A Fast Memetic Multi-objective Differential Evolution for Multi-tasking Optimization

Yongliang Chen, Jinghui Zhong (Corresponding Author)
School of Computer Science and Engineering
South China University of Technology
Guangzhou, China
Email: jinghuizhong@gmail.com

Mingkui Tan
School of Software Engineering
South China University of Technology
Guangzhou, China

***Abstract*—Multi-tasking optimization has now become a promising research topic that has attracted increasing attention from researchers. In this paper, an efficient memetic evolutionary multi-tasking optimization framework is proposed. The key idea is to use multiple subpopulations to solve multiple tasks, with each subpopulation focusing on solving a single task. A knowledge transferring crossover is proposed to transfer knowledge between subpopulations during the evolution. The proposed framework is further integrated with a multi-objective differential evolution and an adaptive local search strategy, forming a memetic multi-objective DE named MM-DE for multi-tasking optimization. The proposed MM-DE is compared with the state-of-the-art multi-tasking multi-objective evolutionary algorithm (named MO-MFEA) on nine benchmark problems in the CEC 2017 multi-tasking optimization competition. The experimental results have demonstrated that the proposed MM-DE can offer very promising performance.**

***Index Terms*—Evolutionary Algorithm, Memetic Algorithm, Local Search, Multi-tasking Optimization, Multi-objective Optimization, Differential Evolution**

I. Introduction

Evolutionary algorithms (EAs) are powerful nature-inspired meta-heuristic search algorithms, which have been widely used in problem-optimization fields [1]–[5]. EAs are manipulated based on groups of individuals carrying different genes, relying on bio-inspired operators such as mutation, crossover and selection.

Most of the existing EAs focus on solving a single task at a time. However, in actual conditions, we often face more than one task, and the tasks are usually related to some extent. One problem often contains useful information that can help solve another related problem. This fact urges us to think of a more efficient EA to handle multiple tasks simultaneously. Multi-tasking is such a proposition that can tackle several tasks at the same time. Since the proposition of multi-tasking has been raised out, it has proved to be very useful and has achieved great success in machine learning research area, including bi-level optimization [6], measuring complementarity [7], modular training [8] and software tests generation [9].

Multi-tasking optimization is now receiving more and more attention in computing intelligence and problem optimization fields. In 2016, multifactorial evolutionary algorithm (MFEA) [10] for single-objective optimization problems (SOOPs) was brought out and gained promising performance in multi-tasking continuous optimization. Later in 2017, new achievements were gained in evolutionary multifactorial optimization based on partial swarm optimization (PSO) and differential evolution (DE) [11], and evolutionary multi-objective optimization problems (MOOPs) [12]. Nevertheless, existing works are mainly designed based on basic multifactorial framework [10], whose efficiency is not high enough to tackle complicated optimization problems. There is still an urgent need for a more general and efficient method to handle different multi-tasking problems.

Local search is an effective tool to improve the search efficiency of EAs on various applications [13]–[15]. In the past decades, local search has been adopted to improve a number of population-based search algorithms, including but not limited to evolutionary programming [16], genetic algorithm [17], genetic programming [18], ant colony optimization [3], and particle swarm optimazation [19]. These EAs embedded with local search are also known as memetic algorithm (MA) [20].

In this paper, an efficient memetic multi-objective evolutionary algorithm is proposed for multi-tasking optimization. First of all, a general evolutionary multi-tasking optimization framework named multi-population multi-tasking evolutionary framework(MMEF) is proposed. The key idea is dividing the whole population into several subpopulations to solve several tasks respectively at the same time. A knowledge transferring crossover operator is proposed, which allows knowledge to transfer between different subpopulations during the evolution process. Furthermore, the proposed MMEF is integrated with a multi-objective differential evolution and an adaptive local search, forming a memetic multi-objective differential evolution algorithm (MM-DE) for multi-tasking optimization. The proposed MM-DE was tested on nine benchmark problems in CEC 2017 multi-tasking optimization competition [21] against the state-of-the-art MO-MFEA [12]. The experimental results have demonstrated that the MM-DE perform much better than MO-MFEA in terms of solution accuracy and search efficiency.

II. Preliminaries

Our proposed method is developed based on a DE variant named JADE [22] and the multi-objective optimization

978-1-5090-6017-7/18/$31.00 ©2018 IEEE

mechanism in NSGA-II [23]. Thus, in this section, we briefly introduce the related background of these algorithms.

A. JADE

Differential evolution (DE), an early EA developed by Storn and Price for global single-objective continuous optimization problem (SOOP) [24], has proved to be a simple yet efficient algorithm in real-world applications [25]–[27]. Except for the canonical DE, extended versions for multi-objective optimization problem (MOOP) (e.g., PDE [28] and MOEA/D [29]) have also been verified powerful. Nowadays, some improved DE variants embedded with self-adaptive technologies have been widely used, such as jDE [30], SaDE [31], JADE [22] and so on. These variants can offer very promising performance in optimization problems.

In this paper, JADE is adopted as the core evolution algorithm. At the beginning of JADE, an initial population is randomly generated. Then, new individuals are created by randomly combining individuals in the previous generation using genetic operators such as mutation and crossover.

