IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 25, 2020, accepted November 27, 2020, date of publication December 2, 2020,
date of current version December 15, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3042004

A Novel Multi-Task Optimization Algorithm Based on the Brainstorming Process

CHAO LYU[1,2], YUHUI SHI[2], (Fellow, IEEE), AND LIJUN SUN[2,3]

[1]Harbin Institute of Technology, Harbin 150001, China
[2]Key Laboratory of Computational Intelligence, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China
[3]Centre for Artificial Intelligence, CIBCI Lab, Faculty of Engineering and Information Technology, University of Technology Sydney, Ultimo, NSW 2007, Australia

Corresponding author: Yuhui Shi (shiyh@sustech.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFC0804002, in part by the National Science Foundation of China under Grant 61761136008, in part by the Shenzhen Peacock Plan under Grant KQTD2016112514355531, in part by the Program for Guangdong Introducing Innovative and Entrepreneurial Teams under Grant 2017ZT07X386, and in part by the Science and Technology Innovation Committee Foundation of Shenzhen under Grant JCYJ20200109141235597.

ABSTRACT Evolutionary multi-task optimization (EMTO) is an emerging research topic in the field of evolutionary computation, which aims to simultaneously optimize several component tasks within a problem and output the best solution for each task. Since EMTO has widespread applications in solving real-world multi-task optimization problems, in recent years, some EMTO algorithms have been proposed. However, most of which are based on the multifactorial evolution framework which has difficulties in independently controlling the optimization of each component task and implementing parallel computing. To tackle this problem and enrich the EMTO algorithms' family, this paper firstly designs a novel EMTO framework inspired by the brainstorming process of human beings when they solve multi-task problems. Under this framework, a novel EMTO algorithm, named as brain storm multi-task optimization (BSMTO), is presented, where the optimization for each component task and the knowledge transfer between different tasks are both implemented by the proposed brainstorming operations. Furthermore, through investigating the knowledge transfer process in the proposed algorithm, an enhanced BSMTO algorithm named as BSMTO-II is further proposed, where the knowledge transfer in each component task can be managed and controlled by our newly designed scheme. Finally, the proposed two algorithms are tested on benchmark problems. Experimental results show that BSMTO-II has a competitive performance compared with both classical and state-of-the-art algorithms. Moreover, the effectiveness of the proposed EMTO framework and the knowledge transfer control scheme is proved through experiments, and the key parameters settings and the algorithmic complexity are also discussed at last.

INDEX TERMS Brainstorming process, evolutionary computation, evolutionary multi-task optimization.

I. INTRODUCTION

Evolutionary computation (EC) is a technique inspired by the process of natural evolution to solve optimization problems which cannot be effectively addressed by classical optimization methods (e.g., the steepest descent method [1]). Due to its flexibility and robustness in solving real-world optimization problems, EC has been a research hotspot in the field of artificial intelligence. In past few decades, many evolutionary algorithms (EAs) have been proposed by researchers, such as genetic algorithm (GA) [2], genetic programming (GP) [3], differential evolution (DE) [4], particle swarm optimization (PSO) [5], brain storm optimization (BSO) [6], etc. Generally speaking, EC can be classified into two paradigms according to the number of optimization objectives it tackles: the single-objective optimization (SOO), which aims to optimize only one objective function, and the multi-objective optimization (MOO) [7], [8], which aims to optimize several conflicting objectives.

However, in recent years, a new EA paradigm called evolutionary multi-task optimization (EMTO) has been proposed for solving the so called multi-task optimization (MTO)

The associate editor coordinating the review of this manuscript and approving it for publication was Gustavo Olague.

problem that consists of several component tasks, all of which should be optimized simultaneously [9]. Therefore, EMTO is supposed to output the optimal solution for each component task within the MTO problem. However, different from MOO, component tasks within an MTO problem are hoped to have some similarities with each other so that the common information can be shared and transferred among them. In this way, the optimization knowledge obtained from one task can also be utilized to solve the other tasks so that the entire MTO problem can be solved with a higher efficiency. In real world, many problems can be formulated as MTO problems [10]–[14], such as the travelling salesman problem (TSP) [10], the vehicle routing problem (VRP) [14], the optimal power flow problem [11], and the cloud service composition problem [13], etc., all of which can be addressed by EMTO algorithms. Due to the potential optimization ability and practical application value, EMTO has attracted much attention from researchers and has been regarded as the third paradigm besides SOO and MOO in the EC research field [15].

The earliest EMTO algorithm was proposed by Gupta *et al.* [9] named as multifactorial evolutionary algorithm (MFEA). It is inspired by the biocultural model of multifactorial inheritance, where component tasks are optimized simultaneously by a single population through a search operator called assortative mating. Afterwards, more and more EMTO algorithms emerged. For example, Feng *et al.* [16] tried to embed particle swarm optimization and differential evolution into EMTO and proposed two EMTO algorithms named MFPSO and MFDE, respectively. Besides, Zheng *et al.* [17] and Yu *et al.* [18] also studied the DE-based EMTO. Chen *et al.* [19] employed the cooperative coevolutionary mechanism and proposed an EMTO algorithm called EMTSO-CCMA for high-dimensional optimization problems. Tang *et al.* [20] proposed a group-based MFEA that groups tasks of similar types and selectively transfers the genetic information only within the groups. Bali *et al.* [21] proposed a linearized domain adaptation (LDA) strategy that transforms the search space of a simple task to the search space similar to its constitutive complex task. Moreover, since the knowledge transfer across tasks is a key operation in EMTO, some researchers tried to facilitate the evolutionary process by modifying the knowledge transfer operations. For example, Wen and Ting [22] embedded parting ways detection and resource reallocation into MFEA and proposed the MFEARR algorithm. Zheng *et al.* [15] proposed a self-regulated EMTO (SREMTO) algorithm which can automatically adapt to the intensity of the knowledge transfer according to the degrees of relatedness between different tasks. Feng *et al.* [23] proposed an explicit cross-task genetic transfer scheme based on the autoencoding [24]. In addition to focusing on modifying EMTO algorithms, there are also many works concentrating on their applications in real-world problems [10]–[14], which illustrate the extensive application prospect of EMTO. However, most existing EMTO algorithms are based on the multifactorial evolution [25], [26] and try to tackle multiple optimization tasks through evolving a single population, which brings difficulties in controlling the optimization process and the knowledge transfer in each component task and implementing parallel computing. Therefore, this paper proposes a novel EMTO framework inspired by the brainstorming (BS) process in human creative problem solving process [27] named as brain storm multi-task problems solver (BSMTPS), hoping that the diversity in inspirations can bring the diversity in performance improvements and emerge better algorithms.

Before the emergency of EMTO algorithms, Shi [6] proposed a population-based optimization algorithm named as brain storm optimization (BSO), which is a typical SOO algorithm. Different from traditional swarm intelligence based optimizers which are inspired by collective behaviours of animals like birds [5], ants [28], and bees [29], etc., BSO is proposed by imitating behaviours of human beings which is the most intelligent animal in this world. On the other hand, BSO adopts a clustering operator to divide the individuals into several groups according to their similarities in each generation, which plays an important role in balancing the exploration with the exploiting of the search space. Since the original BSO was proposed, it attracts more and more attention from researchers due to its potential optimization ability so that many modified algorithms have been designed. For example, some researchers tried to replace the k-means clustering method that the original BSO employed with different clustering algorithms such as the simple grouping method [30] and the affinity propagation clustering [31] to improve the performance of BSO. Cheng *et al.* [32] modified BSO to make it suitable for solving multimodal optimization problems. Moreover, based on BSO, Shi *et al.* [33] proposed a multi-objective optimization algorithm, which expands the application field of BSO from SOO into MOO. In addition to modifying algorithms, researchers have employed BSO to solve a variety of real-world optimization problems such as the satellite formation reconfiguration problem [34], where a novel algorithm called closed-loop BSO was proposed, the wireless sensor networks deployment problem [35], and the DC brushless motor optimization [36], etc.

Besides, it should be noted that the BS process has the natural potential to solve several different but similar problems simultaneously. For example, holding seminars by inviting scholars from different research fields can be seen as a BS process, where good ideas can be generated through cross-domain discussions. Although some existing EAs are also inspired by human beings' behaviours [37], the cluster-based search scheme adopted by BSO provides a natural framework for solving multi-task problems since the individuals can be naturally clustered by different component tasks. Based on the inspiration, the BSMTPS framework can be designed. Under this framework, a novel EMTO algorithm named as brain storm multi-task optimization (BSMTO) is proposed, which can tackle multiple optimization tasks and achieve knowledge transfer through implementing the proposed BS operations on multiple populations. Moreover, the

optimization process and knowledge transfer in each task can be independently observed and controlled so that the evolutionary efficiency can be improved while the parallelization can be naturally achieved. Therefore, BSMTO can be seen as an extension of BSO from SOO to MTO. However, it should be pointed out that proposing BSMTO is not just embedding BSO into the existing EMTO frameworks, but designing a novel methodology to solve MTO problems.

The proposed BSMTO is a population-based EMTO algorithm with multiple sub-populations, each of which optimizes a single component task. In BSMTO, searches in decision spaces are implemented by two types of BS operators: the internal brain storm (IBS), which is designed to search the decision space of each of the component tasks, and the cross-task brain storm (CBS), which is designed to transfer optimization knowledge between two tasks. Through these two operators, new individuals can be generated, then they will be evaluated and assigned to specific sub-populations according to the proposed directional assignment (DA) scheme. Furthermore, we experimentally investigate the knowledge transfer scheme in BSMTO. Although similar work have been done in reference [22], in this paper, we try to evaluate the performance of new individuals generated by different BS operators in a more straightforward way. Based on the experimental results, we define the saturation point in knowledge transfer (SPKT) and propose a novel knowledge transfer control (KTC) scheme. By combining it with BSMTO, we propose an enhanced algorithm named as BSMTO-II to improve the knowledge transfer efficiency.

The main contributions of this paper can be summarized as follows:

- This paper designs a novel framework for solving MTO problems inspired by the BS process in human beings and proposes two novel EMTO algorithms, where the optimization process in each component task can be implemented and controlled independently through the proposed BS operators.
- This paper investigates the knowledge transfer process in EMTO from a new perspective and proposes a new knowledge transfer control scheme to improve the evolutionary efficiency in multi-task optimization.
- Through conducting experiments on benchmarks, this paper demonstrates the superiority of the proposed algorithm compared with existing EMTO algorithms and state-of-the-art SOO algorithms, and illustrates the effectiveness of the proposed EMTO framework and the knowledge transfer control scheme.

