An Empirical Study of Multifactorial PSO and Multifactorial DE

L. Feng[1], W. Zhou[1], L. Zhou[1], S. W. Jiang[2]
J. H. Zhong[3], B. S. Da[4], Z. X. Zhu[5], Y. Wang[6]

College of Computer Science, Chongqing University, China[1]
Singapore Institute of Manufacturing Technology (SIMTech), Singapore[2].
School of Computer Science and Engineering, South China University of Technology[3]
School of Computer Engineering, Nanyang Technological University, Singapore[4]
College of Computer Science and Software Engineering, Shenzhen University[5]
Beijing DiDi Infinity Tech. & Dev. Co,. Ltd.[6]

***Abstract*—Recently, the notion of *Multifactorial Optimization* (*MFO*) has emerged as a promising approach for evolutionary multi-tasking by automatically exploiting the latent synergies between optimization problems, simply through solving them together in an unified representation space [1]. It aims to improve convergence characteristics across multiple optimization problems at once by seamlessly transferring knowledge between them. In [1], the efficacy of *MFO* has been studied by a specific mode of knowledge transfer in the form of *implicit genetic transfer* through chromosomal crossover. Here we further explore the generality of *MFO* when diverse population based search mechanisms are employed. In particular, in this paper, we present the first attempt to conduct *MFO* with the popular particle swarm optimization and differential evolution search. Two specific multi-tasking paradigms, namely multifactorial particle swarm optimization (*MFPSO*) and multifactorial differential evolution (*MFDE*) are proposed. To evaluate the performance of *MFPSO* and *MFDE*, comprehensive empirical studies on 9 single objective *MFO* benchmark problems are provided.**

I. Introduction

It is well established that problems seldom exist in isolation, and related problems often contain useful information that can be utilized to improve problem-solving efficiency. With the fast growth of machine learning techniques, the idea of exploiting commonalities and differences across tasks for performing efficient and effective learning has been widely studied in the field of multi-task learning. Many successful multi-task learning paradigms have been proposed in the literature for handling different learning tasks. For example, Jinbo *et al.* presented a family of multi-task learning algorithms for collaborative computer aided diagnosis which aims to diagnose multiple clinically-related abnormal structures from medical images [2]. Kang *et al.* introduced a multi-task learning algorithm which can detect the relatedness among tasks in a shared feature representations, for hand-written digit recognition [3]. Further, Wang *et al.* proposed a multi-task learning framework, called Boosted MTL, for face verification with limited training data [4]. Xu *et al.* provided a survey of multi-task learning in bioinformatics [5].

In spite of the accomplishments made in multi-task learning, the attempts to conduct multi-tasking for efficient optimization in computational intelligence, population based search in particular, have to date received far less attention. A common practice in population based search is to solve only a single optimization problem at a time [6–8], which is unproductive in today's cloud computing industry wherein multiple optimization tasks can be received from multiple users at the same time. Taking this cue, recently, Gupta *et al.* proposed a multifactorial optimization (*MFO*) paradigm for evolutionary multi-tasking [1, 9]. In contrast to traditional optimization search, *MFO* contends to conduct evolutionary search on multiple search spaces corresponding to different tasks or optimization problems concurrently. The efficacy of performing multifactorial optimization has been preliminarily studied by designing a multifactorial evolutionary algorithm (*MFEA*) in [1].

In this paper, we further contribute to multifactorial optimization by embarking a study on multi-tasking with particle swarm optimization (PSO) and differential evolution (DE), namely *MFPSO* and *MFDE*, respectively. The aim of this paper is to explore the generality of the *MFO* paradigm proposed in [1] when diverse population based search mechanisms are employed. To the best of our knowledge, this is the first attempt in the literature to conducting multi-task optimization with *PSO* and *DE*. In particular, by exploiting the search mechanisms of *PSO* and *DE*, we first present the designs of mating approaches for multi-task optimization in *MFPSO* and *MFDE*, respectively. Subsequently, to evaluate the performance of both *MFPSO* and *MFDE*, comprehensive empirical studies are conducted with 9 single objective *MFO* benchmarks reported in [10].

The rest of this paper is organized as follows. Section II presents the brief introduction of the concept of multifactorial optimization. The subsections describing the *PSO* and *DE* for search are provided therein. The proposed *MFPSO* and *MFDE* for multi-task optimization are detailed in Section III. Section IV contains empirical studies to verify the efficacy of the two proposed search paradigms for multi-task optimization. Lastly, concluding remarks are drawn in Section V.

978-1-5090-4601-0/17/$31.00 ©2017 IEEE

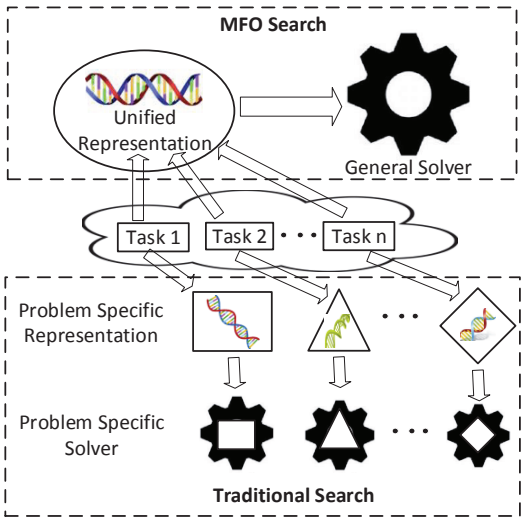


Fig. 1. Illustration of comparison between the multifactorial optimization and the traditional search method.

