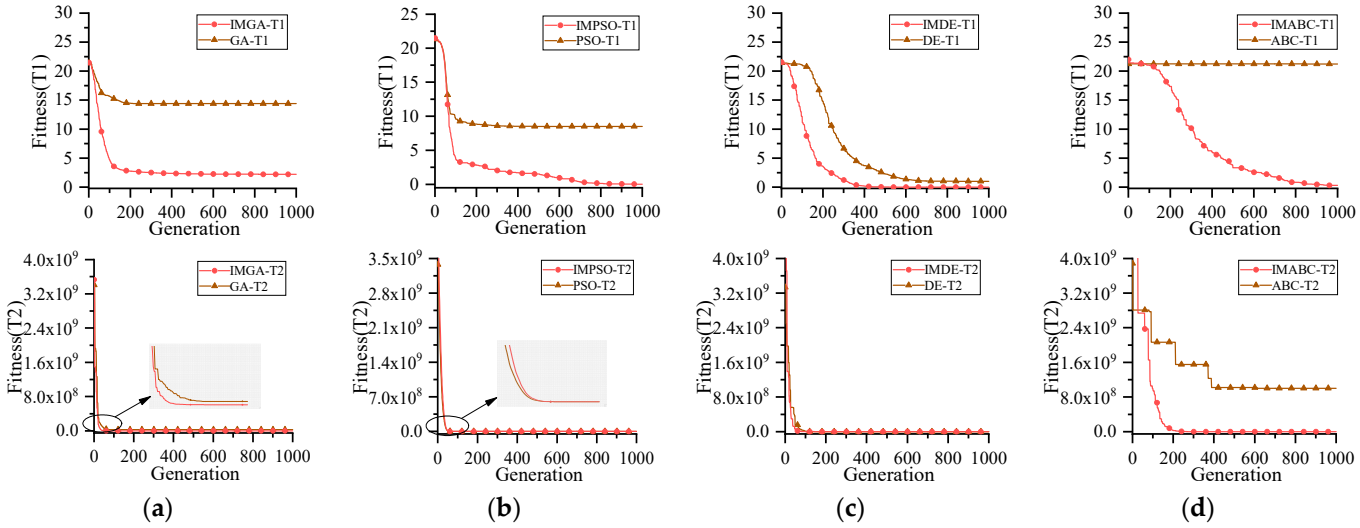

Figure 5. Convergence traces of IMTO and baseline solvers on multitasking problem P_5 in test suite 1. (**a**) Convergence traces of IMGA and B-GA, **(b)** Convergence traces of IMPSO and B-PSO, (**c**) Convergence traces of IMDE and B-DE, and (**d**) Convergence traces of IMABC and B-ABC.

5. Discussion

In this work, we propose a novel framework to handle multi-task problems. It focuses on choosing more suitable and potential individuals to solve each task. The experiments on benchmark problems show that the inter-task learning method used in IMTO can significantly improve the solving accuracy. This is because the target task can learn superior experience from other tasks and avoid being trapped into local optima.

Clearly, the inter-task learning method decides the performance of MTO algorithms. However, the solution distributions of multiple tasks may be different. If the target task learns the knowledge from the assistant task directly, the algorithm performance may be even degraded. So, the inter-task learning faces a challenge that is how to learn knowledge correctly. Consequently, the knowledge transfer proposed in this work may have a limitation caused by a direct learning method. So, to further improve the solution accuracy, we need to minimize the discrepancy of different sub-populations, which can avoid that the learned individuals are not suitable for the original task. Besides, there is much information existing in the assistant task. If we can choose more useful knowledge for a target task, the effect of knowledge on solving the target task can be further improved.

6. Conclusions

This paper has presented a novel high-efficiency multi-task optimization framework focusing on choosing the most suitable individuals to handle each task, which can improve the optimization accuracy via knowledge transfer. It divides the initial population into different task groups according to individuals' skill membership values. Each task group cannot only do the vertical evolution, but also horizontal evolution that contains knowledge transfer among different tasks. The knowledge transfer includes population information sharing and inter-task learning. To further confirm when the knowledge should transfer, the convergence rate is used to guide horizontal knowledge transfer. When a solution process is trapped into local optima, a knowledge transfer mechanism is triggered. Four representative ingle-objective optimization algorithms are used to combine with IMTO. The experimental results show that the proposed framework can significantly outperform baseline solvers and MFO.

