九州大学学術情報リポジトリ
Kyushu University Institutional Repository

Local Fitness Landscape from Paired Comparison-Based Memetic Search in Interactive Differential Evolution and Differential Evolution

Pei, Yan
Graduate School of Design, Kyushu University

髙木, 英行
九州大学大学院芸術工学研究院

https://hdl.handle.net/2324/1807799

出版情報：International Journal of Ad Hoc and Ubiquitous Computing. 25 (1/2), pp.17-30, 2017. Inderscience Enterprises
バージョン：
権利関係：

KYUSHU UNIVERSITY

Local Fitness Landscape from Paired Comparison-Based Memetic Search in Interactive Differential Evolution and Differential Evolution

Yan Pei

School of Computer Science and Engineering
the University of Aizu
Tsuruga, Ikki-machi, Aizu-Wakamatsu, Fukushima, 965-8580 Japan
email: peiyan@u-aizu.ac.jp

Hideyuki Takagi

Faculty of Design
Kyushu University
4-9-1 Shiobaru, Minamiku, Fukuoka, 815-8540, Japan
email: takagi@design.kyushu-u.ac.jp

Abstract: We propose a triple comparison-based interactive differential evolution (IDE) algorithm and differential evolution (DE) algorithm. The comparison of target vector and trial vector supports a local fitness landscape for IDE and DE algorithms to conduct a memetic search. In addition to the target vector and trial vector used in canonical IDE and DE algorithm frameworks, we conduct a memetic search around whichever vector has better fitness. We use a random number from a normal distribution generator or a uniform distribution generator to perturb the vector, thereby generating a third vector. By comparing the target vector, the trial vector, and the third vector, we implement a triple comparison mechanism in IDE and DE algorithms. A Gaussian mixture model is used as a pseudo-IDE user for evaluating the IDE and 25 benchmark functions from the CEC2005 test suite are employed to evaluate the DE. We compare our proposals with canonical IDE and triple comparison-based IDE implemented by opposite-based learning and apply several statistical tests to investigate the significance of our proposed algorithms. We also compare our proposals with several evaluation metrics, such as number of function calls, success rate and acceleration rate. Our proposed triple comparison-based IDE and DE algorithms show significantly better optimization performance arising from the evaluation results. We also investigate potential issues arising from our proposal and discuss some open topics and future opportunities.

Keywords: evolutionary computation, interactive evolutionary computation, interactive differential evolution, paired comparison, triple comparison

Reference to this paper should be made as follows: Yan Pei and Hideyuki Takagi, 'Local Fitness Landscape from Paired Comparison-Based Memetic Search in Interactive Differential Evolution and Differential Evolution', *International Journal of Ad Hoc and Ubiquitous Computing*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Yan PEI is an Assistant Professor at School of Computer Science and Engineering, the University of Aizu, Japan. He received a doctorate from Kyushu University, Japan, in 2014. He received the B. Eng. and M. Eng. Degrees from Northeastern University, China in 2006 and 2009, respectively. Dr. Pei's research interests involve in computational intelligence and machine learning. He is a member of IEEE SMC, IEEE CIS and Japanese society for evolutionary computation.

Hideyuki TAKAGI received the doctoral degree in 1991. He worked for the Panasonic Central Research Labs in 1981 - 1995, was a visiting researcher at the UC Berkeley in 1991 - 1993 and moved to Kyushu Institute of Design in 1995. Now, he is a Professor of Kyushu University, Japan. Dr. Takagi is quite active in interactive evolutionary computation research. He received several awards such as best paper awards in 1997, 1998, 2001, 2004 and 2012. He was/is a volunteer for IEEE SMC Society such as Vice-President, Board of Governors member, TC Chair, Chapter Chair, Associate Editor, and Distinguish Lecturer.

1 Introduction

Evolutionary computation (EC) is a meta-heuristic technique that is used to solve complex problems which are hard to solve using conventional optimization methods [13]. Interactive EC (IEC) is a niche research field within the EC community that embeds the feeling, knowledge, and experience of a real human into EC optimization, so as to make IEC algorithm converge to a real human's preference rather than to the fitness function(s) of an optimized problem. Extending the range of IEC applications scale and enhancing IEC algorithm performance (including improving IEC interface) are two primary research subjects within the IEC field. One of these involves applying IEC optimization principles and techniques to a variety of industrial and commercial applications that require the assistance of a real human in the optimization process. The other pursues the discovery of more effective and efficient IEC algorithms or interfaces to obtain a better optimization result, while at the same time relieving human fatigue due to psychological and physiological limitations of real humans when they interact with an IEC algorithm.

There are three research perspectives in IEC for obtaining a more effective and efficient IEC algorithm and interface [5]. The first is to approximate the fitness landscape of a real human's subjective evaluation space, attempting to build optimized problem structures to assist the IEC search. Several methods have been proposed which deal with this aspect, such as dimensionality reduction techniques [7], Fourier analysis [6], and support vector regression [8]. The second perspective is to develop a new search mechanism within a canonical IEC algorithm to enhance its performance or to design a better interface, which improves human-computer communication. A kernel method-based human model is studied to assist the IEC user in reducing his/her fatigue [10]. The third research approach is to create a new IEC/EC algorithm in order to achieve better performance of IEC/EC optimization [3, 4]. This paper takes the second perspective and tries to develop a new search mechanism in canonical differential evolution to enhance its optimization performance.

Differential evolution (DE) is a type of population-based optimization algorithm [11]. It searches for the global optimum using a differential vector between two individuals for which the length is in proportion to the distribution size of the individuals in general and for which each parent individual generates its offspring. DE has the characteristics of a paired comparison scheme where there is a competition between the target vector and the trial vector. Its benefit for IEC application is that it allows a real human to give fitness based on the paired comparison of two objects rather than to give multiple fitness values for several objects at the same time. Paired comparison-based IDE does not modify any parts of its algorithm because the algorithm already includes paired comparison by nature. Since pairs of individuals are presented to the IDE user for comparison without modifying the canonical DE algorithm, the IDE is expected to be a promising IEC method. Because each human has his or her limitations for assigning fitness to IEC algorithms from one generation to the next, the paired comparison can significantly relieve IEC user fatigue [14]. Reference [9] uses the opposite point(s) of a target vector and/or a trial vector from opposition-based learning for implementing triple and quadruple comparison schemes in canonical IDE and DE. This can significantly enhance optimization performance of canonical IDE and DE, especially for relieving user fatigue in IDE applications.

By drawing inspiration from multiple comparison implementation in IDE algorithm, fitness landscapes can support information on search conditions and problem structures to develop a new multiple comparison-based IDE algorithm. There are a variety of representations of the fitness landscape. One specific representation from an IDE algorithm is that of the fitness of a target vector and trial vector. When the fitness of the target vector is better than that of the trial vector, it indicates that searching around the target vector has the potential to obtain the global optimum with higher probability, and vice versa. Based on this hypothesis, in this paper, we propose a new triple comparison-based IDE, which conducts a memetic search around whichever is the better vector, be it the target vector or the trial vector. We implement the memetic search by perturbation with a normal distribution generator or a uniform distribution generator. The novel memetic search of the local fitness landscape obtained from a paired comparison of target vector and trial vector highlights the originality of this work. We use a Gaussian mixture model (GMM) as a pseudo IDE user to evaluate our proposal, and 25 benchmark problems to evaluate our proposal in DE. From the evaluation of our proposal, this new triple comparison-based IDE/DE algorithm can obtain a better optimization performance than a canonical IDE/DE algorithm.

Following this introductory section, we briefly review conventional paired comparison-based IDE and DE, triple and quadruple comparison-based IDE and DE by using opposition-based learning in section 2. Section 3 presents our new triple comparison-based IDE and DE by memetic search with the local fitness landscape obtained from a paired comparison of target vector and trial vector. The memetic search is implemented by conducting a perturbation from a normal distribution generator or a uniform distribution generator. In sections 4 and 5, we evaluate our proposal by using GMM and 25 benchmark functions for IDE and DE, respectively We analyze and discuss some open topics and issues arising from the evaluation results of IDE and DE in each section. Finally, in section 6, we conclude the current work and present some future opportunities, which invite investigation.

Copyright © 2015 Inderscience Enterprises Ltd.
Copyright © 2009 Inderscience Enterprises Ltd.

2 A Brief Review of Related Works

2.1 Notations

In this section, we present a brief review on canonical DE, paired comparison-based IDE, triple and quadruple comparison-based IDE, and related concepts, such as optimization and opposition-based learning. We make some notations in advance.