II. Preliminary

This section begins with a brief introduction to the multifactorial optimization (*MFO*) paradigm proposed in [1]. The background of the particle swarm optimization (*PSO*) and the differential evolution (*DE*) search technique are also presented in this section.

A. Multifactorial Optimization

In [1], multifactorial optimization (*MFO*) has been defined as an evolutionary multi-tasking paradigm that builds on the implicit parallelism of population-based search with the aim of finding the optimal solutions for multiple task simultaneously. As depicted in Fig. 1, to solve multiple tasks, the traditional search methods (e.g., genetic algorithm, particle swarm optimization, etc.) requires different solvers with unique problem specific representation for each task, while *MFO* employs an unified problem representation and solves multiple tasks within one single solver. In *MFO*, each task is treated as an additional factor influencing the evolution of a single population of individuals.

To evaluate individuals in a multi-tasking scenario, the following properties for each individual have been defined in [1].

- *Factorial Cost:* The factorial cost f_p of an individual p denotes its fitness or objective value on a particular task T_i. For K tasks, there will be a vector with length K, in which each dimension gives the fitness of p on the corresponding task.
- *Factorial Rank:* The factorial rank r_p simply denotes the index of individual p in the list of population members sorted in ascending order with respect to their factorial costs on one specific task.
- *Scalar Fitness:* The scalar fitness φ_p of an individual p is defined based on its best rank over all tasks, which is given by $\varphi_p = \frac{1}{\min_{j\in\{1,\ldots,K\}} r_p^j}$.
- *Skill Factor:* The skill factor τ_p of individual p denotes the task, amongst all other tasks in *MFO*, on which p is most effective, i.e., $\tau_p = argmin\{r_p^j\}$, where $j \in \{1,\ldots,K\}$.

As individuals in *MFO* are encoded in the unified search space of all the tasks, the *Factorial Cost* of each individual is obtained through a problem dependent decoding process. Further, with the properties given above, performance comparison between solutions in *MFO* can be carried out based on scalar fitness φ. In particular, individual p_a is considered to dominate p_b in a multifactorial sense simply if $\varphi_{p_a} > \varphi_{p_b}$. Therefore, suppose all the tasks are minimization problems, the definition of multifactorial optimality is given as [1]:

> *Multifactorial Optimality*: An individual p^*, with a list of fitness or objective value $\{f_1^*, f_2^*, \ldots, f_K^*\}$ on K tasks, is considered optimum in a multifactorial sense if and only if $\exists j \in [1,\ldots,K]$, such that $f_j^* \leq f(\mathbf{x}_j)$, where $\mathbf{x}_j$ denotes any feasible solution in the search space of task T_j.

B. Particle Swarm Optimization

Particle swarm optimization (*PSO*) is a nature-inspired swarm intelligence algorithm proposed by Kennedy and Eberhart in 1995 [11]. In *PSO*, each single individual solution is considered as a particle in the swarm. The swarm changes its search direction based on the environmental impacts or the leader particles command. According to this concept, the best solution of an interested problem can be found by changing the direction of the swarm.

In particular, for solving a given optimization problem, each particle is often represented by a position in the problem search space, which is given by a vector $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{id})$, where d is the dimensionality of the search space. All of the particles are equipped with the velocity vectors represented by $\mathbf{v}_i = (v_{i1}, v_{i2}, \ldots, v_{id})$ to direct their movements. Further, each particle has a fitness value that is evaluated by the fitness function to be optimized. The initial swarm is typically generated randomly over the problem search space. At every iteration, each particle keeps track of its coordinates in the problem space associated with the best fitness value and the particle is updated by following the personal best *pbest* and the best position obtained by the population so far, i.e., *gbest* [11]. The update of velocity and position of the ith particle are mathematically defined as:

$$v_{id} = w \times v_{id} + c_1 \times r_1 \times (p_{id} - x_{id}) + c_2 \times r_2 \times (p_{gd} - x_{id}) \quad (1)$$

$$x_{id} = x_{id} + v_{id} \quad (2)$$

where v_{id} and x_{id} denote the dth dimension of the velocity vector and position vector of particle i, respectively. p_{id} represents the dth dimension of the best solution found by the ith particle, while p_{gd} gives the dth dimension of the best particle found by the swarm so far. w is the inertia weight, which decreases linearly during a run. c_1 and c_2 are positive constants called acceleration coefficients, which are used to

balance the impact of p_{id} and p_{gd} in changing particles' velocities. The algorithm either stops when a predefined fitness value is obtained or when maximum number of iterations are met.

C. Differential Evolution

Differential Evolution (*DE*) is another popular population based optimization algorithm that originally proposed by Price and Storn in 1995 [12] to solve continuous optimization problems. The basic idea of *DE* is based on composing a temporary population by exploiting the individual differences of the current population. In particular, *mutation* and *crossover* are the two key components in *DE*. These two operators produces a new candidate solution component based on the weighted difference between two randomly selected population individuals that is added to a third individual. This results in the perturbation of the population individuals relative to the spread of the entire population. Further, solution selection then kicks in for self-organizing the sample space into areas of interest.

