International Journal of Computational Intelligence Systems
Vol. **12(2)**, 2019, *pp.* 795–808
DOI: https://doi.org/10.2991/ijcis.d.190711.001; ISSN: 1875-6891; eISSN: 1875-6883
https://www.atlantis-press.com/journals/ijcis/

A Novel Memetic Framework for Enhancing Differential Evolution Algorithms via Combination With Alopex Local Search

Miguel Leon[1,*], Ning Xiong[1], Daniel Molina[2], Francisco Herrera[2]

[1]*IDT School of Innovation, Design and Engineering, Mälardalen University, Västerås, Sweden*
[2]*DaSCI Andalusian Institute of Data Science and Computational Intelligence, University of Granada, Granada, Spain*

ARTICLE INFO

Article History
Received 15 Oct 2018
Accepted 22 May 2019

Keywords
Differential evolution
L-SHADE
Memetic algorithm
Alopex
Local search
Optimization

ABSTRACT

Differential evolution (DE) represents a class of population-based optimization techniques that uses differences of vectors to search for optimal solutions in the search space. However, promising solutions/regions are not adequately exploited by a traditional DE algorithm. Memetic computing has been popular in recent years to enhance the exploitation of global algorithms via incorporation of local search. This paper proposes a new memetic framework to enhance DE algorithms using Alopex Local Search (MFDEALS). The novelty of the proposed MFDEALS framework lies in that the behavior of exploitation (by Alopex local search) can be controlled based on the DE global exploration status (population diversity and search stage). Additionally, an adaptive parameter inside the Alopex local search enables smooth transition of its behavior from exploratory to exploitative during the search process. A study of the important components of MFDEALS shows that there is a synergy between them. MFDEALS has been integrated with both the canonical DE method and the adaptive DE algorithm L-SHADE, leading to the MDEALS and ML-SHADEALS algorithms, respectively. Both algorithms were tested on the benchmark functions from the IEEE CEC'2014 Conference. The experiment results show that Memetic Differential Evolution with Alopex Local Search (MDEALS) not only improves the original DE algorithm but also outperforms other memetic DE algorithms by obtaining better quality solutions. Further, the comparison between ML-SHADEALS and L-SHADE demonstrates that applying the MFDEALS framework with Alopex local search can significantly enhance the performance of L-SHADE.

1. INTRODUCTION

Memetic computing is a hot research topic that has received increasing attention for more than one decade [1]. Memetic algorithms (MAs) are essentially global optimization methods such as evolutionary algorithms (EAs) that have been hybridized with some local search (LS) techniques. Inspired by Dawkin's notion of memetic units capable of local refinements, MAs usually apply an improvement scheme like a LS method on members of the population after evolutionary operators, with the purpose of exploiting the most promising search regions gathered during the global sampling by EA. A very important advantage of MAs is that they provide an effective and efficient way in which EAs and LS techniques can complement each other to compensate for both the deficiency of EAs in local exploitation and the inadequacy of LS in global exploration [2]. Some successful MAs [3] were proposed that exhibited their strength in obtaining reliable and precise solutions in complex and high dimensional spaces for continuous optimization problems.

Differential evolution (DE) represents a class of EAs that perform metaheuristic and population-based searches to solve many optimization problems [4]. In DE, the direction and magnitude of search is determined by the distribution of solutions in the population rather than a predefined probability density function. DE has been shown to offer a competitive tool for handling complex and high dimensional solution spaces, especially in real-parameter optimization tasks. Interesting surveys on recent advances in DE are given in [5].

Over the years, the usage of DE to solve real world problems is increasing. One reason for this is that unlike traditional optimization methods that utilize gradient information (that is not always available), DE only requires fitness values, thereby being applicable to a wider range of problems [6]. DE has shown its good performance in different real world scenarios like designing optimal harmonic filters [6], process modeling for greenhouses [7], as well as enhancing the contrast and brightness of satellite images [8]. However, DE also encounters two problems in some practical applications: slow convergence speed and stagnation in local optima [9]. To overcome these problems, a number of adaptive DE algorithms have been developed by adapting mutation strategies and control parameters in the optimization processes [10–15]. An alternative way of improving the performance of DE is to merge it with LS for enhancing the exploitation capability, leading to memetic DE.

Memetic DE algorithms have been proposed in recent years as enhancement of standard DE technique by incorporation of LS

*Corresponding author. Email: miguel.leonortiz@mdh.se

[9,16,17]. Some LS techniques and hybridization of them [18] have been introduced and utilized in this context to strengthen local refinement of individuals in the global search procedure. A successfully incorporated LS function is expected to not only speed up the convergence but also produce results of higher precision in situations when a traditional DE algorithm suffers from local stagnation [19].

Alopex (algorithm of pattern extraction) was originally proposed by Harth and Tzanakou for optimization and pattern matching in visual receptive fields [20]. Alopex can be considered as a derivative-free gradient descent method since it utilizes the correlation between changes of variables and changes in objective function values to estimate the gradient of the landscape. Alopex also adopts the similar idea of simulated annealing by using a temperature parameter to adaptively control the stochasticity of the moves.

This paper proposes a new memetic framework referred to as MFDEALS, which uses Alopex as the basic LS mechanism in cooperation with a DE algorithm. Unlike conventional memetic computing paradigms which implement exploration and exploitation independently, the novelty of the proposed work lies in its bridge in which the behavior of exploitation (by Alopex LS) can be controlled by the status (population diversity and search stage) of the global exploration by a DE algorithm. We believe that a seamless coupling of Alopex search with the DE cycle will offer valuable guidance to adapt the LS characteristics in favor of more cooperative performance of exploitation and exploration functions, which share limited computing resources in an evolutionary optimization process.

MFDEALS has been applied with the canonical DE and the adaptive DE algorithm L-SHADE [15] (the winner of the CEC'14 competition), leading to the MDEALS and ML-SHADEALS algorithms, respectively. Both algorithms were tested on the set of benchmark problems from IEEE CEC'2014 [21]. The results demonstrate that the MDEALS algorithm not only improves the canonical DE but also achieves superiority to other memetic DE algorithms employing LS. Moreover, ML-SHADEALS has been shown to outperform L-SHADE, indicating the benefit of the proposed framework when integrated with an adaptive DE algorithm.

The remainder of the paper is organized as follows. Section 2 explains the basis of Differential Evolution and L-SHADE. In Section 3, a review of memetic computing in DE is given. The description of the proposed MFDEALS framework is given in Section 4. The results of evaluations and comparison with other related algorithms are presented in Section 5. Finally, Section 6 gives the conclusion and future work.

2. BACKGROUND

In this section, we are going to briefly describe the original DE and L-SHADE algorithms, in Subsections 2.1 and 2.2, respectively.

2.1. Differential Evolution

DE is a stochastic algorithm maintaining a population with N_p individuals. Every individual in the population stands for a possible solution to the problem. An individual in the population is represented by vector $X_{i,g}$ with $i = 1, 2, \cdots, N$ and g referring to the index of the generation. A cycle in DE consists of three consecutive steps of operations: mutation, crossover, and selection which are described as follows:

2.1.1. Mutation

In this first step, N mutant vectors are created using individuals randomly selected from the current population. Indeed there are a few mutation strategies which can be used to generate mutant vectors, but only the random mutation strategy will be explained below. The other mutation strategies and their performance are discussed in [22]. The calculation of the mutant vector $V_{i,G}$ using the random mutation strategy is given in Equation (1).

$$V_{i,g} = X_{r_1,g} + F \cdot \left(X_{r_2,g} - X_{r_3,g} \right) \tag{1}$$

where $V_{i,g}$ represents the mutant vector, i stands for the index of the vector, g denotes the generation, $r_1, r_2, r_3 \in \{1, 2, \ldots, N\}$ are random integers and F is the mutation factor which usually lies in the interval [0, 2].

2.1.2. Crossover

This operation combines each individual in the population with the corresponding mutant vector created in the mutation stage. These new solutions created are called trial vectors, and we use $U_{i,g}$ to represent the trial vector corresponding to individual i in generation g. Every parameter in the trial vector is decided in terms of Equation (2).

$$U_{i,g}[j] = \begin{cases} V_{i,g}[j] & \text{if } rand[0,1] < CR \text{ or } j = j_{rand} \\ X_{i,g}[j] & \text{otherwise} \end{cases} \tag{2}$$

where j stands for the index of a parameter in the vector, J_{rand} is a randomly selected integer between 1 and N to ensure that at least one parameter from the mutant vector will be included in the trial vector, and CR is the probability of recombination.

2.1.3. Selection

This operation compares a trial vector and its parent solution in the current population to decide the winner to be propagated into the next generation. Therefore, if the problem of interest is minimization, the individuals in the new generation are chosen according to Equation (3).

$$X_{i,g+1} = \begin{cases} U_{i,g} & \text{if} f\left(U_{i,g}\right) < f\left(X_{i,g}\right) \\ X_{i,g} & \text{otherwise} \end{cases} \tag{3}$$

where $X_{i,g}$ is an individual in the population, $X_{i,g+1}$ is the individual in the next generation, $f\left(U_{i,g}\right)$ represents the objective value of the trial vector and $f\left(X_{i,g}\right)$ stands for the objective value of the individual in the current population.