In general, the optimization process can be described as follows. Given the following single objective function $\{f : R^n \longrightarrow R\}$, the optimization algorithm seeks the point $\mathbf{x} \in R^n$, for which $f(\mathbf{x})$ has the minimal value.

There are some vector concepts in IDE/DE algorithm, i.e., base vector ($base_{i,j}$), target vector ($target_{i,j}$), trial vector ($trial_{i,j}$), and mutant vector ($mutant_{i,j}$), i and j is the index of individual and index of dimension, respectively. There are two special parameters in DE, scale factor F and crossover rate Cr. The same concepts and notations also appear in the IDE algorithm.

2.2 Differential Evolution

Differential evolution (DE) is a population-based optimization algorithm [11]. DE uses a differential vector from two random individuals to perturb a base vector (the vector with best fitness value (DE-best) or a third random vector (DE-rand) from a population) to implement a mutation operation and obtain a mutant vector. It conducts a crossover operation between the mutant vector and the target vector to create a trial vector. Following this, it compares the fitness of the target vector and trial vector to enable the better one to survive into the next generation. The formal expression of this search mechanism is shown in Eq. (1). F is a scale factor that needs to be set whose range is usually within $(0, 2]$ from the discussion of [11]. Note that the target vector, base vector, and two random vectors are four different vectors, so the minimum size of a population is four in DE.

$$mutant_{i,j} = base_{i,j} + F * (x1_{i,j} - x2_{i,j}). \tag{1}$$

2.3 Paired Comparison-Based Interactive Differential Evolution

When the individuals of an IEC optimization are voice or video, i.e., time series objects, IEC users have to compare an individual with others in their memory. As a result, IEC users' mental stress and fatigue become overwhelming. It has been pointed out that human beings have limited memory and cannot process more than five to nine different pieces of information simultaneously [2]. The population sizes of many IEC systems frequently exceed this memory limitation. Consequently, displaying $10 - 20$ voices, images or videos to an IEC user is not practical.

Algorithm 1 Paired comparison-based interactive differential evolution and differential evolution algorithm. PS: population size; Dim: dimension; G: generation; $maxIter$: maximum generation; i: index of individual; j: index of dimension; f(*) is a fitness function.

```
Generate an initial population.
Evaluate the fitness for each individual.
for $G = 1$ to $maxIter$ do
  for $i = 1$ to $PS$ do
    $k$=rand(1,Dim)
    for $j = 1$ to $Dim$ do
      if $rand[0,1) < C_r$ or $j == k$ then
        $mutant_{i,j} = base_{i,j} + F * (x1_{i,j} - x2_{i,j})$
        $trial_{i,j} = mutant_{i,j}$
      else
        $trial_{i,j} = target_{i,j}$
      end if
    end for
  end for
  /*Paired Comparison Mechanism*/
  for $i = 1$ to $PS$ do
    if $f(trial_i) < f(target_i)$ then
      replace $target_i$ with $trial_i$
    end if
  end for
end for
return the optimum
```

Paired comparison-based IEC solves this problem by replacing comparison of all individuals with paired comparisons. This is expected to reduce IEC user fatigue. One of the concrete implementations of paired comparison is a tournament interactive genetic algorithm (IGA) [1]. The obtained fitness of tournament IGA has noise because the tournament is not a round robin competition against the canonical IGA algorithm. The noise influences an IGA selection operation so as to reduce IGA search performance. One promising subject for future research is to improve the method by which efficient paired comparison-based IEC algorithms are implemented.

The IDE/DE algorithm includes paired comparison naturally as part of Algorithm 1. Paired comparison-based IDE does not revise any parts of its canonical DE algorithm [14]. Since it displays paired comparisons of individuals to an IDE user with the canonical DE algorithm, the IDE algorithm is expected to be a promising paired comparison IEC method.

2.4 Multiple Comparison-Based Interactive Differential Evolution

2.4.1 Opposite Point

Opposition-based learning (OBL) [15] is used for machine learning [16] and acceleration of optimization search (OBL optimization). The philosophy of OBL is that if the original hypothesis is not adequate, we should consider it in respect to its opposite hypothesis. Suppose that $x \in [a, b]$ is a real number, the opposition point of x is given as $OP(x) = a + b - x$. By extending this principle into a multi-dimensional space, opposition point, $OP(X)$, of one point on a n-dimensional real space, $X = (x_1, x_2, ..., x_n)$ $(x_i \in [a_i, b_i]; i = 1, 2, ..., n; a_i, b_i \in R)$, is given by Eq.s (2) and (3).

$$OP(x_i) = a_i + b_i - x_i. \tag{2}$$

$$OP(X) = \{OP(x_1), OP(x_2), ..., OP(x_n)\}. \tag{3}$$

2.4.2 Triple and Quadruple Comparison-Based Interactive Differential Evolution

The triple and quadruple comparison-based IDE uses not only a target vector and a trial vector, but also their opposition vector(s) at every comparison in the IDE search [9]. There are three implementations for combining target vector, trial vector, and their opposition vector(s). Two triple comparison-based IDEs are implemented by comparing a target vector, a trial vector and either the opposite point of the target vector or the opposite point of the trial vector. A quadruple comparison-based IDE is implemented by comparing a target vector, a trial vector, and the opposite points of the target vector and the trial vector.

Two different mirror points for calculating opposition points can be used. One is the center gravity point of an individual distribution, the other is the whole searching space because a big shift of individuals may accelerate DE convergence especially in the early generations. From the empirical study of these two methods, there is not a significant difference between the two methods [9]. Accordingly, we use the latter one, whole searching ranges, in our experimental evaluation.

3 Memetic Search in Interactive Differential Evolution and Differential Evolution for Implementing a Multiple Comparison Mechanism

3.1 Local Fitness Landscape from Paired Comparison

The fitness landscape is originally a biological concept that is used to visualize the relationship between a biological entity and its evolutionary process. In the evolutionary optimization field, it presents the solution of optimized problems and the extent to which these problems are capable of being solved. Most such problems can be represented by fitness function(s). In IEC, the fitness landscape can act as a tool to analyze human models of physiology or psychology, which present a human's preference, according to the optimized objective of an IEC application.

In the IDE/DE, the fitness of the target vector and the trial vector supports a local fitness landscape when comparing their paired comparison fitness value. The individual (either target vector or trial vector) with better relative fitness indicates a search region where there may be a global optimum. The implied information about search conditions and landscape provide the opportunity to improve IDE/DE optimization performance. Figure 1 illustrates the local fitness landscape for the paired comparison of a target vector and trial vector in the IDE/DE. If the fitness of the trial vector is better than that of the target vector, it means that by searching around the trial vector, there would be a great probability of finding the global optimum, and vice versa. The promising search direction is from the vector with lower fitness to the vector with higher fitness, and the promising search range is around the vector with higher fitness.

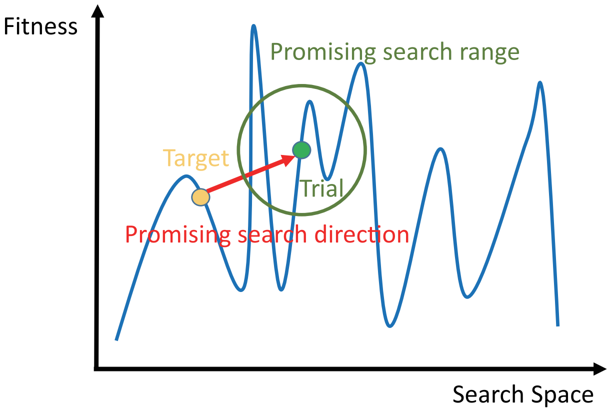


Figure 1 The local fitness landscape for the paired comparison of the target vector and the trial vector in IDE/DE, taking a maximum optimization problem as an example. The fitness of the trial vector is higher than that of the target vector in this figure, meaning that the promising search range is around the trial vector, the promising search direction is from the target vector to the trial vector, and vice versa. We apply memetic search by perturbing the vector with better fitness by a generator of some distribution (we use a uniform distribution generator or a normal distribution generator in this paper) to implement a triple comparison-based IDE/DE algorithm.