Further, in the literature, five mutation strategies have been commonly used in *DE* [13], which are given by:

$$DE/rand/1 : \mathbf{v}_i^g = \mathbf{x}_{r1}^g + F \times (\mathbf{x}_{r2}^g - \mathbf{x}_{r3}^g) \quad (3)$$

$$DE/best/1 : \mathbf{v}_i^g = \mathbf{x}_{best}^g + F \times (\mathbf{x}_{r1}^g - \mathbf{x}_{r2}^g) \quad (4)$$

$$DE/current-best/2 : \mathbf{v}_i^g = x_i^g + F \times (\mathbf{x}_{best}^g - \mathbf{x}_i^g + \mathbf{x}_{r2}^g - \mathbf{x}_{r3}^g) \quad (5)$$

$$DE/best/2 : \mathbf{v}_i^g = \mathbf{x}_{best}^g + F \times (\mathbf{x}_{r1}^g - \mathbf{x}_{r2}^g + \mathbf{x}_{r3}^g - \mathbf{x}_{r4}^g) \quad (6)$$

$$DE/rand/2 : \mathbf{v}_i^g = x_{r1}^g + F \times (\mathbf{x}_{r2}^g - \mathbf{x}_{r3}^g + \mathbf{x}_{r4}^g - \mathbf{x}_{r5}^g) \quad (7)$$

where $\mathbf{v}_i^g = \{v_1, v_2, \ldots, v_d\}$ (d is the dimension of the problem encountered) denotes the mutant vector generated for each individual $\mathbf{x}_i$ at generation g. $r1$, $r2$, $r3$, $r4$, $r5$ are random and mutually exclusive integers generated within the range $[1, NP]$. NP is the population size. F is the scaling factor which controls the amplitude of the difference vector. $\mathbf{x}_{best}$ gives the solution with the best fitness found so far.

For crossover, the following operator has been defined:

$$\mathbf{u}_i^{g+1} = \begin{cases} \mathbf{v}_i^g & if(rand \leq CR) or (j = rand_j) \\ \mathbf{x}_i^g & otherwise \end{cases} \quad (8)$$

where $rand$ is uniformly drawn from $[0, 1]$, and $rand_j$ is randomly selected from $[1, \ldots, d]$ which is used to ensure that at least one dimension of trial vector is changed. CR is the crossover rate that takes the value in the range of $[0, 1]$. It helps the *DE* in controlling the portion of offspring solutions that are copied from the mutant vectors.

III. Multifactorial Optimization with PSO and DE

In this section, we first recall the multifactorial evolutionary algorithm (*MFEA*) proposed in [1]. Further, the proposed multifactorial optimization paradigms with *PSO* and *DE*, i.e., *MFPSO* and *MFDE*, are detailed.

In particular, the *MFEA* is inspired by the bio-cultural models of multifactorial inheritance. In contrast to traditional evolutionary algorithms, *assortative mating* and *vertical cultural transmission* are the two key components for conducting multi-tasking with implicit knowledge transfer among individuals. As outlined in Fig. 2, the work-flow of *MFEA* can be summarized in Alg. 1. The *MFEA* starts with a population of individuals generated in an unified search space, e.g., random key representation [1], defined on all the tasks to be solved. The *assortative mating* serves as the novel cultural reproduction operator, which plays the key role in implicit knowledge transfer among tasks for offspring generation. Further, the *vertical cultural transmission* governs the fitness evaluation by taking the cultural *skill factor* of individual into consideration. For more detailed introduction of *MFEA* and the corresponding operators, the reader may refer to [1].

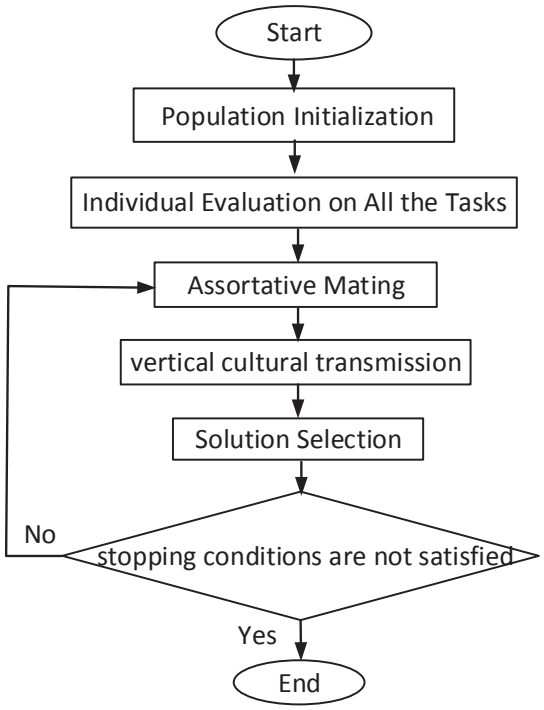


Fig. 2. The outline of *MFEA*

In contrast to genetic algorithm, *PSO* and *DE* possess unique solution reproduction or update operations. To conduct multifactorial optimization with *PSO* and *DE*, new *assortative mating* schemes are required. In what follows, we present the detailed designs of *assortative mating* in *MFPSO* and *MFDE*, respectively. With respect to the other components, such as *unified individual representation*, *vertical cultural transmission*, etc., consistency to the original *MFEA* [1] is maintained in *MFPSO* and *MFDE* as this study takes a focus on differences in the reproduction or update mechanisms of *PSO* and *DE*.

