Neural Computing and Applications (2020) 32:3923–3937
https://doi.org/10.1007/s00521-018-3822-5

ORIGINAL ARTICLE

Weighted differential evolution algorithm for numerical function optimization: a comparative study with cuckoo search, artificial bee colony, adaptive differential evolution, and backtracking search optimization algorithms

Pinar Civicioglu[1] · Erkan Besdok[2] · Mehmet Akif Gunen[2] · Umit Haluk Atasever[2]

Received: 23 February 2018 / Accepted: 15 October 2018 / Published online: 26 October 2018

Abstract
In this paper, weighted differential evolution algorithm (WDE) has been proposed for solving real-valued numerical optimization problems. When all parameters of WDE are determined randomly, in practice, WDE has no control parameter but the pattern size. WDE can solve unimodal, multimodal, separable, scalable, and hybrid problems. WDE has a very fast and quite simple structure, in addition, it can be parallelized due to its non-recursive nature. WDE has a strong exploration and exploitation capability. In this paper, WDE's success in solving CEC' 2013 problems was compared to 4 different EAs (i.e., CS, ABC, JADE, and BSA) statistically. One 3D geometric optimization problem (i.e., GPS network adjustment problem) and 4 constrained engineering design problems were used to examine the WDE's ability to solve real-world problems. Results obtained from the performed tests showed that, in general, problem-solving success of WDE is *statistically better* than the comparison algorithms that have been used in this paper.

Keywords Cuckoo search algorithm · Artificial bee colony algorithm · Differential evolution algorithm · Backtracking search optimization · Particle swarm optimization

1 Introduction

Evolutionary algorithms (EA) are commonly used for solving complex numerical optimization problems (i.e., multimodal, non-differentiable, highly nonlinear, and constrained design problems) [1–7]. EAs are population based, iterative, stochastic search mechanisms, searching for optimum solutions that belong to the related problem [8–11]. Random solution of a numerical optimization problem is demonstrated with a pattern containing individuals as the dimension of the problem [8, 12–15]. EAs can find the near optimum solution of a problem by using limited number of patterns. Patterns are kept in a pattern matrix having different patterns as the rows [8, 15]. Interaction models defined between the pattern matrix elements bring in collective search capability to EAs. EAs can be classified as *swarm-inspired* [1–4, 7–13], *bio-inspired* [16, 17], or *nature-inspired* [18, 19] algorithms. Bio-inspired EAs, the most frequently used EAs, simulate various genetic processes such as *selection*, *mutation*, and *crossover* analogically, with the help of the interaction models that they are using. EAs use interaction models to generate trial patterns by using the present patterns. If a trial pattern produces more feasible results than the pattern that it corresponds, then it is added to the pattern matrix instead of the related pattern at the next iteration. Problem-solving successes of EAs are generally sensitive to several parameters; i.e., structural properties

✉ Erkan Besdok
ebesdok@erciyes.edu.tr

Pinar Civicioglu
civici@erciyes.edu.tr

Mehmet Akif Gunen
akif@erciyes.edu.tr

Umit Haluk Atasever
uhatasever@erciyes.edu.tr

[1] Department of Aircraft Electrics and Electronics, Faculty of Aeronautics and Astronautics, Erciyes University, Kayseri, Turkey

[2] Department of Geomatics Engineering, Engineering Faculty, Erciyes University, Kayseri, Turkey

of the problem, dimension of problem, initial form of the pattern matrix, random number generator that is being used, total number of function evaluation value.

EAs have been used in the solution of various structural engineering design problems [20, 21], such as communication applications [7], image processing applications [22], solution of some speech recognition problems [23], solution of sensor deployment problems [24], various data mining applications [25], design of IIR filters [26], video processing [27], and solution of other several engineering problems [28–35].

The nature of a numerical optimization problem defines whether it is a unimodal, multimodal, separable, non-separable, scalable or hybrid problem, or not [1, 10–12]. Unimodal problems have a single local solution, which is the same with the global solution. Multimodal problems have several global solutions. Therefore, in the solution of multimodal problems, EAs using interaction models that do not trap the local solutions should be used. In separable problems, since each variable is independent from the other variables, in the solution of this type of problems, each variable can be optimized independently. In non-separable problems, all variables have to be optimized together. In scalable problems, computational complexity of problem changes with changing dimension of problem. Problem-solving success of EAs is sensitive to whether they have trial pattern generation strategy appropriate to the nature of the related problem, or not.

EAs generally produce a trial pattern with the usage of a crossover process defined between the patterns or a mutation process based on a statistical distribution model. Pattern generation strategies used in EAs are frequently based on interaction between patterns. (i.e., *swarming*). EAs benefit from a trial pattern to evolve a pattern to a pattern that iteratively provides a better fitness value. In order to generate a trial pattern, patterns selected as the *raw genetic material* among the present patterns are mixed by using various pseudo-genetic operators. System equations defining the trial pattern generation processes used in EAs generally show important similarities [8]. Besides, usage style of the related system equations shows significant differences in EAs. The strategy that an EA uses for trial pattern generation influences its problem-solving success and speed. Researches continue to develop new EAs that can solve complex engineering problems.

This paper introduces WDE that can solve different types of numerical problems (i.e., separable, non-separable, unimodal, multimodal, and hybrid problems). WDE is a non-recursive, iterative algorithm; the swarming process of WDE is very efficient. Therefore, WDE is generally not trapped easily with local solution. This increases WDE's success in solving numerical problems considerably. WDE uses a random crossover-based *swarming* strategy to generate interaction pattern. While evolving a pattern into a more feasible pattern, WDE produces a trial pattern by changing some randomly selected or all individuals of the related pattern with the related individuals of an interaction pattern. Though trial pattern generation strategy of WDE is very productive, there is a possibility of trapping to local solution in hybrid problems. Every pattern matrix of WDE evolves into a randomly selected and permuted pattern matrix to provide swarming in every iteration. While WDE generates a trial pattern by allowing the changing of only one individual for the first pattern matrix, it produces a trial pattern by allowing a randomly selected number of individuals for the second pattern matrix. WDE generates a trial pattern by allowing the changing of all individuals for the third pattern matrix.