For each individual at each generation, the mutation vector w^i_{G+1} can be generated based on the current parent population $\{v^i_G | i = 1, 2, ..., NP\}$, where NP is the population size and G means the current generation. Different mutation strategies [2], [24] can be used in this process, e.g., *"DE/rand/1"*, *"DE/current-to-best/1"*, *"DE/best/1"* and so on. In JADE, the *"DE/current-to-pbest"* strategy is used, which is expressed by

$$w^i_{G+1} = v^i_G + F_i * (v^{p,best}_G - v^i_G) + F_i * (v^a_G - \widetilde{v^b_G}) \quad (1)$$

where $a, b, c \in [1, NP]$ are random integers with $a \neq b \neq i$, $v^{p,best}_G$ is chosen randomly from the top $p\%$ individuals of the population in generation G, and $\widetilde{v^b_G}$ is selected randomly from a joint set $\mathbf{P} \cup \mathbf{A}$, where $\mathbf{P}$ is the current population and archive $\mathbf{A}$ is a set storing inferior individuals. The update of $\mathbf{A}$ is very simple. $\mathbf{A}$ is initiated as empty at the beginning. When selection operation is conducted through DE, obsoleted parent individual will be picked out and put into $\mathbf{A}$ if $\mathbf{A}$ is not full yet, otherwise, it will then randomly replace one of the individual in $\mathbf{A}$ to keep $\mathbf{A}$ full. Using interference from the archive to guide the search process can somehow avoid the population being trapped in local optimum and premature scene. F_i is the scaling factor, in conventional DE, $F_i = F$ is a preset constant, while in JADE, each individual v^i is associated with its own scaling factor F_i. For solutions with more than one dimension, we have

$$\begin{aligned} w^i_{G+1}(j) =& v^i_G(j) + F_i * (v^{p,best}_G(j) - v^i_G(j)) \\ &+ F_i * (v^a_G(j) - \widetilde{v^b_G}(j)), \\ & j = 1, 2, ..., D \end{aligned} \quad (2)$$

where D represents the dimension number of the solution vector. To guarantee more diversity, a crossover operation is generated by combining individuals from previous generation v_G and newly produced individuals w_{G+1}, following the rule of

$$p^i_G(j) = \begin{cases} w^i_{G+1}(j) & \text{if } rand(0,1) < CR_i \text{ or } j = k \\ v^i_G(j) & otherwise \end{cases} \quad (3)$$

where $k \in [1, D]$ is a random integer to ensure that at least one element in v^i_G is replaced, CR_i is crossover rate which is predefined in traditional DE and associated to certain individual in JADE, and $rand(a, b)$ returns a random number between a and b. At last, evaluation and selection can be described as

$$v^i_{G+1} = \begin{cases} p^i_G & \text{if } f(p^i_G) < f(v^i_G) \\ v^i_G(j) & otherwise \end{cases} \quad (4)$$

where f is the objective function and we suppose that the smaller objective value is better.

In JADE, The CR_i and F_i are updated adaptively based on the CRs and Fs obtained from the last generation. Mark S_{CR} and S_F to be sets storing former CR and F values respectively, and the CR_i associated to individual x_i is generated according to a normal distribution of mean value μ_{CR} and standard deviation 0.1

$$CR_i = rand_{ni}(\mu_{CR}, 0.1) \quad (5)$$

then truncated to its range $[L_{CR}, U_{CR}]$. The mean μ_{CR} is initialized to be 0.5 at the first generation and updated at the end of each generation by

$$\mu_{CR} = (1 - c) * \mu_{CR} + c * mean_A(S_{CR}) \quad (6)$$

where $c \in (0, 1)$ is a predefined constant and $mean_A(\cdot)$ returns the usual arithmetic mean.

Similarly, $F_i \in [L_F, U_F]$ for individual x_i is generated according to a Cauchy distribution with location parameter μ_F and scale parameter 0.1

$$F_i = rand_{ci}(\mu_F, 0.1) \quad (7)$$

and then truncated to be U_F if $F_i > U_F$ or regenerated if $F_i < L_F$. μ_F is initialized to be 0.5 at the first generation and then updated at the end of each generation by

$$\mu_F = (1 - c) * \mu_F + c * mean_L(S_F) \quad (8)$$

where $mean_L(\cdot)$ returns the Lehmer mean

$$mean_L(S_F) = \frac{\sum_{F \in S_F} F^2}{\sum_{F \in S_F} F} \quad (9)$$

The whole population improves through iteration of mutation, crossover and selection together with the self-adaptive updates of parameters, until the terminate condition is met.

B. NSGA-II for MOOPs

The nondominated sorting genetic algorithm II (NSGA-II) proposed in [23] is a well-known algorithm for MOOP. MOOP is about multi-criteria decision making, that is to give consideration to multiple objectives while solving the encountered problem. In MOOPs, several objective functions need to be optimized in one solution space, which can be formulated as

$$min(f_1(x), f_2(x), ..., f_k(x)),\ x \in X \tag{10}$$

where x is a solution vector, k is the number of objective functions, and X represents the decision space.

As functions in MOOPs are conflicting with each other, finding a solution that can optimize all objective functions is almost impossible. Thus, the goal of an MOOP is to find a set of trade-off solutions which are all equally good. This set is called Pareto set (PS) and solutions in it are called Pareto optimal solutions (POS).

To apply EA for MOOPs, a fundamental problem is to evaluate the quality of individuals in the population, so that the algorithm can select the better ones. In NSGA-II, two core strategies are used to solve this problem, i.e., the nondominated sorting strategy (NSS) and crowding distance sorting strategy (CDSS). These two approaches set up a criterion to judge solutions of a population.

Specifically, for two solution vectors x_1 and x_2 in the solution space, we can define $x_1 \prec x_2$ to represent that x_1 dominates x_2, and the dominated relation can be described as

$$\begin{aligned} x_1 \prec x_2 \Leftrightarrow f_i(x_1) \leq f_i(x_2),\ i = 1, 2, ..., k \\ and\ \exists j \in [1, k],\ s.t.\ f_j(x_1) < f_j(x_2) \end{aligned} \tag{11}$$

From (11), it can be observed that solution a dominates b iff a is not worse than b on all objectives and a is better than b on at least one objective. If $x_1 \not\prec x_2$ and $x_2 \not\prec x_1$, we regard them to be nondominated, which means they are equally good.