2.2. L-SHADE

L-SHADE is an adaptive DE algorithm proposed in 2014 by Tanabe and Fukunaga [15]. L-SHADE is based on its predecessor SHADE [13], which adapts two of the main parameters in DE,

mutation factor (F) and crossover rate (CR). L-SHADE improves SHADE by a linear reduction of population size during search.

In L-SHADE, the first operation of mutation is performed by following the mutation strategy termed as DE/current-to-*p*Best/1. This mutation strategy was proposed in JADE [23] and it is described in Equation (4).

$$V_i = X_i + F_i * \left(X_{pBest} - X_i + X_{r_1} - Y_{r_2}\right) \tag{4}$$

where X_i represents the ith individual in the population, X_{pBest} is an individual randomly selected from the p best individuals of the population. p is a parameter that falls in the range $[0, 1]$, representing the percentage of the best individuals from the population. X_{r_1} is a random individual from the population and Y_{r_2} is randomly selected from the combination of the population (X) and the archive (A). This archive is used to store a group of several individuals that were removed from the population, to enforce diversity in the mutation. In cases the archive reaches the limit in size, it will be updated with the replacement of a random entity in it by the new one. Additionally, F_i is the mutation factor that is calculated as follows:

$$F_i = randc\left(M_{F,r_i}, 0.1\right) \tag{5}$$

where $randc\,(mean, std)$ stands for a random number generated using the Cauchy distribution with the mean equal to M_{F,r_i} and std equal to 0.1. M_{F,r_i} is a random element selected from the memory (M_F) that reflects the experiences of successful F values.

The second and third steps, crossover and selection, follow Equations (2) and (6), respectively. However, the parameter CR_i is not fixed in L-SHADE. It is generated according to a Normal distribution as follows:

$$CR_i = \begin{cases} 0 & \text{if } M_{CR,r_i} = \perp \\ randn\left(M_{CR,r_i}, 0.1\right) & \text{otherwise} \end{cases} \tag{6}$$

where $randn\,(mean, std)$ represents a random value that is created according to the Normal distribution with the mean equal to M_{CR,r_i} and std equal to 0.1. M_{CR,r_i} is a random element selected from the memory (M_{CR}) that reflects the experiences of successful CR values. If M_{CR,r_i} has been assigned with the terminal value $\perp$, then CR_i is set to 0.

Initially all the entities in both M_F and M_{CR} are set to 0.5 and then they are updated based on successful experiences in the optimization process. After each generation, one entity in M_F is replaced by the weighted Lehmer mean of the successful F values in the generation. Likewise, an entity in M_{CR} is replaced by the weighted Lehmer mean of the successful CR values in the generation. The calculation of the weighted Lehmer mean is given in Equations (7–9).

$$mean_{WL}(S) = \frac{\sum_{a=1}^{|S|} w_a \cdot S_a^2}{\sum_{a=1}^{|S|} w_a \cdot S_a} \tag{7}$$

$$w_a = \frac{\triangle f_a}{\sum_{b=1}^{|S|} \triangle f_b} \tag{8}$$

$$\triangle f_a = \left|f\left(U_{a,g}\right) - f\left(X_{a,g}\right)\right| \tag{9}$$

where S refers to the set of successful F or CR values depending on which memory to update. Specifically in case of no successful CR values from the generation, $M_{CR,k}$ is set to the terminal value $\perp$. Besides M_F and M_{CR} are updated cyclically with successive generations.

Finally, the population size is linearly reduced, depending on the status of the search. The population size is initialized to N^{init}, then it is reduced linearly with the number of evaluations. The population size of the next generation (N_{G+1}) is determined according to Equation (10).

$$N_{G+1} = round\left(NFE \cdot \frac{N^{min} - N^{init}}{MAX\_NFE} + N^{init}\right) \tag{10}$$

where N^{min} is set to 4, which is the minimum number of individuals in a population, NFE is the number of fitness evaluations conducted so far and MAX_NFE is the maximum number of fitness evaluations. For $N_{G+1} < N_G$, the population is sorted and the worst $(N_G - N_{G+1})$ individuals are deleted from the population.

3. MEMETIC COMPUTING WITH DE

When designing global search methods, exploration and exploitation are the two major issues of consideration [3,9]. Algorithms that explore too much will fail to find optimal solutions, while algorithms with too much exploitation can get stuck into a local optimum. A powerful algorithm is expected to explore the search space effectively while fine-tuning some promising solutions simultaneously.

Population based algorithms [24] such as DE usually trend to focus more on the exploration of the search space, whereas their convergence speed is not satisfying given limited computing resources [9]. To overcome this problem, some investigations were conducted by combining global search with various LS operators [3,25,26] including the local surrogate model based search [27] to improve the search performance. The LS operators, unlike population-based algorithms, trend to exploit the promising regions inside a search space.

Nevertheless, hybridization of global and LS algorithms is not a trivial task and a number of different issues have to be addressed. In Subsection 3.1, the computational expense allocated to LS algorithm is discussed. In Subsection 3.2, the different types of LS operators are reviewed.

3.1. Strategies of Allocating Resources to LS

Computational cost for LS, in this context of optimization, refers to the number of fitness evaluations used by the LS operator. Evidently, the amount of computation allocated to the LS operator will affect the trade-off between exploration and exploitation. In [28], the local/global search ratio $\left(\frac{L}{G}\right)$ is defined as the the percentage of the number of evaluations that have been used by the LS operator. This ratio is related to the following three factors that have to be considered when designing a memetic algorithm.

The first factor concerns which individual(s) from the population are going to be affected by LS. This is problem dependent, since the convergence speed will change depending on the design. Applying LS to a lot of individuals may increase the convergence speed, however, if not many individuals are actually modified, the whole population will not converge fast enough. Different methods have been studied to handle this factor. The most commonly used scheme is to modify only the best individual in the population [29–31]. An alternative approach is to apply LS to a part of the population. In [32], the LS operator was applied to a percentage of the best individuals of the population. Poikolainen [25] proposed to apply LS to the individuals with their fitness better than the average of the fitness of the whole population. In [33], a different option was presented, in which LS was applied with a certain probability to all individuals in the population.

The second factor is about how often the LS operator is applied in the DE cycle. Some works conducted LS after every generation [29,32,33], while others applied LS after a specified number of generations [25,17].

The third factor is associated with the length of the LS operator, which is defined as the number of trial solutions that are tested before the termination of the LS. According to [9], the methods of deciding search length can be divided into three groups:

- *Fixed length LS*: The length of LS is defined beforehand and it remains unchanged through the whole search. In [29], the length is set to dimension divided by 5. In the other algorithm [33], the length of the LS is set to one.
- *Dynamic length LS*: A predefined rule is used to decide the intensity of LS without using any feedback from the search. For instance, it was suggested in [34] to apply LS more at the beginning yet less at the end of the search.
- *Adaptive length LS*: Despite some similarity to the second group, the approaches in this group utilize feedback from the search in order to decide the intensity of the LS operator. Some examples can be found in [9,31,35].

3.2. LS Methods in DE

Many different LS methods have been proposed for being integrated into DE. Dominguez-Isidro *et al.* [36] tested some traditional LS methods, such as Nelder-Mead method (NM) [37], Hooke-Jeeves [38] and hill-climbing [39] within the DE framework. A hill climber was also used by Mandal *et al.* [40]. Chaos search was proposed by Jiang [41] as a LS method based on chaos theory. According to [9,28], the various LS methods that have been applied in DE can be divided into two categories:

- *Local improvement process (LIP) oriented LS (LLS)*: Local improvements are performed on one or more individuals from the population. Some well-known techniques in this category include gradient descent and hill-climbing.
- *Crossover-based LS (XLS)*: A predefined number of individuals from the population are recombined in order to create new trial solutions. XLS methods have an attractive property of self-adaptation [42], since the population is evolving as the search progresses.

The LS methods that belong to the categories LLS and XLS are reviewed in the following:

LLS in DE: There are many methods that belong to this category. One of the trends is to apply Chaos search [41], in which local perturbations are made. Pei-Chong proposed DECH [32], a memetic algorithm that hybridizes Chaos search together with DE. In DECH, small perturbations are made to 20% of the individuals of the population. A new generated solution will replace the original one if the new one is better. In a similar work conducted by Jia *et al.* [29], chaotic LS was only applied to the best individual of the population after each generation.

A different strategy was addressed in [25] with the proposal of the algorithm called DE with concurrent fitness-based local search (DEcfbLS). The mechanism of LS used therein is very similar to Hooke-Jeeves algorithm [38]. The basic idea is to perform separated searches on single variables with iterations. If the search fails (without finding improvement) in one direction or after one iteration, the step size of search will be halved for subsequent trials. This LS method was applied in [25] to multiple promising solutions of the population, following a fitness-based selection rule.