The rest of this paper is organized as follows. In Section II, some backgrounds about this research work are introduced. Then the proposed algorithms, BSMTO and BSMTO-II with the KTC scheme, are introduced in detail in Section III. In Section IV, the proposed algorithms are tested on benchmark problems and compared with both classical and state-of-the-art algorithms, while the key parameters settings and the algorithmic complexity are also discussed in this section. Section V summarizes the whole paper and gives the conclusion and future work.

II. BACKGROUNDS

A. EVOLUTIONARY MULTI-TASK OPTIMIZATION

A general multi-task optimization problem can be defined as follows: Suppose a MTO problem with k component tasks: T_1, T_2,…,T_k. Without loss of generality, suppose that each task is a minimization problem. For the jth task T_j, $j \in \{1, 2, \ldots, k\}$, the search apace is X_j, which is a D_j-dimensional space defined on $\mathbb{R}$, and the objective function is f_j, which defines the map: $X_j \to \mathbb{R}$. The goal of the MTO problem is to find the optimal solution for each task T_j, that is, find k solutions: $x_1, x_2, \ldots, x_k$, that satisfy: $x_j = \arg\min f_j(x)$ and $x_j \in X_j, j \in \{1, 2, \ldots, k\}$.

It should be noted that there indeed exist some similarities between MTO and MOO since both of which tackle more than one objective functions simultaneously. However, they are essentially different types of optimization problems. In general, MOO requires to find non-dominated solutions as many as possible to form the Pareto front [7], from which decision makers can select suitable solutions according to their requirements. Hence, the essence of MOO is to find a trade-off among conflicting objectives. However, MTO requires to provide the best solution for each optimization task, which aims to solve multiple optimization problems by making use of their similarities.

B. MFEA ALGORITHM

Among EMTO algorithms, MFEA [9] is the earliest and most representative one. In MFEA, four important concepts are defined to evaluate the individuals:

- Definition 1 (*Factorial Cost*): Given a task T_j and an individual p_i, the factorial cost Ψ_i^j is defined as:

$$\Psi_i^j = \lambda \cdot \delta_i^j + f_i^j \tag{1}$$

  where δ_i^j and f_i^j are the total constraint violation and the objective function value of p_i on T_j, respectively, and λ is the penalizing multiplier. For non-constrained optimization problems, we have $\Psi_i^j = f_i^j$.
- Definition 2 (*Factorial Rank*): The factorial rank r_i^j of p_i on task T_j is defined as its index in the list of all individuals sorted in the ascending order according to their factorial costs on T_j.
- Definition 3 (*Scalar Fitness*): The scalar fitness φ_i of p_i is defined as the reciprocal of the minimum value of r_i^j among all tasks, that is:

$$\varphi_i = \frac{1}{\min_{j\in\{1,2,\ldots,k\}}\{r_i^j\}} \tag{2}$$

- Definition 4 (*Skill Factor*): The skill factor τ_i of p_i is defined as the index of the task on which p_i performs best, that is:

$$\tau_i = \arg\min_{j\in\{1,2,\ldots,k\}}\{r_i^j\} \tag{3}$$

The implementation process of MFEA is similar to that of a standard EA: After initialization, all individuals are evaluated on all tasks to compute their skill factors. Then the genetic operators (i.e. *crossover* and *mutation*) are applied on the current population to generate offspring. However, different from a standard EA, MFEA employs the *Assortative Mating* to determine which kind of genetic operator should be applied on the parental individuals according to their skill factors, and the skill factors of offspring are assigned by the *Vertical Cultural Transmission* scheme through inheriting their parents' skill factors. To reduce the computational complexity, each offspring in MFEA is only evaluated on the single task determined by its current skill factor. Then the current population and the offspring are concatenated to form a *intermediate-pop* and the individuals with the best scalar fitness are selected to form the next population. The above process is recursively implemented until the stopping conditions are satisfied.

C. BRAIN STORM OPTIMIZATION

Brain storm optimization (BSO) is a classical population-based optimization algorithm which is proposed by Shi [6] through simulating the brainstorming (BS) process in human beings. BS is an effective method for solving very difficult problems, which requires a group of people with different backgrounds getting together to conduct the brainstorming process. People in the group should propose their ideas for solving the problem based on the ideas proposed by others, and problem owners should evaluate these ideas and select several better ideas which are used as seeds to generate more ideas. The above process can be performed for several iterations until a satisfactory solution is generated.

BSO imitates the above process to solve optimization problems. In the BSO algorithm, all individuals are divided into several clusters for implementing BS and the idea generation process is implemented by genetic operators. To be specific, an offspring can be generated based on two parental individuals from different clusters or based on a single individual. Moreover, a parental individual can be either the best individual in its cluster or an individual that is randomly selected from the cluster. A standard BSO algorithm can be described in Algorithm 1, where *rand* is a randomly generated number, P_1, P_2, P_3, and P_4 are four predefined parameters, lb_j and ub_j are the lower bound and the upper bound for the jth dimension of the search space, respectively.

Algorithm 1 Brain Storm Optimization (BSO)

for each individual x_i, $(i \in 1:n)$ **do**
  for each decision variable x_{i_j}, $(j \in 1:D)$ **do**
    x_{i_j} = a uniform random number from $[lb_j, ub_j]$.
  end for
  Add x_i into the initial population.
end for
while the stopping criterion is not met **do**
  Divide all individuals into m clusters.
  Evaluate each individual and record the best individual in each cluster as the cluster center.
  if $rand < P_1$ **then**
    Randomly select a cluster and replace its center with a randomly generated individual.
  end if
  while n offspring have not been generated **do**
    if $rand < P_2$ **then**
      Randomly select a cluster c_p.
      if $rand < P_3$ **then**
        Apply mutation operator on the cluster center of c_p to generate offspring.
      else
        Apply mutation operator on a randomly selected individual from c_p to generate offspring.
      end if
    else
      Randomly select two clusters c_p and c_q.
      if $rand < P_4$ **then**
        Apply crossover operator on the two centers of c_p and c_q to generate offspring and then apply mutation operator on the offspring.
      else
        Randomly select an individual from each of c_p and c_q and perform crossover and mutation operators on which to generate offspring.
      end if
    end if
  end while
  Evaluate the n offspring and update the population.
end while

III. THE PROPOSED BRAIN STORM MULTI-TASK OPTIMIZATION

In this section, we firstly introduce the BS process into the MTO problem solving and propose a framework named brain storm multi-task problems solver (BSMTPS). Under this framework, a novel EMTO algorithm called brain storm multi-task optimization (BSMTO) is proposed. Then a knowledge transfer control scheme is proposed through investigating the performance of different types of offspring in BSMTO and then combined with the proposed algorithm to form an enhanced BSMTO algorithm called BSMTO-II.

A. THE BRAIN STORM EMTO FRAMEWORK

It is well known that the BS process is firstly designed for solving a single difficult problem based on the wisdom of crowds. Therefore, the BSO algorithm is proposed originally for solving single-task optimization (STO) problems. However, it should be noted that the BS possesses a natural potential to solve MTO problems, since solving a MTO problem not only requires to optimize the component tasks

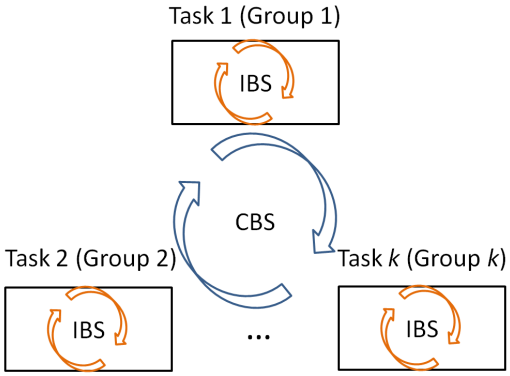


FIGURE 1. Illustration of the brain storm multi-task problems solver (BSMTPS).

simultaneously, but also hopes to transfer common knowledge across different tasks to facilitate the optimization of all tasks. Suppose we have a MTO problem with several component tasks, for solving a single task, we can gather a group of people who are skilled in solving this task and implement the BS process within them. And for realizing knowledge transfer, we can gather the representative members from each group to from a joint group and implement the BS process within the joint group. In this way, not only can we solve each component task, but also facilitate the optimization of difficult tasks with the help of the shared knowledge obtained from some easier tasks. Based on the above idea, in this paper, we propose an EMTO framework called brain storm multi-task problems solver (BSMTP) by dividing individuals into several groups and perform BS operations inner and inter groups, as illustrated in FIGURE 1.

It can be seen that each component task is assigned with a group of individuals and two types of BS operations are designed, i.e., the internal brain strom (IBS) and the cross-task brain storm (CBS). IBS is executed within individuals belonging to the same group for solving the corresponding task while CBS is executed within representative individuals selected from different groups for sharing common knowledge. Through the cooperation of IBS and CBS, component tasks can be solved with a higher efficiency than solving each of them independently.

B. BRAIN STORM MULTI-TASK OPTIMIZATION

1) AN OVERVIEW OF BSMTO

Based on the designed BSMTPS framework, we propose a novel EMTO algorithm called brain storm multi-task optimization (BSMTO) which is summarized in Algorithm 2, where *rand* is a randomly generated number. The proposed BSMTO is a population-based EA with k sub-populations, each of which is responsible for solving a single component optimization task through implementing the IBS operation (see Algorithm 3) which is proposed by simplifying a standard BSO. And individuals in each sub-population are only evaluated on the corresponding task. As for the knowledge transfer, we perform the CBS process (see Algorithm 4) on individuals selected from two different sub-populations. Both of these two types of BS are implemented by genetic operators.

In detail, after initializing k sub-populations with size of n, offspring can be generated in two ways, and we call offspring generated through IBS as pure-offspring and call offspring generated through CBS as hybrid-offspring. IBS and CBS operations are randomly chosen by BSMTO according to a predefined possibility p_2, whose role is similar to that of P_2 in BSO. In each generation, after nk offspring are generated, each of them is evaluated and assigned according to the proposed directional assignment (DA) scheme (see Algorithm 5), then each sub-population can be updated. The above process is iteratively performed until the stopping condition is satisfied.