In this paper, numerical optimization problem-solving capability of WDE is examined in detail by using CEC' 2013 [36] numerical functional optimization problems. Cuckoo search algorithm (CS) [3, 8], artificial bee colony (ABC) [1], adaptive differential evolution algorithm (JADE) [37], and backtracking search optimization algorithm (BSA) [7] have been successful in the solution of many different engineering problems. Therefore, the success of WDE in the solution of functional optimization problems has been examined with statistical comparison with CS, ABC, JADE, and BSA. GPS-Based Geodesic network adjustment problem has been solved by using classical least square adjustment method and WDE. Engineering design problems have been solved by using PSO2011 [9], CPI-JADE [38], A+ [39], CS, ABC, JADE, BSA, and WDE.

Differential evolutionary algorithm (DE) is a statistical, powerful, and widely used evolutionary computation algorithm developed to solve real-valued numerical optimization problems. DE is robust, its implementation is relatively easy, and its structure is simple. DE is used in the solution of unimodal, multimodal, and hybrid problem types. DE also has some disadvantages. The success of DE shows the sensitivity to the problem type. DE suffers from premature convergence, especially when solving multimodal and hybrid problems. The global search capability of DE is sufficient for many problem types. However, the local search ability of DE is weak. Also, DE suffers from low convergence speed. The standard DE tends to rapidly reduce the diversity of the population. This restricts DE's search ability. The performance of DE generally decreases rapidly as the search space size increases. DE is sensitive to the selection of control parameters, and it is time-consuming and difficult to tune them for different problems. The success of DE depends on the crossover and mutation strategy it uses at the moment. DE is not strong enough to determine efficient evolutionary direction and evolutionary step size [7, 8, 15].

The need to develop a method that does not have the inherent limitations of DE has motivated the development of

WDE. The problem-solving success of the WDE is, to a large extent, not dependent on the problem type, as opposed to the DE. Also, WDE does not need to use a different mutation process like DE for each problem type. The crossover and mutation process of WDE is different from the crossover and mutation processes of DE; they are simpler and much more efficient. Compared to DE, WDE has the ability to determine very efficient evolutionary search direction and evolutionary step. In addition, WDE does not allow the diversity of the population to decline rapidly. For this reason, it may continue to do efficient searches in progressive iterations, contrary to DE. Because WDE does not have a control parameter in practice, the time-consuming and difficult parameter tuning process of DE is not available in WDE.

This paper is organized as follows: In Sect. 2, nomenclature is given. In Sects. 3 and 4, weighted differential evolution algorithm (WDE) and experiments are presented, respectively. In Sect. 5, conclusions and possible research directions are given.

2 Nomenclature

Symbol	Meaning/definition
$\mathcal{F}$	Objective function
low, up	Search Limits
N	Size of population
D	Dimension of problem
$MaxCycle$	Maximum number of iterations
$gmin$	Global minimum
$gbest$	Global minimizer
$\kappa_{(\cdot)} \sim U(0,1), \kappa_{(\cdot)} \neq 0$	Uniform random number
$\lambda_{(\cdot)} \sim N(0,1)$	Normal random numbers
$\alpha, \beta \sim U(0,1)$	Uniform random numbers
$P_{(i0,j0)} \mid P_{(i0,j0)} \sim \mathbf{U}(low_{(j0)}, up_{(j0)})$	Patterns of pattern matrix
$fitP_{(i0)}$	Fitness values of $P_{i0=1:N}$
permute()	Permuting function
$\circ$	Hadamart operator

3 Weighted differential evolution algorithm (WDE)

WDE is a bi-population based, iterative, evolutionary search algorithm developed to solve real-valued numerical optimization problems. WDE has been designed as a global minimizer algorithm. WDE can perform bounded or unbounded search. The elitist EAs are generally successful in solving the unimodal problems. However, the elitist behavior that an EA exhibits may cause it to be trapped with local solutions. Therefore, partially elitist functioning of an EA accelerates its convergence to solution and prevents it from being trapped with local solutions.

The initialization process of WDE includes defining the initial population (i.e., pattern matrix; P). P is computed by using Eq. 1;

$$P_{(i0,j0)} \sim \mathbf{U}\left(low_{(j0)}, up_{(j0)}\right) \;\mid\; \left(\underbrace{2N}_{\text{rows}}, \underbrace{D}_{\text{colums}}\right) \leftarrow size(P) \tag{1}$$

Here, $i0 = [1:2N],\ j0 = [1:D], where\ \ i0,\ j0 \in \mathrm{Z}^{+}$. In Eq. 1, N is the pattern vector number, and D is the problem dimension. low_{j0}, up_{j0} are the lower and upper search limits of the $j0th$ parameter. $U(\cdot)$ denotes the continuous uniform distribution. The objective function value of P_{i0} is computed by using Eq. 2;

$$fitP_{(i0)} = \mathcal{F}\left(P_{(i0)}\right) \tag{2}$$

In Eq. 2, $\mathcal{F}$ denotes the objective function. The *first selection process* of WDE generates a sub-pattern matrix, *SubP*, from *P*. WDE generates *SubP* by randomly selecting N pattern vectors from *P* in each iteration where $\left(\underbrace{N}_{\text{rows}}, \underbrace{D}_{\text{colums}}\right) \leftarrow size(SubP)$. *SubP* is defined by using Eq. 3;

$$SubP = P_{(k)} \;\mid\; \{k = j_{(1:N)} \mid j = permute\,(1:2N) \tag{3}$$

In Eq. 3, $permute(\cdot)$ denotes the permuting function. The objective function values, *fitSubP*, of the pattern vectors of *SubP* are defined in Eq. 4;

$$fitSubP = fitP_{(k)} \tag{4}$$

The mutation process aims to generate new pattern vectors, i.e., *TempP*. WDE regenerates $TempP_{index=1:N} = \begin{bmatrix} TempP_1 \\ \ldots \\ TempP_N \end{bmatrix}$ in each iteration by using Eq. 5;

$$TempP_{(index)} = \sum \left(w \circ P_{(l)}\right) \;\mid\; l = j \backslash k \;\;(see\ Eq.3\ for\ j,k) \tag{5}$$

where $index = 1:N$ and $index \in \mathrm{Z}^{+}$. In Eq.5, $w^{*} = \kappa^{3}_{(N)} \mid [N,1] = size(w^{*})$ where $w^{*} := \frac{w^{*}}{\sum w^{*}}$ and $w = w^{*} \times \varDelta$ where $\varDelta = [1]_{(1,D)}$.