Our future work aims to extend the proposed framework to solve multi-objective MTO problems [51–56] and make this kind of individually guided knowledge transfer more general. How to improve the effect of knowledge transfer maximally and avoid negative transfer minimally [57,58] deserve further research efforts.

Author Contributions: X.W., Method development, Writing and original draft preparation, and Performing experiments and result analysis; Q.K., Idea development, and Review and editing; M.Z., Paper revision and research supervision; Z.F., Data analysis, and Paper revision; A.A., Paper review and revision. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China (51775385); in part by the Strategy Research Project of Artificial Intelligence Algorithms of Ministry of Education of China; in part by the Shanghai Industrial Collaborative Science and Technology Innovation Project (2021-cyxt2-kj10); in part by Innovation Program of Shanghai Municipal Education Commission (202101070007E00098); and in part by The Deanship of Scientific Research (DSR) at King Abdulaziz University (KAU), Jeddah, Saudi Arabia has funded this project, under grant no. (RG-11-135-43). We are also grateful for the efforts from our colleagues in the Sino-German Center of Intelligent Systems, Tongji University.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The test problems used in this work come from standard test suites.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hua, Y.; Liu, Q.; Hao, K.; Jin, Y. A survey of evolutionary algorithms for multi-objective optimization problems with irregular pareto fronts. *IEEE/CAA J. Autom. Sin.* **2021**, *8*, 303–318. [CrossRef]
2. Tian, Y.; Li, X.; Ma, H.; Zhang, X.; Tan, K.C.; Jin, Y. Deep reinforcement learning based adaptive operator selection for evolutionary multi-objective optimization. *IEEE Trans. Emerg. Topics Comput. Intell.* **2022**, 1–14. [CrossRef]
3. Deng, Q.; Kang, Q.; Zhang, L.; Zhou, M.; An, J. Objective space-based population generation to accelerate evolutionary algorithms for large-scale many-objective optimization. *IEEE Trans. Evol. Comput.* **2022**. [CrossRef]
4. Nguyen, T.T.; Yang, S.; Branke, J. Evolutionary dynamic optimization: A survey of the state of the art. *Swarm Evol. Comput.* **2012**, *6*, 1–24. [CrossRef]
5. Wang, B.-C.; Li, H.-X.; Li, J.; Wang, Y. Composite differential evolution for constrained evolutionary optimization. *IEEE Trans. Syst. Man Cybern. Syst.* **2019**, *49*, 1482–1495. [CrossRef]
6. Zhang, L.; Wang, K.; Xu, L.; Sheng, W.; Kang, Q. Evolving ensembles using multi-objective genetic programming for imbalanced classification. *Knowl.-Based Syst.* **2022**, *255*, 109611. [CrossRef]
7. Sun, Y.; Yen, G.G.; Yi, Z. IGD Indicator-based Evolutionary Algorithm for Many-objective Optimization Problems. *IEEE Trans. Evol. Comput.* **2019**, *23*, 173–187. [CrossRef]
8. Xu, H.; Zeng, W.; Zeng, X.; Yen, G.G. An evolutionary algorithm based on minkowski distance for many-objective optimization. *IEEE Trans. Cybern.* **2019**, *49*, 3968–3979. [CrossRef]
9. Hong, W.; Tang, K.; Zhou, A.; Ishibuchi, H.; Yao, X. A scalable indicator-based evolutionary algorithm for large-scale multiobjective optimization. *IEEE Trans. Evol. Comput.* **2019**, *23*, 525–537. [CrossRef]
10. Chen, S.-Y.; Song, M.H. Energy-saving dynamic bias current control of active magnetic bearing positioning system using adaptive differential evolution. *IEEE Trans. Syst. Man, Cybern. Syst.* **2019**, *49*, 942–953. [CrossRef]
11. Wang, X.; Choi, T.-M.; Liu, H.; Yue, X. A novel hybrid ant colony optimization algorithm for emergency transportation problems during post-disaster scenarios. *IEEE Trans. Syst. Man Cybern. Syst.* **2018**, *48*, 545–556. [CrossRef]
12. Wang, J.; Zhou, Y.; Wang, Y.; Zhang, J.; Chen, C.L.P.; Zheng, Z. Multiobjective vehicle routing problems with simultaneous delivery and pickup and time windows: Formulation, instances, and algorithms. *IEEE Trans. Cybern.* **2016**, *46*, 582–594. [CrossRef] [PubMed]
13. Pan, Z.; Wang, L.; Wang, J.; Lu, J. Deep reinforcement learning based optimization algorithm for permutation flow-shop scheduling. *IEEE Trans. Emerg. Topics Comput. Intell.* **2021**, 1–12. [CrossRef]
14. Kang, Q.; Song, X.; Zhou, M. A collaborative resource allocation strategy for decomposition-based multiobjective evolutionary algorithms. *IEEE Trans. Syst. Man Cybern. Syst.* **2019**, *49*, 2416–2423. [CrossRef]
15. Fu, Y.; Ding, M.; Zhou, C.; Hu, H. Route planning for unmanned aerial vehicle (UAV) on the sea using hybrid differential evolution and quantum-behaved particle swarm optimization. *IEEE Trans. Syst. Man Cybern. Syst.* **2013**, *43*, 1451–1465. [CrossRef]
16. Lin, Q.; Fang, Z.; Chen, Y.; Tan, K.C.; Li, Y. Evolutionary architectural search for generative adversarial networks. *IEEE Trans. Emerg. Top. Comput. Intell.* **2022**, *6*, 783–794. [CrossRef]
17. Chen, Y.; Zhong, J.; Feng, L.; Zhng, J. An adaptive archive-based evolutionary framework for many-task optimization. *IEEE Trans. Emerg. Topics Comput. Intell.* **2020**, *4*, 369–384. [CrossRef]
18. Yao, S.; Kang, Q.; Zhou, M.; Rawa, M.; Albeshri, A. Discriminative manifold distribution alignment for domain adaptation. *IEEE Trans. Syst. Man, Cybern. Syst.* **2022**. [CrossRef]