Then, we can rank all solutions in a solution set by using NSS. We classify solutions which can't be dominated by other solutions in the population into rank 1. Then, we remove solutions of rank 1, and the non-dominated solutions in the remaining solution set is classified to rank 2 by using the same method. The above procedures are repeated until all solutions have been ranked. The set of solutions in rank 1 is called approximation Pareto-front (PF).

After NSS, solutions are distributed into different ranks. Then, the CDSS can be used to sort the solutions in the same rank.

To get an estimation of the density of a certain solution within its nondominated rank, we can calculate the size of the region that restricts it. The definition of the region can be a cuboid in Fig. 1, and the crowding distance of a solution in its rank is the average side length of the cuboid. For solutions in the same rank, we prefer those with larger crowding distance values, which means there are fewer other solutions around them, which can increase the diversity of the PS.

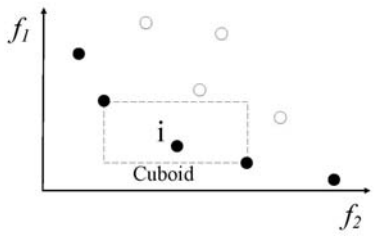

Fig. 1. Crowding-distance calculation. Solid points are in the same nondominated rank.

At last, having gained the ranks and crowding distances of all solutions, we can choose NP best solutions of them to form a new population. It should be emphasized that we first select the solutions with low rank numbers, if solutions in the previous i ranks can't reach NP and previous $i+1$ ranks are beyond NP, then we will select the rest of the solutions of rank $(i+1)$ according to CDSS.

III. The Proposed Method

In global search algorithms, exploration and exploitation are two competing goals [32]. To find correct global optima, information of the search space needs to be collected and estimated. Exploration is such a process to ensure the reliability of the final global optima. Exploration can sometimes locate a precise optimum, however, it takes a relatively long time and huge computing cost (and it does not always succeed). In this sense, Exploitation is important, for it concentrates on searching for better and more precise solutions around current solutions. To this end, the balance between exploration and exploitation can lead to discrepancies in performance of search methods.

Memetic computing (MC) is a term derived from "meme" in [33], which is defined as *a paradigm that uses the notion of meme(s) as units of information encoded in computational representations for the purpose of problem-solving* [34]. The memetic algorithm (MA) [35] is one of the earliest research field in MC. MAs usually adopt a population-based heuristic search algorithm to perform global search, and a local search algorithm as meme. Thus, MAs are algorithms achieving the two goals mentioned above.

This section first introduces the memetic multi-population multi-tasking evolutionary framework (MMEF), and then a memetic multi-objective DE developed on MMEF with JADE and an adaptive local search strategy [36] is presented.

A. The Proposed Memetic MMEF

Figure 2 illustrates the proposed memetic MMEF. Specifically, suppose there are N tasks to be solved, then the whole population is divided into N subpopulations, with each subpopulation focusing on solving one task. Each subpopulation is evolved by genetic operators, including a newly proposed Transfer Learning Crossover (TLC). In traditional multi-tasking EAs, the whole population is evolved as the whole, which may reduce the search efficiency due to the negative effects raised from learning different tasks. Inspired by this drawback,

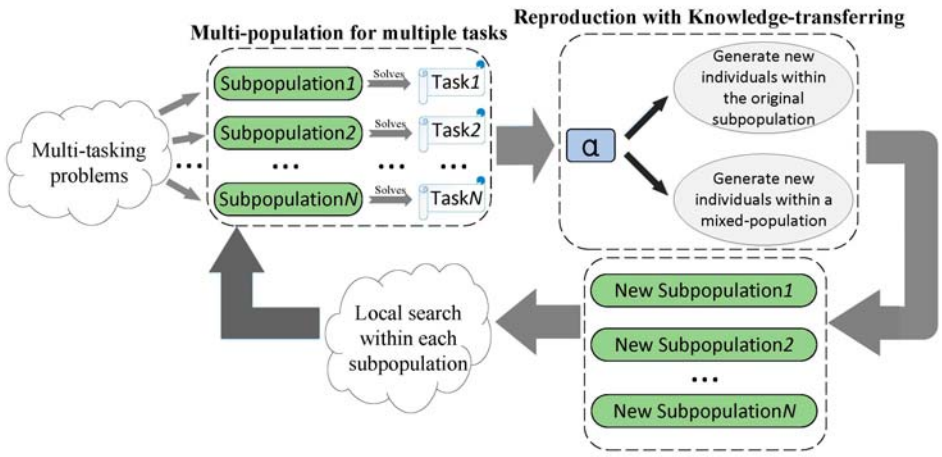

Fig. 2. The proposed memetic multi-population multi-tasking evolutionary framework.

we propose to use multi-populations to evolve multiple tasks simultaneously. In this way, each subpopulation can concentrate on one task, which can avoid the extra disturbance. To allow knowledge transferring, TLC is proposed. The key idea is to use a new parameter α to determine the parents that used to generate offsprings. Specifically, to generate an offspring for a subpopulation, we first generate a random value $r \in [0, 1]$. If $r < \alpha$, then, the parents are selected from a mixed-population of two tasks. Otherwise, the parents are selected from the subject subpopulation. At last, local search is performed to improve the promising individuals in each subpopulation.

The above MMEF has three main advantages. First, the mechanism of multi-population reduces the distraction when handling MT problems, which makes it more efficient. Second, by selecting parents from different subpopulations, the framework can maintain a high population diversity. Third, the gene transferring rate between subpopulations can be controlled by α. When the tasks to be solved are not similar, which means knowledge-transferring can do little to improve solving the problems, a little α may be more proper to make subpopulation evolved more separately. On the other hand, if the tasks to be solved are related intimately, which means solving one task can effectively help solve other tasks, then a bigger α may be suitable. However, as creatures tend to communicate within their surrounding area, α should not be too large to ensure individuals primarily evolve in their own subpopulations.