A. Assortative Mating for MFPSO

The pseudo code of the *assortative mating* in *MFPSO* is given in Alg. 2. To update the velocity of particle i in the population, a random value *rand* is first generated in the

1 **Generate** an initial population of individuals with size NP.
2 **Evaluate** all individuals on every tasks in the multi-tasking environment, and obtain the *skill factor* of each individual accordingly.
3 **while** *stopping conditions are not satisfied* **do**
4 **Perform** *assortative mating* on the current population to generate the offsprings.
5 **Evaluate** offspring individuals on selected tasks based on *vertical cultural transmission*.
6 **Update** the *scalar fitness* and *skill factor* of all the parent and offspring individuals.
7 **Select** the fittest NP individuals from both parent and offspring solutions to survive for the next generation.

Algorithm 1: Pseudo code of *MFEA*.

range of $[0, 1]$ (see line 1 in Alg. 2). If *rand* is less than the random mating probability *rmp* (user defined value to control the probability of knowledge transfer across solutions with different *skill factors*), the global best solution (i.e., p'_{gd} in line 3 of Alg. 2) which has different *skill factor* from particle i, is used in the velocity update of particle i. Otherwise, the velocity update will proceed as usual given in Eq. 1 (see line 5 in Alg. 2).

1 **Generate** a random number *rand* between 0 and 1.
2 **if** $rand < rmp$ **then**
3 $v_{id} = w \times v_{id} + c_1 \times r_1 \times (p_{id} - x_{id}) + c_2 \times r_2 \times (p_{gd} - x_{id}) + c_3 \times r_3 \times (p'_{gd} - x_{id})$
4 **else**
5 $v_{id} = w \times v_{id} + c_1 \times r_1 \times (p_{id} - x_{id}) + c_2 \times r_2 \times (p_{gd} - x_{id})$

Algorithm 2: Pseudo code of the *assortative mating* in *MFPSO*.

B. Assortative Mating for MFDE

Further, the *assortative mating* of *MFDE* is summarized in Alg. 3. For easy of understanding, here we consider the $DE/rand/1$ mutation strategy in the original *DE* as discussed in Section II-C.

1 **Generate** a random number *rand* between 0 and 1.
2 **if** $rand < rmp$ **then**
3 $\mathbf{v}_i^g = \mathbf{x}_{r1}^g + F \times (\mathbf{x}_{r2}^{'g} - \mathbf{x}_{r3}^{'g})$
4 **else**
5 $\mathbf{v}_i^g = \mathbf{x}_{r1}^g + F \times (\mathbf{x}_{r2}^g - \mathbf{x}_{r3}^g)$

Algorithm 3: Pseudo code of the *assortative mating* in *MFDE*.

In particular, like the *assortative mating* in the *MFPSO*, a random value *rand* is first generated in the range of $[0, 1]$ (see line 1 in Alg. 3). If *rand* is less than the predefined mating probability value *rmp*, two random solutions, i.e., $\mathbf{x}_{r2}^{'g}$ and $\mathbf{x}_{r3}^{'g}$, which have different *skill factor* from $\mathbf{x}_i^g$, are selected to generate the trial vector $\mathbf{v}_i^g$. Please note that $\mathbf{x}_{r1}^g$ in line 3 of Alg. 3 is a randomly selected individual that shares common *skill factor* with $\mathbf{x}_i^g$. Otherwise, the generation of trial vector $\mathbf{v}_i^g$ will proceed like the original $DE/rand/1$ strategy, in which $\mathbf{x}_{r2}^g$ and $\mathbf{x}_{r3}^g$ (see line 5 in Alg. 3) are two randomly selected individuals that share common *skill factor* with $\mathbf{x}_i^g$.

IV. Empirical Study

In this section, a comprehensive empirical study is conducted to study the performance of *MFPSO* and *MFDE*. In particular, we first give the details of the experimental setup, which is followed by the analysis and discussions of the empirical results obtained.

A. Experimental Setup

According to the evolutionary multi-task optimization technical report [10], 9 single objective *MFO* benchmark problems are considered in this study. A summary of the *MFO* problem properties is given in Table I. The *MFO* benchmark set comprises of test problems of diverse properties, that are built by pairing classical single objective functions, such as *Griewank*, *Rastrigin*, and *Schwefel*, etc., with the consideration of the function landscape type, degree of intersection between tasks, and inter-task similarity[1]. For more details of the single objective *MFO* benchmark set, the reader is refer to [10].

The configurations of the search operators and parameters in *MFPSO* and *MFDE* used in this study are summarized as follows:

1) Population size: Population size NP is configured as 100 in both *MFPSO* and *MFDE*.
2) Maximum number of generations: Max_FES = 1000
3) Independent number of runs: $runs = 20$
4) Parameter settings in *MFPSO* and *MFDE* [1, 12, 14]:
   - w in *MFPSO*: decreases linearly from 0.9 to 0.4;
   - c_1, c_2, and c_3 in *MFPSO*: $c_1 = c_2 = c_3 = 0.2$;
   - rmp in *MFPSO* and *MFDE*: $rmp = 0.3$;
   - F and CR in *MFDE*: $F = 0.5$, $CR = 0.9$;

Further, the baselines considered for comparison in this study include the traditional single task *PSO* and *DE* algorithm. For fair comparison, *PSO* and *DE* share the same parameter setting and search operators of *MFPSO* and *MFDE*, respectively. Lastly, the empirical studies presented in this paper are conducted under window10 OS using the computer with a 3.2 GHz Intel Corei5 processor and 8 GB RAM.