Random perturbation based strategies were proposed in [31] for applying LS on partial dimensions of the best individual in the population. The Uniform, Normal and Cauchy distributions were employed there for generating random moves, leading to the following three variant DE algorithms, respectively: DE with random local search (DERLS), DE with Normal local search (DENLS), and DE with Cauchy local search (DECLS).

XLS in DE: Crossover-based LS has attracted much attention. Ali *et al.* [30] proposed two LS strategies: Trigonometric Local Search (TLS) and Interpolated Local Search (ILS) for being used in a basic DE cycle. TLS is based on the Trigonometric Mutation Operator [43], which shifts the center point of a hyper geometric triangle along the three legs (of the same triangle). ILS attempts to build a parabolic curve to fit into the randomly selected points from the population and then to identify the point on the curve with the minimum objective value. It was shown in the paper that using these LS schemes in DE could improve the quality of obtained solutions while not affecting the convergence speed.

Differential Evolution with Orthogonal Local Search (OLSDE) was proposed in [44], in which orthogonal design (OD) is used to generate a set of trial solutions around two randomly selected parents and the best trial solution then replaces the worst individual in the population. Prior to performing OD, the variables have to be divided into groups based on statistic information from the population and a set of discrete levels are determined for each variable in order to construct the orthogonal array. A similar method, called Taguchi LS (TLS) was proposed in [16]. In TLS, two random individuals are taken from the population and the middle point between them is calculated. These three individuals are divided into four different parts, then, nine new individuals are created as a combination of these parts. Using the fitness of the nine new individuals the effect of each parent on each part is calculated. A final offspring is created combining the parts from the parents that has a better effect.

Noman and Iba [9] proposed an enhanced DE algorithm with crossover-based adaptive LS, which is called DEachSPX. Simplex search is used in DEachSPX as the LS method, and the length of the LS is adapted by using a hill-climbing heuristic. Indeed

DEachSPX is an improvement of the previous algorithm DEfirSPX [45], which uses a fixed length for the crossover-based LS. An even earlier work in this direction was presented in [28], in which the DExhcSPX algorithm was developed with the use of hill-climbing search for DE.

4. ALOPEX BASED MEMETIC FRAMEWORK TO ENHANCE DE ALGORITHMS

In this paper we propose a new Memetic framework, referred to as MFDEALS, to enhance DE algorithms using Alopex Local Search (ALS). First, the ALS mechanism used inside a DE cycle is described in Subsection 4.1, which is followed by the presentation of the proposed MFDEALS framework together with its differences from our preceding work in Subsection 4.2.

4.1. ALS in DE

Alopex is originally an algorithm to solve combinatorial optimization and pattern matching problems. This method measures the local correlation between changes in variables and the change of the objective function value, in order to decide the direction of move in search. The ALS function that is integrated into the DE cycle will be described in the following.

Let $A_i = (a_{i,1}, a_{i,2}, \dots, a_{i,N})$ be the ith solution in generation g that will undergo LS, we first select a reference solution $B_i = (b_{i,1}, b_{i,2}, \dots, b_{i,N})$ from the population as the reference point. B_i is selectedas either the best individual or a random individual from the population, with the probabilities γ and $(1-\gamma)$, respectively. Then the correlations between the variable and objective value changes for A_i with respect to B_i are calculated according to Equation (11) as follows:

$$C^g_{i,j} = (a_{i,j} - b_{i,j}) \cdot [f(A_i) - f(B_i)] \quad for\, j = 1, 2 \dots, D \tag{11}$$

where f denotes the objective function to optimize, D is the number of variables. $C^g_{i,j}$ measures the relation of the local change in variable j and the change in the objective function. A positive correlation indicates that a move of the variable toward the selected individual would lead to an increase of the objective value, and vice versa.

The next step is to decide the probabilities of move directions for all variables using the derived correlations. Considering a minimization problem without loss of generality, the probability for a positive move on the jth variable in A_i is yielded as

$$P_{i,j} = \frac{1}{1 + e^{\frac{C^g_{i,j}}{T_g}}} \tag{12}$$

where T_g is an evolving parameter as temperature to reflect the diversity of the population. T_g is set to 1 in the first generation and later it is calculated as the mean of correlations in the previous generation. Hence the temperature for Generation g $(g > 1)$ is written as

$$T_g = \frac{1}{D} \cdot \frac{1}{|P_S|} \cdot \sum_{i \in P_S} \sum_{j=1}^{N} |C^{g-1}_{i,j}| \tag{13}$$

where P_S denotes the set of indices of individuals that received LS in Generation $g-1$.

The ratio of the correlation between one variable and the temperature controls the stochasticity of search in the corresponding dimension. A very small ratio will give a probability near to 0.5, so that the search is substantially biased toward a completely randomized walk in that dimension. On the contrary, a large correlation (relative to the temperature) will make the search to have a more deterministic nature on the associated variable. It follows that, large T in the early generations of the DE cycle will contribute to increase the stochasticity of the LS, while small T when the population tends to converge will enforce the chance to move in the direction to decrease the objective value.

The trial solution $Q_i = (q_{i,1}, q_{i,2}, \dots, q_{i,D})$, which is hopefully to be superior to A_i, is generated by

$$q_{i,j} = a_{i,j} + \delta_{i,j} \cdot |b_{i,j} - a_{i,j}| \cdot \alpha \; for\; j = 1, 2, \dots, D \tag{14}$$

where α is a scaling factor generated by the normal distribution $N(0, \lambda)$ with its standard deviation being λ. In the implementation, α is truncated or regenerated, respectively if the generated value is larger than one or negative. $\delta_{i,j}$ represents the direction of the move as given by

$$\delta_{i,j} = \begin{cases} 1 & \text{if } P_{i,j} \geqslant rand(0,1) \\ -1 & \text{otherwise} \end{cases} \tag{15}$$

where rand(0,1) is a uniform random number between 0 and 1. If this generated random number is smaller than the probability $P_{i,j}$, set $\delta_{i,j} = +1$ meaning a positive move in variable j, otherwise a negative movein the variable is decided.

If the trial solution Q_i is better than A_i, it will replace A_i and become a member of the population. Otherwise the search procedure moves on to the next individual selected for local improvement. The ALS function: ALS (OldT, Population) is formally described in Algorithm 1, where OldT represents the temperature calculated from the previous generation.

According to the categories of LS as described in Subsection 3.2, ALS employed in DE can be considered as a crossover-based method. The reason is that ALS takes two individuals from the population and the new trial solution is created via recombination of both. This method also has a self-adaptive property, as individuals of the population are evolving such that the magnitude of perturbations will vary depending on the current state of the search.

4.2. Combining ALS With DE

In this subsection, we illustrate how ALS can be combined with a DE algorithm in the proposed MFDEALS framework. There are four parameters that have to be set with this framework. The first one is the parameter γ, which defines the probability of selecting the best individual of the population as the reference point for ALS. It controls the greediness of the LS. If γ is close to 1, there will be a high chance to move toward the best individual and exploit the region nearby, while if γ is close to 0, a random search will be performed instead. The other three parameters correspond to the three

Algorithm 1: Function ALS(oldT, population, λ, γ, *threshold*)

Data: oldT, Population, λ, γ, *threshold*
Result: X, newT, *numEval*
$X = Population$;
$\alpha = -1$;
while $\alpha \leqslant 0$ **do**
  $\alpha = min(1, N(0, \lambda))$;
end
$numEval = 0$;
for i = *1 to N* **do**
  if $f(X_{i,g}) <= threshold$ **then**
    $A_i = X_{i,g}$;
    if *rand(0,1)* $< \gamma$ **then**
      Select B_i as the best individual from Population$\backslash\{A_i\}$;
    else
      Select B_i randomly from Population$\backslash\{A_i\}$;
    end
    for j = *1 to N* **do**
      $C^g_{i,j} = (a_{i,j} - b_{i,j}) \cdot [f(A_i) - f(B_i)]$;
      $P_{i,j} = \frac{1}{1+e^{\frac{C^g_{i,j}}{oldT}}}$;
      $\delta_{i,j} = \begin{cases} 1 & \text{if } P_{i,j} \geqslant rand(0,1) \\ -1 & \text{otherwise} \end{cases}$;
      $q_{i,j} = a_{i,j} + \delta_{i,j} \cdot |a_{i,j} - b_{i,j}| \cdot \alpha$;
    end
    if $f(Q_i) < f(A_i)$ **then**
      $X_{i,g} = Q_i$;
    end
    $numEval = numEval + 1$;
  end
end
newT = average of the calculated correlations as shown in Eq (13);
Return: $[X, newT, numEval]$

different factors that affect the Local/Global ratio, as mentioned in Subsection 3.1:

- The amount of individuals affected by ALS
- The frequency to apply ALS
- The length of the ALS operator

Regarding the first factor, the main idea in MFDEALS is to apply ALS to a part of promising individuals in the population to promote their further refinements. In order to decide which individuals will receive ALS, we introduce a variable parameter *threshold*, which is defined based on fitness of the individuals of the population. We suggest five alternative ways to set the value of this parameter :

- $threshold = f(X_{best})$
- $threshold = \frac{average(fitnesses)+f(X_{best})}{2}$
- $threshold = average\,(fitnesses)$
- $threshold = \frac{average(fitnesses)+f(X_{worst})}{2}$
- $threshold = f(X_{worst})$

where $f(X_{best})$ is the fitness of the best individual of the population, $f(X_{worst})$ is the fitness of the wort individual of the population, and $average\,(fitnesses)$ stands for the average of all the individuals of the population. The parameter *frequencyLS* is used to determine after how many generations ALS will be applied. If, for instance, *frequencyLS* is set to 1, then ALS will be applied at each generation. *lengthLS* controls the third factor, indicatingthe number of iterations when ALS is applied to a generation.