B. The Proposed MM-DE

Our MM-DE algorithm is developed by integrating the multi-objective JADE for global search and an adaptive local search proposed in [36], together with the MMEF. In the following parts, we first introduce the chromosome representation for evolution and the local search of the proposed method, then detailed evolution procedures are described in a step-by-step manner.

Chromosome Representation In MOOPs and MTEAs, one priority is to confirm the uniform coding strategy. In some situations, the dimensions of solutions of the functions in one MOOP can be different, then we need to put them into a uniform solution space. For MOOP having functions with different dimensions, the largest dimension number is chosen as the length of the chromosome for coding, in which the smaller one takes the front part while the larger one takes relatively more dimensions and the largest takes all. When it comes to solving multi-tasking problems, the dimension number of each task is set to be the largest one. Though some dimensions are needless for tasks of low dimensions, they are necessary for transferring knowledge.

To represent solutions in functions with different domains, all solutions in the solution space are varied in the range of $[0, 1]$. The method to map the solutions to the domains can be expressed as

$$X' = L + X * (U - L) \tag{12}$$

where X' is the solution in the domain, X is the solution in the solution space, and U and L are the upper bound and the lower bound of the function respectively.

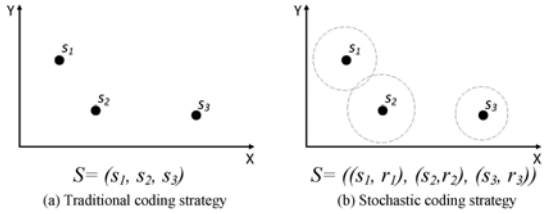

Fig. 3. Comparison of two different coding strategies.

The MM-DE adopts a normal form proposed in [36] to represent chromosome. In traditional EAs, as illustrated in Fig. 3(a), each individual represents a candidate solution in the search space, containing no information about their surrounding area. This coding strategy is not effective enough to exploit the neighboring regions of individuals. To improve the search efficiency, a stochastic coding strategy is adopted as shown in Fig. 3(b), each solution contains not only the solution of the encountered problem, but also a region which is defined by a

radius value r. Specifically, each individual can be expressed by

$$S = ((s_1, r_1), (s_2, r_2), ..., (s_D, r_D)) \tag{13}$$

where the vector $\{s_1, s_2, ..., s_D\}$ represents the center values of the regions in each solution, vector $\{r_1, r_2, ..., r_D\}$ determines the size of the neighboring region. During the evolution, individuals may be updated by sampling solutions within the region, this fine-tuning strategy can efficiently improve the search efficiency.

Based on the above coding strategy, four steps are utilized in the proposed method to evolve chromosomes.

Step 1 - Initialization In this step, N initial subpopulations with NP random individuals are generated. Let P_0 be one of the initial subpopulations, then P_0 can be expressed as

$$P_0 = \{S_1, S_2, ..., S_{NP}\} \tag{14}$$

where S_i is the ith individual, which can be expressed as

$$S_i = \{(s_1, r_1), (s_2, r_2), ..., (s_D, r_D)\},\ i = 1, 2, ..., NP \tag{15}$$

For the ith individual S_i, its jth solution center value is randomly generated within the search space, i.e.,

$$s_{i,j} = rand(L_i, U_i),\ j = 1, 2, ..., D \tag{16}$$

where L_i and U_i are the lower bound and the upper bound of the search space, respectively. The region radius $r_{i,j}$ is initialized by

$$r_{i,j} = (U_i - L_i)/10 \tag{17}$$

In the same way, we generate $N - 1$ other subpopulations. Then, for each individual in all subpopulations, its fitness value (i.e., the objective value of the corresponding task assigned to the subpopulation) is evaluated.

Step 2 - New Individual Generation In this step, JADE is applied to generate $N * NP$ new individuals. Notice that, except using traditional mutation operators in JADE, TLC is applied in this step. Specifically, for each target individual solution $S_i = ((s_{i,1}, r_{i,1}), (s_{i,2}, r_{i,2}), ..., (s_{i,D}, r_{i,D}))$ focusing on task k in the current generation, a new individual $S'_i = ((s'_{i,1}, r'_{i,1}), (s'_{i,2}, r'_{i,2}), ..., (s'_{i,D}, r'_{i,D}))$ is generated by the following processes. First, a new temporary population $\mathbf{P}$ is generated by

$$\mathbf{P} = \begin{cases} \mathbf{P}_k \cup \mathbf{P}_l & if\ rand(0,1) < \alpha_{k,l} \\ \mathbf{P}_k & otherwise \end{cases} \tag{18}$$

where $l \in [1, N]$ is a random integer with $l \neq k$, $\mathbf{P}_k$ and $\mathbf{P}_l$ are subpopulation k and l, $\alpha_{k,l} \in (0, 1)$ is the TLC rate between $\mathbf{P}_k$ and $\mathbf{P}_l$ set by users. Once $\mathbf{P}$ has been generated, the new individual S'_i can be generated by

$$s'_{i,j} = \begin{cases} s_{i,j} + F_i * (s_{rbest,j} - s_{i,j}) + F_i * (s_{a,j} - \widetilde{s_{b,j}}) \\ \qquad if\ rand(0,1) < CR_i\ or\ j = q \\ s_{i,j} \qquad if\ rand(0,1) \geq CR \end{cases} \tag{19}$$

$$r'_{i,j} = \begin{cases} r_{i,j} + F_i * (r_{rbest,j} - r_{i,j}) + F_i * (r_{a,j} - \widetilde{r_{b,j}}) \\ \qquad if\ rand(0,1) < CR_i\ or\ j = q \\ r_{i,j} \qquad if\ rand(0,1) \geq CR \end{cases} \tag{20}$$