B. Results and Discussions

The performance of *MFPSO* and *PSO* on the 9 single objective *MFO* benchmark problems are summarized in Table II. In the table, the averaged fitness values obtained by *MFPSO* and *PSO* over 20 independent runs are presented. Values in

[1]The inter-task similarity is defined based on the Pearson correlation [1]. A higher value denotes greater similarity between tasks.

TABLE I
SUMMARY OF PROPERTIES OF PROBLEM PAIRS FOR EVOLUTIONARY MULTITASKING

Category	Task	Landscape	Degree of intersection	Inter-task similarity
CI+HS	Griewank (T_1)	multimodal, nonseparable	Complete intersection	1.0000
	Rastrigin (T_2)	multimodal, nonseparable		
CI+MS	Ackley (T_1)	multimodal, nonseparable	Complete intersection	0.2261
	Rastrigin (T_2)	multimodal, nonseparable		
CI+LS	Ackley (T_1)	multimodal, nonseparable	Complete intersection	0.0002
	Schwefel (T_2)	multimodal, separable		
PI+HS	Rastrigin (T_1)	multimodal, nonseparable	Partial intersection	0.8670
	Sphere (T_2)	unimodal, separable		
PI+MS	Ackley (T_1)	multimodal, nonseparable	Partial intersection	0.2154
	Rosenbrock (T_2)	multimodal, nonseparable		
PI+LS	Ackley (T_1)	multimodal, nonseparable	Partial intersection	0.0725
	Weierstrass (T_2)	multimodal, nonseparable		
NI+HS	Rosenbrock (T_1)	multimodal, nonseparable	No intersection	0.9434
	Rastrigin (T_2)	multimodal, nonseparable		
NI+MS	Griewank (T_1)	multimodal, nonseparable	No intersection	0.3669
	Weierstrass (T_2)	multimodal, nonseparable		
NI+LS	Rastrigin (T_1)	multimodal, nonseparable	No intersection	0.0016
	Schwefel (T_2)	multimodal, separable		

the bracket give the corresponding standard deviation. T_1 and T_2 denote the two tasks contained in the *MFO* benchmark, respectively.

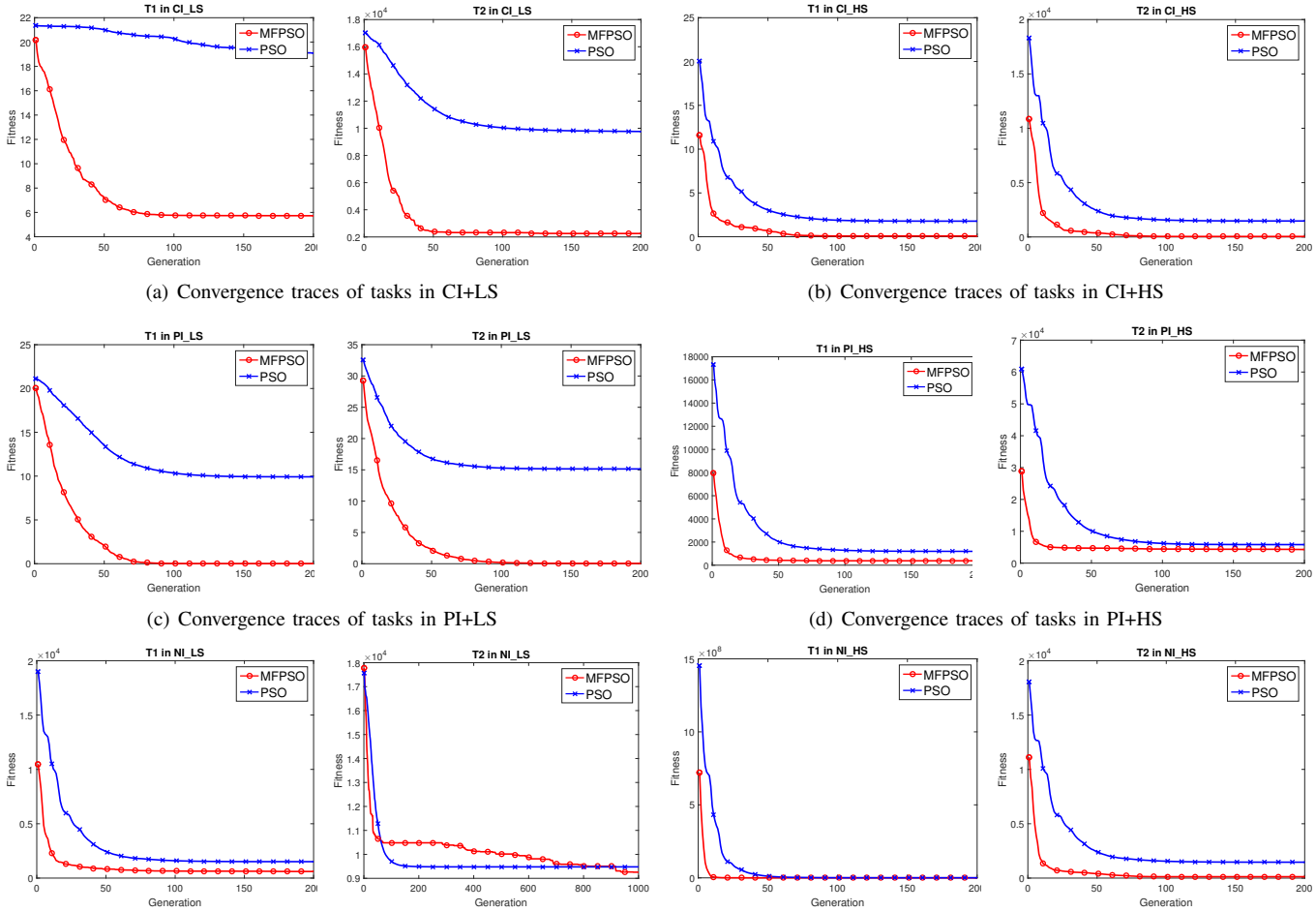
As can be observed in Table II, *MFPSO* achieved superior performance in terms of solution quality on all the *MFO* benchmarks. In particular, on T_2 of benchmark PI+MS and task T_1 of benchmark NI+HS, the single task *PSO* stagnated at the local optimum with fitness values 969566.16 and 659108.88 obtained, respectively. However, with multitasking, *MFPSO* obtained significantly improved solutions with the fitness values of 123.67 and 43.92 achieved. Further, with the implicit knowledge transfer across tasks in multi-tasking, on T_1 of benchmark CI+HS, T_1 of benchmark CI+HS, and T_1 of benchmark NI+MS, etc., *MFPSO* obtained solutions approaching to the corresponding global optimum 0.