Additionally, in the ALS function there is a special parameter λ, which is the standard deviation (std) of the normal distribution that generates the scaling factor used in Equation (14). λ is designed to be adaptive to the current state of the search by DE. At early stages, λ is a value slightly smaller than 0.5 such that the LS operator will have an exploratory ability, while in the later stages λ will be close to 0 leading to more fine-tuning ability of ALS. *lambda* is adapted according to Equation (16).

$$\lambda = 0.5 - \frac{currentEval}{maxEval} \cdot 0.5 + 0.0001 \tag{16}$$

where *currentEval* is the current number of evaluations and *maxEval* denotes the maximum number of evaluations, and 0.0001 is a tiny constant added to ensure that the standard deviation is always more than 0. The pseudo-code of the whole MFDEALS algorithm is given in Algorithm 2.

There are several clear differences between MFDEALS and the preceding algorithm Differential Evolution with Alopex Local Search (DEALS) [46]. These differences can be outlined in two aspects: differences in the LS schemes and differences in the choices on the factors related to the $\frac{L}{G}$ ratio. In the first aspect, there are three points of differences that need to noted.

- **In relation to parameter α:** This parameter is used in Equation (14). In DEALS, the parameter α is calculated by creating a random number following a uniform distribution in the range $(0, 1)$ ($rand\,(0, 1)$), while in MFDEALS α is generated using a Normal distribution with 0 as the mean and λ as the std. Further, the variation of the λ parameter will make the magnitude of the move adaptable to the state of the search. To be known, ALS will exploremore at the beginning of the search, while it will focus more on exploitation of promising regions in the later stages.
- **In relation to the temperature (T):** In MFDEALS the temperature is derived from correlations from the preceding generation using Equation (13). However, in DEALS, T is calculated at each iteration as the mean of the local correlations along all dimensions. Obviously more correlations are involved in MFDEALS for the temperature calculation.
- **In relation to the choice of reference solution for ALS:** DEALS always selects a random individual from the

Algorithm 2: MFDEALS

```
Initialize the population S(0) with randomly
  created individuals;
g=0; T = 1;
Define γ, frequencyLS, lengthLS;
while NFE < MAX_NFE do
    for i = 1 to N do
        Perform mutation on the ith individual;
        Perform crossover on the ith
          individual;
        Perform selection on the ith individual;
    end
    NFE = NFE + N;
    if g%frequencyLS == 0 then
        for i = 1 to lenghLS do
            Calculate threshold;
            λ = 0.5 − NFE/MAX_NFE * 0.5 + 0.0001;
            [S(g+1), T, numEval] = ALS(T,
              S(g+1), λ, γ threshold);
            NFE = NFE + numEval;
        end
    end
    g = g + 1;
end
```

population as the reference point for LS, while in MFDEALS a probabilistic chance will decide whether the best individual or a random individual from the population is selected.

Regarding the $\frac{L}{G}$ ratio, MFDEALS is different from DEALS in the following two issues:

- **In relation to the number of individuals to be improved by LS:** DEALS applies LS only to the best individual in the population, while MFDEALS applies ALS to all the individuals that have their fitness better than a threshold. This change means that more individuals will undergo local improvement in MFDEALS.
- **In relation to the length of the LS operator:** In DEALS, the LS operator is assigned with a minimum search length. But the counter will restart when an improvement is found within this minimum length, resulting in the possibility of over-long search lengths in practice. MFDEALS attempts to avoid excessive usages of resources in LS by setting the length of ALS to a fixed value.

5. RESULTS OF EXPERIMENTS

To evaluate the efficacy of the proposed framework, we have combined it with the canonical DE and L-SHADE algorithms, with the latter being the winner of the CEC'2014 competition. The new memetic DE algorithms yielded from such combinations are termed as MDEALS and ML-SHADEALS, respectively.

A number of experiments have been carried out. First, the important components of MDEALS were studied together with its parameters. Second, MDEALS was compared with the canonical DE algorithm as well as our previous work (DEALS algorithm). Third, MDEALS was compared with the state-of-the-art memetic DE algorithms. Lastly, ML-SHADEALS was compared with L-SHADE to demonstrate the potential of our framework to further improve top DE algorithms.

5.1. Experimental Settings

The CEC'2014 benchmark [21] is used here to examine the performance of our proposed framework. The benchmark is composed of 30 functions, which are grouped into four categories: Unimodal functions (F1–F3), multimodal functions (F4–F16), Hybrid Functions (F17–F22), and Composition functions (F23–F30).

The various DE variants that were tested on the benchmark includes: MDEALS and ML-SHADEALS (created due to our framework), canonical DE, DEALS, L-SHADE, as well as the other nine memetic DE algorithms. Every algorithm was run 30 independent times on each of the 30 functions. The maximum number of function evaluations (FEs) is set to 300 000 given the dimension of the problems as 30. If the objective value obtained by an algorithm is better than 1.00E-08, it is considered that the optimal solution of the function is found.

5.2. Parameter Study Inside MDEALS

To perform the parameter study, the DE parameters were set as: $F = 0.5$, $CR = 0.5$, and $N = 100$. The proposed MDEALS algorithm has one important parameter to study: γ (probability of selecting the best individual from the population when applying ALS). Eleven different values of γ have been tested and the results are shown in Table 1. In this table comparisons are given in terms of two different metrics: the average of the ranks on all the functions (avg. rank) and the sums of the relative errors with respect to the worst option on each function (s.r.e.). It can be observed that values close to 0 or 1 do not give good results. The reason is simple: the values close to 1 make all the different individuals converge to the best one such that the population will gradually get stuck. On the other hand, values close to 0 will make ALS focus on exploring the search space instead of exploiting promising regions. The test results also show that the best values for γ are in the interval [0.3, 0.4].

Table 1 Study of the parameter gamma (γ).

Metrics	γ= 0	γ= 0.1	γ= 0.2	γ= 0.3	γ= 0.4	γ= 0.5	γ= 0.6	γ= 0.7	γ= 0.8	γ= 0.9	γ= 1
avg. rank	7.35	5.82	5.15	4.28	**4.02**	4.88	4.62	5.92	7.05	7.75	9.17
s.r.e.	20.41	18.06	17.03	**16.74**	17.59	17.96	**16.74**	17.94	20.06	21.49	25.01

We also conducted tests on the three factors that are related to the ratio between local and global searches. In order to study the first factor about the amount of individuals that are affected by the LS operator, different methods of deciding the value for *threshold* were used and tested in MDEALS. This means that ALS has been applied to different parts of the population as suggested in Subsection 4.2, for example, the best individual, individuals with the fitness better than the averaged fitness, individuals with the fitness better than the average between the best and averaged fitness, individuals with the fitness better than the average between the worst and averaged fitness, as well as the whole population. The results are shown in Table 2. It can be observed from the table that applying ALS to the individuals with the fitness better than the average is the best option. If ALS is applied to a small number of individuals, the population does not converge properly and only a few solutions progress fast. On the other hand, applying ALS to all the individuals may consume too many resources for LS and result in an improper local/global ratio.

The second factor is about how often the LS operator is applied, that is, after how many generations the LS operator is to be applied (*frequencyLS*). The results obtained from testing different generation intervals are shown in Table 3. It can be observed that applying ALS after every generation gives the best results. Performing ALS with a lower frequency will make the population converge slowly.

The third factor concerns the length of LS (*lengthLS*). We did tests of LS with different numbers of iterations and the results are illustrated in Table 4. It can be seen that a small length of LS is the proper choice. If the LS operator has a higher search length, it will use too many resources and consequently the global search by DE will not be able to evolve through enough generations.