3.2 Memetic Search for Implementing a New Triple Comparison Mechanism

With the local fitness landscape obtained from paired comparison in IDE, we propose a memetic search method in the IDE algorithm to implement a new triple comparison-based IDE/DE. When we find a vector with higher fitness from comparison of target and trial vectors, the IDE/DE algorithm can conduct a memetic search by perturbing the vector to generate a third vector in order to implement a triple comparison mechanism. Here the originality of our proposal can be seen. The promising search direction is from the one with lower fitness to the other, and perturbation can be implemented by adding a random number from a generator. In this paper, we use a normal distribution (Eq. (4), $\mu = 0, \sigma = 1$) or a uniform distribution (Eq. (5), $a = 0, b = 1$) as generators in our experimental evaluation.

$$N(x, \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} exp(\frac{-|x-u|^2}{2\sigma^2}). \qquad (4)$$

$$U(x, a, b) = \frac{1}{b-a}. \qquad (5)$$

Besides a target vector and a trial vector in the canonical IDE/DE framework, the third vector is from perturbation on whichever the better of the target and trial vectors. Eq.s (6) and (7) show the two implementations of the third vector ($third_{i,j}$) from a normal distribution generator and a uniform distribution generator, respectively. Abbreviations, $better_{i,j}$ and $worse_{i,j}$, are whichever the vector with better and worse fitness from target vector and trial vector in Eq.s (6) and (7). Thus we can perform a memetic search in the IDE/DE algorithm. After we obtain a third vector from Eq. (6) or Eq. (7), the IDE/DE algorithm compares the target vector, the trial vector, and the third vector to implement a triple comparison mechanism in IDE/DE.

Boundary condition handling presents a problem when we conduct a memetic search in IDE and DE. The third vector produced by the memetic search can be occasionally generated beyond the current search range. When this condition happens, we should replace it with a reasonable individual or give up the memetic search at this time to make sure the IDE/DE searches in the search range of the problem under optimization. In the evaluation, we generate a uniform random number within the search range when this condition occurs.

$$third_{i,j} = better_{i,j} + (better_{i,j} - worse_{i,j}) * N(0,1).(6)$$

$$third_{i,j} = better_{i,j} + (better_{i,j} - worse_{i,j}) * U(0,1).(7)$$

The evaluation metric for IDE is user fatigue extent rather than the number of fitness calculation, which is related with user fatigue but is not in proportion to it. Suppose to compare the user fatigues of choosing the best IDE object between two objects and that among three or four objects. The mental load from few comparisons must be less than that from more comparisons, but this does not that the mean mental load from the triple comparison is 1.5 times greater than that from a paired comparison. Even when IDE tasks are time series optimization problems, such as music or movies, where we cannot compare spatially and simultaneously, an IDE user's mental load must increase, but its ratio may not become 1.5 times as well. Generally speaking, when the number of individual comparisons is within the number that an IDE user can memorize, IEC user fatigue is lower; when it exceeds the maximum memory capacity, user fatigue drastically increases. Paying attention to this fact, we develop our proposed methods requiring triple comparisons and aim to reduce the total user fatigue by accelerating IDE search even if the user fatigue of each comparison increases. This is the philosophy motivating our use of a multiple comparison mechanism in our proposed IEC algorithm.

4 Optimization Evaluation and Discussion of Interactive Differential Evolution

4.1 Benchmark Functions and Experimental Conditions

User fatigue is an important evaluation factor for IEC. When mental loads for evaluating individuals are the same, we may say that the IEC user fatigue is in proportion to the total time until the IEC user finds a satisfactory individual. However, when mental loads for evaluating one individual are different due to different IEC interfaces, this relation is not always true. There are cases where IEC user fatigue is low thanks to easy evaluation even if total evaluation time until to the goal is long. There are opposite cases where IEC user fatigue is low thanks to short total evaluation time even though the mental load for one evaluation is high. We need to evaluate acceleration methods by analyzing the load of one evaluation and convergence characteristics through IEC simulation, and then we need to conduct a human subjective evaluation to confirm the simulation results. This paper performs an IEC simulation of the former stage. We use Gaussian mixture models (GMM) as pseudo-IDE user for evaluation in this section. Concretely, we combine four Gaussian functions ($k = 4$) and implement the characteristics expressed by Eq. (8) in 3 dimensions (3-D), 5-D, 7-D, and 10-D. Figure 3 is a 3-D representation of a Gaussian mixture model.

$$GMM(\mathbf{x}) = \sum_{i=1}^{k} a_i \exp(-\sum_{j=1}^{n} \frac{(x_{ij} - \mu_{ij})^2}{2\sigma_{ij}^2}). \qquad (8)$$

where

$$\sigma = \begin{pmatrix} 1.5 & 1.5 & 1.5 & 1.5 & 1.5 & 1.5 & 1.5 & 1.5 & 1.5 & 1.5 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \end{pmatrix}.$$

$$\mu = \begin{pmatrix} -1 & 1.5 & -2 & 2.5 & -1 & 1.5 & -2 & 2.5 & -1 & 1.5 \\ 0 & -2 & 3 & 1 & 0 & -2 & 3 & 1 & 0 & -2 \\ -2.5 & -2 & 1.5 & 3.5 & -2.5 & -2 & 1.5 & 3.5 & -2.5 & -2 \\ -2 & 1 & -1 & 3 & -2 & 1 & -1 & 3 & -2 & 1 \end{pmatrix}.$$

$$a_i = \left(3.1, 3.4, 4.1, 3.0\right)^T.$$

Table 1 Abbreviations used for the proposed algorithms and the algorithms used for comparison in the experimental evaluations.

Abbreviations	Meaning
(I)DE-best	standard (I)DE/best/1/bin [11].
(I)DE-best-target	triple comparison-based (I)DE with opposite point of target vector, the base vector is the best vector, there are three points used in this algorithm, i.e., target vector, trial vector, and opposite point of target vector [9].
(I)DE-best-trial	triple comparison-based (I)DE with opposite point of trial vector, the base vector is the best vector, there are three points used in this algorithm, i.e., target vector, trial vector, and opposite point of trial vector [9].
(I)DE-best-normal	triple comparison-based (I)DE by memetic search with normal distribution, the base vector is the best vector, there are three points used in this algorithm, i.e., target vector, trial vector, and a point from disturbing whichever the better one of the target and trial vectors with a normal distribution.
(I)DE-best-rand	triple comparison-based (I)DE by memetic search with uniform distribution, the base vector is the best vector, there are three points used in this algorithm, i.e., target vector, trial vector, and a point from disturbing whichever the better one of the target and trial vectors with a uniform distribution.
(I)DE-rand	standard (I)DE/rand/1/bin [11].
(I)DE-rand-target	triple comparison-based (I)DE with opposite point of target vector, the base vector is the random vector, there are three points used in this algorithm, i.e., target vector, trial vector, and opposite point of target vector [9].
(I)DE-rand-trial	triple comparison-based (I)DE with opposite point of trial vector, the base vector is the random vector, there are three points used in this algorithm, i.e., target vector, trial vector, and opposite point of trial vector [9].
(I)DE-rand-normal	triple comparison-based (I)DE by memetic search with normal distribution, the base vector is the random vector, there are three points used in this algorithm, i.e., target vector, trial vector, and a point from disturbing whichever the better one of the target and trial vectors with a normal distribution.
(I)DE-rand-rand	triple comparison-based (I)DE by memetic search with uniform distribution, the base vector is the random vector, there are three points used in this algorithm, i.e., target vector, trial vector, and a point from disturbing whichever the better one of the target and trial vectors with a uniform distribution.

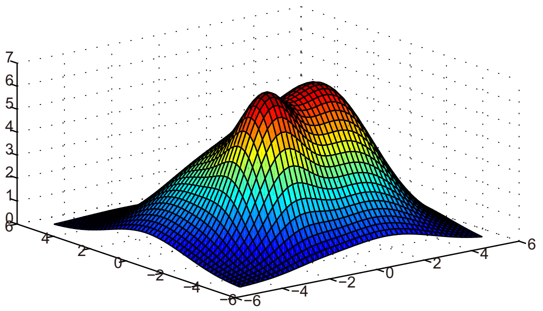


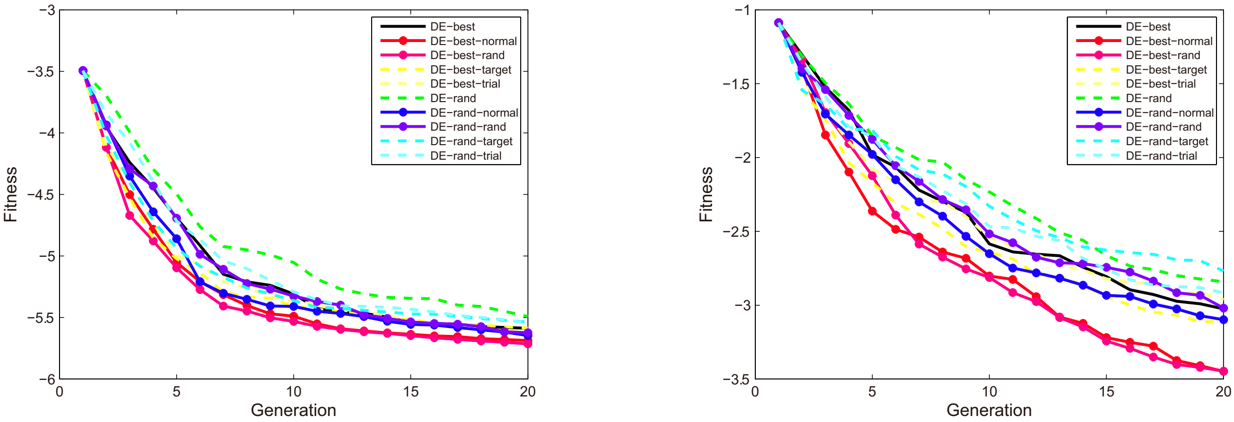
Figure 3 3-D view of a Gaussian mixture model.