To access the search efficiency of *MFPSO*, Fig. 3 depicts the convergence graphs of *MFPSO* and *PSO* on the representative multi-tasking *MFO* benchmarks. In the figures, Y-axis depicts the fitness value, and X-axis denotes the number of generations. As can be observed, in terms of convergence rate, *MFPSO* exhibited superior performance over the single task *PSO* on most of the *MFO* benchmarks. In particular, on benchmark CI+HS Fig. 3(b), PI+HS Fig. 3(d), etc., *MFPSO* takes less than 50 generations to arrive at solutions of higher quality than those obtained by *PSO* where the latter exhausted over 100 generation to arrive at the same solution quality. However, on T_2 of benchmark NI+LS (see Fig. 3(e)), negative transfer of *MFPSO* can be observed. The multi-tasking *MFPSO* converges slower than single task *PSO* before generation 800. This may due to the low similarity between T_1 and T_2 in benchmark NI+LS (inter-task similarity $= 0.0016$ in Table I). Further, it is worth noting that, as in each generation, *MFPSO* optimizes two tasks simultaneously, while *PSO* tackles only one task, the computational cost used by *MFPSO* to arrive at the solution of each task depicted in Fig. 3 is probably about half of that incurred by *PSO*.

TABLE II
AVERAGE PERFORMANCES (MEAN AND BRACKETED STANDARD DEVIATION) OF *MFPSO* AND *PSO* ON DIFFERENT TASKS. BETTER SCORES ARE SHOWN IN BOLD.

Category	*MFPSO*		*PSO*	
	T_1	T_2	T_1	T_2
CI+HS	**0.21** (0.05)	**8.11** (33.59)	1.78 (0.95)	1468.85 (1005.36)
CI+MS	**0.06** (0.25)	**6.26** (27.16)	10.31 (2.90)	1477.86 (931.74)
CI+LS	**5.60** (9.30)	**2226.33** (4257.62)	18.08 (2.97)	9726.86 (855.26)
PI+HS	**205.73** (145.10)	**3841.81** (158.64)	1194.14 (823.38)	5816.55 (6514.39)
PI+MS	**3.58** (0.34)	**123.67** (150.76)	10.29 (3.03)	969566.16 (1392162.51)
PI+LS	**0.01** (0.05)	**0.05** (0.15)	9.92 (2.56)	15.14 (3.96)
NI+HS	**43.92** (28.57)	**39.58** (108.57)	659108.88 (546768.94)	1459.88 (837.34)
NI+MS	**0.48** (0.33)	**12.07** (2.13)	2.01 (0.91)	41.24 (4.22)
NI+LS	**332.64** (135.80)	**9256.22** (7133.19)	1510.29 (879.87)	9478.12 (843.47)

Since *MFPSO* and *PSO* share the same configuration of search operators and parameters, and the only difference is the implicit knowledge transfer across tasks in the former, the superior performance obtained by *MFPSO* in terms of both solution quality and search efficiency confirmed the efficacy



(a) Convergence traces of tasks in CI+LS

(b) Convergence traces of tasks in CI+HS

(c) Convergence traces of tasks in PI+LS

(d) Convergence traces of tasks in PI+HS

(e) Convergence traces of tasks in NI+LS

(f) Convergence traces of tasks in NI+HS

Fig. 3. Convergence traces of *MFPSO* versus the *PSO* on representative multi-tasking benchmarks. Y-axis: Fitness Value; X-axis: Generation.

of the automated implicit information transfer in the *MFO* paradigm.