Algorithm 2 Brain Storm Multi-Task Optimization (BSMTO)

1: **for** each component task T_i, $(i \in 1:k)$ **do**
2: Initialize a sub-population sp_i with n individuals based on X_i according to lines 1-6 of Algorithm 1.
3: Evaluate each individual in sp_i on T_i and record the best individual as the center of sp_i.
4: **end for**
5: **for** $g = 1 : gmax$ **do**
6: **while** nk offspring have not been generated **do**
7: **if** $rand < p_2$ **then**
8: Generate two pure-offspring using IBS.
9: **else**
10: Generate two hybrid-offspring using CBS.
11: **end if**
12: **end while**
13: Evaluate the new individuals and add them to the corresponding sub-populations according to DA algorithm.
14: **for** each sub-population sp_i **do**
15: Select the best n individuals to form the next generation of sp_i
16: Record the best individual as the center.
17: **end for**
18: **end for**

2) INTERNAL BRAIN STORM (IBS) AND CROSS-TASK BRAIN STORM (CBS)

The IBS and the CBS are two operators proposed to generate new individuals. IBS is proposed by simplifying a standard BSO algorithm and is implemented within a sub-population. The process of IBS is described as follows. From the k sub-populations, we randomly select a sub-population from which the offspring will be generated. Similar to the offspring generation scheme in BSO, IBS selects parents in two different ways. That is, a pair of parental individuals can be either two randomly selected individuals from the sub-population or the combination of a randomly selected individual with the best individual (i.e. center individual) of the sub-population.

These two ways are randomly chosen by BSMTO according to a predefined probability p_3, whose role is similar to that of P_3 in BSO. Finally two offspring can be generated based on the selected parents by applying genetic operators. The process of IBS is described in Algorithm 3, where *rand* is a randomly generated number.

Algorithm 3 Internal Brain Storm (IBS)

1: Randomly select a sub-population sp_i.
2: **if** $rand < p_3$ **then**
3: Randomly select an individual from sp_i.
4: Apply crossover and mutation operators on the combination of the selected individual with the center individual of sp_i to generate two offspring.
5: **else**
6: Randomly select two individuals from sp_i.
7: Apply crossover and mutation operators on the two selected individuals to generate two offspring.
8: **end if**

Algorithm 4 Cross-Task Brain Storm (CBS)

1: Randomly select two sub-populations sp_i and sp_j.
2: **for** each selected sub-population **do**
3: Randomly select an individual as a parental individual.
4: **end for**
5: Apply crossover and mutation operators on the two selected individuals to generate two offspring.

CBS is the other offspring generation scheme which is proposed for achieving knowledge transfer across different tasks. Similar to the process of IBS, CBS uses two individuals from different sub-populations to create offspring. It works as follows. Firstly we randomly select two sub-populations from the k sub-populations. For each selected sub-population, we randomly select an individual as a parental individual. After two parental individuals have been selected, we apply the genetic operators on them to generate two offspring. The process of CBS is described in Algorithm 4.

3) DIRECTIONAL ASSIGNMENT (DA) SCHEME

As described above, during the optimization process, two types of individuals are generated in BSMTO: the pure-offspring which are created by IBS and the hybrid-offspring which are created by CBS. However, it should be noted that these two types of new individuals have distinct responsibilities. Pure-offspring are supposed to search in the decision space of a single task, which naturally should be evaluated on the task corresponding to their parents' sub-population. While the hybrid-offspring are generated by individuals from two different sub-populations and supposed to transfer common knowledge between the two corresponding tasks. Hence, evaluating a hybrid-offspring on either task is meaningful. However, as pointed out in [9], evaluating each individual on all component tasks may cost too much computing resource.

Algorithm 5 Directional Assignment (DA)

1: **if** the new individual x is a pure-offspring (which parents are selected from sp_i) **then**
2: Evaluate x on T_i.
3: Add x into sp_i.
4: **else**
5: Randomly select a sub-population sp_m from the two sub-populations: sp_i and sp_j to which its parents belong.
6: Evaluate x on T_m.
7: Add x into sp_m.
8: **end if**

Therefore, we propose a directional assignment (DA) scheme to determine the task and the sub-population for an offspring to be evaluated on and added in.

The DA scheme works as follows. For a pure-offspring, we simply evaluate it on the task corresponding to its parents' sub-population and then add it into the sub-population. For a hybrid-offspring, we firstly randomly choose a sub-population from the two sub-populations to which its parents belong, then it will be evaluated on the task corresponding to the selected sub-population and then added into the sub-population. The proposed DA scheme is described in Algorithm 5.

4) BSMTO FOR MTO PROBLEMS WITH MULTIPLE CONTINUOUS SEARCH SPACES

In this section, we consider a special circumstance that the component tasks within a MTO problem have multiple continuous search spaces. That is to say, the component optimization tasks have their own dimensionalities and search ranges. For example, suppose a MTO problem that consists of two tasks: T_1 and T_2, where T_1 has an objective function with 30 dimensions, and the value range for each dimension is $[-5, 5]$, while the dimensionality of the objective function of T_2 is 50, and the value range for each dimension is $[-10, 10]$. This type of MTO problems widely exist in practical optimization problems. To solve such problems and improve the generality of BSMTO, we propose a method which is embedded into CBS and DA to achieve knowledge transfer across tasks with multiple continuous search spaces.

Suppose CBS selects two tasks T_1 and T_2 with dimensionalities of D_1 and D_2, respectively. Without loss of generality, assume that $D_1 \leq D_2$, and the search ranges for each dimension are $[lb_1, ub_1]$ and $[lb_2, ub_2]$, respectively. After selecting two parental individuals x_1 and x_2 from the corresponding sub-populations sp_1 and sp_2, respectively, CBS firstly extracts the decision variables in the first D_1 dimensions of x_2 to form a shorter individual cx_2, that is, $cx_2 = [x_{2_1}, x_{2_2}, \ldots, x_{2_{D_1}}]$. Then we standardize cx_2 and x_1 as $c\tilde{x}_2$ and $\tilde{x_1}$ by the following equations:

$$\tilde{x_1} = (x_1 - lb_1)/(ub_1 - lb_1) \tag{4}$$

$$c\tilde{x}_2 = (cx_2 - lb_2)/(ub_2 - lb_2) \tag{5}$$

Then we apply genetic operators on $\tilde{x_1}$ and $c\tilde{x}_2$ to generate two offspring $\tilde{o_1}$ and $\tilde{o_2}$ which can be decoded into o_1 and o_2 by the following equations:

$$o_1 = lb_1 + (ub_1 - lb_1)\tilde{o_1} \tag{6}$$

$$o_2 = lb_2 + (ub_2 - lb_2)\tilde{o_2} \tag{7}$$

Then we complement o_1 and o_2 as O_1 and O_2, respectively for evaluating them on the corresponding tasks. To be specific, we firstly choose the evaluation task for each hybrid-offspring. For example, if o_1 will be evaluated on T_1, it remains unchanged, that is, $O_1 = o_1$. However, if o_1 will be evaluated on T_2, we complement it with the remaining decision variables in x_2, that is, $O_1 = [o_1, x_{2_{D_1+1}}, x_{2_{D_1+2}}, \ldots, x_{2_{D_2}}]$. The same operation is also implemented on o_2 to form O_2. Finally, the generated hybrid-offspring O_1 and O_2 will be evaluated and added into the corresponding sub-populations.

C. KNOWLEDGE TRANSFER CONTROL (KTC) SCHEME AND BSMTO-II ALGORITHM

As mentioned above, through the IBS and CBS operations, two types of offspring are generated: the pure-offspring and the hybrid-offspring. In this section, we analyse the performance of these two types of offspring through experiments and propose the knowledge transfer control (KTC) scheme.

To be specific, we use the proposed BSMTO algorithm to optimize 9 benchmark MTO problems [38], each of which contains two tasks. In each generation, we calculate the following two metrics for each sub-population before updating it:

$$app = \frac{\sum_{j=1}^{np} rp_j}{np * m} \tag{8}$$

$$aph = \frac{\sum_{j=1}^{nh} rh_j}{nh * m} \tag{9}$$

where j is the individual index; rp_j and rh_j are the ranks of the jth pure-offspring and the jth hybrid-offspring generated during the current generation among all individuals of the current sub-population in the ascending order according to their objective values, respectively; np and nh are the numbers of the pure-offspring and the hybrid-offspring generated during the current generation, respectively; and m is the total number of individuals in the current sub-population. It can be seen that app is the average relative rank of the pure-offspring in the sub-population, which reflects the overall performance of the pure-offspring generated during this generation, and aph is the average relative rank of the hybrid-offspring in the sub-population, which reflects the overall performance of the hybrid-offspring generated during this generation. The lower the values of these two metrics are, the better the corresponding types of offspring perform.