4.2 Algorithm Parameter Setting and Evaluation Metrics

We test each benchmark function for 20 generations and 30 independent runs. The parameter settings of canonical paired comparison-based IDE, triple comparison-based IDE and our proposed new triple comparison-based IDE by a memetic search are listed in Table 2. Figure 2 shows the average convergence curves of the best fitness values over 30 runs for all 4 benchmark functions. Table 3 shows their mean values.

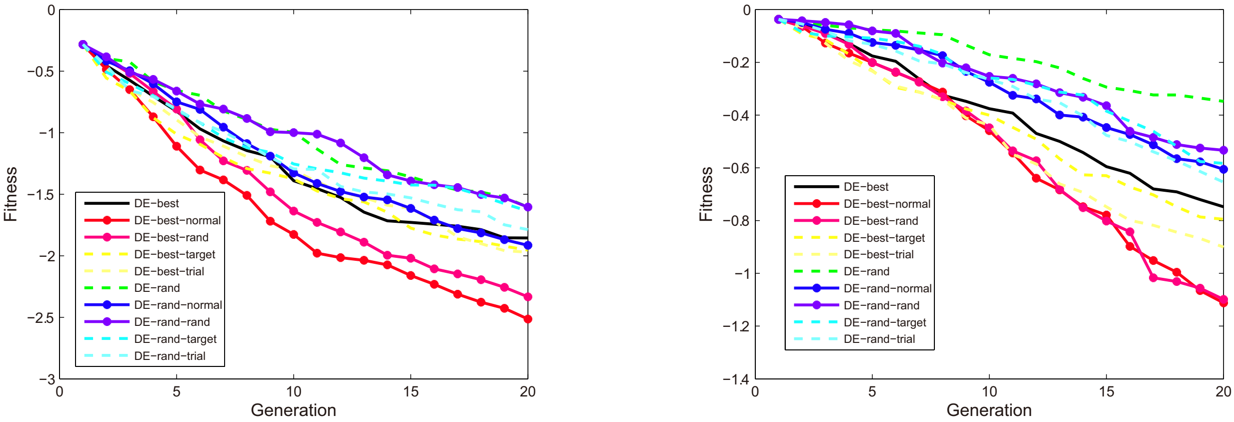
In IDE/DE algorithm frameworks, there are two parameters that should be specially considered, i.e., scale factor F and crossover rate Cr. Regarding scale factor F, we set it as 1 following from the discussion of [11], which suggested that F should lie within (0,2]. If F is too large, the convergence of IDE/DE becomes slow. On the contrary, if it is too small, it leads to local optimum for the multi-modal problems. So we choose it as 1, neither too large nor too small. Regarding crossover rate Cr, if it is 1, which means the trial vector is equal to mutant vector, it reduces the processing time of the crossover operation. The same considerations are taking in setting the parameters for the DE evaluation. Abbreviations used in Figure 2 and Table 3 are given in Table 1.

We apply the Wilcoxon signed-rank test on our proposed algorithm and the algorithms used to evaluate the significant difference of the two algorithms. Additionally, we apply the Friedman test and Bonferroni-Dunn test on one of our proposed algorithms and the comparision algorithms to rank them and



3-D Gaussian mixture model

5-D Gaussian mixture model



7-D Gaussian mixture model

10-D Gaussian mixture model

Figure 2 Average convergence curves for 30 running with 3-D, 5-D, 7-D, and 10-D Gaussian mixture models.

Table 2 IDE experiment parameters setting.

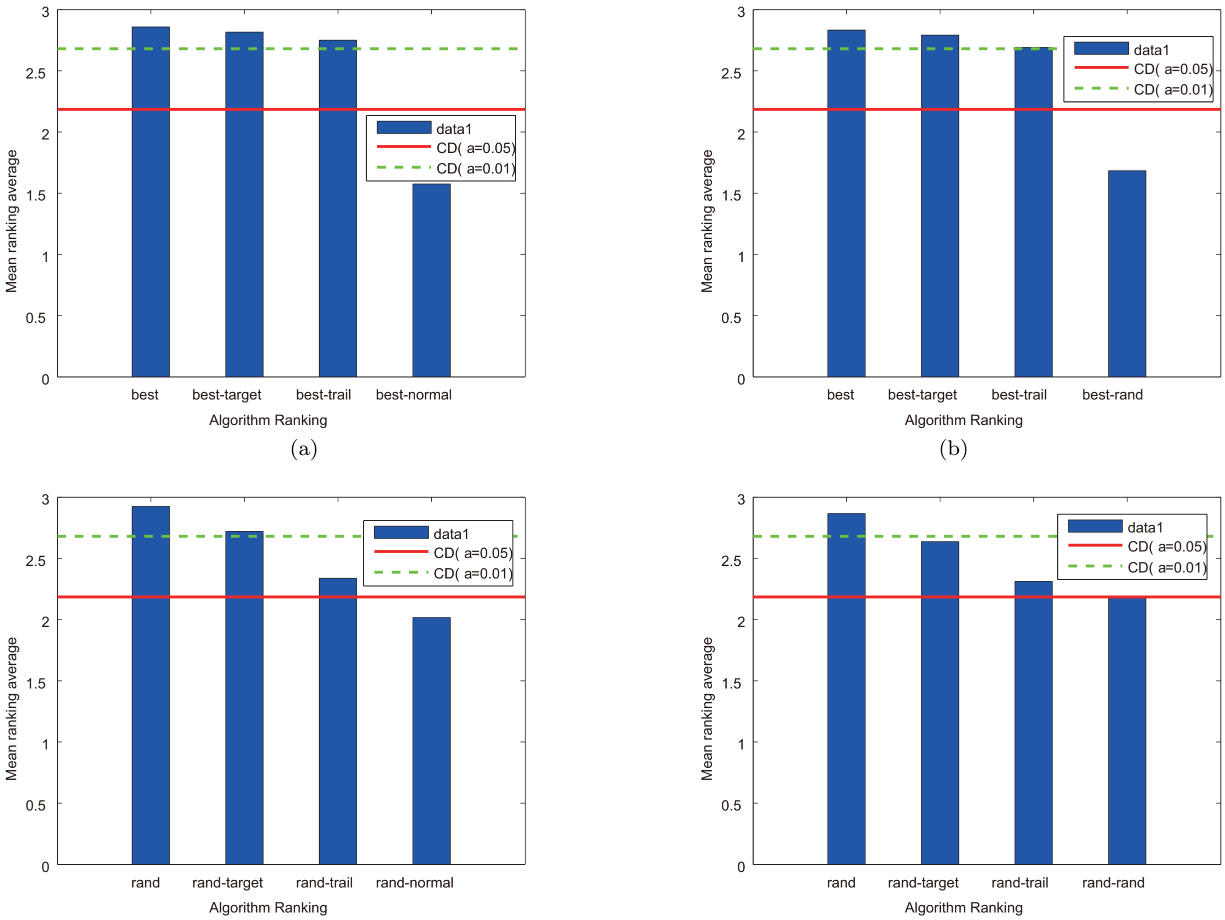
population size	20
max. search generation	20
dimensions of benchmark functions, D	3, 5, 7, 10
# of trial runs	30
scale factor, F	1
crossover rate, Cr	1

evaluate the significance of their differences. Note that we take our proposed algorithm as a control algorithm in the Bonferroni-Dunn test.