where F_i and CR_i are parameters associated to individual S_i, $a \neq b \in [1, NP]$ ($[1, 2 * NP]$, if $\mathbf{P} = \mathbf{P}_k \cup \mathbf{P}_l$) are random individual indexes and $q \in [1, D]$ is a random integer to ensure at least one dimension is replaced. We extend the "DE/current-to-$p$best" strategy to "DE/current-to-$rank$best", which means the individuals of rank 1 can be candidates of S_{rbest}. It should be emphasized that, if $\mathbf{P}$ is a mixed-population, then the basic vector S_i is from $\mathbf{P}_l$, which gives the chance for individuals in $\mathbf{P}_l$ to handle task k. What's more, if F_i goes beyond its range, it will be regenerated until it is valid. If CR_i exceeds its range, it will be set to be a random number within the range. It can be observed that when TLC takes place, JADE operations are conducted with the mixed-population $\mathbf{P}$ to generate new individuals. If $s'_{i,j}$ exceeds the search range, it will be replaced by the nearest value in the search range. Simultaneously, there is a limit for $r'_{i,j}$ by adaptively setting the maximum value of $r_{i,j}$ according to the evaluation times $evals$

$$r_{max} = \frac{U - L}{10} * \frac{eval_{max} - evals + 1}{eval_{max}} \tag{21}$$

where $eval_{max}$ is the maximum evaluation times the algorithm can conduct. r_{max} will reduce as the evolution goes on, which is good for improving the fine-tuning accuracy. Finally, we combine all $s'_{i,j}$s together to form a new individual S'_i. Because S_i comes from $\mathbf{P}_k$, S'_i and S_i will compete under the standard of task k. S_i will be substituted by the better one that dominates the other and become a new individual of the new population. If either of them can dominate each other, then S'_i will be added to a new set $\mathbf{A}_k$.

Step 3 - Adaptive Local Search In this step, an adaptive local search is performed to fine-tune the solutions by sampling solutions from their neighboring regions.

In MM-DE, we set a variable T to control the frequency of performing the local search. To reduce computational cost, the local search is only performed on the LM individuals of rank 1. A constant $\beta \in (0, 1)$ is set to control the local search rate of each dimension value in the LM solutions. Also, there is a random integer $l \in [1, D]$ to guarantee at least one dimension value in one solution is updated. For the LM chosen individuals, LN neighboring individuals are generated to compete with the original individual by

$$s'_{i,j} = \begin{cases} s_{i,j} + Gauss(0,1) * r_{i,j} & \text{if rand(0,1)} < \beta\ or\ j = l \\ s_{i,j} & \text{otherwise} \end{cases}, \tag{22}$$

where $Gauss(0, 1)$ returns a random number with standard Gaussian distribution.

To ensure $s'_{i,j} \in [L_i, U_i]$, we use the method of

$$s'_{i,j} = \begin{cases} rand(s_{i,j}, Ui) & \text{if } s'_{i,j} > U_i \\ rand(L_i, s_{i,j}) & \text{if } s'_{i,j} < L_i \end{cases} \tag{23}$$

When a new individual S_i' is generated by involving all $s_{i,j}'$, it will be compared with the previous individual S_i and replace S_i with nondominated relations. If none of them can dominate each other, then S_i' will be added to $\mathbf{A}_k$. The radius values are updated adaptively after the local search. If S_i' dominates S_i, then the radius values of S_i will be extended. Otherwise, if S_i dominates S_i', then the radius values of S_i will be reduced

$$r_{i,j} = \begin{cases} r_{i,j} * \lambda & reduce\ case \\ r_{i,j}/\lambda & extend\ case \end{cases} \quad j = 1, 2, ..., D \tag{24}$$

where $\lambda \in (0, 1)$ is the shrinking rate.

Algorithm 1: Pseudocode of local search in MM-DE

```
Select LM individuals of rank 1
for every S_i in the LM individuals do
  for 1 to LN do
    Generate a random integer l ∈ [1, D]
    for every center value s_{i,j} in S_i do
      if rand(0,1) < β or j = l then
        s'_{i,j} = s_{i,j} + Gauss(0,1) * r_{i,j}
      else
        s'_{i,j} = s_{i,j}
      end if
      Ensure s'_{i,j} ∈ [L_i, U_i]
      S'_i ← S'_i ∪ {(s'_{i,j}, r_{i,j})}
    end for
    Substitute S_i with the better one between S_i and S'_i
  end for
  Update radius values in S_i
end for
```

Step 4 - New Subpopulations Updating This step aims to update all subpopulations. Specifically, for the kth subpopulation, the NP best individuals are selected from $\mathbf{P}_k \cup \mathbf{A}_k$ by using NSS and CDSS.

There is a loop from *Step2* and *Step 4* until termination conditions are met.

IV. Experimental Studies

A. Experimental Settings

In this section, the proposed MM-DE is validated by testing nine benchmark problems in [21]. In the problems, there are three levels of similarity between the tasks, i.e., high, medium and low. Similarly, the intersection levels of the global optima between tasks can be ranked as complete, partial and no. Then, the nine problems can be differentiated by combining these two criteria arbitrarily (i.e., Complete Intersection with High Similarity (CIHS)). The nine benchmark problems are all multi-tasking problems with two tasks. Except the first tasks in NIMS and NILS are 3-objective, the others are all 2-objective.

To evaluate the performance of MOOP, we use the average inverted generational distance (IGD) as performance metric for comparison. The IGD is defined as follows: *Suppose P is a set of solutions which is uniformly distributed along the PF of an MOOP, and A is a set of approximation Pareto solutions. The IGD from A to P can be expressed by:*

$$IGD(A, P) = \frac{1}{|P|}\sqrt{\sum_{x \in P}(min_d(x, A))^2} \tag{25}$$

where $|P|$ is the number of solutions in P, and $min_d(x, A)$ returns the distance between x and the point in A which is closest to x. The IGD can efficiently judge whether a solution is good enough by measuring both diversity and convergence. Generally, a small $IGD(A, P)$ value means that A is close to P. For these nine problems, the accurate PFs are known.