WDE updates the initial $M_{(1:N,1:D)} = 0$ in each iteration by using Eq. 6, and uses the updated M to control the mutation process;

$$\mathrm{M}_{(index,J)} := 1 \tag{6}$$

In Eq. 6, $J = V(1 : \lceil K \times D \rceil) \ \mid \ \mathrm{V} = \text{ permute } (j0)$. K is defined with the rule in Eq. 7;

$$\text{If } \alpha < \beta \text{ then } K = \kappa^3_{(1)} \text{ else } K = (1 - \kappa^3_{(1)}) \tag{7}$$

In Eq. 7, $\alpha, \beta, \kappa \sim U(0,1)$, $(\cdot) = size(\kappa_{(.)}$.

In WDE, $\kappa_{(\cdot)}$ generates $(\cdot)$-sized real-valued uniform random number, each time it is used. In WDE, the evolutionary step size (i.e., scale factor), F, is computed by using the rule given in Eq. 8;

$$\begin{aligned} &\text{if } \alpha < \beta \,\text{then} F = [\lambda^3_{(D)}]' \ \mid \ [\underbrace{1}_{\text{rows}}, \underbrace{D}_{\text{columns}}] = size(F) \\ &\qquad \text{else} F = (\lambda^3_{(N)} \times \varDelta) \ \mid \ [\underbrace{N}_{\text{rows}}, \underbrace{D}_{\text{columns}}] = size(F) \end{aligned} \tag{8}$$

In WDE, trial vector, T, is generated by using Eq. 9;

$$\begin{aligned} &T = SubP + F \times M \circ \left(TempP - SubP_{(m)}\right) \\ &\quad |m = permute(i)|m \neq [1 : N] \end{aligned} \tag{9}$$

here, $i = 1 : N \mid i \in \mathrm{Z}^+$. $T \notin [\, low \quad up \,]$ values are updated by using Eqs. 10 and 11;

$$\begin{aligned} &\text{if } \left(T_{(i,j0)} < low_{(j0)}\right) \text{ then} \\ &\quad T_{(i,j0)} = low_{(j0)} + \kappa^3_{(1)}\left(up_{(j0)} - low_{(j0)}\right) \end{aligned} \tag{10}$$

$$\begin{aligned} &\text{if } \left(T_{(i,j0)} > up_{(j0)}\right) \\ &\quad \text{then } T_{(i,j0)} = up_{(j0)} + \kappa^3_{(1)}\left(low_{(j0)} - up_{(j0)}\right) \end{aligned} \tag{11}$$

The objective function values of $T_{i=1:N}$ vectors are computed by using Eq. 12;

$$fitT = \mathcal{F}(T) \tag{12}$$

T and $fitT$ are used in order to update $SubP$ and $fitSubP$ as per the greedy-selection rule. The relevant updating process of WDE is defined by using the rule given in Eq. 13;

$$\begin{aligned} &\text{if } \left(fitT_{(i^*)} < fitSubP_{(i^*)}\right) \text{ then } [SubP_{(i^*)}, fitSubP_{(i^*)}] : \\ &\quad = [T_{(i^*)}, fitT_{(i^*)}] \mid i^* \in i \end{aligned} \tag{13}$$

The updated $SubP$ and $fitSubP$ are used in order to update $P_{(l)}$ and $fitP_{(l)}$ values. The relevant updating process of WDE is shown in Eq. 14;

$$[P_{(l)}, fitP_{(l)}] := [SubP, fitSubP] \tag{14}$$

Here, see Eq. 5 for l. WDE achieves the searched global solution by using Eq. 15;

$$[gmin, gbest] = [fitP_{(\gamma)}, P_{(\gamma)}] \ \mid fitP_{(\gamma)} = \min(fitP) \ , \quad \gamma \in i \tag{15}$$

The pseudo-code of the WDE is given in Fig. 1. The values of the parameters used in WDE can be determined randomly. In that case, WDE theoretically has no control parameter. Since the values of the parameters used in WDE are determined randomly, WDE has no parameter tuning process. Therefore, it is easy to use.

The novelties of WDE introduced herein are as follows:

- The swarming process of WDE uses new stochastic mutation control mechanism.
- WDE' s system equation is partially similar to the system equation of differential evolution algorithm [15], but WDE' s direction vector generation strategy is different.
- The direction vectors generated in WDE are composed of the mixed vectors of different pattern vectors.
- WDE uses a new method for boundary control.
- The values of all parameters used in WDE are determined randomly. Therefore, WDE does not waste time to initial parameter tuning.
- Since it has a non-recursive structure, WDE can be parallelized easily. Therefore, it is rather fast.

Similarities and differences between WDE and comparison algorithms are listed below:

- WDE does not partially or totally act elitist like ABC, CS, and JADE.
- WDE uses sub-populations for interacting pattern matrices.
- Mutation and crossover strategies of WDE are different than those of comparison algorithms.
- WDE is a bijective search algorithm like BSA.
- WDE does not have any control parameters.
- Boundary control mechanism of WDE is unique to itself.
- Though basic system equations of WDE are somehow similar to *DE/rand/1/bin* [15], BSA, ABC, and CS, usage style of these equations are quite different than other algorithms.
- WDE is structurally non-recursive as different from *DE/rand/1/bin* [15], ABC, and JADE [37]. Therefore, it can be parallelized without being modified.
- Functioning of WDE is analogically based on the cooperation of *bio-interacting* sub-populations.
- Functioning of WDE may be explained with the usage of *bio-inspired* evolutionary optimization processes (i.e., *initialization*, *selection*, *mutation*, and *recombination/crossover*) just like other EAs.