4.3 Discussion on Optimization Performance of Our Proposal

In our proposed algorithm, when we find the promising search region from a comparison of target vector and trial vector, we conduct a memetic search in the region to implement a triple comparison-based IDE so as to relieve IDE user fatigue. Although the IDE user must compare three objects, as opposed two in the original IDE, this does not mean that user fatigue is 1.5 times greater than the original IDE. This is one factor motivating our proposed memetic search in the triple comparison-based IDE.

Figure 2 demonstrates that the convergence speed of the proposed algorithms is faster than the corresponding IDE algorithm and triple comparison-based IDE algorithms in [9]. We observe that all of our proposed algorithms significantly outperform the canonical paired comparison-based IDE and triple comparison-based IDE algorithms of [9] for the IDE with the best vector as base vector from Table 3 according to Wilcoxon signed-rank test. However, for the IDE algorithms where the random vector is used as the base vector, our proposed algorithm performance acceleration is not obvious, and the proposed algorithm applied in a lower dimensional problem is better than when applied to a higher dimensional problem. This may be because the memetic search in our proposal can enhance the exploitation capability of IDE and its search range influences our proposal's performances. In our experimental evaluation, we only investigate the performance of our proposal using a normal distribution generator $(N(\mu,\sigma), \mu = 0, \sigma = 1)$ and a uniform distribution generator $(U(a,b), a = 0, b = 1)$. We will further investigate the issue of generator selection and the influence of its parameters in our future work.



(a) (b)

(c) (d)

Figure 4 Bonferroni-Dunn test using our proposed algorithm as the control algorithm in each comparison group. From this evaluation, we can conclude that our proposed algorithm is significantly better than the comparison algorithms in each group.

Table 3 Mean value of all the comparison algorithms at 20th generation. Marks †, ‡, § indicate our proposed algorithms are significantly better than canonical paired comparison-based IDE, triple comparison-based IDE with opposite point of target vector, and triple comparison-based IDE with opposite point of trial vector, respectively, from Wilcoxon signed-rank ($p < 0.05$).

Algorithm	3-D	5-D	7-D	10-D
IDE-best	-5.58818	-3.02461	-1.85508	-0.74791
IDE-best-target	-5.58665	-3.11135	-1.95467	-0.79504
IDE-best-trial	-5.62214	-2.96281	-1.97154	-0.90041
IDE-best-normal	-5.68788†‡§	-3.44824†‡§	-2.51374†‡§	-1.11205†‡§
IDE-best-rand	-5.71373†‡§	-3.44642†‡§	-2.33407†‡§	-1.09868†‡§
IDE-rand	-5.49153	-2.84135	-1.60453	-0.34802
IDE-rand-target	-5.5353	-2.76868	-1.637	-0.583
IDE-rand-trial	-5.54368	-2.91589	-1.78787	-0.65426
IDE-rand-normal	-5.64681†‡§	-3.09815†‡	-1.91501†‡	-0.6054†
IDE-rand-rand	-5.62434†‡§	-3.01962‡	-1.60471	-0.53337†

We conduct Wilcoxon signed-rank tests on our proposed algorithm between the case of normal distribution perturbation and uniform distribution perturbation. With the exception of rand-normal and rand-rand algorithms applied on 3-D and 7-D problems, there is not any significant difference between these two algorithms. It can be concluded that a memetic search with different distribution may obtain the same evaluation result.

Our proposed algorithm needs fitness evaluation 1.5 times more than that of canonical paired comparison-based IDE and the same as triple comparison-based IDE with opposition-based learning. For IDE/best algorithm, in every generation, it needs $(2 \times \mathit{populationsize} + \mathit{populationsize}) \times \mathit{generation}$ times evaluations, $2 \times \mathit{populationsize}$ times evaluations are for paired comparison and $\mathit{populationsize}$ times evaluations are for choosing the best vector as base vector. For IDE/rand algorithm, in every generation, $(2 \times \mathit{populationsize}) \times \mathit{generation}$ times evaluations and $2 \times \mathit{populationsize}$ times evaluations are required for paired comparison. In our evaluation experiments, IDE/best needs 1200 fitness evaluations at the 20th generation, and IDE/best + our

Table 4 Mean value of all the comparision algorithms at the same fitness evaluation times (IDE-best algorithm group is up to 1200 evaluation times, and IDE-rand algorithm group is up to 600 evaluation times.). Marks †, ‡, § indicate our proposed algorithms are significantly better than canonical paired comparison-based IDE, triple comparison-based IDE with opposite point of target vector and triple comparison-based IDE with opposite point of trial vector, respectively, from Wilcoxon signed-rank ($p < 0.05$).

Algorithm	3-D	5-D	7-D	10-D
IDE-best	-5.58818	-3.02461	-1.85508	-0.74791
IDE-best-target	-5.51597	-2.94033	-1.77527	-0.62989
IDE-best-trial	-5.53919	-2.82137	-1.61431	-0.74783
IDE-best-normal	-5.64594‡§	-3.14348‡§	-2.15022†‡§	-0.9563‡
IDE-best-rand	-5.64783††‡§	-3.24266‡§	-2.02005§	-0.80181
IDE-rand	-5.34486	-2.66371	-1.35938	-0.29417
IDE-rand-target	-5.36312	-2.33027	-1.25474	-0.2598
IDE-rand-trial	-5.29119	-2.46786	-1.29446	-0.26509
IDE-rand-normal	-5.35965	-2.55776	-1.15553	-0.27792
IDE-rand-rand	-5.32643	-2.51741	-0.9993	-0.25344

proposal reaches the same number of fitness evaluations at the 15th generation. For the same reason, IDE/rand needs 600 fitness evaluations by the 15th generation, and IDE/rand + our proposal reaches this number of fitness evaluations at the 10th generation. Table 4 presents the Wilcoxon signed-rank test in this condition, and the result indicates that our proposal IDE/best + our proposal is significantly better than some of its competitors, and IDE/rand + our proposal seems the same as the canonical IDE and proposals of [9].

4.4 Discussion on Algorithms Ranking

We apply the Friedman test and Bonferroni-Dunn test on our proposed algorithm and their comparison algorithms. The metric evaluation of critical difference is calculated by Eq. (9). Figure 4 illustrates the critical difference between the algorithm ranks. $k = 4$ for each comparison group (one for our proposed algorithm and 3 for the comparison algorithms; note that our proposed algorithm is the control method.), and $N = 4$ (4 benchmark problems), q is equal to $q(\alpha = 0.01) = 2.936$, and $q(\alpha = 0.05) = 2.394$ from Appendix Table B.16 of reference [17].

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6 * N}}. \tag{9}$$

In all sub-figures of Figure 4, our proposed algorithms are the control algorithm. This indicates that our proposed algorithms can obtain significantly better performance than canonical paired comparison-based IDE and triple comparison-based IDE by OBL with a significance level of $\alpha < 0.05$. It also demonstrates that the memetic search method used to implement a triple comparison mechanism in IDE is better than that implemented by OBL. We will investigate these two implementations of the triple comparison method theoretically in the future.

5 Evaluation and Discussion of Differential Evolution Optimization

5.1 Benchmark Functions and Experimental Conditions

We use 25 benchmark functions from [12] to evaluate our proposed algorithms with the newly introduced triple comparison by memetic search in DE. Table 7 presents the benchmark functions' type, characteristic, search bound and optimum fitness values. Their landscapes have a variety of characteristics. They include both uni-modal and multi-modal, shifted, rotated, and global optimum on the bounds.

The evaluated algorithms (Table 1) are the same as in section 4. The DE experimental parameters are set as shown in Table 2. The evaluation is conducted under difficult search conditions; only 50 individuals are used to search 5 dimensions (5-D) and 10-D functions.