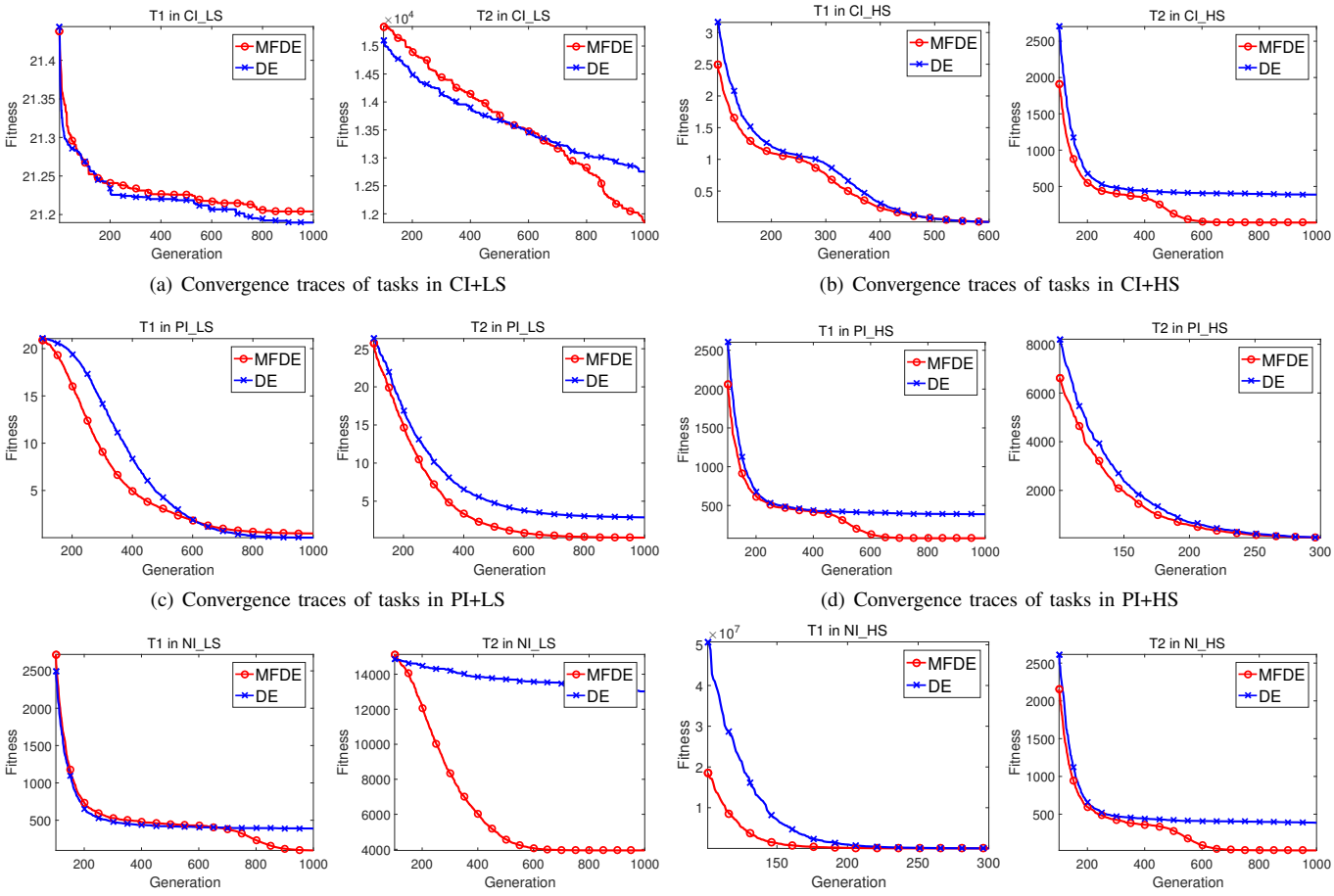
Further, Table 4 tabulates the averaged performance of *MFDE* and *DE* on all the single objective *MFO* benchmarks. In the table, competitive or superior performance is obtained by *MFDE* when compared to *DE* on most of the benchmarks in terms of solution quality. In particular, on problems like T_1 of benchmark PI+HS, T_2 of benchmark NI+HS, and T_2 of benchmark CI+HS, etc., *MFDE* achieved great improved high quality solutions against *DE*. On T_2 of benchmark CI+MS, *DE* stagnated at local optimum 393.616, while the multi-tasking *MFDE* has found the solution close to the global optimum 0. On the other hand, on the benchmarks which have low similarity or no interactions between tasks, deteriorated solutions have been obtained by *MFDE*, such as T_1 of benchmark PI+LS, T_1 of benchmark NI+HS, etc.

Subsequently, Fig. 4 then provides the convergence graphs of *MFDE* and *DE* on the representative multi-tasking *MFO* benchmarks. As can be observed, in Fig. 4, *MFDE* converges faster over *DE* on most of the benchmarks, e.g. T_2 of benchmark CI+HS (Fig. 4(b)), T_1 of benchmark PI+HS (Fig. 4(d)), and T_2 of benchmark NI+HS (Fig. 4(f)), etc. On T_1 of both benchmark CI+LS and PI+LS, due to the low similarity shared between tasks (i.e., 0.0002 and 0.0725 in Table I), *MFDE* obtained competitive performance against *DE*.

Further, on T_1 of benchmark NI+HS, where deteriorated solution has been obtained by *MFDE* against *DE* (see Table III), faster convergence of *MFDE* over *DE* can be observed, i.e., Fig. 4(f). This is due to the high similarity between T_1 and T_2 in benchmark NI+HS (0.9434 in Table I). Lastly, similar to *MFPSO* discussed above, as *MFDE* optimizes two tasks in one single run, the computational cost incurred by *MFDE* on each task is almost halved when compared to the single task *DE*.