The appropriate choices concerning the different factors in our MDEALS algorithm are stated below:

- $\gamma : [0.3, 0.4]$.
- $threshold = average\,(fitnesses)$
- $frequencyLS = 1$
- $lenghtLS = \{1, 2\}$

5.3. Comparison of MDEALS With DEALS

In this subsection, a comparison is made between the proposed MDEALS algorithm and our preceding method DEALS [46]. In the experiments the DE parameters were set as: $F = 0.5$, $CR = 0.5$, and $N = 100$. Additionally, γ was set to 0.3, *frequencyLS* set to 1, *lengthLS* set to 1, and *threshold* set to $average\,(fitnesses)$ in MDEALS as the best parameter values obtained from the experiment results as shown in Subsection 5.2. Table 5 shows the results of MDEALS, DEALS, and Basic DE, respectively, where the numbers are the mean errors from the 30 independent runs of each algorithm. Sum. rel. error stands for the sum of the relative errors and average rank is the mean of the ranks of an algorithm across all the benchmark functions. The bottom part of the table shows the one-to-one comparisons of MDEALS with respect to DEALS and DE, respectively. It is evident that MDEALS outperforms the other two methods in 24 out of the 30 functions. In Table 6, the results of the Wilcoxon's statistical tests are indicated. We can see clearly that MDEALS is statistically better than basic DE and DEALS algorithms.

Apart from the aforementioned experiments, we also checked the effectiveness of ALS in comparison with DE. This was done by calculating the success rates of the two parts (DE and ALS) of the MDEALS algorithm. The results are shown in Table 7. In this table, Columns 1 and 4 represent the number of evaluations that DE and ALS used, respectively, Columns 2 and 5 represent how many trials were successful and Columns 3 and 6 show the success rates for DE and ALS, respectively. It can be observed that the success rates of ALS were higher in all the functions except F14, on which the success rates of both were similar.

5.4. Comparison of MDEALS With Other Memetic DE Algorithms

In this subsection, we present the experimental evaluations of comparison of MDEALS with the state-of-art memetic DE algorithms. They are summarized in the following:

- DE [4]: The canonical DE algorithm introduced in Section 2.1.
- DETLS [30]: A variant DE algorithm that uses a trigonometric mutation operator by calculating the center point of a triangle.
- DEILS [30]: A variant DE algorithm that builds a parabolic curve in order to yied a new solution.
- LSDE [33]: A DE variant employing LS operator as a weighted combination of an individual with the best individual from the population to produce a new solution.
- DEachSPX [9]: A DE algorithm using NM as the LS operator.
- DEChaosLS [29]: A memetic algorithm hybridizing DE with a LS operator based on chaos search.

Table 2 Study of the variable *threshold*.

Metrics	$f\,(X_{best})$	$\frac{average\,(fitnesses) + f\,(X_{best})}{2}$	$average\,(fitnesses)$	$\frac{average\,(fitnesses) + f\,(X_{worst})}{2}$	$f\,(X_{worst})$
avg. rank	4.08	2.90	**2.15**	2.90	2.97
s.r.e.	25.41	19.43	**17.53**	19.43	19.75

Table 3 Study of the parameter *frequencyLS*.

Metrics	1	2	3	4	5	6	7	8	9
avg. rank	**3.08**	3.45	3.75	4.52	5.02	5.10	6.20	6.70	7.18
s.r.e	**20.54**	**20.54**	21.41	22.62	23.50	23.76	24.95	26.04	25.87

Table 4 Study of the parameter *lengthLS*.

Metrics	1	2	3	4	5	6	7
avg. rank	3.47	**3.43**	3.57	3.83	4.07	4.83	4.80
s.r.e.	22.32	**21.94**	23.34	23.42	23.01	24.54	25.77

Table 5 Results of MDEALS, DEALS, and DE on the benchmark problems from CEC 2014 with dimension 30.

Func.	MDEALS	DEALS	DE
avg. rank	**1.3**	2	2.7
s.r.e.	**17.46**	25.87	27.35
Better than (+)	-	**24**	**24**
Similar to (=)	-	2	2
Worse than (−)	-	4	4

DE, differential evolution.

Table 6 Wilcoxon's statistical test (p value = 0.05) of MDEALS against DEALS and DE.

MDEALS vs	R+	R−	Exact *P*-value	Diff.?
DEALS	381.5	53.5	1.651E-04	**Yes**
DE	381.5	53.5	1.651E-04	**Yes**

DE, differential evolution.

Table 7 Comparison of the success rates between two parts (DE and ALS) within MDEALS.

Func.	DE Eval	DE Succ	DE Succ Rate (%)	ALS Eval	ALS Succ	ALS Succ Rate (%)
F1	169220	1422.8	0.84	130878.4	15292.8	**11.68**
F2	116600	20536.2	17.61	67336.6	21211.2	**31.50**
F3	124900	9818.4	7.86	77469.6	10062.8	**12.99**
F4	193720	58040.8	29.96	106337.2	50385.8	**47.38**
F5	206960	720.0	0.35	93103.0	345.2	**0.37**
F6	191920	16277.4	8.48	108137.2	19137.0	**17.70**
F7	56000	16008.4	28.59	31730.6	11801.2	**37.19**
F8	196640	2119.4	1.08	103394.6	4987.2	**4.82**
F9	201100	1306.2	0.65	98984.6	4632.4	**4.68**
F10	198980	1627.4	0.82	101097.4	3114.8	**3.08**
F11	209240	725.2	0.35	90804.6	2217.4	**2.44**
F12	200200	701.8	0.35	99898.0	594.8	**0.60**
F13	202060	1753.0	0.87	98020.8	931.8	**0.95**
F14	191620	2014.6	**1.05**	108485.8	1097.2	1.01
F15	203580	2079.4	1.02	96524.0	3703.6	**3.84**
F16	208820	788.4	0.38	91234.2	1052.0	**1.15**
F17	165640	790.2	0.48	134438.8	6278.4	**4.67**
F18	163340	2473.2	1.51	136719.4	6494.0	**4.75**
F19	196760	3374.8	1.72	103324.6	4856.4	**4.70**
F20	183940	1611.4	0.88	116120.8	3279.6	**2.82**
F21	168380	1075.6	0.64	131670.0	5247.0	**3.98**
F22	201020	1754.8	0.87	99016.2	2257.4	**2.28**
F23	171400	14978.2	8.74	128668.8	13934.6	**10.83**
F24	191100	4246.0	2.22	108948.0	3984.4	**3.66**
F25	168280	1345.0	0.80	131810.4	18604.2	**14.11**
F26	200900	1763.4	0.88	99159.6	1406.8	**1.42**
F27	193880	4968.4	2.56	106261.6	11053.4	**10.40**
F28	222080	11951.4	5.38	77966.0	13418.8	**17.21**
F29	173600	2300.0	1.32	126507.4	8297.6	**6.56**
F30	187240	2947.6	1.57	112836.6	11142.2	**9.87**

ALS, Alopex local search; DE, differential evolution.

- DETaguchiLS [16]: A DE variant that combines different parts of individuals to create new solutions.
- DEcfbLS [25]: A DE variant that uses the deterministic Hooke-Jeeves algorithm as the LS operator.
- DENM [36]: A DE algorithm employing the NM for LS.

A more detailed description of the above algorithms can be found in Section 3 or the corresponding references.

The parameters of DE were set as follows: $F = 0.5$, $CR = 0.5$, population size (N) = 100. Then various LS methods were adopted and integrated into DE by following the respective papers and the recommended parameter values. The extra parameters for some of the algorithms are given below:

- MDEALS: $\gamma = 0.3$, $threshold = average\left(fitnesses\right)$, $frequencyLS = 1$ and $lengthLS = 1$.
- LSDE: $p = 0.1$.
- DEahcSPX: $\varepsilon = 1$ and $n_p = 3$.
- DEChaosLS: m = 1500.
- DEcfbLS: iter = 12 and $\rho = 0.4$

The results of the experiments are given in Table 8, in which the numbers are the average errors of the solutions found by the

Table 8 | Results of MDEALS and other memetic DE algorithms on the benchmark suit from CEC 2014.