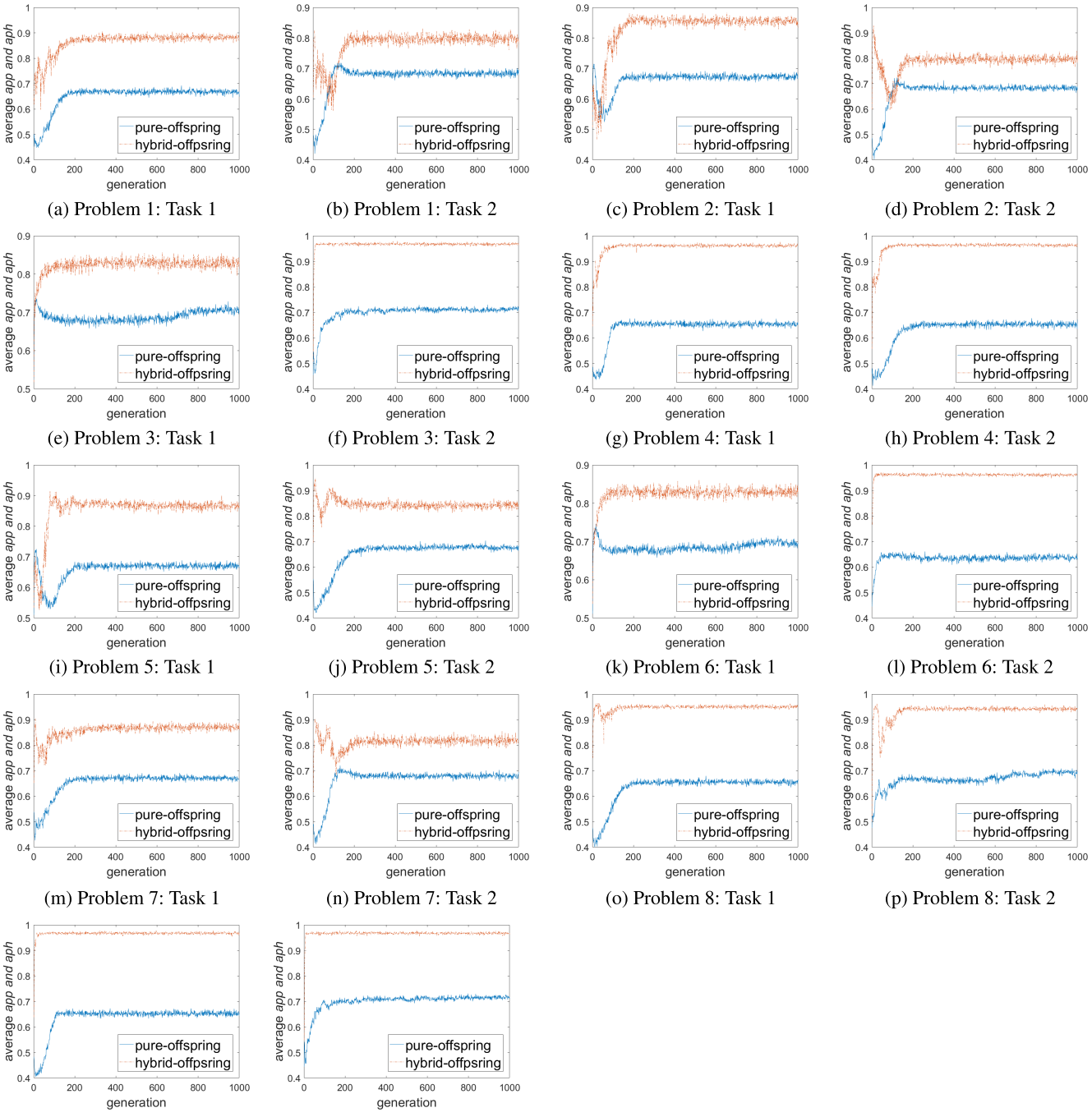
For each MTO problem, we implement 20 independent experiments and record the mean values of app and aph for each sub-population (component task) in each generation, the experimental setup can be seen in Section IV.B and the results are shown in FIGURE 2.

It can be seen that the overall performance of the pure-offspring is different from that of the hybrid-offspring during the whole optimization process in these MTO problems. Generally speaking, the pure-offspring outperform the hybrid-offspring in most optimization time. However, in early stages, the performance of the hybrid-offspring is close to that of the pure-offspring, and for some tasks, the hybrid-offspring can even outperform the pure-offspring, which means that the hybrid-offspring have a tendency to make a positive role in the optimization of the task, since they may guide correct search directions for the sub-population. With the optimization process going on, the performance of the hybrid-offspring deteriorate constantly and the values of the aph will ultimately exceed 0.8 in most tasks, which means that the hybrid-offspring have a tendency to play a negative role, since they may misguide the optimization process of the task. From these observations, it can be concluded that the knowledge transfer implemented by CBS tends to make positive effect in early stages of optimization; however, in late stages, it tends to have negative influence on the task optimization.

Based on the above experimental results and analysis, we propose the following knowledge transfer control scheme (KTC) for BSMTO, which is used to control the generation of hybrid-offspring and improve the MTO efficiency. Before introducing it, we firstly define the saturation point of the knowledge transfer (SPKT) in BSMTO in Definition 5.

- Definition 5 (*The Saturation Point of Knowledge Transfer*): For the BSMTO algorithm, the saturation point of knowledge transfer of a component task T_i is the moment that the hybrid-offspring in the corresponding sub-population sp_i can no longer make a significant positive effect on the optimization of T_i.

In this paper, we try to detect the SPKT of each task by evaluating the overall performance of the hybrid-offspring in the corresponding sub-population during a period of time. And the KTC scheme is designed by detecting the SPKT for each task and adopting measures to prevent the optimization from being influenced by the harmful transferred knowledge. The KTC works as follows. During the optimization process of BSMTO, for each sub-population sp_i, we calculate and record its aph value in every generation. For every dg generations, the metric Ph_i is calculated and updated, which is the average value of aph over the last dg generations in sp_i. Once $Ph_i > \delta$, where δ is a predefined threshold, which means the corresponding task T_i meets SPKT, the knowledge transfer is no loner open for sp_i. That is, for the following generations, once two hybrid-offspring are created based on parental individuals from sp_i and sp_j, as long as task j does not meet the SPKT, i.e. $Ph_j < \delta$, the two hybrid-offspring are both evaluated on T_j and then added into sp_j; However, if $Ph_j > \delta$, which means that T_i and T_j both meet their SPKTs, CBS will no longer select sp_i and sp_j together for creating hybrid-offspring. Once all component tasks meet their SPKTs, that



(a) Problem 1: Task 1 (b) Problem 1: Task 2 (c) Problem 2: Task 1 (d) Problem 2: Task 2

(e) Problem 3: Task 1 (f) Problem 3: Task 2 (g) Problem 4: Task 1 (h) Problem 4: Task 2

(i) Problem 5: Task 1 (j) Problem 5: Task 2 (k) Problem 6: Task 1 (l) Problem 6: Task 2

(m) Problem 7: Task 1 (n) Problem 7: Task 2 (o) Problem 8: Task 1 (p) Problem 8: Task 2

(q) Problem 9: Task 1 (r) Problem 9: Task 2

FIGURE 2. **Changes of the average values of *app* and *aph* during the optimization process for each component task of the 9 benchmark MTO problems over 20 independent experiments.**

is, $Ph_i > \delta$ for $i \in \{1, 2, \ldots, k\}$, we just terminate the CBS operation by setting $p_2 = 1$ in the following generations.

Based on the proposed KTS scheme, we modify the previously proposed CBS and DA as CBS-II and DA-II, respectively. Based on these two modified algorithms, we modify the previously proposed BSMTO as BSMTO-II. These three modified algorithms are described in Algorithms 6-8.

D. EXTENSION OF THE PROPOSED ALGORITHMS

It should be pointed out that the proposed algorithms can provide a novel framework for designing more multi-population based EMTO algorithms. In both BSMTO and BSMTO-II, the IBS operation can be treated as a traditional single-task optimizer that searches within the decision space of a task. For simplicity, in this paper, the IBS is just designed by

Algorithm 6 Brain Storm Multi-Task Optimization-II (BSMTO-II)

1: **for** each component task T_i, $(i \in 1:k)$ **do**
2: Initialize a sub-population sp_i with n individuals based on X_i according to lines 1-6 of Algorithm 1.
3: Evaluate each individual in sp_i on T_i and record the best individual as the center of sp_i.
4: Initialize the SPKT flag $S_i = 0$.
5: **end for**
6: **for** $g = 1 : gmax$ **do**
7: Generate nk new individuals according to Algorithm 2 (lines 6-12) by using IBS and CBS-II.
8: Evaluate and assign offspring according to DA-II.
9: **for** each sub-population sp_i **do**
10: Calculate and record the value of aph for the current generation.
11: Select the best n individuals to form the next generation and record the best individual as the center.
12: **if** g/dg is an integer **then**
13: Calculate and update Ph_i.
14: **if** $Ph_i > \delta$ **then**
15: Set $S_i = 1$.
16: **end if**
17: **end if**
18: **end for**
19: **if** $S_i = 1$, $for\ i \in \{1, 2, \ldots k\}$ **then**
20: Set p_2=1.
21: **end if**
22: **end for**

Algorithm 7 Cross-Task Brain Storm-II (CBS-II)

1: Randomly select a sub-population sp_i whose S_i is equal to 0.
2: Randomly select another sub-population sp_j.
3: Generate two hybrid-offspring according to Algorithm 4 (lines 2-5).

simplifying a standard BSO. However, the IBS can be replaced with more advanced individual generation operators so that pure-offspring with higher quality may be generated. In other words, new EMTO algorithms can be proposed by embedding state-of-the-art STO algorithms into the IBS under the framework of BSMTO or BSMTO-II. Moreover, under the proposed BSMTPS framework, new CBS schemes can be proposed to further improve the efficiency of the knowledge transfer by generating hybrid-offspring with higher quality.

IV. EXPERIMENTS AND DISCUSSION

In order to test the performance of the BSMTO and BSMTO-II algorithms, in this section, we use the two proposed algorithms to optimize 9 MTO problems and compare the experimental results with both the classical and state-of-the-art algorithms.

Algorithm 8 Directional Assignment-II (DA-II)

1: **if** the new individual x is a pure-offspring **then**
2: Evaluate x and add it to the corresponding sub-population according to Algorithm 5 (lines 2-3).
3: **else if** $S_i = 0$ && $S_j = 0$ (where i and j are the indexes of the sub-populations to which its parents belong) **then**
4: Evaluate x and add it to the corresponding sub-population according to Algorithm 5 (lines 5-7).
5: **else**
6: Select the task T_m from T_i and T_j whose SPKT flag is 0.
7: Evaluate x on T_m and add it to sp_m.
8: **end if**

A. BENCHMARK INTRODUCTION

The benchmark used in this paper is from the CEC 2017 Evolutionary Multi-Task Optimization Competition [38], which has been widely used by researchers to test the performance of EMTO algorithms. This benchmark contains 9 MTO problems, each of which has two unconstrained single-objective minimization tasks, and each task is represented by a classical objective function with its own dimensionality and search range. According to the degree of intersection of the global optima of the component tasks, these MTO problems are classified into three categories: the complete intersection (CI) problems, the partial intersection (PI) problems, and the no intersection (NI) problems. Further, the problems in each category can be classified into three sub-categories according to the Spearman's rank correlation similarity metric, that is, the high similarity (H) problem, the medium similarity (M) problem, and the low similarity (L) problem. Through the above two classifications, each problem in the benchmark represents a typical MTO problem so that the performance of EMTO algorithms can be fully tested. The details of these MTO problems are given in TABLE 1.

B. EXPERIMENTAL SETUP

We firstly introduce some details in implementing the proposed algorithms and then give the parameter settings. The crossover operator employed by BSMTO is the simulated binary crossover (SBX) [39]. As for the mutation operation, BSMTO employs two mutation operators: the Gaussian mutation [40] and the polynomial mutation [41] which are uniformly randomly selected by a generated offspring. The parental individuals selection scheme employed by BSMTO is the Roulette Wheel Selection [42]. Moreover, since the evolutionary operators may not fully exploit the search space, here we employ the BFGS quasi-Newton method [43] on the newly generated individuals to further improve their quality, which is a widely-used local search scheme in the field of EA to fully exploit the search space based on the gradient information of the fitness function. To be specific, once an offspring has been generated, the BFGS quasi-Newton method will be applied on it with a small possibility p_4. The above operators and schemes are also applied to BSMTO-II.