19. Kang, Q.; Yao, S.; Zhou, M.; Zhang, K.; Abusorrah, A. Effective visual domain adaptation via generative adversarial distribution matching. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 3919–3929. [CrossRef]
20. Yao, S.; Kang, Q.; Zhou, M.; Rawa, M.; Abusorrah, A. A survey of transfer learning for machinery diagnostics and prognostics. *Artif. Intell. Rev.* **2022**, 1–52. [CrossRef]
21. Gupta, A.; Ong, Y.-S.; Feng, L. Multifactorial evolution: Toward evolutionary multitasking. *IEEE Trans. Evol. Comput.* **2016**, *20*, 343–357. [CrossRef]
22. Wang, X.; Kang, Q.; Zhou, M.; Yao, S.; Abusorrah, A. Domain adaptation multitask optimization. *IEEE Trans. Cybern.* **2022**. [CrossRef] [PubMed]
23. Zheng, X.; Qin, A.K.; Gong, M.; Zhou, D. Self-regulated evolutionary multitask optimization. *IEEE Trans. Evol. Comput.* **2020**, *24*, 16–28. [CrossRef]
24. Huang, S.; Zhong, J.; Yu, W. Surrogate-assisted evolutionary framework with adaptive knowledge transfer for multi-task optimization. *IEEE Trans. Emerg. Top. Comput.* **2021**, *9*, 1930–1944. [CrossRef]
25. Dang, Q.; Gao, W.; Gong, M. An efficient mixture sampling model for gaussian estimation of distribution algorithm. *Inf. Sci.* **2022**, *608*, 1157–1182. [CrossRef]
26. Gupta, A.; Ong, Y.-S.; Feng, L. Insights on transfer optimization: Because experience is the best teacher. *IEEE Trans. Emerg. Topics. Comput. Intell.* **2018**, *2*, 51–64. [CrossRef]
27. Zhou, L.; Feng, L.; Tan, K.C.; Zhong, J.; Zhu, Z.; Liu, K.; Chen, C. Toward adaptive knowledge transfer in multifactorial evolutionary computation. *IEEE Trans. Cybern.* **2021**, *51*, 2563–2576. [CrossRef]
28. Ding, J.; Yang, C.; Jin, Y.; Chai, T. Generalized multitasking for evolutionary optimization of expensive problems. *IEEE Trans. Evol. Comput.* **2019**, *23*, 44–58. [CrossRef]
29. Feng, L.; Zhou, W.; Zhou, L.; Jiang, S.W.; Zhong, J.H.; Da, B.S.; Zhu, Z.X.; Wang, Y. Evolutionary multitasking via explicit autoencoding. *IEEE Trans. Cybern.* **2019**, *49*, 3457–3470. [CrossRef]
30. Gupta, A.; Ong, Y.-S.; Feng, L.; Tan, K.C. Multiobjective multifactorial optimization in evolutionary multitasking. *IEEE Trans. Cybern.* **2017**, *47*, 1652–1665. [CrossRef]
31. Dang, Q.; Gao, W.; Gong, M. Multiobjective multitasking optimization assisted by multidirectional prediction method. *Complex Intell. Syst.* **2022**, *8*, 1663–1679. [CrossRef]
32. Dang, Q.; Gao, W.; Gong, M. Dual transfer learning with generative filtering model for multiobjective multitasking optimization. *Memetic Comput.* **2022**, 1–27. [CrossRef]
33. Feng, L.; Zhou, W.; Zhou, L.; Jiang, S.W.; Zhong, J.H.; Da, B.S.; Zhu, Z.X.; Wang, Y. An empirical study of multifactorial PSO and multifactorial DE. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), Donostia, Spain, 5–8 June 2017; pp. 921–928.