Func.	MDEALS	DEachSPX	DEChaosLS	DEcfbLS	DEILS	DETLS	DENM	DETaguchiLS	LSDE
F1	**1.76E+06**	1.27E+07(+)	3.48E+07(+)	2.81E+07(+)	2.86E+07(+)	1.75E+07(+)	1.81E+07(+)	1.88E+07(+)	5.85E+07(+)
F2	**0.00E+00**	**0.00E+00**(=)	**0.00E+00**(=)	**0.00E+00**(=)	**0.00E+00**(=)	**0.00E+00**(=)	**0.00E+00**(=)	**0.00E+00**(=)	**0.00E+00**(=)
F3	**0.00E+00**	9.62E-08(+)	5.35E-06(+)	2.72E-05(+)	7.71E-07(+)	1.13E-07(+)	6.45E-06(+)	4.92E-07(+)	1.26E-05(+)
F4	**6.73E+01**	6.84E+01(+)	7.09E+01(+)	7.27E+01(+)	7.00E+01(+)	6.92E+01(+)	7.11E+01(+)	6.84E+01(+)	7.10E+01(+)
F5	**2.07E+01**	2.09E+01(+)	2.09E+01(+)	2.07E+01(+)	2.09E+01(+)	2.09E+01(+)	2.09E+01(+)	2.09E+01(+)	2.09E+01(+)
F6	2.16E+00	9.27E+00(+)	9.47E+00(+)	1.77E+01(+)	1.12E+01(+)	4.06E+00(+)	4.47E+00(+)	**9.47E-01**(−)	2.53E+01(+)
F7	1.64E-03	**0.00E+00**(−)	**0.00E+00**(−)	**0.00E+00**(−)	**0.00E+00**(−)	**0.00E+00**(−)	**0.00E+00**(−)	**0.00E+00**(−)	**0.00E+00**(−)
F8	1.92E+01	8.69E+01(+)	7.73E+01(+)	3.26E+01(+)	8.90E+01(+)	9.01E+01(+)	2.32E+01(+)	**1.40E+00**(−)	9.02E+01(+)
F9	**4.82E+01**	1.66E+02(+)	1.21E+02(+)	5.60E+01(+)	1.68E+02(+)	1.66E+02(+)	7.41E+01(+)	1.56E+02(+)	1.67E+02(+)
F10	1.51E+03	3.28E+03(+)	2.90E+03(+)	1.52E+03(+)	3.26E+03(+)	3.33E+03(+)	4.82E+02(−)	**4.74E+01**(−)	3.30E+03(+)
F11	3.84E+03	6.41E+03(+)	5.44E+03(+)	**3.26E+03**(−)	6.46E+03(+)	6.47E+03(+)	4.20E+03(+)	6.50E+03(+)	6.36E+03(+)
F12	1.07E+00	1.97E+00(+)	1.58E+00(+)	**8.79E-01**(−)	1.95E+00(+)	1.98E+00(+)	1.54E+00(+)	1.98E+00(+)	1.96E+00(+)
F13	**2.25E-01**	3.63E-01(+)	3.03E-01(+)	3.03E-01(+)	3.75E-01(+)	3.46E-01(+)	3.36E-01(+)	3.57E-01(+)	3.79E-01(+)
F14	**2.16E-01**	2.77E-01(+)	2.48E-01(+)	2.51E-01(+)	2.65E-01(+)	2.63E-01(+)	2.62E-01(+)	2.67E-01(+)	2.66E-01(+)
F15	**6.97E+00**	1.51E+01(+)	1.43E+01(+)	9.90E+00(+)	1.57E+01(+)	1.52E+01(+)	1.55E+01(+)	1.53E+01(+)	1.56E+01(+)
F16	1.12E+01	1.23E+01(+)	1.18E+01(+)	**1.12E+01**(−)	1.23E+01(+)	1.23E+01(+)	1.20E+01(+)	1.22E+01(+)	1.24E+01(+)
F17	**8.70E+04**	4.13E+05(+)	6.86E+05(+)	6.03E+05(+)	8.03E+05(+)	4.63E+05(+)	5.39E+05(+)	1.12E+06(+)	1.55E+06(+)
F18	1.10E+03	**8.68E+02**(−)	2.56E+03(+)	3.91E+03(+)	1.11E+03(+)	1.13E+03(+)	4.53E+03(+)	2.01E+03(+)	2.80E+03(+)
F19	**5.43E+00**	5.95E+00(+)	6.16E+00(+)	6.18E+00(+)	6.12E+00(+)	5.74E+00(+)	6.19E+00(+)	5.93E+00(+)	6.14E+00(+)
F20	6.11E+01	7.40E+01(+)	1.20E+02(+)	1.49E+02(+)	1.11E+02(+)	**5.59E+01**(−)	1.51E+02(+)	7.76E+01(+)	1.37E+02(+)
F21	**6.88E+03**	1.95E+04(+)	5.44E+0(+)	6.18E+04(+)	4.40E+04(+)	2.75E+04(+)	3.97E+04(+)	3.34E+04(+)	8.53E+04(+)
F22	9.76E+01	1.84E+02(+)	1.17E+02(+)	**7.84E+01**(−)	1.71E+02(+)	1.56E+02(+)	1.21E+02(+)	1.62E+02(+)	2.14E+02(+)
F23	3.15E+02	3.15E+02(=)	3.15E+02(=)	3.15E+02(−)	3.15E+02(=)	3.15E+02(=)	3.15E+02(−)	3.15E+02(=)	**2.00E+02**(−)
F24	2.26E+02	2.23E+02(−)	2.23E+02(−)	2.23E+02(−)	2.23E+02(−)	2.23E+02(−)	2.23E+02(−)	2.23E+02(−)	**2.00E+02**(−)
F25	2.04E+02	2.11E+02(+)	2.04E+02(−)	2.14E+02(+)	2.14E+02(+)	2.11E+02(+)	2.12E+02(+)	2.13E+02(+)	**2.00E+02**(−)
F26	**1.00E+02**	1.00E+02(+)	1.00E+02(+)	1.00E+02(+)	1.00E+02(+)	1.00E+02(+)	1.00E+02(+)	1.00E+02(+)	1.00E+02(+)
F27	3.33E+02	3.00E+02(−)	3.01E+02(−)	3.00E+02(−)	3.00E+02(−)	3.00E+02(−)	3.01E+02(−)	3.00E+02(−)	**2.00E+02**(−)
F28	8.66E+02	7.82E+02(−)	7.82E+02(−)	7.83E+02(−)	7.94E+02(−)	7.83E+02(−)	7.89E+02(−)	7.88E+02(−)	**2.46E+02**(−)
F29	**1.90E+03**	3.97E+03(+)	6.18E+03(+)	5.19E+03(+)	3.70E+03(+)	2.21E+03(+)	6.03E+03(+)	4.82E+03(+)	6.46E+03(+)
F30	**1.92E+03**	2.47E+03(+)	2.72E+03(+)	2.92E+03(+)	2.58E+03(+)	2.72E+03(+)	2.78E+03(+)	1.96E+03(+)	2.92E+03(+)
s.r.e.	**16.86**	21.58	22.39	21.42	22.86	21.16	20.81	20.13	25.20
avg. rank	**2.8**	4.9	4.9	4.8	6.0	4.9	5.3	5.1	6.3
Better than (+)	-	**23**	**23**	**20**	**24**	**23**	**23**	**21**	**23**
Similar to (=)	-	2	2	1	2	2	1	2	1
Worse than (−)	-	5	5	9	4	5	6	7	6

DE, differential evolution.

algorithms in 30 independent runs for each function. The best result on each function is marked with boldface. The bottom of this table summarizes the results by giving the sums of relative errors, the average rank, as well as the numbers of functions on which MDEALS is better(+), equal(=) or worse(−). Table 9 shows, for every group of functions, the numbers of functions in which MDEALS is better than, equal to or worse than its counterparts. From these two tables, we can claim that MDEALS is the best algorithm for all the unimodal functions. Regarding the composite functions, MDEALS outperforms the other methods in almost all the functions. In the multimodal functions, MDEALS is better than its counterparts in between 69% and 92% of the functions. On the hybrid functions, LSDE is the best algorithm and MDEALS is similar to all other algorithms.

In addition, the Friedman's statistical test, using Holms correction, has been performed to compare the proposed MDEALS algorithm against its counterparts. The results are given in Table 10. It can be observed that, according to this statistical test, MDEALS achieved significant superiority to each of the approaches in comparison, with a maximum statistical error of 5%.

Moreover, the convergence speeds of the compared algorithms are shown in Figure 1. This figure illustrates the evolution of the averaged fitness of the population vs. the numbers of evaluations for each of the algorithms in comparison. It can be observed that MDEALS converges much faster than the other memetic DE algorithms on the majority of the functions.

Table 9 | MDEALS vs other memetic DE algorithms in terms of function groups.

	Unimodal Functions			Multimodal Functions		
Algorithm	**Better**	**Equal**	**Worst**	**Better**	**Equal**	**Worse**
DEachSPX	**2**	1	0	**12**	0	1
DEChaosLS	**2**	1	0	**12**	0	1
DEcfbLS	**2**	1	0	**9**	0	4
DEILS	**2**	1	0	**12**	0	1
DETLS	**2**	1	0	**12**	0	1
DENM	**2**	1	0	**11**	0	2
TETaguchiLS	**2**	1	0	**9**	0	4
LSDE	**2**	1	0	**12**	0	1
	Composite Functions			**Hybrid Functions**		
Algorithm	**Better**	**Equal**	**Worst**	**Better**	**Equal**	**Worse**
DEachSPX	**5**	0	1	**4**	1	3
DEChaosLS	**6**	0	0	3	1	**4**
DEcfbLS	**5**	0	1	**4**	0	**4**
DEILS	**6**	0	0	**4**	1	3
DETLS	**5**	0	1	**4**	1	3
DENM	**6**	0	0	**4**	0	**4**
TETaguchiLS	**6**	0	0	**4**	1	3
LSDE	**6**	0	0	3	0	**5**

DE, differential evolution.

Table 10 | Friedman's statistical test and Post-Hoc Holm's Test ($\alpha = 0.05$) between MDEALS and other memetic DE algorithms.