TABLE 1. Details of the MTO benchmark.

Problem	Property	Task	Function name	Dimension	Search range
Problem 1	CI+H	Task 1	Griewank	50	[-100, 100]
		Task 2	Rastrigin	50	[-50, 50]
Problem 2	CI+M	Task 1	Ackley	50	[-50, 50]
		Task 2	Rastrigin	50	[-50, 50]
Problem 3	CI+L	Task 1	Ackley	50	[-50, 50]
		Task 2	Schwefel	50	[-500, 500]
Problem 4	PI+H	Task 1	Rastrigin	50	[-50, 50]
		Task 2	Sphere	50	[-100, 100]
Problem 5	PI+M	Task 1	Ackley	50	[-50, 50]
		Task 2	Rosenbrock	50	[-50, 50]
Problem 6	PI+L	Task 1	Ackley	50	[-50, 50]
		Task 2	Weierstrass	25	[-0.5, 0.5]
Problem 7	NI+H	Task 1	Rosenbrock	50	[-50, 50]
		Task 2	Rastrigin	50	[-50, 50]
Problem 8	NI+M	Task 1	Griewank	50	[-100, 100]
		Task 2	Weierstrass	50	[-0.5, 0.5]
Problem 9	NI+L	Task 1	Rastrigin	50	[-50, 50]
		Task 2	Schwefel	50	[-500, 500]

The parameters in the proposed algorithms are set as follows. In BSMTO, the sub-population size n is 50, the *gmax* is 1000, the three possibilities p_2, p_3, and p_4 are set as 0.85, 0.8, and 0.02, respectively. The distribution index η in SBX is set as 1.0, and the mutation possibilities for the pure-offspring and the hybrid-offspring are set as 0.07 and 0.02, respectively. The distribution index η_m in polynomial mutation is 5 and the standard deviation in Gaussian mutation is 1. In BSMTO-II, dg is 20, the threshold δ is 0.8, and the other parameters are the same as BSMTO. All the experiments are conducted in MATLAB R2016b run on a PC with a Intel Core i7-6700 3.4GHz CPU and 16GB RAM.

C. EXPERIMENTAL RESULTS AND DISCUSSIONS

1) COMPARISON AND DISCUSSION OF NUMERICAL RESULTS

We firstly test BSMTO-II on the 9 MTO problems. For each problem, we employ the algorithm to optimize each component task and record the minimum value of the objective function. We do 20 independent optimization experiments for each problem and the mean results with the standard deviations (in parentheses) are given in TABLE 2. For comparison, we also list the results of some classical and state-of-the-art EMTO algorithms which are tested on the same benchmark with the same fitness function evaluation cost. Furthermore, we calculate the average rank of each algorithm among all comparative algorithms over all tasks in the 9 problems and emphasize the best result of each task in bold.

From TABLE 2, it can be seen that the proposed BSMTO-II outperforms all the other algorithms on 9 tasks and gets the best mean rank, which demonstrates that the proposed EMTO framework BSMTPS has a competitive potential in coping with MTO problems compared with the multifactorial inheritance employed by MFEA-based algorithms and the other evolutionary multitasking schemes. In addition, benefiting from the brain storm multi-task optimization with the knowledge transfer control scheme, BSMTO-II has a superior EMTO performance.

To further illustrate the effectiveness of the proposed optimization schemes, several additional experiments are designed. Firstly, to demonstrate the superiority of the proposed EMTO framework compared with the state-of-the-art EAs, we combine two DE-based algorithms: LSHADE-SPACMA [44] and LSHADE-cnEpSin [45] which are two winning STO algorithms in CEC2017 to form a hybrid EA called HSHADE. Then we employ HSHADE to optimize all tasks of the benchmark MTO problems under the same fitness evaluation cost. Without loss of generality, for each problem, we use LSHADE-SPACMA to optimize the first task and use LSHADE-cnEpSin to optimize the second task, independently. In other words, HSHADE is employed to cope with each of the MTO problems by optimizing the two component tasks independently with two single-task EAs. On the other hand, the effectiveness of the proposed KTC scheme can be proved by comparing BSMTO with BSMTO-II since only the latter incorporates the scheme. We test the three algorithms: HSHADE, BSMTO, and BSMTO-II on the benchmark MTO problems through conducting the same experiments and compare their optimization results. Moreover, to show the significance of the difference between results obtained by different algorithms, for each task, we conduct two t-tests based on the output results over all independent experiments: one is conducted between the results obtained by HSHADE and BSMTO-II, and the other is conducted between the results obtained by BSMTO and BSMTO-II. The significance level is set as 0.05 and the results of the two t-tests are denoted as h_{bh} and h_{bb}, respectively, where the value 1 means the two sets of data have significant difference and 0 means they have no significant difference. TABLE 3 summarizes the mean optimization results (with standard deviations in parentheses) of each component task in the 9 MTO problems over 20 independent experiments obtained by the three algorithms and the corresponding t-test results, where the best result of each task is emphasized in bold.

From TABLE 3, it can be seen that HSHADE obtains the best result on 5 tasks, while BSMTO-II obtains the best result on 10 tasks. On the other hand, BSMTO-II can obtain better

TABLE 2. Comparison of the mean optimization results obtained by BSMTO-II and some existing EMTO algorithms over 20 independent experiments on the benchmark MTO problems.

Problem	Task	MFEA	MFPSO	MFEARR	SREMTO	MFDE	EMTA	BSMTO-II
Problem 1	Task 1	3.33e-01 (3.55e-02)	2.10e-01 (5.00e-02)	2.96e-01 (NA)	6.28e-03 (7.73e-03)	1.00e-03 (3.05e-03)	1.20e-05 (4.70e-05)	**2.41e-12** (4.33e-12)
	Task 2	1.87e+02 (4.25e+01)	8.11e+00 (3.36e+01)	2.15e+02 (NA)	2.49e+01 (2.67e+01)	2.61e+00 (7.96e+00)	2.16e-02 (8.54e-02)	**0** (0)
Problem 2	Task 1	4.79e+00 (8.19e-01)	6.00e-02 (2.50e-01)	3.90e+00 (NA)	3.39e+00 (7.03e-01)	1.00e-03 (3.00e-03)	2.77e-01 (5.18e-01)	**2.20e-14** (3.90e-15)
	Task 2	2.51e+02 (5.79e+01)	6.26e+00 (2.72e+01)	2.30e+02 (NA)	6.56e+01 (2.71e+01)	**3.00e-03** (1.20e-02)	1.37e+01 (2.46e+01)	5.52e+01 (2.66e+01)
Problem 3	Task 1	2.02e+01 (7.69e-02)	**5.60e+00** (9.30e+00)	2.01e+01 (NA)	2.11e+01 (2.11e-01)	2.12e+01 (4.00e-02)	2.10e+01 (4.14e-01)	1.98e+01 (1.93e-01)
	Task 2	3.70e+03 (6.16e+02)	**2.23e+03** (4.26e+03)	3.57e+03 (NA)	6.50e+03 (9.44e+02)	1.18e+04 (1.58e+03)	6.73e+03 (9.17e+02)	3.41e+03 (6.84e+02)
Problem 4	Task 1	5.78e+02 (1.35e+02)	2.06e+02 (1.45e+02)	4.68e+02 (NA)	2.81e+02 (1.10e+02)	7.83e+01 (1.54e+01)	**3.15e+01** (5.54e+00)	6.99e+01 (1.23e+01)
	Task 2	8.38e+00 (1.50e+00)	3.84e+03 (1.59e+02)	4.47e+00 (NA)	2.60e-06 (3.59e-06)	2.20e-05 (2.90e-05)	1.74e+02 (1.99e+02)	**2.18e-13** (5.29e-14)
Problem 5	Task 1	3.42e+00 (4.83e-01)	3.58e+00 (3.40e-01)	3.34e+00 (NA)	2.22e+00 (6.31e-01)	1.00e-03 (1.00e-03)	1.77e+00 (4.84e-01)	**1.49e-08** (1.95e-09)
	Task 2	6.30e+02 (1.91e+02)	1.24e+02 (1.51e+02)	6.44e+02 (NA)	1.17e+02 (4.16e+01)	6.03e+01 (2.05e+01)	1.57e+02 (5.40e+01)	**1.76e+01** (3.66e+01)
Problem 6	Task 1	1.92e+01 (3.51e+00)	1.00e-02 (5.00e-02)	1.45e+01 (NA)	3.54e+00 (8.55e-01)	4.60e-01 (5.80e-01)	**2.30e-05** (4.80e-05)	1.32e+01 (2.16e+00)
	Task 2	1.93e+01 (4.37e+00)	5.00e-02 (1.50e-01)	1.56e+01 (NA)	3.32e+00 (1.06e+00)	2.20e-01 (4.70e-01)	**2.35e-03** (3.93e-03)	2.43e+01 (2.48e+00)
Problem 7	Task 1	7.97e+02 (3.38e+02)	4.39e+01 (2.86e+01)	1.18e+03 (NA)	1.33e+02 (4.86e+01)	8.93e+01 (4.86e+01)	6.00e+01 (2.96e+01)	**3.89e+00** (1.70e+01)
	Task 2	2.71e+02 (8.10e+01)	3.96e+01 (1.09e+02)	2.69e+02 (NA)	7.01e+01 (2.44e+01)	2.05e+01 (1.54e+01)	2.84e+01 (2.25e+01)	**1.22e+01** (1.84e+01)
Problem 8	Task 1	4.01e-01 (6.81e-02)	4.80e-01 (3.30e-01)	3.28e-01 (NA)	1.06e-02 (1.19e-02)	2.03e-03 (4.28e-03)	8.07e-01 (7.13e-02)	**5.92e-12** (3.08e-12)
	Task 2	2.69e+01 (3.67e+00)	1.21e+01 (2.13e+00)	2.71e+01 (NA)	1.99e+01 (2.16e+00)	2.97e+00 (1.08e+00)	**2.63e+00** (5.15e-01)	2.44e+01 (2.23e+00)
Problem 9	Task 1	6.30e+02 (1.11e+02)	3.33e+02 (1.36e+02)	4.65e+02 (NA)	2.81e+02 (5.14e+01)	9.62e+01 (2.00e+01)	**3.43e+01** (5.84e+00)	6.40e+01 (1.41e+01)
	Task 2	3.85e+03 (4.79e+02)	9.26e+03 (7.13e+03)	**3.49e+03** (NA)	5.88e+03 (9.64e+02)	3.94e+03 (7.31e+02)	7.08e+03 (9.85e+02)	3.60e+03 (5.56e+02)
Mean rank		5.89	3.83	5.28	4.33	3.11	3.33	**2.22**

results than HSHADE on most tasks, and the t-test results show that their difference is significant. It can be concluded that compared with the combination of the state-of-the-art STO EAs, BSMTO-II, which employs the BSMTPS framework, has a superior overall optimization performance. Due to the hybrid-offspring generated through the CBS process, BSMTO-II is able to implement knowledge transfer and coordinate the optimization between different component tasks so that the entire MTO problem can be solved with higher efficiency than solving each task independently. On the other hand, it can be seen that on most tasks, BSMTO-II can outperform BSMTO, while the difference is significant for most of these tasks, which intuitively illustrates the effectiveness of the proposed KTC scheme. With the help of KTC, the SPKT of each task can be detected by BSMTO-II and the knowledge transfer that tends to be harmful for the task optimization is then terminated. Therefore, BSMTO-II is able to implement the knowledge transfer with a higher efficiency compared with BSMTO and shows a superior performance.