34. Bali, K.K.; Ong, Y.S.; Gupta, A.; Tan, P.S. Multifactorial evolutionary algorithm with online transfer parameter estimation: MFEA-II. *IEEE Trans. Evol. Comput.* **2020**, *24*, 69–83. [CrossRef]
35. Bali, K.K.; Gupta, A.; Ong, Y.-S.; Tan, P.S. Cognizant multitasking in multiobjective multifactorial evolution: MO-MFEA-II. *IEEE Trans. Cybern.* **2021**, *51*, 1784–1796. [CrossRef]
36. Liu, D.; Huang, S.; Zhong, J. Surrogate-assisted multi-tasking memetic algorithm. In Proceedings of the 2018 IEEE Congress on Evolutionary Computation (CEC), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.
37. Bali, K.K.; Gupta, A.; Feng, L.; Ong, Y.S.; Siew, T.P. Linearized domain adaptation in evolutionary multitasking. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), Donostia, Spain, 5–8 June 2017; pp. 1295–1302.
38. Tang, J.; Chen, Y.; Deng, Z.; Xiang, Y.; Joy, C.P. A group-based approach to improve multifactorial evolutionary algorithm. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, Stockholm, Sweden, 13–19 July 2018; pp. 3870–3876.
39. Zhang, J.; Zhou, W.; Chen, X.; Yao, W.; Cao, L. Multisource Selective Transfer Framework in Multiobjective Optimization Problems. *IEEE Trans. Evol. Comput.* **2020**, *24*, 424–438.
40. Martinez, A.D.; Osaba, E.; Ser, J.D.; Herrera, F. Simultaneously evolving deep reinforcement learning models using multifactorial optimization. In Proceedings of the 2020 IEEE Congress on Evolutionary Computation (CEC), Glasgow, UK, 19–24 July 2020.
41. Feng, L.; Huang, Y.; Zhou, L.; Zhong, J.; Gupta, A.; Tang, K.; Tan, K.C. Explicit evolutionary multitasking for combinatorial optimization: A case study on capacitated vehicle Routing Problem. *IEEE Trans. Cybern.* **2021**, *51*, 3143–3156. [CrossRef]
42. Li, G.; Zhang, Q.; Gao, W. Multipopulation evolution framework for multifactorial optimization. In Proceedings of the Genetic and Evolutionary Computation Conference Companion, Kyoto, Japan, 6 July 2018; pp. 215–216.
43. Cheng, M.-Y.; Gupta, A.; Ong, Y.-S.; Ni, Z.-W. Coevolutionary multitasking for concurrent global optimization: With case studies in complex engineering design. *Eng. Appl. Artif. Intell.* **2017**, *64*, 13–24. [CrossRef]
44. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
45. Ghahramani, M.; Qiao, Y.; Zhou, M.; O'Hagan, A.; Sweeney, J. AI-based modeling and data-driven evaluation for smart manufacturing processes. *IEEE/CAA J. Autom. Sin.* **2020**, *7*, 1026–1037. [CrossRef]
46. Wang, Y.; Zuo, X. An Effective Cloud Workflow Scheduling Approach Combining PSO and Idle Time Slot-Aware Rules. *IEEE/CAA J. Autom. Sin.* **2021**, *8*, 1079–1094. [CrossRef]