Fig. 1 Pseudo-code of the WDE. The unoptimized MATLAB code of the WDE is publicly available at [40]

```
   Input: Objective function: $\mathcal{F}$, Search Limits: $(low,\ up)$, Size of Population: $N$,
          Dimension of Problem: $D$, and Maximum Number of Iterations: $MaxCycle$
   Output: $gmin$: Global minimum, $gbest$: Global minimizer
1  $i = [1, ..., \gamma, i^*, ..., N],\ \ i0 = [1:2N],\ \ j0 = [1:D]\ ,where\ i, i0, j0 \in \mathbb{Z}^+$
2  $\kappa_{(\cdot)} \sim U(0,1), \kappa_{(\cdot)} \neq 0,\ \ \lambda_{(\cdot)} \sim N(0,1)$
   // $\kappa_{(\cdot)}$, $\lambda_{(\cdot)}$ generates $(\cdot, 1)$ sized new random numbers at each call.
3  $\alpha, \beta \sim U(0,1)$
   // $\alpha, \beta$ generates a new random number at each call.
   // INITIALIZATION
4  $P_{(i0,j0)} \sim \mathbf{U}(low_{(j0)}, up_{(j0)})$
5  $fitP_{(i0)} = \mathcal{F}(P_{(i0)})$
6  for $iteration$=1 to $MaxCycle$ do
7      $j = permute(i0)$ ;  // permute$(\cdot)$ function denotes the vector permutation function
8      $k = j_{(1:N)}$ ;                                                    // see line 10 for $P_{(k)}$
9      $l = j_{(N+1:2N)}$ ;                                                 // see line 16 for $P_{(l)}$
       // GENERATION OF SUBPOPULATION; $SubP$ and $TempP$
10     $SubP = P_{(k)}$
11     $fitSubP = fitP_{(k)}$
12     for $index$=1:$N$ do
13         $w^* = \kappa^3_{(N)}\ |\ [N,1] = size(w^*)$
14         $w^* := \frac{w^*}{\sum w^*}$ ;            // (:=) denotes ‘update of variable value’ operation
15         $w := w^* \times \varDelta\ |\ \varDelta = [\mathbf{1}]\ \ |\ \ [1, D] = size(\varDelta)$
16         $TempP_{(index)} = \sum(w \circ P_{(l)})$ ;                 // $\circ$ denotes Hadamart operator
17     end
       // GENERATION OF BINARY-MAP M
18     $M_{(1:N,1:D)} = 0$
19     for $index$=1:$N$ do
20         if $\alpha < \beta$ then $K = \kappa^3_{(1)}$ else $K = 1 - \kappa^3_{(1)}$
21         $J = V(1 : \lceil K \times D \rceil)\ |\ V = permute(j0)$ ;       // $\lceil\rceil$ denotes ceiling function
22         $M_{(index,J)} := 1$
23     end
       // GENERATION OF EVOLUTIONARY STEP SIZE
       // see lines 2, 15 for $\lambda, \varDelta$, respectively.
24     if $\alpha < \beta$ then $F = [\lambda^3_{(D)}]'\ |\ [1, D] = size(F)$ else $F = (\lambda^3_{(N)} \times \varDelta)\ |\ [N, D] = size(F)$
          // TRIAL VECTOR GENERATION (MORPHOGENESIS: (Mutation & Random-Crossover)
25     $\varGamma = TempP - SubP_{(m)}\ |\ m = permute(i) | m \neq [1:N]$ ;      // see lines 10-17 for
       $TempP$ and $SubP$, respectively.
26     $T = SubP + F \times M \circ \varGamma$ ;                              // $\circ$ denotes Hadamart operator
       // BOUNDARY CONTROL
27     if $T_{(i,j0)} < low_{(j0)}$ then $T_{(i,j0)} = low_{(j0)} + \kappa^3_{(1)}(up_{(j0)} - low_{(j0)})$
28     if $T_{(i,j0)} > up_{(j0)}$ then $T_{(i,j0)} = up_{(j0)} + \kappa^3_{(1)}(low_{(j0)} - up_{(j0)})$
       // UPDATE
29     $fitT = \mathcal{F}(T)$
       // For each $i^* \in i$ run line 30. See line 1 for $i^*$.
30     if $fitT_{(i^*)} < fitSubP_{(i^*)}$ then $[SubP_{(i^*)}, fitSubP_{(i^*)}] = [T_{(i^*)}, fitT_{(i^*)}]$
31     $[P_{(l)}, fitP_{(l)}] = [SubP, fitSubP]$ ;                                 // see line 9 for $l$
       // GET THE SOLUTIONS
32     $[gmin, gbest] = [fitP_{(\gamma)}, P_{(\gamma)}]\ |\ fitP_{(\gamma)} = \min(fitP)\ |\gamma \in i$ ;    // see line 1 for $\gamma$
33 end
```

4 Experiments

Numerical optimization problem-solving capability of WDE that is introduced in this paper is examined by using CEC' 2013s benchmark problems (i.e., F1–F28) [36] that consist of very complex problems.

In this paper, in order to examine the success of WDE in the solution of real-world engineering problems, one geometric optimization problem (i.e., GPS network adjustment problem), F29 [41–43], and 4 engineering design problems have been used [2, 31].

Benchmark problems (i.e., F1–F28) have been solved for 50 trials by using a different initial population each time. The same initial population has been used for each algorithm in the experiments. Dimension of pattern matrix is 30, and stopping conditions used in the experiments are given below:

1. Stop if the absolute value of the solution obtained for the algorithm is smaller than 10^{-16}.
2. Stop if a better solution at the end of the last 200,000 function evaluations has not been obtained.
3. Stop when the function evaluation number reaches to 2,000,000.

In the tests performed in this paper, solutions obtained with WDE and comparison algorithms were pairwisely compared by using two-tailed Wilcoxon signed-rank test [44]. For Wilcoxon signed-rank tests that have been carried out in this paper, H_0 hypothesis is defined as 'data come from distributions with equal medians.' 'Significance level' is used as $\alpha = 0.05$. If a corrected p value lower than or equal to α value is produced in a test, then H_0 hypothesis is rejected for that test. Alternative hypothesis is determined as $F_A < F_B$. The validity of the alternative hypothesis is decided on by looking at whether Algorithm A provides a statistically better solution than Algorithm B or not, and the sizes of ranks provided by Wilcoxon signed-rank test (i.e., R^+, R^- as in [43]).

Initial values of control parameters of the proposed algorithm and the comparison algorithms used in this paper are given in Table 1.

4.1 Numerical function optimization problems: F1–F28

In this section, success of WDE in numerical function optimization problems has been examined with detailed applications. Basic statistical evaluations of the results (i.e., mean value of global minimum values (Ave), standard deviation value of global minimum values (Std), and runtime value in *seconds* (t (s)) obtained in the test performed by using F1–F28 are given in Table 2. The best 'Ave' values are marked with bold font in Table 2.

Mersenne Twister has been used in the tests as pseudo-random number generator [45].

Results that belong to the comparison of CEC' 2013 [36] benchmark problem (i.e., F1–F28)-solving successes of WDE and comparison algorithms by using Wilcoxon signed-rank test [39] ($p = 0.05$) are given in Table 3.