2) COMPARISON AND DISCUSSION OF OPTIMIZATION CURVES

To intuitively illustrate the influence of the proposed EMTO framework on the MTO process, in this subsection, we abandon the CBS operation in BSMTO by setting p_2 as 1 and thus interdict the knowledge transfer, then a simplified BSMTO can be generated and we can name it as naive BSMTO (NBSMTO), which is essentially a single-task optimization algorithm. We test NBSMTO and BSMTO-II on the MTO benchmark through conducting the same experiments and compare their optimization curves. That is, for each component task of each of the problems, we record the mean objective function values output by the two algorithms in every generation over all runs, as shown in FIGURE 3.

From FIGURE 3, it can be seen that due to the BS-based knowledge transfer scheme, BSMTO-II has a faster convergence speed in early stages of optimization than that of NBSMTO which only employs the IBS to create new individuals. Based on the BS-based knowledge transfer, two component tasks within a MTO problem are able to share common knowledge with each other so that the information carried by a parental individual from one task can also be used to optimize the other task through the generated hybrid-offspring. In this way, the explorations of the two component tasks are able to cooperate so that the convergence of both tasks can be accelerated. It should also be noted that due to the KTC scheme, BSMTO-II degrades into a NBSMTO after all component tasks meeting their SPKTs. However,

TABLE 3. Comparison of the mean optimization results obtained by HSHADE, BSMTO, and BSMTO-II over 20 independent experiments on each Component Task (T) of the benchmark MTO problems (P).

P	T	HSHADE	BSMTO	BSMTO-II	h_{bh}	h_{bb}
1	1	2.08e-05 (1.19e-05)	**9.52e-13** (2.97e-12)	2.41e-12 (4.33e-12)	1	0
	2	2.16e+02 (2.27e+01)	**0** (0)	**0** (0)	1	0
2	1	7.84e-04 (2.38e-04)	2.40e-11 (6.45e-11)	**2.20e-14** (3.90e-15)	1	0
	2	2.18e+02 (2.21e+01)	1.26e+02 (7.00e+01)	**5.52e+01** (2.66e+01)	1	1
3	1	2.12e+01 (6.54e-02)	2.00e+01 (1.77e-02)	**1.98e+01** (1.93e-01)	1	1
	2	7.91e+03 (8.12e+02)	4.02e+03 (7.39e+02)	**3.41e+03** (6.84e+02)	1	1
4	1	**1.92e+01** (2.83e+01)	1.69e+02 (4.61e+01)	6.99e+01 (1.23e+01)	1	1
	2	8.90e-04 (4.98e-04)	2.55e-13 (4.33e-14)	**2.18e-13** (5.29e-14)	1	1
5	1	7.63e-04 (1.92e-04)	**1.43e-08** (2.15e-09)	1.49e-08 (1.95e-09)	1	0
	2	4.82e+01 (1.68e+00)	**9.35e+00** (2.87e+01)	1.76e+01 (3.66e+01)	1	0
6	1	**7.87e-04** (1.80e-04)	1.82e+01 (7.92e-01)	1.32e+01 (2.16e+00)	1	1
	2	**1.84e-02** (1.75e-02)	2.25e+01 (2.09e+00)	2.43e+01 (2.48e+00)	1	1
7	1	4.66e+01 (8.39e-01)	2.22e+01 (4.43e+01)	**3.89e+00** (1.70e+01)	1	1
	2	2.18e+02 (2.25e+01)	2.83e+01 (3.57e+01)	**1.22e+01** (1.84e+01)	1	0
8	1	3.69e-05 (2.52e-05)	6.29e-12 (3.69e-12)	**5.92e-12** (3.08e-12)	1	0
	2	**2.09e+00** (4.80e-01)	2.58e+01 (2.31e+00)	2.44e+01 (2.23e+00)	1	1
9	1	**2.09e+01** (2.51e+01)	1.61e+02 (2.88e+01)	6.40e+01 (1.41e+01)	1	1
	2	7.95e+03 (8.72e+02)	3.89e+03 (7.76e+02)	**3.60e+03** (5.56e+02)	1	1

benefiting from the efficient exploration driven by CBS in early optimization stages, the convergence speed of BSMTO-II in later optimization stages can also be faster that that of the NBSMTO on some tasks (such as the first task in problems 3 and 6). Moreover, it should be noted that because of the different degrees of optima intersection and similarity between two component tasks in different MTO problems, BSMTO-II has distinct performances on these problems. In general, although BSMTO-II can obtain better optimization results than NBSMTO regardless of the properties of the problems, with the decrease of the degrees of the optima intersection and the similarities between component tasks, the superiority of BSMTO-II over NBSMTO shows a decreasing trend, which shows that the effectiveness of the proposed EMTO scheme can be influenced by the quality of the transferred knowledge.

On the other hand, it should be pointed out that the knowledge transfer scheme employed by the proposed algorithms is a little bit naive, where the transferred knowledge is directly utilized by other tasks through the CBS operation without any processing. Due to the difference of the degrees and types of similarities between the component tasks in different MTO problems, the proposed knowledge transfer scheme may fail in some cases. From FIGURE 3 and TABLE 3, it can be seen that BSMTO-II can not obtain better results with significant difference than NBSMTO on some tasks, and obtain even worse results on some specific problems.

3) KEY PARAMETER SENSITIVITY ANALYSIS

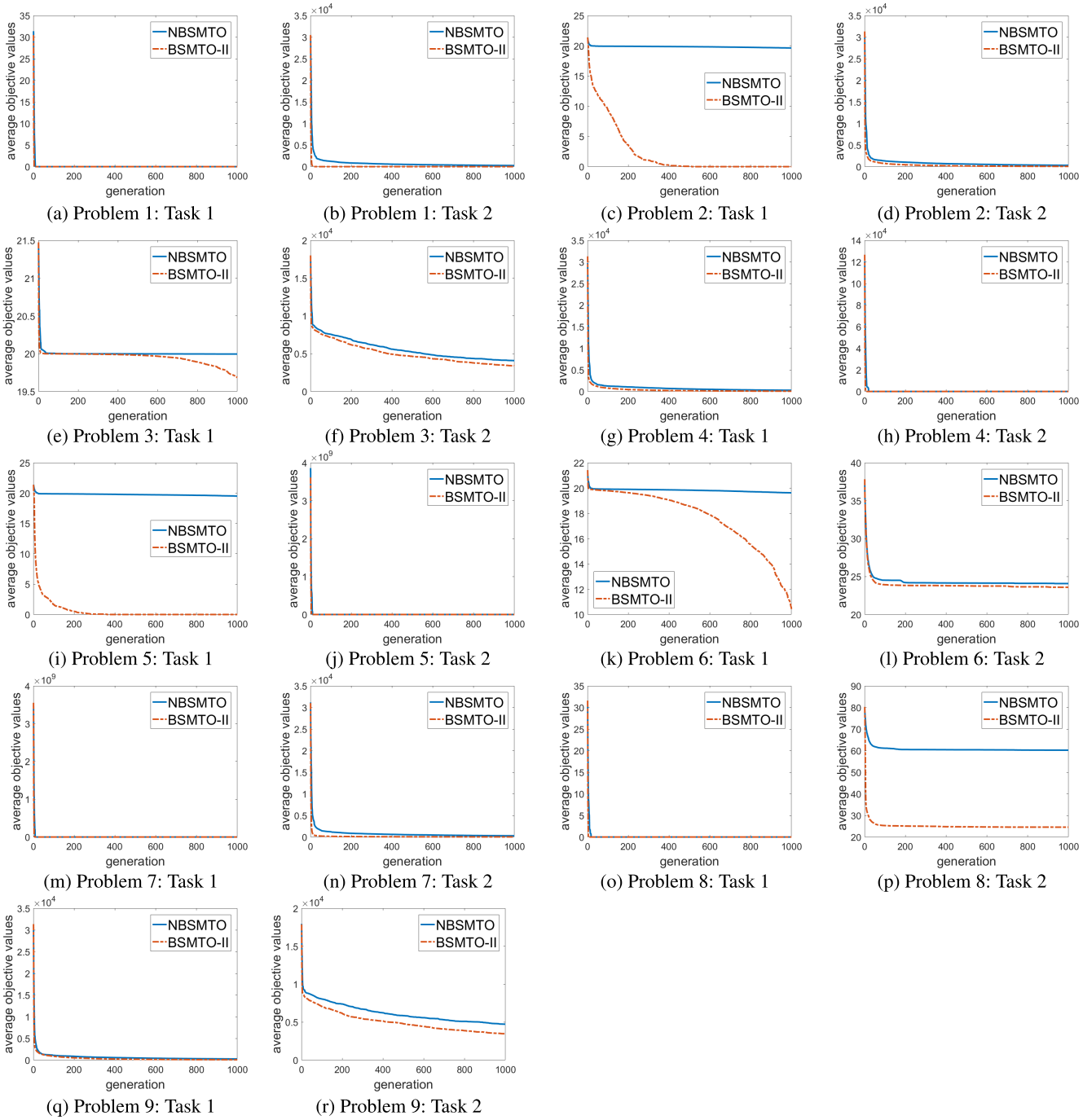
There are several parameters in our proposed algorithms, among which, p_2 is a vital one, which determines the ratio of the number of generated hybrid-offspring over the number of generated pure-offspring in a sub-population. To investigate the influence of p_2 on the performance of the proposed algorithms, we test BSMTO-II on the 9 benchmark MTO problems by setting p_2 as five values of 0.2, 0.4, 0.6, 0.8, and 1.0, respectively. For each value of p_2, we conduct 20 independent optimization experiments and record the mean objective function value of each task in each problem. We also record its mean rank among all values over all tasks and the number of tasks on which it outperforms all the other values (#BP tasks). Experimental results are summarized in TABLE 4, where the best result of each task is emphasized in bold.