V. Conclusion

In this paper, we presented two specific *MFO* search algorithms, i.e., *MFPSO* and *MFDE*, to explore the generality of the *MFO* for multi-task optimization. In particular, based on the multifactorial evolutionary algorithm search paradigm proposed in [1], the detailed designs of the implicit knowledge transfer, which plays the key role for multi-tasking, under the search mechanisms of *PSO* and *DE* have been presented. Further, the 9 single objective *MFO* benchmark problems reported



(a) Convergence traces of tasks in CI+LS

(b) Convergence traces of tasks in CI+HS

(c) Convergence traces of tasks in PI+LS

(d) Convergence traces of tasks in PI+HS

(e) Convergence traces of tasks in NI+LS

(f) Convergence traces of tasks in NI+HS

Fig. 4. Convergence traces of *MFDE* versus the *DE* on representative multi-tasking benchmarks. Y-axis: Fitness Value; X-axis: Generation.

TABLE III
AVERAGE PERFORMANCES (MEAN AND BRACKETED STANDARD DEVIATION) OF *MFDE* AND *DE* ON DIFFERENT TASKS. BETTER SCORES ARE SHOWN IN BOLD.

Category	*MFDE*		*DE*	
	T_1	T_2	T_1	T_2
CI+HS	1.00E-03 (3.05E-03)	**2.608** (7.96)	**1.80E-05** (9.00E-06)	387.61 (12.11)
CI+MS	**0.001** (0.003)	**0.003** (0.012)	0.007 (0.004)	393.616 (14.47)
CI+LS	21.20 (0.04)	**11843.03** (1578.16)	21.19 (0.03)	12748.38 (1199.50)
PI+HS	**78.26** (15.37)	2.20E-05 (2.90E-05)	387.96 (10.39)	4.00E-06 (2.00E-06)
PI+MS	**0.001** (0.001)	60.34 (20.53)	0.026 (0.08)	**57.16** (21.89)
PI+LS	0.46 (0.58)	**0.22** (0.47)	**0.01** (0.01)	2.88 (2.01)
NI+HS	89.28 (48.60)	**20.54** (15.41)	**55.27** (21.86)	386.47 (14.23)
NI+MS	2.03E-03 (4.28E-03)	**2.97** (1.08)	**1.40E-05** (5.00E-06)	3.84 (3.27)
NI+LS	**96.15** (20.02)	**3938.15** (730.99)	388.67 (16.22)	13017.51 (731.19)

in [10], have been used to evaluate the search performance of *MFPSO* and *DE*. The obtained results confirmed the efficacy of *MFO* for optimizing multiple tasks simultaneously.

ACKNOWLEDGMENT

The research reported in this paper is financially supported by the National Natural Science Foundation of China (Grant No. 61603064).

REFERENCES

[1] A. Gupta, Yew-Soon Ong, and L. Feng. Multifactorial evolution: Towards evolutionary multitasking. *Evolutionary Computation, IEEE Transactions on*, PP(99):1–1, 2015.

[2] J. Bi, T. Xiong, S. Yu, M. Dundar, and R. Rao. An improved multi-task learning approach with applications in medical diagnosis. *Machine Learning and Knowledge Discovery in Databases*, pages 117–132, 2008.

[3] Z. Kang, K. Grauman, and F. Sha. Learning with whom to share in multi-task feature learning. *In Proceedings of the 28th International Conference on Machine Learning*, pages 521–528, 2011.

[4] X. Wang, C. Zhang, Z. Zhang, M. Dundar, and R. Rao. Boosted multi-task learning for face verification with applications to web image and video search. *IEEE Conference Computer Vision and Pattern Recognition*, pages 142–149, 2009.

[5] Q. Xu and Q. Yang. A Survey of Transfer and Multitask Learning in Bioinformatics. *Journal of Computing Science and Engineering*, 2011.

[6] T. Back, U. Hammel, and H. P. Schwefel. Evolutionary computation: Comments on the history and current state. *IEEE Transactions on Evolutionary Computation*, 1(1):3–17, 1997.

[7] S. M. Guo and C. C. Yang. Enhancing differential evolution using eigenvector-based crossover operator. *IEEE Transactions on Evolutionary Computation*, 19(1):31–49, 2014.
[8] M. R. Bonyadi and Z. Michalewicz. Analysis of stability, local convergence, and transformation sensitivity of a variant of the particle swarm optimization algorithm. *IEEE Transactions on Evolutionary Computation*, 20(3):370–385, 2016.
[9] Y. S. Ong and A. Gupta. Evolutionary multitasking: A computer science view of cognitive multitasking. *Cognitive Computation*, 8(2):2, 2016.
[10] B. S. Da, Y. S. Ong, L. Feng, A. K. Qin, A. Gupta, Z. X. Zhu, C. K. Ting, K. Tang, and X. Yao. Evolutionary multitasking for single-objective continuous optimization: Benchmark problems, performance metrics and baseline results. *Technical Report, Nanyang Technological University*, 2016.
[11] J. Kennedy and R. C. Eberhart. Particle swarm optimization. *In: IEEE Internat. Conf. on Neural Networks*, pages 1942–1948, 1995.
[12] R. Storn and K. Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. *Internation Computer Science Institute, Berkley, Tech. Rep.*, 1995.
[13] A. K. Qin and P. N. Suganthan. Self-adaptive differential evolution algorithm for numerical optimization. *Proc. of the 2005 IEEE Congress on Evolutionary Computation (CEC05)*, pages 1785–1791, 2005.
[14] R. C. Eberhart and Y. Shi. Comparing inertia weights and constriction factors in particle swarm optimization. *Proc. of the 2000 IEEE Congress on Evolutionary Computation (CEC00)*, pages 84–88, 2000.