TABLE I
Parameter settings.

parameter	*value*	*summary*
CR_i	$[0.1, 0.9]$	Crossover rate
F_i	$[0.1, 2]$	Scaling factor
α	0.05	K-crossover rate
β	0.002	Rate to local-search each dimension
LM	3	Individuals for local search
LN	10	Neighboring points generated around
T	10	Frequency of local search
λ	0.5	Shrinking rate
$eval_{max}$	$300,000$	Maximum evaluation times
NP	120 (NIMS,NILS) 100 (Others)	Population size

Some initial parameter settings of the proposed MM-DE are listed in Table I. Notice that, NP is set differently for NIMS and NILS, because individuals selected to conduct IGD calculation are set to be 100 for 2-objective tasks and 120 for 3-objective tasks. Algorithms are conducted for 30 independent times with different random seeds. The average IGD values are used as the performance metric for comparison.

To demonstrate the capability of MM-DE, it is compared with MO-MFEA, a state-of-the-art MTEA for MOOP. The parameter settings of MO-MFEA are set according to [21]. Furthermore, to show the effectiveness of the parameter adaptation component and the local search component, we simplify MM-DE and generate two other algorithms, namely, MM-DE/A (MM-DE without parameter adaptation) and MM-DE/L (MM-DE without local search). Notice that, for MM-DE/A, CR_i and F_i are generated randomly within their ranges. The other parameter settings of MM-DE/A and MM-DE/L are set the same as MM-DE.

B. Experimental Results

Table II shows the average IGD values of the two algorithms in 30 runs, where the better ones are marked in bold format. Also, a Wilcoxon signed rank test is performed to better perceive the significant differences between the two algorithms. It can be observed clearly that the proposed MM-DE not only has successfully found all the better results , but also is significantly better than MO-MFEA. What's more, the performance on some problems (i.e., CIMS, PIHS and PIMS), MM-DE distinctly outperformed MO-MFEA on both two tasks, which can prove that MM-DE has the better capability of finding solutions of high quality.

TABLE II
COMPARISON RESULTS OF MO-MFEA AND MM-DE.

		MO-MFEA	MM-DE
CIHS	T_1	3.99E-04 $-$	**1.36E-04**
	T_2	2.65E-03 $-$	**1.44E-04**
CIMS	T_1	4.57E-02 $-$	**1.46E-04**
	T_2	8.77E-03 $-$	**1.37E-04**
CILS	T_1	2.71E-04 $-$	**1.36E-04**
	T_2	1.90E-04 $-$	**1.37E-04**
PIHS	T_1	1.10E-03 $-$	**1.37E-04**
	T_2	3.04E-02 $-$	**1.36E-04**
PIMS	T_1	2.62E-03 $-$	**9.82E-04**
	T_2	1.09E+01 $-$	**2.29E+00**
PILS	T_1	3.24E-04 $-$	**1.37E-04**
	T_2	1.10E-02 $-$	**1.38E-04**
NIHS	T_1	1.55E+00 $-$	**1.47E+00**
	T_2	5.02E-04 $-$	**1.37E-04**
NIMS	T_1	2.79E-01 $-$	**1.43E-01**
	T_2	2.86E-02 $-$	**2.29E-04**
NILS	T_1	8.35E-04 $-$	**5.88E-04**
	T_2	6.43E-01 $-$	**6.42E-01**

Symbols $+$, $\approx$ and $-$ represent that the competitor is respectively significantly better than, similar to and worse than MM-DE according to the Wilcoxon signed-rank test at $\alpha = 0.05$.

TABLE III
COMPARISON OF MM-DE, MM-DE/A AND MM-DE/L.

		MM-DE	MM-DE/A	MM-DE/L
CIHS	T_1	**1.36E-04**	1.40E-04 $-$	**1.36E-04** $\approx$
	T_2	**1.44E-04**	**1.44E-04** $\approx$	**1.44E-04** $\approx$
CIMS	T_1	**1.46E-04**	1.47E-04 $-$	1.47E-04 $\approx$
	T_2	**1.37E-04**	1.42E-04 $-$	**1.37E-04** $\approx$
CILS	T_1	**1.36E-04**	1.40E-04 $-$	1.37E-04 $-$
	T_2	**1.37E-04**	1.53E-04 $-$	**1.37E-04** $\approx$
PIHS	T_1	**1.37E-04**	1.44E-04 $-$	5.32E-03 $-$
	T_2	**1.36E-04**	1.45E-04 $-$	6.05E-03 $-$
PIMS	T_1	9.82E-04	1.35E-03 $-$	**9.19E-04** $\approx$
	T_2	2.29E+00	1.93E+00 $+$	**1.65E+00** $+$
PILS	T_1	1.37E-04	1.39E-04 $-$	**1.36E-04** $\approx$
	T_2	**1.38E-04**	2.18E-04 $-$	2.16E-04 $-$
NIHS	T_1	1.47E+00	**1.39E+00** $+$	1.48E+00 $\approx$
	T_2	**1.37E-04**	1.44E-04 $-$	1.37E-04 $-$
NIMS	T_1	**1.43E-01**	1.67E-04 $-$	1.45E-01 $\approx$
	T_2	2.29E-04	**2.13E-04** $\approx$	2.37E-04 $\approx$
NILS	T_1	5.88E-04	9.90E-04 $-$	**5.86E-04** $\approx$
	T_2	**6.42E-01**	6.43E-01 $-$	6.51E-01 $-$

Symbols $+$, $\approx$ and $-$ represent that the competitor is respectively significantly better than, similar to and worse than MM-DE according to the Wilcoxon signed-rank test at $\alpha = 0.05$.