On the last row of Table 3, results obtained from WDE and comparison algorithms have been compared as $(+, =, -)$ where $(+)$ is the benchmark function number that WDE obtains a statistically better result than the related comparison algorithm, (=) is the benchmark function number that the performances of WDE and the related comparison algorithm are statistically equal and $(-)$ is the

Table 1 Initial values of control parameters of the proposed algorithm and the comparison algorithms

#	Algorithm	Initial values of control parameters
1	ABC [1]	$\text{Limit} = N \cdot D\,\text{Sizeofempoyedbee} = \text{Sizeofcolony}/2$
2	CS [3, 8]	$\beta = 1.50, p_0 = 0.25$
3	JADE [37]	$p_1 = p_2 = 0.30 \cdot \kappa \vert\, \kappa \sim \mathbf{U}(0,1)$
4	BSA [7]	$mixrate = 1.00$

benchmark function number that the related comparison algorithm obtains a statistically better result than WDE.

In the solution of CEC' 2013 benchmark problems, when results of WDE and comparison algorithms were examined in $(+, =, -)$ format, the following results are obtained as given in Table 3: CS (22,1,5), ABC (18,6,4), JADE (22,4,2), BSA (13,8,7). Accordingly, WDE had statistically better results (66.96%) than comparison algorithms in 75 out of a total of 112 piecewise comparisons. Successes of WDE and comparison algorithms are statistically similar in 19 (16.96%) comparisons. Comparison algorithms achieved statistically better results than WDE only in 18 comparisons (16.07%).

4.2 GPS baseline network adjustment problem (F29)

GPS network adjustment problem is a geometric optimization problem widely encountered in Geodesy [41–43]. In this paper, in order to review WDE's capability of solving the geometric optimization problems, the GPS Network defined in [41] has been adjusted by using WDE . The relevant GPS network includes 6 geodesic points defined as $\text{point}_\text{i} = \langle x, y, z \rangle \mid i = 1:6$. $\text{point}_{1,2}$ are the fixed values in solution of the GPS network adjustment problem. $\text{points}_{1,2}$ are given in Table 4. The baseline values of the relevant GPS Network are given in Table 5.

The covariance values of the relevant observations are necessary for least squares adjustment (LSA) [41]. The related covariance values are given in page 323 of [41] for F29. WDE does not need covariance values in order to adjust the GPS network. On the relevant GPS Network, 13 baselines have been observed. Therefore, totally $13 \times 3 = 39$ observation equations have been acquired. $\text{points}_{3:6}$ have totally $4 \times 3 = 12$ unknowns. Consequently, the relevant GPS network includes $39 - 12 = 27$ redundant observations. Therefore, in this problem $\text{degree-of-freedom} = 27$ [41]. In the adjustment process, the optimal values of totally 12 coordinate values of $\text{point}_{3:6}$ have been searched. Therefore, the problem dimension of the GPS baseline network adjustment problem is 12. According to definition, $C_{1:6} = [\ \text{point}_1 \quad \text{point}_2 \quad \ldots \quad \text{point}_6]^\text{T}$. In the objective function used for WDE, the baseline values have been computed by using $f = C_{from} - C_{to} \mid \{from, to\} \in \{1:6\}$. The relevant residual values, v, have been acquired by using Eq. 16;

$$v = f - [\varDelta x \varDelta y \varDelta z]_{1:13} \tag{16}$$

In order to protect the centeroid of the GPS network during the search process, $v := v - mean(v)$ update has been made. $mean(v)$ denotes the mean values of the residuals.