Table 7 Test Functions (Uni=Uni-modal, Multi=Multi-modal, Sh=Shifted, Rt=Rotated, GB=Global on Bounds, HC=Hybrid Composition, NM=Number Matrix)

No.	Type	Characteristic	Bounds	Optimum fitness
$F1$	Uni	Sh Sphere	$[-100, 100]$	-450
$F2$		Sh Schwefel 1.2		-450
$F3$		Sh Rt Elliptic		-450
$F4$		f_2 with Noise		-450
$F5$		Schwefel 2.6 GB		-310
$F6$	Multi	Sh Rosenbrock	$[-100, 100]$	390
$F7$		Sh Rt Griewank	$[0, 600]$	-180
$F8$		Sh Rt Ackley GB	$[-32, 32]$	-140
$F9$		Sh Rastrigin	$[-5, 5]$	-330
$F10$		Sh Rt Rastrigin	$[-5, 5]$	-330
$F11$		Sh Rt Weierstrass	$[-0.5, 0.5]$	90
$F12$		Schwefel 2.13	$[\pi, \pi]$	-460
$F13$		Sh Expanded F8F2	$[-3, 1]$	-130
$F14$		Sh Rt Scaffer F6	$[-100, 100]$	-300
$F15$	Hybrid	HC Function	$[-5, 5]$	120
$F16$		Rt HC Function 1		120
$F17$		f_{16} with Noise		120
$F18$		Rt HC Function 2		10
$F19$		$f_{18}$with Basin		10
$F20$		f_{18} with GB		10
$F21$		Rt HC Funtion 3		360
$F22$		f_{21} with NM		360
$F23$		NC Rt f_{21}		360
$F24$		Rt HC Function 4		260
$F25$		f_{24} without Bounds		260

5.2 Algorithm Parameter Setting and Evaluation Metrics

For all algorithms, we run up to 1000 generations with 30 independent runs. The population size is 50. The scale

Table 5 Mean value of all the comparison algorithms at 1000th generation. Marks †, ‡, § show our proposed algorithms are significantly better than canonical DE, triple comparison-based DE with opposite point of target vector and triple comparison-based DE with opposite point of trial vector, respectively.

	DE-best	DE-best-target	DE-best-trial	DE-best-normal	DE-best-rand	DE-rand	DE-rand-target	DE-rand-trial	DE-rand-normal	DE-rand-rand	Ave.
F1	193.4554	-33.8007	267.6453	-449.853†‡§	-450†‡§	-414.47	-420.162	-418.083	-449.999†‡§	-449.981†‡§	-262.525
F2	-30.5252	-92.0418	-149.129	-337.633†‡§	-449.716†‡§	-421.736	-417.616	-410.76	-447.812†‡§	-449.974†‡§	-320.694
F3	2237212	2355786	2302850	987614.1	270925.9	215058.7	195402.5	192488.5	96892.44†‡§	57317.3†‡§	891154.811
F4	-40.0224	52.08288	26.45469	-391.352†‡§	-445.256†‡§	-413.826	-420.179	-423.46	-448.037†‡§	-449.759†‡§	-295.335
F5	-275.547	-307.68	-308.901	-310†‡§	-310†‡§	-309.972	-309.969	-309.964†‡§	-309.993	-305.931	-305.796
F6	3576282	7612198	3830604	24934.29†‡§	491.467†‡§	35496.74	25629.46	22811.32	398.3847†‡§	413.0122†‡§	1512925.912
F7	-148.704	-150.031	-144.839	-178.768†‡§	-179.14†‡§	-177.865	-177.722	-177.635	-179.226†‡§	-179.278†‡§	-169.321
F8	-119.446	-119.44	-119.55	-119.621†‡§	-119.621†‡§	-119.893	-119.912	-119.91	-119.631	-119.674	-119.670
F9	-277.863	-279.649	-280.894	-298.761†‡§	-304.227†‡§	-306.239	-307.217	-307.377	-316.092†‡§	-315.959†‡§	-299.428
F10	-259.986	-259.882	-259.949	-276.744†‡§	-277.489†‡§	-282.316	-281.646	-284.125	-296.192†‡§	-299.442†‡§	-277.777
F11	99.70575	99.75439	99.70235	96.85224†‡§	95.12038†‡§	94.5809	95.46711	95.43329	98.83612	98.05506	97.351
F12	40129.9	44939.13	32233.67	24411.69†‡	18986.78†‡§	4968.164	3683.398	3058.884	8090.315	9904.339	19040.628
F13	-122.327	-122.15	-123.027	-125.836†‡§	-126.557†‡§	-126.213	-126.176	-126.237	-127.684†‡§	-128.004†‡§	-125.421
F14	-295.901	-295.948	-295.974	-296.094†‡§	-296.157†‡§	-296.567	-296.528	-296.644	-296.348	-296.281	-296.244
F15	682.9748	681.0848	665.1281	530.3528†‡§	505.8682†‡§	557.9363	574.885	608.8543	565.9005	535.0564	590.804
F16	397.4698	393.2986	390.0029	354.7217†‡§	355.0738†‡§	337.3688	328.7351	334.7326	297.6289†‡§	290.0357†‡§	347.907
F17	394.8716	389.054	394.2845	349.8379†‡§	366.9631†‡§	330.2865	336.7578	322.3315	312.0499†‡§	309.8791†‡§	350.632
F18	929.0445	951.4049	951.1026	749.3402†‡§	786.655 †‡§	578.9428	579.7486	562.1065	398.4057†‡§	602.3826	708.913
F19	918.7944	948.9641	940.677	745.4966†‡§	787.6388†‡§	591.2802	583.9887	553.0895	426.8287†‡§	618.9495	711.571
F20	931.4755	939.004	931.474	775.7908†‡§	789.7336†‡§	610.5591	581.8392	544.3689	463.1416†‡§	640.3234	720.771
F21	1337.38	1364.828	1300.608	1047.535†‡§	1240.305†‡§	929.7804	909.7599	911.1893	1026.82	1387.776	1145.598
F22	1233.855	1232.115	1227.136	1184.359†‡§	1174.511†‡§	1164.546	1147.519	1147.206	1131.453†‡§	1056.947†‡§	1169.965
F23	1366.119	1390.442	1378.996	1145.157†‡§	1253.834‡	970.7627	965.0779	981.2438	1146.44	1360.522	1195.859
F24	829.3185	978.665	909.5946	674.743†‡§	704.022†‡§	497.6214	496.5886	492.813	677.7884	685.0006	694.616
F25	1992.419	1990.693	1989.407	689.1103†‡§	709.6839†‡§	2000.03	2001.975	1999.778	700.119†‡§	733.7863†‡§	1480.700

Table 6 Mean value of all the comparison algorithms at the same fitness evaluation time's generation (DE-best and DE-rand at the 900 generation and others at the 600th generation). Marks †, ‡, § show our proposed algorithms are significantly better than canonical DE, triple comparison-based DE with opposite point of target vector and triple comparison-based DE with opposite point of trial vector, respectively.

	DE-best	DE-best-target	DE-best-trial	DE-best-normal	DE-best-rand	DE-rand	DE-rand-target	DE-rand-trial	DE-rand-normal	DE-rand-rand
F1	193.4554	-33.8007	267.6453	-449.853†‡§	-450†‡§	-414.47	-419.905	-417.853	-449.747†‡§	-448.304†‡§
F2	-30.5252	-92.0418	-149.129	-337.633†‡§	-449.716†‡§	-421.736	-417.239	-410.76	-446.126†‡§	-447.029†‡§
F3	2237212	2355786	2302850	1000404†‡§	278393.1†‡§	215076.4	203060	205373.8	137327.1†‡§	89565.01†‡§
F4	-40.0197	52.11947	26.57674	-391.346†‡§	-440.478†‡§	-413.826	-417.162	-421.005	-444.133†‡§	-441.353†‡§
F5	-275.547	-307.68	-308.901	-310†‡§	-310†‡§	-309.913	-304.928	-304.701	-307.862§	-244.126†‡§
F6	3576282	7612198	3830604	24934.29†‡§	491.4867†‡§	35496.74	26923.43	23231.23	784.3448†‡§	2660.284†‡§
F7	-148.704	-150.031	-144.839	-178.768†‡§	-179.14†‡§	-177.865	-177.722	-177.601	-179.129†‡§	-179.054†‡§
F8	-119.446	-119.44	-119.55	-119.569†‡	-119.619†‡	-119.881	-119.702	-119.71	-119.599	-119.652
F9	-277.863	-279.649	-280.894	-298.761†‡§	-304.223†‡§	-306.239	-307.201	-307.061	-314.899†‡§	-314.413†‡§
F10	-259.986	-259.882	-259.949	-276.744†‡§	-277.477†‡§	-282.316	-280.664	-283.744	-295.969†‡§	-298.482†‡§
F11	99.70575	99.75439	99.70235	97.27338†‡§	95.2624†‡§	94.63565	96.46316	96.72212	99.18288	98.79825
F12	40129.9	44939.13	32233.67	24417.19†‡	18987.17†‡§	5437.609	10248.61	9868.917	20227.67	19610.54
F13	-122.327	-122.15	-123.027	-125.836†‡§	-126.557†‡§	-126.213	-126.175	-126.213	-127.608†‡§	-127.869†‡§
F14	-295.901	-295.948	-295.974	-296.094†	-296.157†‡§	-296.52	-296.308	-296.399	-296.138	-296.174
F15	682.9748	681.0848	665.1281	530.3528†‡§	505.8699†‡§	557.9545	575.1057	609.1143	567.0017	545.6756
F16	397.4698	393.2986	390.0029	354.7314†‡§	355.0741†‡§	337.3689	328.7351	334.74	300.89†‡§	292.8369†‡§
F17	394.8722	389.0568	394.2876	349.8702†‡§	366.9973†‡§	330.2959	336.7789	324.1745	316.4441†‡	321.6932‡
F18	929.0445	951.4049	951.1026	752.5895†‡§	789.0976†‡§	578.9428	579.7486	562.1071	405.9448†‡§	614.2533
F19	918.7944	948.9641	940.677	747.8883†‡§	789.3667†‡§	591.2802	583.9887	553.0998	433.4875†‡§	628.5128
F20	931.4755	939.004	931.474	778.0431†‡§	791.7951†‡§	610.5591	581.8392	544.3689	470.2662†‡	661.1316
F21	1337.38	1364.828	1300.608	1048.843†‡§	1243.305	929.7804	909.9198	911.2944	1073.532	1440.504
F22	1233.855	1232.115	1227.136	1185.364†‡§	1175.073	1164.546	1147.519	1147.206	1134.368†‡§	1079.894†‡§
F23	1366.119	1390.442	1378.996	1145.692†‡§	1257.182‡	970.7627	965.0779	981.2438	1194.684	1411.144
F24	829.3951	981.358	912.3688	680.907†‡§	706.3744†‡§	497.6254	496.6826	492.8677	683.9339	691.3242
F25	1993.253	1994.792	1995.784	698.3492†‡§	721.5549†‡§	2001.062	2009.588	2008.884	709.135†‡§	745.9672†‡§