It can be seen that the value of p_2 can influence the performance of BSMTO-II to some extents. If p_2 is set as small as 0.2, BSMTO-II has a relative poor performance. In this case, too many hybrid-offspring are generated through CBS, which can disturb the optimization performed within each sub-population. With the increase of p_2, the performance of BSMTO-II starts to improve. When p_2 reaches a certain level, the performance of BSMTO-II becomes optimal and tends to be stable, where the single-task optimization and cross-task knowledge transfer tend to achieve a balance. However, if p_2 continues to increase, the performance of BSMTO-II deteriorates, because the algorithm can hardly conduct the knowledge transfer and gradually degrades to a STO algorithm.

From the above analysis, it can be seen that the parameter p_2 plays an important role in controlling the intensity of the knowledge transfer in the proposed algorithms. Setting a suitable value for p_2 can balance the intra-optimization within each component task and the knowledge transfer across different tasks. According to the experimental results, we suggest that p_2 should be set as a value roughly between 0.6 and 0.9, on which the performance of the proposed algorithms become robust to p_2 and reaches a satisfactory level.

4) DISCUSSION ABOUT THE COMPLEXITY AND RUNTIME OF THE PROPOSED ALGORITHM

It is well known that in practical applications of EAs, most of the computation time is consumed on calculating individuals' fitness values. Therefore, in general, the computation complexity of an EA can be estimated through counting the number of fitness evaluation times in a run cycle. As for BSMTO (also for BSMTO-II), the number of fitness evaluation times *fes* can be evaluated as follows. Assume we have a MTO problem with k component tasks, for each task, we assign a sub-population with size of n. For each generation, nk new



(a) Problem 1: Task 1 (b) Problem 1: Task 2 (c) Problem 2: Task 1 (d) Problem 2: Task 2

(e) Problem 3: Task 1 (f) Problem 3: Task 2 (g) Problem 4: Task 1 (h) Problem 4: Task 2

(i) Problem 5: Task 1 (j) Problem 5: Task 2 (k) Problem 6: Task 1 (l) Problem 6: Task 2

(m) Problem 7: Task 1 (n) Problem 7: Task 2 (o) Problem 8: Task 1 (p) Problem 8: Task 2

(q) Problem 9: Task 1 (r) Problem 9: Task 2

FIGURE 3. Comparison of the mean optimization curves obtained by NBSMTO and BSMTO-II over 20 independent experiments on each component task of the 9 benchmark MTO problems.

individuals (offspring) are generated and evaluated. Consequently, during the *gmax* iterations, the number of fitness evaluation times is approximately equal to the product of n, k, and *gmax*, that is:

$$fes(\text{BSMTO}) \approx nk * gmax \tag{10}$$

Specifically, for the benchmark MTO problems, the problem size (dimension of the jth component task) D_j is also a key parameter to determine the total computation time, since D_j can determine the dimension of an individual and thus determine the computational complexity of both the evolutionary operators and the fitness evaluation. Based on the above analysis, we design several experiments to investigate the influence of the three parameters: D_j, n, and *gmax* on the runtime of the proposed algorithms. Here, we select Problem 3 as the benchmark, where the two component tasks have the same dimension (denoted as D). For each of the three parameters, we set it as several different values while the other parameters

TABLE 4. Comparison of the performance of BSMTO-II with different values of p_2.

Problem	Task	p_2=0.2	p_2=0.4	p_2=0.6	p_2=0.85	p_2=1.0
Problem 1	Task 1	**0**	**0**	**0**	2.41e-12	8.62e-12
	Task 2	**0**	**0**	**0**	**0**	2.71e+02
Problem 2	Task 1	9.04e-11	8.99e-12	1.88e-13	**2.20e-14**	1.96e+01
	Task 2	**0**	**0**	1.71e+01	5.52e+01	2.73e+02
Problem 3	Task 1	2.00e+01	2.00e+01	1.99e+01	**1.98e+01**	2.00e+01
	Task 2	4.94e+03	4.73e+03	4.05e+03	**3.41e+03**	4.34e+03
Problem 4	Task 1	3.28e+02	1.86e+02	7.04e+01	**6.99e+01**	2.88e+02
	Task 2	3.66e-13	2.91e-13	2.24e-13	**2.18e-13**	2.37e-13
Problem 5	Task 1	1.49e-08	1.42e-08	**1.33e-08**	1.49e-08	1.97e+01
	Task 2	2.50e+01	8.63e+00	**4.93e+00**	1.76e+01	2.91e+01
Problem 6	Task 1	1.91e+01	1.90e+01	1.88e+01	**1.32e+01**	1.97e+01
	Task 2	2.15e+01	**2.10e+01**	2.27e+01	2.43e+01	2.48e+01
Problem 7	Task 1	2.64e+01	2.12e+01	4.30e+00	**3.89e+00**	2.94e+01
	Task 2	**1.74e+00**	1.24e+01	1.81e+01	1.22e+01	2.55e+02
Problem 8	Task 1	**4.04e-12**	5.72e-12	4.50e-12	5.92e-12	1.28e-11
	Task 2	**2.35e+01**	2.42e+01	2.46e+01	2.44e+01	5.91e+01
Problem 9	Task 1	2.42e+02	1.02e+02	7.66e+01	**6.40e+01**	2.54e+02
	Task 2	3.92e+03	2.54e+03	**3.46e+03**	3.60e+03	4.71e+03
Mean rank		3.06	2.44	**2.05**	2.17	4.72
#BP tasks		6	4	5	**9**	0

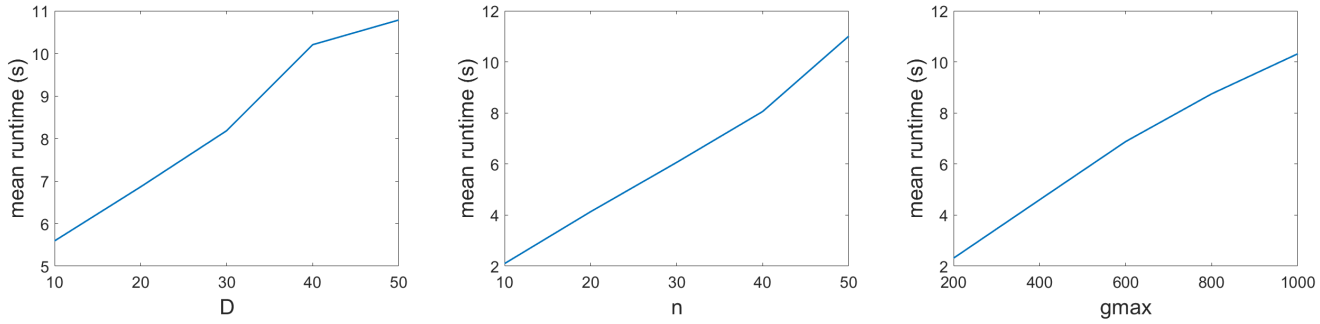


FIGURE 4. Mean runtime of BSMTO on Problem 3 under different values of *D*, *n*, and *gmax*.

remain unchanged and record the mean run time of BSMTO running on the problem for 30 times. The experimental results are shown in FIGURE 4.

It can be seen that with the increases of D, n, and $gmax$, the runtime of BSMTO indeed grows, however, the relation between the runtime with each of the three parameters is approximate linear. In practical use, for a specific MTO problem, the parameter D_j of each component task has a certain value, in this case, the computation time of BSMTO and BSMTO-II will be mainly determined by the other two parameters.

V. CONCLUSION

In this paper, we have proposed two novel EMTO algorithms inspired by the multi-task brainstorming process. Firstly, we have proposed a new EMTO framework named as brain storm multi-task problems solver (BSMTPS), where the optimization process and the knowledge transfer in each component task can be independently observed and controlled. Under this framework, we have proposed the brain storm multi-task optimization (BSMTO) algorithm through designing both the internal and cross-task brain storm operators. Afterwards, we have experimentally investigated the performance of the two types of offspring generated by BSMTO and defined the saturation point of knowledge transfer. Based on the experimental results, we have proposed the knowledge transfer control (KTC) scheme and incorporated it into BSMTO to form the BSMTO-II algorithm. The proposed algorithms have been tested on benchmark MTO problems. Experimental results have showed that BSMTO-II has a superior EMTO performance compared with both classical and state-of-the-art algorithms, while the proposed EMTO framework and the KTC scheme are both effective. Furthermore, we have discussed about the influence of the key parameter setting on the algorithms' performance and the computational complexity of the proposed method.

Due to the limitations of the proposed algorithms, in future work, the knowledge transfer scheme (generation of the hybrid-offspring) can be further improved. For example, the crossover operator adopted by CBS can be modified to achieve knowledge sharing between two tasks without sufficient high similarity. Meanwhile, new CBS operations can be designed by sharing optimization knowledge among three or more component tasks. Moreover, under the proposed BSMTPS framework, new EMTO algorithms can be proposed by modifying or replacing the IBS operation with more powerful STO operators. On the other hand, it is meaningful to employ the proposed algorithms to solve some

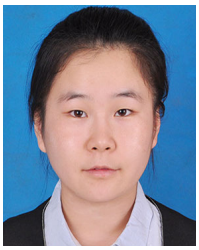
real-world MTO problems such as the TSP problems, the vehicle routing problems (VRP), and the swarm robots coordination problems.