As for the convergence trend, which is shown in Fig. 4, on most of the tasks, MM-DE reaches the smallest IGD values earlier than MO-MFEA. In some cases (i.e., CIHS, PIMS and NIHS), MM-DE converges faster all the time and gains better results. Though in some problems (i.e., PILS, CILS and NILS), MM-DE converges slower than MO-MFEA on early stage, it can finally reach better IGD values, which indicates the better global search capability of MM-DE.

Table III shows the comparisons results of the MM-DE and its two simplified versions. It can be observed that both of the parameter adaptation component and the local search component are effective to improve the search performance. We can find that the parameter adaptation component significantly impacts the capability of the algorithm, for more than half of the tasks, MM-DE significantly outperformed MM-DE/A. Meanwhile, with the help of the local search component, MM-DE performs much better than or is at least competitive to MM-DE/L on most problems (e.g., both tasks of PIHS, the first task of CILS, the second task of PILS and NIHS).

V. CONCLUSION

In this paper, a memetic multi-population based evolutionary framework is proposed for solving multi-tasking multi-objective optimization problems. Based on the proposed framework, a fast memetic algorithm named MM-DE is developed. In the proposed MM-DE, an adaptive local search strategy is adopted to capture the features of individuals in the surrounding regions to exploit better solutions. The parameter adaptation strategy used in JADE is adopted in MM-DE to further improve the performance of the algorithm. The experimental results on nine benchmark problems on multi-tasking multi-objective optimization demonstrate that the proposed algorithm can offer very promising performance.

ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China (Grant No. 61602181), and by the Fundamental Research Funds for the Central Universities (Grant No. 2017ZD053).

REFERENCES

[1] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments-a survey," *IEEE Trans. Evolut. Comput.*, vol. 9, no. 3, pp. 303–317, 2005.

[2] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evolut. Comput.*, vol. 15, no. 1, pp. 4–31, 2011.

[3] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE computational intelligence magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[4] J. Zhong, L. Feng, and Y. S. Ong, "Gene expression programming: A survey [review article]," *IEEE Computational Intelligence Magazine*, vol. 12, no. 3, pp. 54–72, Aug 2017.

[5] J. Zhong, W. Cai, M. Lees, and L. Luo, "Automatic model construction for the behavior of human crowds," *Applied Soft Computing*, vol. 56, pp. 368 – 378, 2017.

[6] A. Gupta, J. Mańdziuk, and Y.-S. Ong, "Evolutionary multitasking in bi-level optimization," *Complex & Intelligent Systems*, vol. 1, no. 1-4, pp. 83–95, 2015.

[7] A. Gupta, Y.-S. Ong, B. Da, L. Feng, and S. Handoko, "Measuring complementarity between function landscapes in evolutionary multitasking," in *IEEE Congress on Evolutionary Computation, accepted*, 2016.

[8] R. Chandra, A. Gupta, Y.-S. Ong, and C.-K. Goh, "Evolutionary multi-task learning for modular training of feedforward neural networks," in *International Conference on Neural Information Processing*. Springer, 2016, pp. 37–46.

[9] R. Sagarna and Y.-S. Ong, "Concurrently searching branches in software tests generation through multitask evolution," in *Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2016, pp. 1–8.

[10] A. Gupta, Y.-S. Ong, and L. Feng, "Multifactorial evolution: toward evolutionary multitasking," *IEEE Trans. Evolut. Comput.*, vol. 20, no. 3, pp. 343–357, 2016.

[11] L. Feng, W. Zhou, L. Zhou, S. Jiang, J. Zhong, B. Da, Z. Zhu, and Y. Wang, "An empirical study of multifactorial pso and multifactorial de," in *IEEE Congress on Evolutionary Computation (CEC), 2017*. IEEE, 2017, pp. 921–928.

[12] A. Gupta, Y. S. Ong, L. Feng, and K. C. Tan, "Multiobjective multifactorial optimization in evolutionary multitasking," *IEEE Transactions on Cybernetics*, vol. 47, no. 7, pp. 1652–1665, July 2017.

[13] A. Auger and N. Hansen, "Performance evaluation of an advanced local search evolutionary algorithm," in *The IEEE Congress on Evolutionary Computation, 2005*, vol. 2. IEEE, 2005, pp. 1777–1784.