Table 2 Results that belong to tests carried out by using F1–F28

F	CS			ABC			JADE			BSA			WDE		
	Ave	Std	*t* (s)	Ave	Std	*t* (s)	Ave	Std	*t* (s)	Ave	Std	*t* (s)	Ave	Std	*t* (s)
F1	− **1400.000**	8.81e−14	3.16	− **1400.000**	0.00	4.99	− **1400.000**	2.27e−13	8.75	− **1400.000**	8.81e−14	2.26	− **1400.000**	0.00	10.67
F2	9.38e+05	4.71e+05	51.97	2.56e+06	7.87e+05	36.68	2.18e+05	1.38e+05	413.83	6.84e+03	5.51e+03	54.30	**−1300.000**	1.44e−13	65.86
F3	1.36e+08	2.55e+08	53.21	1.14e+07	8.56e+06	35.42	1.42e+08	1.80e+08	408.01	**2.11e+04**	7.09e+04	53.81	1.13e+05	1.14e+05	108.10
F4	219.507	1.47e+03	48.45	38,231.404	4.22e+03	30.67	−1073.517	4.59e+01	649.37	− **1100.000**	1.100e−06	51.98	− **1100.000**	1.24e−13	68.67
F5	− **1000.000**	9.51e−14	6.32	− **1000.000**	1.19e−13	17.03	− **1000.000**	1.22e−13	59.80	− **1000.000**	1.14e−13	1.83	− **1000.000**	0.00	17.46
F6	− 896.408	3.41	43.06	− 899.916	1.03e−01	25.88	− 887.471	2.09e+01	573.27	− **900.000**	1.08e−13	13.47	− **900.000**	0.00	51.42
F7	− 728.928	1.18e+01	37.96	− 750.511	1.16e+01	32.56	− 756.180	1.92e+01	548.48	− **786.287**	7.36	62.24	− 783.078	5.03	120.76
F8	− 679.283	4.39e−02	5.57	− 679.273	5.19e−02	25.54	− 679.305	7.19e−02	24.01	− 679.287	9.17e−02	4.67	− **679.313**	7.82e−02	16.14
F9	− 583.951	1.57	68.73	− 584.044	1.20	30.17	− 587.429	2.24	212.90	**− 587.909**	2.04	131.55	− 585.635	1.08	174.67
F10	− 499.995	5.73e−03	49.98	− 499.791	6.98e−02	49.21	− 497.559	2.04	35.04	− 499.948	5.57e−02	14.25	− **500.000**	2.46e−09	108.93
F11	− 399.851	3.55e−01	15.50	− **400.000**	4.58e−14	10.08	− 394.180	6.01	18.58	− **400.000**	1.80e−14	3.71	− **400.000**	3.22e−11	112.48
F12	− 231.668	2.47e+01	60.74	− 211.882	1.98e+01	33.90	− 240.141	1.62e+01	233.69	− 268.261	6.68	16.17	− **283.416**	3.37	116.99
F13	− 126.694	1.36e+01	50.75	− 80.966	1.47e+01	31.34	− 113.184	2.35e+01	189.50	− 135.234	1.16e+01	18.09	− **170.488**	9.32	115.97
F14	− 56.783	4.90e+01	53.47	− **99.887**	4.11e−02	41.71	163.811	1.36e+02	44.51	− 96.799	8.21	22.57	− 99.463	5.60e−01	112.44
F15	2241.617	3.71e+02	37.39	1798.467	2.26e+02	29.79	2145.751	3.74e+02	199.12	1869.153	2.67e+02	44.67	**1705.872**	2.27e+02	114.41
F16	200.937	1.58e−01	15.55	200.798	9.43e−02	33.58	201.145	3.09e−01	114.86	200.587	1.13e−01	39.30	**200.368**	7.89e−02	138.06
F17	320.589	1.82	52.59	320.342	2.06e−03	41.09	320.436	1.16e−01	55.44	318.331	6.03	20.13	**310.350**	1.00e+01	108.94
F18	475.076	1.28e+01	41.07	526.096	1.43e+01	30.32	492.532	1.89e+01	179.49	444.290	6.49	59.99	**431.403**	2.45	111.91
F19	501.719	1.98e−01	46.49	**500.076**	3.49e−02	40.30	509.711	5.48	221.27	500.287	1.28e−01	50.64	500.450	9.12e−02	107.32
F20	607.551	6.84e−01	23.69	609.447	3.24e−01	25.07	607.080	1.52	103.74	**606.164**	8.00e−01	55.34	606.628	5.82e−01	114.02
F21	895.832	6.70e+01	49.59	**819.902**	2.11e+01	27.94	1040.000	8.00e+01	19.53	1040.000	9.17e+01	5.84	855.007	5.89e+01	130.57
F22	1137.105	2.41e+02	85.81	808.500	5.85	57.17	1046.627	1.21e+02	144.97	**800.000**	1.08e−13	54.01	816.969	3.99	142.86
F23	3780.819	3.61e+02	64.72	3499.165	3.10e+02	33.36	3453.046	4.08e+02	160.52	3144.507	2.23e+02	78.81	**2898.065**	3.15e+02	145.00
F24	1245.374	6.03	134.78	**1231.720**	1.43e+01	37.04	1239.778	5.65	292.58	1236.889	5.49	144.50	1244.293	4.90	223.27
F25	1353.138	4.31	103.14	1357.813	4.74	32.37	1345.338	6.14	290.54	**1344.466**	4.66	146.44	1352.629	4.53	223.59
F26	1400.201	1.20e−01	209.75	**1397.840**	6.43	72.99	1420.577	4.89e+01	300.96	1400.002	2.36e−04	195.24	1398.931	4.66	164.95
F27	2056.377	2.63e+01	94.06	**1700.000**	7.49e−13	45.35	1909.972	6.30e+01	299.81	1938.630	3.89e+01	171.58	1959.547	1.12e+02	242.51
F28	2696.670	2.81e+02	105.38	2648.656	6.04e+02	37.01	2870.237	2.72e+02	236.45	2078.173	5.43e+02	44.16	**1712.140**	3.72e+02	154.15

Table 3 Comparison of CEC' 2013 benchmark problem (i.e., F1–F28)-solving successes of WDE and comparison algorithms by using Wilcoxon signed-rank test ($p = 0.05$)

Fnc	CS				ABC				JADE				BSA			
	p value	zval	Ranksum	Winner	*p* value	zval	Ranksum	Winner	*p* value	zval	Ranksum	Winner	*p* value	zval	Ranksum	Winner
F1	4.02E−02	− 1.75	380	−	2.34E−10	− 6.23	210	+	9.64E−06	− 4.27	280	+	4.02E−02	− 1.75	380	−
F2	1.98E−08	− 5.49	210	+	1.98E−08	− 5.49	210	+	1.98E−08	− 5.49	210	+	1.98E−08	− 5.49	210	+
F3	7.88E−07	− 4.80	232	+	3.40E−08	− 5.40	210	+	9.59E−08	− 5.21	217	+	1.00	4.56	578	=
F4	1.65E−08	− 5.52	210	+	1.65E−08	− 5.52	210	+	1.65E−08	− 5.52	210	+	1.65E−08	− 5.52	210	+
F5	1.86E−09	− 5.90	210	+	1.63E−09	− 5.92	210	+	1.57E−09	− 5.92	220	+	2.34E−10	− 6.23	210	+
F6	1.50E−08	− 5.54	220	+	4.00E−09	− 5.77	210	+	4.00E−09	− 5.77	210	+	2.73E−09	− 5.83	210	+
F7	3.40E−08	− 5.40	210	+	3.95E−08	− 5.37	211	+	2.27E−07	− 5.04	223	+	9.55E−01	1.69	472	−
F8	1.14E−01	− 1.20	365	−	5.99E−02	− 1.56	352	−	4.09E−01	− 0.23	401	−	7.39E−02	− 1.45	356	−
F9	6.14E−04	− 3.23	290	+	1.69E−04	− 3.58	277	+	9.95E−01	2.58	505	=	1.00	3.83	551	=
F10	2.19E−02	− 2.02	335	+	3.40E−08	− 5.40	210	+	3.40E−08	− 5.40	210	+	3.39E−08	− 5.40	210	+
F11	1.00	3.89	550	=	1.00	5.53	610	=	7.84E−06	− 4.32	250	+	1.00	5.69	610	=
F12	3.40E−08	− 5.40	210	+	3.40E−08	− 5.40	210	+	3.40E−08	− 5.40	210	+	1.11E−07	− 5.18	218	+
F13	4.59E−08	− 5.34	212	+	3.40E−08	− 5.40	210	+	3.40E−08	− 5.40	210	+	6.17E−08	− 5.29	214	+
F14	3.40E−08	− 5.40	210	+	1.00	5.40	609	=	3.40E−08	− 5.40	210	+	9.80E−01	2.05	485	−
F15	5.19E−05	− 3.88	266	+	1.14E−01	− 1.20	365	−	1.37E−04	− 3.64	275	+	2.83E−02	− 1.91	339	−
F16	3.40E−08	− 5.40	210	+	3.40E−08	− 5.40	210	+	3.40E−08	− 5.40	210	+	1.03E−06	− 4.75	234	+
F17	1.48E−07	− 5.13	220	+	5.05E−E−01	0.01	410	−	8.12E−04	− 3.15	293	+	6.35E−01	0.34	422	−
F18	3.40E−08	− 5.40	210	+	3.40E−08	− 5.40	210	+	3.40E−08	− 5.40	210	+	9.59E−08	− 5.21	217	+
F19	3.40E−08	− 5.40	210	+	1.00	5.42	610	=	3.40E−08	− 5.40	210	+	1.00	3.77	549	=
F20	1.11E−04	− 3.69	273	+	3.40E−08	− 5.40	210	+	2.20E−01	− 0.77	381	−	9.40E−01	1.56	467	−
F21	3.47E−01	− 0.39	395	−	7.95E−01	0.83	440	−	9.93E−06	− 4.27	254	+	1.09E−05	− 4.25	254	+
F22	3.40E−08	− 5.40	210	+	1.00	4.34	570	=	3.40E−08	− 5.40	210	+	1.00	5.49	610	=
F23	4.56E−07	− 4.91	228	+	5.52E−06	− 4.40	247	+	6.47E−05	− 3.83	268	+	6.66E−03	− 2.48	318	+
F24	3.38E−01	− 0.42	394	−	9.99E−01	3.02	521	=	9.96E−01	2.66	508	=	1.00	3.96	556	=
F25	4.30E−01	− 0.18	403	−	2.35E−03	− 2.83	305	+	1.00	3.72	547	=	1.00	4.29	568	=
F26	1.22E−08	− 5.58	210	+	3.08E−04	− 3.42	287	+	1.22E−08	− 5.58	210	+	1.22E−08	− 5.58	210	+
F27	1.75E−06	− 4.64	238	+	1.00	5.45	610	=	9.97E−01	2.77	512	=	9.97E−01	2.77	512	=
F28	2.24E−07	− 5.05	223	+	4.83E−06	− 4.42	246	+	8.18E−08	− 5.24	216	+	2.18E−02	− 2.02	335	+
Total		+		22	18				22				13			
		=		1	6				4				8			
		−		5	4				2				7			