factor (F) of the DE is set to 1 and the crossover rate (Cr) is set to 1 as well as discussed in section 4.

When the proposed methods are applied to a non-interactive DE, the number of fitness calculations for one comparison becomes three times from two times in the canonical paired comparison-based DE, i.e., the calculation time of the proposal becomes 1.5 times of the canonical DE. The point of our proposed method for canonical non-interactive DE is whether the acceleration performance of the proposed methods exceeds the increase of fitness calculation time so that the total calculation time is surely reduced.

Because there is no need to evaluate IDE user fatigue, as was done in section 4, the evaluation indices used in this section for comparing our proposed methods applied to DE are the number of fitness calculations (NFC) until the convergence threshold (CT) is reached and the success rate (SR) for those that reach the CT. The lower the NFC is, the faster convergence is. A successful convergence is counted when a convergence reaches the CT defined by Eq. (10). All benchmark functions are evaluated by an acceleration rate, AR, too, to evaluate convergence speed. It is defined using NFC at the maximum generation, the 1,000th generation, in Eq. (13), and $AR > 1$ means that the proposed method converges faster than canonical DE. Success rate, SR, is defined by the number of trials that reached the convergence threshold, CT, in Eq. (12). Furthermore, average acceleration rate and average success rate are calculated and used for the final evaluation results.

$$\begin{aligned} converg.\,threshold,\ VTR &= average\,fitness\,of\,each \\ &\quad method\,at\,MAX_{NFC}-th \\ &\quad generation. \qquad (10) \\ NFC &= average\,\#\,of\,fitness \\ &\quad calculation\,until\,convergence \\ &\quad reaches\,CT. \qquad (11) \\ success\,rate,\ SR &= \frac{\#\,of\,reached\,to\,CT}{\#\,of\,MAX_{NFC}}. \qquad (12) \\ acceleration\,rate,\ AR &= \frac{NFC_{canonicalDE}}{NFC_{proposal}}. \qquad (13) \end{aligned}$$

5.3 Optimization Performance of Our Proposal

We apply a Wilcoxon signed-rank test on our evaluation results of mean value at the final generation (Table 3) and at the generation when the compared algorithms achieve the same comparison times, i.e., the 900 generation for canonical DE and the 600th generation for the proposals of reference [9] and this paper (Table 4). From Table 3, except a few cases (such as F3), DE/best + proposed methods are significantly better than canonical DE and the proposals of reference [9]. However, the performance of DE/rand + proposed methods are not as the same as that of DE/best + proposed methods. Because the memetic search is a local search method, it definitely can enhance the exploitation capability of the algorithm around the best individual; this is the reason why the performance of DE/best + proposed methods is better than that of the DE/rand + proposed methods.

Our original proposal is to develop a new triple comparison-based IDE, and this search scheme can also be applied to DE. The comparison time is a suitable evaluation metric to compare the proposed methods with their comparison algorithms. Table 4 presents these results, i.e., when the total number of comparisons is 1800 (when it is the 600th generation in the methods of reference [9] and the proposal of this paper, and the 900th generation in canonical DE). We conduct a Wilcoxon signed-rank test at this condition. From the result of this evaluation, the proposed triple comparison DE significantly outperforms that of its competitors as well. This illustrates the advantage of our proposal.

From the NFC, SR and AR metrics, DE-rand-trial and DE-rand-normal obtain the first and second winners in SR, and DE-best-normal and DE-best-rand obtain the first and second winners in AR. This indicates the exploitation capability of our proposed methods can enhance DE search by introducing a memetic search scheme. Although it is not a statistical result, we can also observe that the average performance of our proposal is better that of the compared algorithms.

6 Conclusion and Future Works

We proposed a new triple comparison-based IDE algorithm and DE algorithm using a memetic search from a fitness landscape obtained by a comparison of target vector and trial vector. The local fitness landscape obtained from the canonical DE algorithm supports information that indicates a promising search region. We implement the memetic search by perturbing the vector with the better fitness with a vector from a normal distribution and a uniform distribution. The originality of implementing a new triple comparison-based IDE for relieving IDE user fatigue motivates this work. We initially studied the performance of our proposed algorithms in both IDE and DE.

In the future, applying the proposed algorithms to a variety of IEC applications will be the primary focus of our study. In this paper, we initially investigated the performance of our proposal with a pseudo-user in IDE and found that the proposed algorithms significantly outperform the comparison algorithms. In the next step, we will study its application issues to discover new knowledge from each IEC application domain. We will also investigate the issue of the parameters settings for the random number generator, and implementations of our proposal using other distribution generators. The question as to why a third vector that comes from memetic search is better than that from OBL in theory needs further research. It will be necessary to establish related mathematical models to explain these differences.

Table 8 The number of function calls (NFC in Eq. (11)), success rate (SR in Eq. (12)), acceleration rate (AR in Eq. (13)) and fitness value to reach (VTR in Eq. (10)) of 5-D benchmark function..

Func.	DE-best			DE-best-target			DE-best-trial			DE-best-normal			DE-best-rand		
	NFC	SR	AR	NFC	SR	AR	NFC	SR	AR	NFC	SR	AR	NFC	SR	AR
F1	56424	0.010		70578	0.108	3.020	87174	0.016	0.988	5205	0.471	11.212	5118	0.472	11.809
F2	56262	0.010		79179	0.060	1.592	79251	0.060	1.559	25593	0.358	7.556	6237	0.465	9.345
F3	48818	0.031		70068	0.111	2.717	70278	0.110	2.864	33939	0.311	5.807	9321	0.448	8.552
F4	46848	0.037		81615	0.047	1.086	81618	0.047	1.167	15078	0.416	6.709	11637	0.435	6.057
F5	13112	0.130		14508	0.419	1.415	16917	0.406	1.266	8442	0.453	1.612	8700	0.452	1.529
F6	37488	0.063		50361	0.220	3.557	53397	0.203	3.924	4023	0.478	9.255	4119	0.477	10.016
F7	56248	0.010		84375	0.031	1.352	84729	0.029	1.006	7593	0.458	11.777	5037	0.472	11.453
F8	56520	0.010		76896	0.073	1.398	68856	0.117	1.950	73707	0.091	1.044	71406	0.103	1.882
F9	56600	0.009		79164	0.060	1.263	78894	0.062	2.104	47718	0.235	6.221	36102	0.299	8.360
F10	56106	0.011		90000	0.000	0.623	90000	0.000	0.623	54027	0.200	4.697	55656	0.191	6.831
F11	48476	0.032		78534	0.064	2.161	84267	0.032	1.508	54273	0.198	2.118	33801	0.312	5.540
F12	56342	0.010		87117	0.016	0.643	61428	0.159	1.992	50196	0.221	3.954	50109	0.222	9.315
F13	58292	0.005		90000	0.000	0.648	87114	0.016	1.210	42318	0.265	6.670	22125	0.377	11.335
F14	56512	0.010		84777	0.029	0.927	84765	0.029	0.934	67995	0.122	3.466	64752	0.140	3.848
F15	46416	0.038		64230	0.143	3.418	69564	0.114	2.677	35328	0.304	9.565	35730	0.302	9.529
F16	52254	0.022		87069	0.016	1.525	87117	0.016	1.128	55479	0.192	7.754	61335	0.159	4.821
F17	60000	0.000		81348	0.048	2.560	78582	0.063	2.433	42099	0.266	7.430	64620	0.141	4.091
F18	50382	0.027		78393	0.064	2.588	81453	0.047	0.970	61131	0.160	4.265	70914	0.106	2.978
F19	50358	0.027		81345	0.048	1.666	78579	0.063	1.744	61200	0.160	4.723	67935	0.123	3.382
F20	50350	0.027		78423	0.064	2.344	78573	0.063	1.742	63990	0.145	4.706	69975	0.111	3.463
F21	41016	0.053		67368	0.126	1.952	67086	0.127	3.054	30177	0.332	8.011	58863	0.173	3.476
F22	60000	0.000		84195	0.032	2.090	84294	0.032	1.535	67287	0.126	4.030	52443	0.209	5.672
F23	45090	0.041		67185	0.127	4.064	73263	0.093	2.150	35703	0.302	6.697	54036	0.200	5.659
F24	37654	0.062		76563	0.075	1.338	67530	0.125	1.971	23742	0.368	2.478	22467	0.375	4.004
F25	60000	0.000		90000	0.000	0.667	90000	0.000	0.667	1137	0.494	66.461	1098	0.494	70.375
Ave.	50302	0.027		75731	0.079	1.864	75389	0.081	1.727	38695	0.285	8.329	37741	0.290	8.933