REFERENCES

[1] J. Fliege and B. F. Svaiter, "Steepest descent methods for multicriteria optimization," *Math. Methods Oper. Res.*, vol. 51, no. 3, pp. 479–494, Aug. 2000.
[2] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1998.
[3] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 82–102, Jul. 1999.
[4] K. Fleetwood, "An introduction to differential evolution," in *Proc. 26th Math. Statist. Complex Syst. (MASCOS) One Day Symp.*, Brisbane, QLD, Australia, Nov. 2004, pp. 785–791.
[5] R. C. Eberhart and Y. Shi, "Particle swarm optimization: Developments, applications and resources," in *Proc. Congr. Evol. Comput.*, vol. 1, May 2001, pp. 81–86.
[6] Y. Shi, "Brain storm optimization algorithm," in *Proc. Int. Conf. Swarm Intell.* Cham, Switzerland: Springer, 2011, pp. 303–309.
[7] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, vol. 16. Hoboken, NJ, USA: Wiley, 2001.
[8] J. Sun, Z. Miao, D. Gong, X.-J. Zeng, J. Li, and G. Wang, "Interval multiobjective optimization with memetic algorithms," *IEEE Trans. Cybern.*, vol. 50, no. 8, pp. 3444–3457, Aug. 2020.
[9] 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.
[10] Y. Yuan, Y.-S. Ong, A. Gupta, P. S. Tan, and H. Xu, "Evolutionary multitasking in permutation-based combinatorial optimization problems: Realization with TSP, QAP, LOP, and JSP," in *Proc. IEEE Region 10 Conf. (TENCON)*, Nov. 2016, pp. 3157–3164.
[11] L. Sampath, A. Gupta, Y.-S. Ong, and H. Gooi, "Evolutionary multitasking to support optimal power flow under rapid load variations," *Southern Power Syst. Technol. China*, vol. 11, no. 10, 2017.
[12] R. Sagarna and Y.-S. Ong, "Concurrently searching branches in software tests generation through multitask evolution," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Dec. 2016, pp. 1–8.
[13] L. Bao, Y. Qi, M. Shen, X. Bu, J. Yu, Q. Li, and P. Chen, "An evolutionary multitasking algorithm for cloud computing service composition," in *Proc. World Congr. Services* Cham, Switzerland: Springer, 2018, pp. 130–144.
[14] L. Zhou, L. Feng, J. Zhong, Y.-S. Ong, Z. Zhu, and E. Sha, "Evolutionary multitasking in combinatorial search spaces: A case study in capacitated vehicle routing problem," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Dec. 2016, pp. 1–8.
[15] X. Zheng, A. K. Qin, M. Gong, and D. Zhou, "Self-regulated evolutionary multitask optimization," *IEEE Trans. Evol. Comput.*, vol. 24, no. 1, pp. 16–28, Feb. 2020.
[16] L. Feng, W. Zhou, L. Zhou, S. W. Jiang, J. H. Zhong, B. S. Da, Z. X. Zhu, and Y. Wang, "An empirical study of multifactorial PSO and multifactorial DE," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2017, pp. 921–928.
[17] X. Zheng, Y. Lei, A. K. Qin, D. Zhou, J. Shi, and M. Gong, "Differential evolutionary multi-task optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2019, pp. 1914–1921.
[18] Y. Yu, A. Zhu, Z. Zhu, Q. Lin, J. Yin, and X. Ma, "Multifactorial differential evolution with opposition-based learning for multi-tasking optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2019, pp. 1898–1905.
[19] 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. (CIS)*, Dec. 2017, pp. 197–201.
[20] J. Tang, Y. Chen, Z. Deng, Y. Xiang, and C. Paul Joy, "A group-based approach to improve multifactorial evolutionary algorithm," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 3870–3876.
[21] K. K. Bali, A. Gupta, L. Feng, Y. S. Ong, and T. P. Siew, "Linearized domain adaptation in evolutionary multitasking," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2017, pp. 1295–1302.
[22] Y.-W. Wen and C.-K. Ting, "Parting ways and reallocating resources in evolutionary multitasking," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2017, pp. 2404–2411.
[23] L. Feng, L. Zhou, J. Zhong, A. Gupta, Y.-S. Ong, K.-C. Tan, and A. K. Qin, "Evolutionary multitasking via explicit autoencoding," *IEEE Trans. Cybern.*, vol. 49, no. 9, pp. 3457–3470, Sep. 2019.
[24] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, no. 12, pp. 3371–3408, Dec. 2010.
[25] J. Rice, C. R. Cloninger, and T. Reich, "Multifactorial inheritance with cultural transmission and assortative mating. i. description and basic properties of the unitary models," *Amer. J. Hum. Genet.*, vol. 30, no. 6, p. 618, 1978.
[26] C. R. Cloninger, J. Rice, and T. Reich, "Multifactorial inheritance with cultural transmission and assortative mating. ii. a general model of combined polygenic and cultural inheritance," *Amer. J. Hum. Genet.*, vol. 31, no. 2, p. 176, 1979.
[27] R. Smith, *The 7 Levels Change*, 2nd ed. Littleton, MA, USA: Tapeslry Press, 2002.
[28] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, Apr. 1997.
[29] M. S. Kiran, H. Hakli, M. Gunduz, and H. Uguz, "Artificial bee colony algorithm with variable search strategy for continuous optimization," *Inf. Sci.*, vol. 300, pp. 140–157, Apr. 2015.
[30] Z.-H. Zhan, J. Zhang, Y.-H. Shi, and H.-L. Liu, "A modified brain storm optimization," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2012, pp. 1–8.
[31] J. Chen, S. Cheng, Y. Chen, Y. Xie, and Y. Shi, "Enhanced brain storm optimization algorithm for wireless sensor networks deployment," in *Proc. Int. Conf. Swarm Intell.* Cham, Switzerland: Springer, 2015, pp. 373–381.
[32] S. Cheng, Q. Qin, J. Chen, G. G. Wang, and Y. Shi, "Brain storm optimization in objective space algorithm for multimodal optimization problems," in *Proc. Int. Conf. Swarm Intell. (ICSI)*. Cham, Switzerland: Springer, 2016, pp. 469–478.
[33] Y. Shi, J. Xue, and Y. Wu, "Multi-objective optimization based on brain storm optimization algorithm," *Int. J. Swarm Intell. Res.*, vol. 4, no. 3, pp. 1–21, Jul. 2013.
[34] C. Sun, H. Duan, and Y. Shi, "Optimal satellite formation reconfiguration based on closed-loop brain storm optimization," *IEEE Comput. Intell. Mag.*, vol. 8, no. 4, pp. 39–51, Nov. 2013.
[35] J. Chen, S. Cheng, Y. Chen, Y. Xie, and Y. Shi, "Enhanced brain storm optimization algorithm for wireless sensor networks deployment," in *Proc. Int. Conf. Swarm Intell.* Cham, Switzerland: Springer, 2015, pp. 373–381.
[36] H. Duan, S. Li, and Y. Shi, "Predator—Prey brain storm optimization for DC brushless motor," *IEEE Trans. Magn.*, vol. 49, no. 10, pp. 5336–5340, May 2013.
[37] S. Biswas, M. A. Eita, S. Das, and A. V. Vasilakos, "Evaluating the performance of group counseling optimizer on CEC 2014 problems for computational expensive optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2014, pp. 1076–1083.
[38] B. Da, Y.-S. Ong, L. Feng, A. K. Qin, A. Gupta, Z. Zhu, C.-K. Ting, K. Tang, and X. Yao, "Evolutionary multitasking for single-objective continuous optimization: Benchmark problems, performance metric, and baseline results," 2017, *arXiv:1706.03470*. [Online]. Available: http://arxiv.org/abs/1706.03470
[39] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Syst.*, vol. 9, no. 2, pp. 115–148, 1995.
[40] R. Hinterding, "Gaussian mutation and self-adaption for numeric genetic algorithms," in *Proc. IEEE Int. Conf. Evol. Comput.*, vol. 1, Nov. 1995, p. 384.
[41] K. Deb and D. Deb, "Analysing mutation schemes for real-parameter genetic algorithms," *Int. J. Artif. Intell. Soft Comput.*, vol. 4, no. 1, pp. 1–28, 2014.
[42] R. Sivaraj and T. Ravichandran, "A review of selection methods in genetic algorithm," *Int. J. Eng. Sci. Technol.*, vol. 3, no. 5, pp. 3792–3797, 2011.
[43] J. Nocedal, "Updating quasi-Newton matrices with limited storage," *Math. Comput.*, vol. 35, no. 151, pp. 773–782, 1980.
[44] A. W. Mohamed, A. A. Hadi, A. M. Fattouh, and K. M. Jambi, "LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2017, pp. 145–152.
[45] N. H. Awad, M. Z. Ali, and P. N. Suganthan, "Ensemble sinusoidal differential covariance matrix adaptation with Euclidean neighborhood for solving CEC2017 benchmark problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2017, pp. 372–379.

CHAO LYU received the B.S. degree from the China University of Mining and Technology, Xuzhou, Jiangsu, China, in 2014, and the M.Eng. degree from Lanzhou University, Lanzhou, Gansu, China, in 2017. He is currently pursuing the Ph.D. degree with the Harbin Institute of Technology, China, and the Southern University of Science and Technology, Shenzhen, Guangdong, China. His research interests include complex networks, evolutionary computation, swarm intelligence, and their applications.

YUHUI SHI (Fellow, IEEE) received the Ph.D. degree in electronic engineering from Southeast University, Nanjing, China, in 1992. From 1998 to 2007, he was with Electronic Data Systems Corporation, Indianapolis, IN, USA, as an Applied Specialist. From 2008 to August 2016, he was a Professor with the Department of Electrical and Electronic Engineering, Xi'an Jiaotong-Liverpool University, Suzhou, Jiangsu, China. Since September 2016, he has been the Chair Professor with the Department of Computer Science and Technology, Southern University of Science and Technology (SUS Tech), Shenzhen, Guangdong, China. He has extensive knowledge on innovation and creative problem-solving skills. He has coauthored the book *Swarm Intelligence* (with Dr. James Kennedy and Prof. Russell Eberhart) and another book entitled *Computational Intelligence: Concept to Implementation* (with Prof. Russell Eberhart). His main research interests include computational intelligence techniques, including swarm intelligence, and their applications.



LIJUN SUN received the B.S. and M.Eng. degrees from Lanzhou University, Lanzhou, Gansu, China, in 2013 and 2016, respectively. She is currently pursuing the Ph.D. degree with the University of Technology Sydney, Australia, and the Southern University of Science and Technology, Shenzhen, Guangdong, China. Her research interests include swarm intelligence and swarm robots coordination.