Table 4 Coordinates of fixed points; 1, 2 [41]

Points	*X* (m)	*Y* (m)	*Z* (m)
1	402.35087	− 4,652,995.30109	4,349,760.77753
2	8086.03178	− 4,642,712.84739	4,360,439.08326

The objective function used by WDE in order to adjust the GPS Network is given in Eq. 17;

$$\underset{\text{point}_{3:6}}{\operatorname{argmin}}\ v^{\mathrm{T}} v \tag{17}$$

While solving the GPS network adjustment problem by using WDE, the pattern matrix dimension has been selected as $N = 20$. The problem dimension is $D = 12$. WDE does not use the search space limits while solving the GPS network adjustment problem.

The initial pattern matrix has been generated by using $P_{(i0,j0)} \sim \mathbf{U}(0, 1)$. The global minimum value WDE has acquired in the final iteration is 0.0026484.

The coordinates of the adjusted network points obtained with solution of the GPS network adjustment problem and the relevant coordinate differences are given in Table 6.

When the adjusted-coordinates given in Table 6 have been compared by using Anova test, p value $= 0.00000$ has been obtained for X, Y, and Z. Therefore, the means values of the relevant adjusted-coordinates obtained by using LSA and WDE for p value < 0.05 are statistically equal. The loop closure values, Δerr_{loop}, computed after the adjustment are given in Table 7.

When the Δerr_{loop} values have been compared statistically by using Anova test, p value $= 4.2603e{-}08$ has been obtained. In that case, it can be said that the differences between err_{LSA} and err_{WDE} are not statistically significant for p value < 0.05. In that case, the mean of err_{LSA} and err_{WDE} belongs to an equal population. Therefore, LSA and WDE have reached statistically the same success for the Δerr_{loop} values in solution of the GPS network adjustment problem.

The mean and standard deviation values of the residuals obtained after completion of the adjustment process are given in Table 8.

After a review of Table 8, it is possible to say that the GPS network adjustment results achieved by using WDE protect the centeroid of the relevant network better than LSA.

The standard deviation values, m_0, of the residuals computed for the baselines are given in Table 9.

After a review of Table 9, it can be said that WDE has obtained a better m_0 value compared to LSA. Therefore, WDE is successful in adjustment of the relevant GPS network.

In solution of the GPS network adjustment problem, WDE provides the following advantages in comparison to LSA:

- There is no statistical difference between the results obtained in solution of the GPS network adjustment problem by using WDE or LSA. However, in order to achieve the solution, LSA uses a mathematical model that is much more complex than the mathematical model used by WDE.
- In order to solve the relevant problem, LSA needs covariance values. However, WDE uses only the observed baseline parameters and fixed point parameters to solve the relevant problem.
- WDE protects the centeroid of the relevant network more accurately.

Table 5 The baseline values used in GPS baseline network adjustment problem (i.e., F29) [41]

Observation	Points		Baseline components (m)		
	From	To	Δx	Δy	Δz
1	1	3	11,644.22320	3601.21650	3399.25500
2	1	5	− 5321.71640	3634.07540	3173.66520
3	2	3	3960.54420	− 6681.24670	− 7279.01480
4	2	4	− 11,167.60760	− 394.52040	− 907.95930
5	4	3	15,128.16470	− 6286.70540	− 6371.05830
6	4	5	− 1837.74590	− 6253.85340	− 6596.66970
7	6	1	− 1116.45230	− 4596.16100	− 4355.89620
8	6	3	10,527.78520	− 994.93770	− 956.62460
9	6	5	− 6438.13640	− 962.06940	− 1182.23050
10	6	4	− 4600.37870	5291.77850	5414.43110
11	6	2	6567.23110	5686.29260	6322.39170
12	2	6	− 6567.23100	− 5686.30330	− 6322.38070
13	1	6	1116.45770	4596.15530	4355.91410

Table 6 The coordinates of the adjusted network points obtained with solution of the GPS network adjustment problem and the coordinate differences

Point#	Adjusted-coordinates by using LSA (m)			Adjusted-coordinates by using WDE (m)			Coordinate differences (m)		
	X (m)	Y (m)	Z (m)	X (m)	Y (m)	Z (m)	$X_{LSA} - X_{WDE}$	$Y_{LSA} - Y_{WDE}$	$Z_{LSA} - Z_{WDE}$
1	402.35087	− 4,652,995.30109	4,349,760.77753	402.35087	− 4,652,995.30109	4,349,760.77,753	0.00000	0.00000	0.00000
2	8086.03178	− 4,642,712.84739	4,360,439.08326	8086.03178	− 4,642,712.84,739	4,360,439.08326	0.00000	0.00000	0.00000
3	12,046.58076	− 4,649,394.08256	4,353,160.06335	12,046.57838	− 4,649,394.08482	4,353,160.05592	0.00238	0.00226	0.00743
4	− 3081.58313	− 4,643,107.36915	4,359,531.12202	− 3081.58523	− 4,643,107.37096	4,359,531.11878	0.00210	0.00181	0.00324
5	− 4919.33908	− 4,649,361.21987	4,352,934.45341	− 4919.34476	− 4,649,361.22199	4,352,934.44937	0.00568	0.00212	0.00404
6	1518.80119	− 4,648,399.14533	4,354,116.68936	1518.79878	− 4,648,399.14653	4,354,116.68679	0.00241	0.00120	0.00257

– WDE is capable of obtaining better m_0 values than LSA.