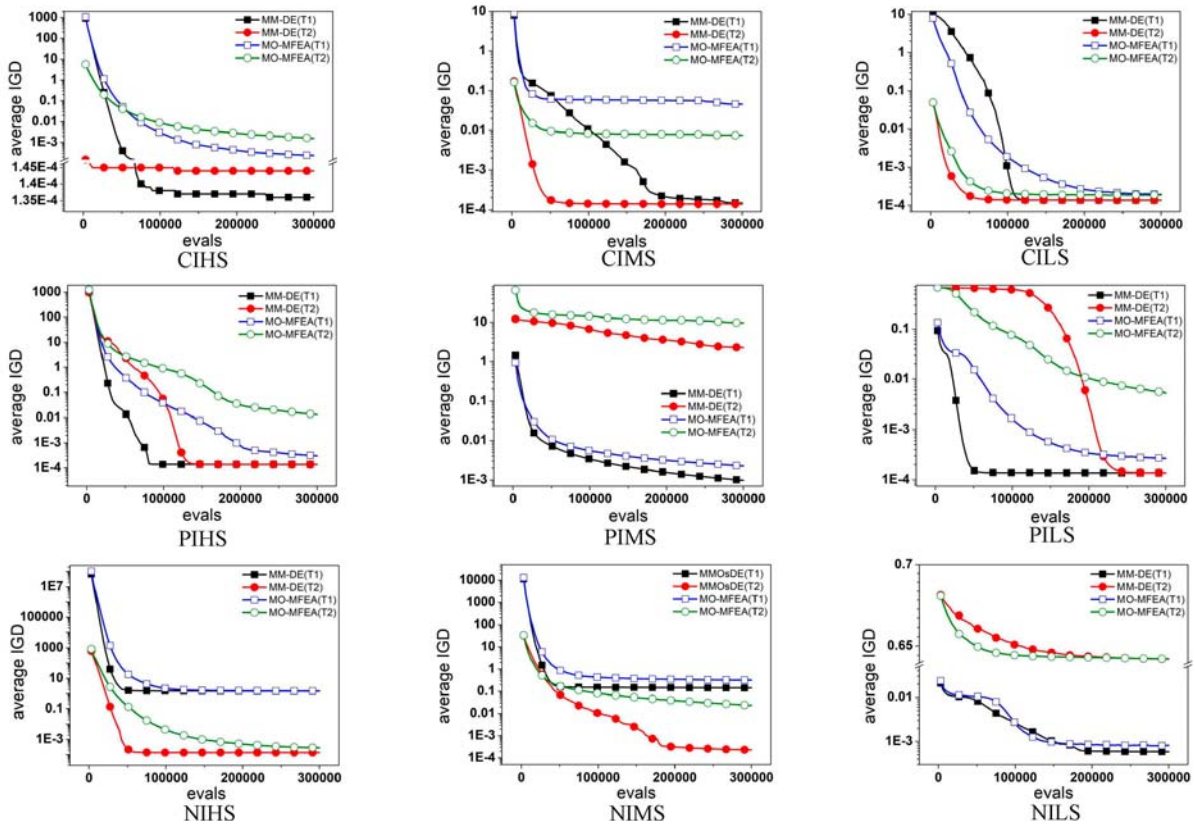

Fig. 4. Evolution average IGD with the number of evaluation times.

[14] F. Neumann and I. Wegener, "Randomized local search, evolutionary algorithms, and the minimum spanning tree problem," *Theoretical Computer Science*, vol. 378, no. 1, pp. 32–40, 2007.

[15] N. Noman and H. Iba, "Accelerating differential evolution using an adaptive local search," *IEEE Trans. Evolut. Comput.*, vol. 12, no. 1, pp. 107–125, 2008.

[16] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies–a comprehensive introduction," *Natural computing*, vol. 1, no. 1, pp. 3–52, 2002.

[17] M. Mitchell, *An introduction to genetic algorithms*. MIT press, 1998.

[18] N. F. McPhee, R. Poli, and W. B. Langdon, "Field guide to genetic programming," 2008.

[19] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *IEEE International Conference on Evolutionary Computation Proceedings, 1998*. IEEE, 1998, pp. 69–73.

[20] X. Chen, Y. S. Ong, M. H. Lim, and K. C. Tan, "A multi-facet survey on memetic computation," *IEEE Trans. Evolut. Comput.*, vol. 15, no. 5, pp. 591–607, 2011.

[21] Y. Yuan, Y.-S. Ong, L. Feng, A. Qin, A. Gupta, B. Da, Q. Zhang, K. C. Tan, Y. Jin, and H. Ishibuchi, "Evolutionary multitasking for multiobjective continuous optimization: Benchmark problems, performance metrics and baseline results," *arXiv preprint arXiv:1706.02766*, 2017.

[22] J. Zhang and A. C. Sanderson, "Jade: adaptive differential evolution with optional external archive," *IEEE Trans. Evolut. Comput.*, vol. 13, no. 5, pp. 945–958, 2009.

[23] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Trans. Evolut. Comput.*, vol. 6, no. 2, pp. 182–197, 2002.

[24] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[25] R. Joshi and A. C. Sanderson, "Minimal representation multisensor fusion using differential evolution," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 29, no. 1, pp. 63–76, 1999.

[26] J. Zhang, V. Avasarala, and R. Subbu, "Evolutionary optimization of transition probability matrices for credit decision-making," *European Journal of Operational Research*, vol. 200, no. 2, pp. 557–567, 2010.

[27] J. H. Zhong, M. Shen, J. Zhang, H. S. H. Chung, Y. H. Shi, and Y. Li, "A differential evolution algorithm with dual populations for solving periodic railway timetable scheduling problem," *IEEE Trans. Evolut. Comput.*, vol. 17, no. 4, pp. 512–527, Aug 2013.

[28] H. A. Abbass, R. Sarker, and C. Newton, "Pde: a pareto-frontier differential evolution approach for multi-objective optimization problems," in *Proceedings of the IEEE Congress on Evolutionary Computation, 2001*, vol. 2. IEEE, 2001, pp. 971–978.

[29] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii," *IEEE Trans. Evolut. Comput.*, vol. 13, no. 2, pp. 284–302, 2009.

[30] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Trans. Evolut. Comput.*, vol. 10, no. 6, pp. 646–657, 2006.

[31] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evolut. Comput.*, vol. 13, no. 2, pp. 398–417, 2009.

[32] A. Torn and A. Zilinskas, *Global optimization*. Springer-Verlag New York, Inc., 1989.

[33] R. Dawkins, *The selfish gene*. Oxford university press, 2016.

[34] X. Chen, Y.-S. Ong, M.-H. Lim, and K. C. Tan, "A multi-facet survey on memetic computation," *IEEE Trans. Evolut. Comput.*, vol. 15, no. 5, pp. 591–607, 2011.

[35] P. Moscato *et al.*, "On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms," *Caltech concurrent computation program, C3P Report*, vol. 826, p. 1989, 1989.

[36] J. Zhong and J. Zhang, "Adaptive multi-objective differential evolution with stochastic coding strategy," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM, 2011, pp. 665–672.