47. Cao, Y.; Zhang, H.; Li, W.; Zhou, M.; Zhang, Y.; Chaovalitwongse, W.A. Comprehensive learning particle swarm optimization algorithm with local search for multimodal functions. *IEEE Trans. Evol. Computat.* **2019**, *23*, 718–731. [CrossRef]
48. Das, S.; Suganthan, P.N. Differential evolution: A survey of the state-of-the-art. *IEEE Trans. Evol. Comput.* **2011**, *15*, 4–31. [CrossRef]
49. Wang, L.; Zhang, X. Antenna array design by artificial bee colony algorithm with similarity induced search method. *IEEE Trans. Magn.* **2019**, *55*, 1–4. [CrossRef]
50. Da, B.; Ong, Y.; Feng, L.; Qin, A.K.; Gupta, A.; Zhu, Z.; Ting, C.; Tang, K.; Yao, X. Evolutionary Multitasking for Single-Objective Continuous Optimization: Benchmark Problems, Performance Metric, and Baseline Results. 2017. Available online: https://arxiv.org/abs/1706.03470 (accessed on 27 December 2022).
51. Gao, K.; Zhang, Y.; Su, R.; Yang, F.; Suganthan, P.N.; Zhou, M. Solving traffic signal scheduling problems in heterogeneous traffic network by using meta-heuristics. *IEEE Trans. Intell. Transp. Syst.* **2019**, *20*, 3272–3282. [CrossRef]
52. Fu, Y.; Zhou, M.; Guo, X.; Qi, L. Scheduling dual-objective stochastic hybrid flow shop with deteriorating jobs via bi-population evolutionary algorithm. *IEEE Trans. Syst. Man Cybern. Syst.* **2020**, *50*, 5037–5048. [CrossRef]
53. Guo, X.W.; Zhou, M.C.; Liu, S.X.; Qi, L. Lexicographic multiobjective scatter search for the optimization of sequence-dependent selective disassembly subject to multiresource constraints. *IEEE Trans. Cybern.* **2020**, *50*, 3307–3317. [CrossRef] [PubMed]
54. Li, Q.; Gravina, R.; Li, Y.; Alsamhi, S.; Sun, F.; Fortino, G. Multi-user activity recognition: Challenges and opportunities. *Inf. Fusion* **2020**, *63*, 121–135. [CrossRef]
55. Li, W.; He, L.; Cao, Y. Many-objective evolutionary algorithm with reference point-based fuzzy correlation entropy for energy-efficient job shop scheduling with limited workers. *IEEE Trans. Cybern.* **2022**, *52*, 10721–10734. [CrossRef]
56. Wang, Y.; Gao, S.; Zhou, M.; Yu, Y. A multi-layered gravitational search algorithm for function optimization and real-world problems. *IEEE/CAA J. Autom. Sin.* **2021**, *8*, 94–109. [CrossRef]
57. Zhang, W.; Deng, L.; Zhang, L.; Wu, D. A survey on negative transfer. *IEEE/CAA J. Autom. Sin.* **2022**, *8*, 94–109. [CrossRef]
58. Huang, Z.; Yang, S.; Zhou, M.; Li, Z.; Gong, Z.; Chen, Y. Feature Map Distillation of Thin Nets for Low-Resolution Object Recognition. *IEEE Trans. on Image Process.* **2022**, *31*, 1364–1379. [CrossRef] [PubMed]