Table 9 The umber of function calls (NFC in Eq. (11)), success rate (SR in Eq. (12)), acceleration rate (AR in Eq. (13)) and fitness value to reach (VTR in Eq. (10)) of 10-D benchmark function..

Func.	DE-rand			DE-rand-target			DE-rand-trial			DE-rand-normal			DE-rand-rand		
	NFC	SR	AR	NFC	SR	AR	NFC	SR	AR	NFC	SR	AR	NFC	SR	AR
F1	37616	0.130	4.462	20532	0.386	2.799	20487	0.386	2.805	13275	0.426	4.338	17472	0.403	3.329
F2	37508	0.121	3.479	24156	0.366	2.385	24963	0.361	2.268	22089	0.377	2.642	23073	0.372	2.455
F3	32545	0.118	3.238	25632	0.358	2.322	27318	0.348	1.997	24891	0.362	2.033	29322	0.337	1.723
F4	31232	0.117	2.694	26460	0.353	1.808	27819	0.345	1.721	24543	0.364	1.965	26172	0.355	1.813
F5	8741	0.072	0.418	50919	0.217	0.253	52221	0.210	0.253	47541	0.236	0.276	79404	0.059	0.179
F6	24992	0.141	4.162	14079	0.422	2.771	14328	0.420	2.730	10056	0.444	3.900	11943	0.434	3.188
F7	37498	0.131	4.412	18648	0.396	3.074	21015	0.383	3.013	15498	0.414	3.735	16374	0.409	3.494
F8	37680	0.069	4.664	53424	0.203	1.266	51324	0.215	1.939	76980	0.072	1.343	59817	0.168	3.151
F9	37733	0.091	2.705	40077	0.277	1.824	42138	0.266	1.794	31098	0.327	2.055	28947	0.339	2.169
F10	37404	0.080	3.960	47379	0.237	2.164	47247	0.238	2.413	28341	0.343	3.338	26898	0.351	3.170
F11	32317	0.100	2.730	48627	0.230	1.490	53604	0.202	1.186	89982	0.000	0.539	86307	0.021	0.589
F12	37561	0.107	3.624	36072	0.300	2.065	35397	0.303	3.428	56766	0.185	1.640	52761	0.207	2.966
F13	38861	0.083	3.053	51048	0.216	1.635	45201	0.249	1.931	27969	0.345	2.535	26748	0.351	2.462
F14	37674	0.060	1.992	56046	0.189	1.402	52428	0.209	1.921	75948	0.078	0.824	64413	0.142	1.283
F15	30944	0.064	2.836	60192	0.166	1.450	66003	0.133	1.227	68850	0.118	2.381	66267	0.132	1.238
F16	34836	0.090	3.777	33060	0.316	2.933	41286	0.271	2.464	18861	0.395	3.237	18429	0.398	3.916
F17	40000	0.086	2.687	43197	0.260	2.049	40371	0.276	2.033	30129	0.333	2.433	35658	0.302	1.936
F18	33588	0.115	7.260	30354	0.331	5.115	19032	0.394	5.787	19368	0.392	5.780	53985	0.200	2.597
F19	33572	0.106	7.249	33144	0.316	4.416	21714	0.379	5.503	24051	0.366	5.569	56454	0.186	2.451
F20	33566	0.101	7.412	29646	0.335	5.621	18801	0.396	5.952	26691	0.352	6.644	59904	0.167	2.266
F21	27344	0.135	4.450	14586	0.419	2.838	14214	0.421	2.994	41112	0.272	2.336	84321	0.032	0.515
F22	40000	0.111	4.532	34044	0.311	2.619	24876	0.362	3.608	23601	0.369	3.353	34731	0.307	4.211
F23	30060	0.144	5.662	13311	0.426	3.590	12444	0.431	3.679	52533	0.208	1.748	80565	0.052	0.720
F24	25102	0.143	4.505	12465	0.431	3.195	13386	0.426	2.962	31677	0.324	1.916	48828	0.229	0.922
F25	40000	0.000	1.000	90000	0.000	0.667	90000	0.000	0.667	1560	0.491	43.205	912	0.495	80.553
Ave.	33535	0.101	3.879	36283	0.298	2.470	35104	0.305	2.651	35336	0.304	4.391	43588	0.258	5.332

This and the other issues and problems arising from the current study will be the subject of ongoing work in the future.

References

[1] I. S. Lim and D. Thalmann. Tournament selection for browsing temporal signals. In *Proceedings of the 2000 ACM symposium on Applied computing-Volume 2*, pages 570–573. ACM, 2000.

[2] G. A. Miller. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological review*, 63(2):81, 1956.

[3] Y. Pei. Chaotic evolution: fusion of chaotic ergodicity and evolutionary iteration for optimization. *Natural Computing*, 13(1):79–96, 2014.

[4] Y. Pei. From determinism and probability to chaos: Chaotic evolution towards philosophy and methodology of chaotic optimization. *The Scientific World Journal*, page Article ID 704587, 2014.

[5] Y. Pei. *Study on Effecient Search in Evolutionary Computation*. PhD thesis, Kyushu University, Japan, 2014.

[6] Y. Pei and H. Takagi. Fourier analysis of the fitness landscape for evolutionary search acceleration. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1–7. IEEE, 2012.

[7] Y. Pei and H. Takagi. Accelerating iec and ec searches with elite obtained by dimensionality reduction in regression spaces. *Evolutionary Intelligence*, 6(1):27–40, 2013.

[8] Y. Pei and H. Takagi. Fitness landscape approximation by adaptive support vector regression with opposition-based learning. In *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, pages 1329–1334. IEEE, 2013.

[9] Y. Pei and H. Takagi. Triple and quadruple comparison-based interactive differential evolution and differential evolution. In *Proceedings of the twelfth workshop on Foundations of genetic algorithms XII*, pages 173–182. ACM, 2013.

[10] Y. Pei, Q. Zhao, and Y. Liu. Kernel method based human model for enhancing interactive evolutionary optimization. *The Scientific World Journal*, page Article ID 185860, 2014.

[11] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[12] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. 2005.

[13] H. Takagi. Interactive evolutionary computation: Fusion of the capabilities of ec optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275–1296, 2001.

[14] H. Takagi and D. Pallez. Paired comparison-based interactive differential evolution. In *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pages 475–480. IEEE, 2009.

[15] H. R. Tizhoosh. Opposition-based learning: A new scheme for machine intelligence. In *CIMCA/IAWTIC*, pages 695–701, 2005.

[16] H. R. Tizhoosh. Opposition-based reinforcement learning. *JACIII*, 10(4):578–585, 2006.

[17] J. H. Zar. *Biostatistical analysis*. Pearson Education India, 1999.