Algorithm	***p* Value**	p_{Holm}	**Statistical Difference?**
LSDE	1.00E-06	5.00E-06	**Yes**
DEILS	1.00E-05	7.30E-05	**Yes**
DENM	3.72E-04	0.002233	**Yes**
DEChaosLS	0.002361	0.011807	**Yes**
DEDLS	0.002979	0.011918	**Yes**
DETaguchiLS	0.004033	0.012099	**Yes**
DETLS	0.005821	0.012099	**Yes**
DEachSPX	0.010909	0.012099	**Yes**

5.5. Combining MFDEALS With L-SHADE

This subsection shows the results of combining MFDEALS with L-SHADE [15], the winner of the CEC'2014 competition. We use ML-SHADEALS to denote this new combined DE variant.

Owing to the L-SHADE characteristics such as large initial population and greedy mutation strategy, a different parameter setting is needed. The number of individuals affected by ALS is reduced such that less exploitation is performed at the beginning of the search. For the same reason, ALS is applied less often in ML-SHADEALS. The parameter specification within ML-SHADEALS is given as follows:

- $\gamma = 0.3$
- $threshold = \frac{average(fitnesses)+f(X_{best})}{2}$
- $frequencyLS = 4$
- $lengthLS = 1$
- Parameters in L-SHADE [15]: $N^{init} = 540$, $N^{arc} = 78$, $p = 0.11$ and $H = 6$, as specified in their paper.

We compared the performance of ML-SHADEALS with L-SHADE on the CEC'2014 benchmark functions. The results, given in Table 11, show that combining MDEALS with L-SHADE can produce even better solutions than L-SHADE. This is also verified by the results of the Wilcoxon's statistical test, as shown in (Table 12), demonstrating that the performance of ML-SHADEALS is significantly better than that of L-SHADE.

Table 11 | Results of ML-SHADEALS and L-SHADE on the benchmark suit from CEC 2014.

Func.	**ML-SHADEALS**	**L-SHADE**
F1	**0.00E+00**	**0.00E+00**
F2	**0.00E+00**	**0.00E+00**
F3	**0.00E+00**	**0.00E+00**
F4	**0.00E+00**	**0.00E+00**
F5	**2.01E+01**	**2.01E+01**
F6	**0.00E+00**	**0.00E+00**
F7	**0.00E+00**	**0.00E+00**
F8	**0.00E+00**	**0.00E+00**
F9	**7.46E+00**	7.62E+00
F10	5.55E-03	**3.47E-03**
F11	**1.18E+03**	1.23E+03
F12	**1.56E-01**	1.57E-01
F13	**1.19E-01**	1.20E-01
F14	2.36E-01	**2.32E-01**
F15	**2.02E+00**	2.08E+00
F16	**8.41E+00**	8.51E+00
F17	2.89E+02	**2.07E+02**
F18	**7.57E+00**	8.06E+00
F19	**3.72E+00**	3.73E+00
F20	3.00E+00	**2.84E+00**
F21	**1.13E+02**	1.19E+02
F22	**2.38E+01**	2.47E+01
F23	**3.15E+02**	**3.15E+02**
F24	**2.24E+02**	**2.24E+02**
F25	**2.03E+02**	**2.03E+02**
F26	**1.00E+02**	**1.00E+02**
F27	**3.00E+02**	**3.00E+02**
F28	**8.44E+02**	8.49E+02
F29	7.20E+02	**7.18E+02**
F30	**1.73E+03**	2.07E+03
Better than	-	12
Similar than	-	13
Worse than	-	5

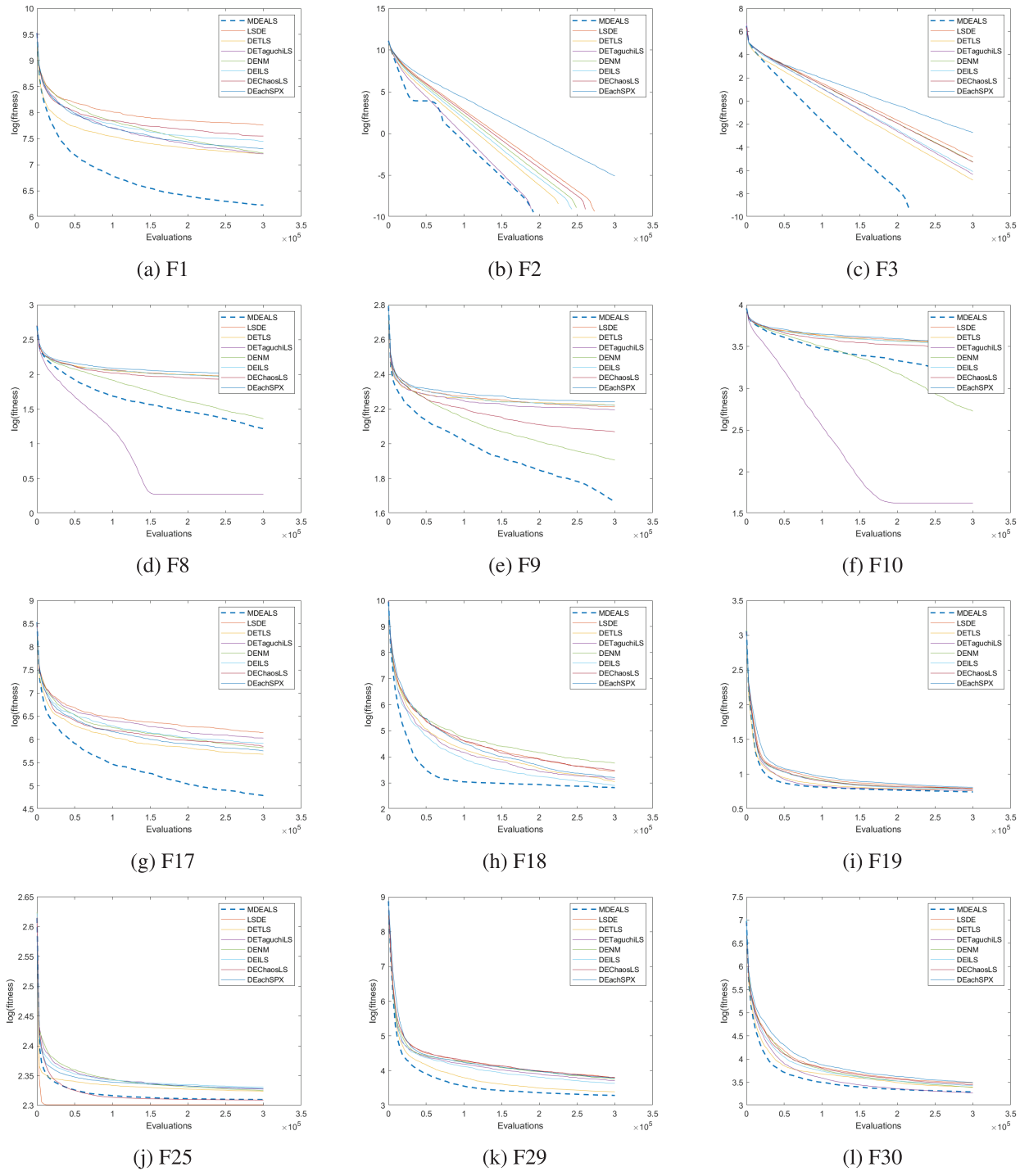

Figure 1 Mean converge characteristics of MDEALS, LSDE, DETLS, DETaguchiLS, DENM, DEILS, DEChaosLS, DEachSPX on 12 functions from CEC2014 benchmark set. All results are mean values of 30 runs.

Table 12 Wilcoxon's statistical test ($\alpha = 0.05$) between ML-SHADEALS and L-SHADE.

ML-SHADEALS vs.	R+	R-	Exact P-value	Statistical difference?
L-SHADE	319	116	0.0274	**Yes**

6. CONCLUSION

DE is a powerful population-based optimization technique, and its hybridization with LS can improve the results. This paper proposes a new MFDEALS. MFDEALS contains an adaptable parameter enabling more exploration at the beginning of the search while more exploitation at a later stage. Moreover, MFDEALS adopts temperature as an internal parameter to affect the directions of moves in ALS based on the correlations of individuals in the population. As information from the global search (by DE) is well utilized to control the behavior of ALS, MFDEALS is deemed to bring a better balance between exploration and exploitation in the overall optimization process.

MFDEALS has been combined with the canonical DE and L-SHADE algorithms. These two algorithms have been tested in the set of benchmark functions proposed in CEC'2014. The results show that the combination between our memetic framework and DE outperformed other memetic DE algorithms in a large majority of the benchmark functions. Additionally, the power of our framework has been demonstrated by improving the results of L-SHADE by combining L-SHADE with our memetic framework.

CONFLICT OF INTEREST

There no conflict of interest.

AUTHORS' CONTRIBUTIONS

Miguel Leon have proposed the main idea of the paper, implemented the algorithm, performed the experiments and wrote the major draft of the paper. Ning Xiong, Daniel Molina and Francisco Herrera have contributed with some writting and helpful comments to further enhance the quality of the paper.

ACKNOWLEDGMENTS

This research was supported by grants from both Swedish Research Council (project number 2016-05431) and Spanish Ministry of Science TIN2016-8113-R.

REFERENCES

[1] F. Neri, C. Cotta, Memetic algorithms and memetic computing optimization: a literature review, Swarm Evol. Comput. 2 (2012), 1–14.
[2] S. Kumar, V.K. Sharma, R. Kumari, Memetic search in differential evolution algorithm, Int. J. Comput. Appl. 90 (2014), 40–47.
[3] D. Molina, M. Lozano, C. Garcia-Martinez, F. Herrera, Memetic algorithms for continuous optimization based on local search chains, Evol. Comput. 18 (2010), 27–63.
[4] R. Storn, K. Price, Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces, Tech report tr-95-012, Computer Science Institute, Berkeley, 1995.
[5] S.S. Mullick, P.N. Suganthan, S. Das, Recent advances in differential evolution - an updated survey, Swarm Evol. Comput. 27 (2016), 1–30.
[6] M. Leon, Y. Zenlander, N. Xiong, F. Herrera, Design optimal harmonic filters in power systems using greedy adaptive differential evolution, in IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), Berlin, 2016, pp. 1–7.
[7] A. Perez-Gonzalez, O. Begovich-Mendoza, J. Ruiz-Leon, Modeling of a greenhouse prototype using PSO and differential evolution algorithms based on a real-time lab view application, Appl. Soft Comput. 62 (2018), 86–100.
[8] D. Suresh, S. Lal, Modified differential evolution algorithm for contrast and brightness enhancement of satellite image, Appl. Soft Comput. 61 (2017), 622–641.
[9] N. Noman, H. Iba, Accelerating differential evolution using an adaptative local search, IEEE Trans. Evol. Comput. 12 (2008), 107–125.
[10] M. Leon, N. Xiong, Adapting differential evolution algorithms for continuous optimization via greedy adjustment of control parameters, J. Artif. Intell. Soft Comput. Res. 6 (2016), 103–118.
[11] L. Cui, G. Li, Q. Lin, J. Chen, N. Lu, Adaptive differential evolution algorithm with novel mutation strategies in multiple sub-populations, Comput. Oper. Res. 67 (2016), 155–173.
[12] G. Li, Q. Lin, L. Cui, Z. Du, Z. Liang, J. Cheng, Z. Ming, A novel hybrid differential evolution algorithm with modified code and jade, Appl. Soft Comput. 47 (2016), 577–599.
[13] R. Tanabe, A. Fukinga, Success-history based parameter adaptation for differential evolution, in IEEE Congress on Evolutionary Computation (CEC), Cancun, 2013, pp. 71–78.
[14] L. Cui, G. Li, Z. Zhu, Q. Lin, K.-C. Wong, J. Chen, N. Lu, J. Lu, Adaptive multiple-elites-guided composite differential evolution algorithm with a shift mechanism, Info. Sci. 422 (2018), 122–143.
[15] R. Tanabe, A.S. Fukunaga, Improving the search performance of shade using linear population size reduction, in IEEE Congress on Evolutionary Computation (CEC), Beijing, 2014, pp. 1658–1665.
[16] H. Peng, Z. Wu, Heterozygous differential evolution with taguchi local search, Soft Comput. 19 (2015), 3273–3291.
[17] A.K. Qin, K. Tang, H. Pang, S. Xia, Self-adaptive differential evolution with local search chains for real-parameter single-objective optimization, in IEEE Congress on Evolutionary Computation (CEC), Beijing, 2014, pp. 467–474.
[18] A. Caponio, F. Neri, V. Tirronen, Super-fit control adaptation in memetic differential evolution, Soft Comput. 13 (2009), 811–831.
[19] J. Lampinen, I. Zelinka, On stagnation of the differential evolution algorithm, in Proceedings of MENDEL, Brno, 2000, pp. 76–83.
[20] E. Harth, E. Tzanakou, Alopex: a stochastic method for determining visual receptive fields, Vis. Res. 14 (1974), 1475–1482.
[21] J.J. Liang, B. Qu, P.N. Suganthan, Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization, Technical report, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, 2013.
[22] M. Leon, N. Xiong, Investigation of mutation strategies in differential evolution for solving global optimization problems,in Artificial Intelligence and Soft Computing, Springer, Zakopane, 2014, pp. 372–383.
[23] J. Zhang, A.C. Sanderson, Jade: adaptive differential evolution with optional external archive, IEEE Trans. Evol. Comput. 13 (2009), 945–958.
[24] N Xiong, D. Molina, M. Leon, F. Herrera, A walk into metaheuristics for engineering optimization: principles, methods, and recent trends, Int. J. Comput. Intell. Syst. 8 (2015), 606–636.

[25] I. Poikolainen, F. Neri, Differential evolution with concurrent fitness based local search, in Proceeding of 2013 IEEE Congress on Evolutionary Computation (CEC), Cancun, 2013, pp. 384–391.
[26] X. Chen, T.S. Ong, M.H. Lim, K.C. Tan, A multi-facet survey on memetic computation, IEEE Trans. Evol. Comput. 15 (2011), 591–607.
[27] A. Zhou, J. Sun, Q. Zhang, An estimation of distribution algorithm with cheap and expensive local search methods, IEEE Trans. Evol. Comput. 19 (2015), 807–822.
[28] M. Lozano, F. Herrera, N. Krasnogor, D. Molina, Real-coded memetic algorithms with crossover hill-climbing, Evol. Comput. 12 (2004), 273–302.
[29] D. Jia, G. Zheng, M.K. Khan, An effective memetic differential evolution algorithm based on chaotical search, Info. Sci. 181 (2011), 3175–3187.
[30] M. Ali, M. Pant, A. Nagar, Two local search strategies for differential evolution, in Proceeding of Bio-Inspired Computing: Theories and Applications (BIC-TA), 2010 IEEE Fifth International Conference, Changsha, 2010, pp. 1429–1435.
[31] M. Leon, N. Xiong, Differential evolution enhanced with eager random search for solving real-parameter optimization problems, Int. J. Adv. Res. Artif. Intell. 4 (2015), 49–57.
[32] W. Pei-chong, Q. Xu, H. Xiao-hong, A novel differential evolution algorithm based on chaos local search, in Proceeding of International Conference on Information Engineering and Computer Science, ICIECS 2009, Wuhan, 2009, pp. 1–4.
[33] J. Gu, G. Gu, Differential evolution with a local search operator, in 2nd International Asia Conference on Informatics in Control and Robotics, Wuhan, 2010, pp. 480–483.
[34] T. Back, Introduction to the special issue: self-adaptation, IEEE Trans. Evol. Comput. 9 (2001), 3–4.
[35] A. LaTorre, S. Muelas, A mos-based dynamic memetic differential evolution algorithm for continuous optimization: a scalability test, Soft Comput. 15 (2011), 2187–2199.
[36] S. Dominguez-Isidro, E. Mezura-Montes, G. Leguizamon, Performance comparison of local search operators in differential evolution for constrained numerical optimization problems, in Differential Evolution (SDE), 2014 Symposium, Orlando, 2014, pp. 1–8.
[37] J. A. Nelder, R.A. Mead, A simplex for function minimization, Comput. J. 7 (1965), 308–313.
[38] K. Deb, Optimization for Engineering Design Algorithms and Examples, New Delhi, Prentice Hall of India, 1995.
[39] S. Russell, P. Norvig, Artificial Intelligence, Pearson Education Limited Prentice-Hall, Malaysia, 2016.
[40] A. Mandal, A.K. Das, P. Mukherjee, S. Das, P.N. Suganthan, Modified differential evolution with local search algorithm for real world optimization, in Proceeding of the 2011 IEEE Congress on Evolutionary Computation (CEC), New Orleans, 2011, pp. 1565–1572.
[41] B. Jiang. Optimizationion complex functions by chaos search, Cybern. Syst. 29 (1998), 409–419.
[42] H.G. Beyer, K. Deb, On self-adaptive feature in real-parameter evolutionary algorithms, IEEE Trans. Evol. Comput. 5 (2011), 250–270.
[43] H.Y. Fan, J. Lampinen, A trigonometric mutation operation to differential evolution, J. Global Optim. 27 (2003), 105–129.
[44] Z. Dai, A. Zhou, A diferential ecolution with an orthogonal local search, in Proceeding of the 2013 IEEE congress on Evolutionary Computation (CEC), Cancun, 2013, pp. 2329–2336.
[45] N. Noman, H. Iba, Enhancing differential evolution performance with local search for high dimensional function optimization, in Proceedings of the 2005 Conference on Genetic and evolutionary computation, GECCO'05, Washington, 2005, pp. 967–974.
[46] M. Leon, N. Xiong, A new differential evolution algorithm with alopex-based local search, in International Conference in Artificial Intelligence and Soft Computing (ICAISC), Zakopane, 2016, pp. 420–431.