4.3 Engineering design problems: F29–F32

F29–F32 are nonlinear constrained engineering optimization problems, and they were widely analyzed in optimization literature [2, 31]. Therefore, F29–F32 also were solved in this study:

– Pressure-vessel design problem, F29 [2, 31],
– Speed-reducer design problem, F30 [2],
– Tension/compression string design problem, F31 [2, 31],
– Welded-beam design problem, F32 [2, 31],

In this paper, the constrained handling method [2, 45, 46] used in [2] is employed to solve F29–F32 problems.

In the solution of F29–F32, problem-solving successes of particle swarm optimization algorithm (PSO2011) [2, 13], cumulative-information-based differential evolution algorithm (CPI-JDE) [38], Advanced Artificial Cooperative Search Algorithm (A+) [39], CS, ABC, JADE, BSA, and WDE (proposed) were compared. CPI-JDE and JADE are relatively new, and they are very successful DE algorithms.

Statistical results that WDE and comparison algorithms have obtained for F29–F32 are given in Table 10. When Table 10 is examined, WDE is seen to have obtained better results than PSO2011, CPI-JADE, ABC, and A+. Related problem-solving successes of WDE, JADE, BSA, and CS are statistically similar to a large extent.

The detailed definitions of the mechanical-engineering-based nonlinear constrained optimization problems (i.e., F29–F32) are given in Sects. 4.4, 4.5, 4.6, and 4.7.

4.4 Pressure-vessel design problem: F29

This problem aims the design of a compressed air storage tank under a working pressure of 3000 psi and a minimum volume of $750 ft^3$, the schematic structure of which is shown in Fig. 2.

The mathematical model of this problem is defined in Eq. 18 [2, 31].

$$\begin{aligned}\underset{x}{\text{argmin}}\ f(x) &= 0.6224 \cdot x_1 \cdot x_3 \cdot x_4 + 19.84 \cdot x_1^2 \cdot x_3 \\ &+ 1.7781 \cdot x_2 \cdot x_3^3 + 3.1661 \cdot x_1^2 \cdot x_4\end{aligned} \tag{18}$$

Subject to:

Table 7 Post-adjustment loop closure values (in m); $\Delta err_{loop} = err_{LSA} - err_{WDE}$

Loop#	Loop edges	err_{LSA} (1)	err_{WDE}	Δerr_{loop}
1	$\overrightarrow{1,3}$, $\overrightarrow{6,3}, \overrightarrow{6,1}$	0.03770	0.04663	-0.00893
2	$\overrightarrow{1,3}$, $\overrightarrow{2,3}$, $\overrightarrow{6,2}$, $\overrightarrow{6,1}$	0.03210	0.03210	0.00000
3	$\overrightarrow{1,3}$, $\overrightarrow{2,3}$, $\overrightarrow{2,6}$, $\overrightarrow{1,6}$	0.03930	0.03930	0.00000
4	$\overrightarrow{1,5}$, $\overrightarrow{6,5}$, $\overrightarrow{6,1}$	0.04900	0.05793	− 0.00893
5	$\overrightarrow{1,5}$, $\overrightarrow{4,5}$, $\overrightarrow{6,4}$, $\overrightarrow{6,1}$	0.06240	0.06240	0.00000
6	$\overrightarrow{1,3}$, $\overrightarrow{2,3}$, $\overrightarrow{2,4}$, $\overrightarrow{4,5}$, $\overrightarrow{1,5}$	0.08030	0.08512	− 0.00482
7	$\overrightarrow{4,5}$, $\overrightarrow{6,4}$, $\overrightarrow{6,5}$	0.02540	0.02595	− 0.00055
8	$\overrightarrow{4,5}$, $\overrightarrow{2,4}$, $\overrightarrow{6,2}$, $\overrightarrow{6,5}$	0.03260	0.03370	− 0.00110
9	$\overrightarrow{2,4}$, $\overrightarrow{6,2}$, $\overrightarrow{6,4}$	0.00980	0.01173	− 0.00193
10	$\overrightarrow{2,4}$, $\overrightarrow{2,6}$, $\overrightarrow{6,4}$	0.01620	0.01138	0.00482
11	$\overrightarrow{2,4}$, $\overrightarrow{2,3}$, $\overrightarrow{4,3}$	0.03660	0.04142	− 0.00482
12	$\overrightarrow{1,5}$, $\overrightarrow{6,5}$, $\overrightarrow{6,3}$, $\overrightarrow{1,3}$	0.04350	0.04350	0.00000
13	$\overrightarrow{2,6}$, $\overrightarrow{6,2}$	0.03260	0.03370	− 0.00110
14	$\overrightarrow{2,4}$, $\overrightarrow{2,3}$, $\overrightarrow{6,3}$, $\overrightarrow{6,4}$,	0.02240	0.02240	0.00000
15	$\overrightarrow{6,5}$, $\overrightarrow{4,5}$, $\overrightarrow{2,4}$, $\overrightarrow{2,3}$, $\overrightarrow{6,3}$	0.03680	0.04162	− 0.00482
16	$\overrightarrow{6,5}$, $\overrightarrow{4,5}$, $\overrightarrow{4,3}$, $\overrightarrow{6,3}$	0.03280	0.03280	0.00000
17	$\overrightarrow{1,6}$, $\overrightarrow{6,1}$	0.02900	0.00856	0.02044
18	$\overrightarrow{1,6}$, $\overrightarrow{2,6}$, $\overrightarrow{6,2}, \overrightarrow{6,1}$	0.01720	0.01720	0.00000

Table 8 Mean and standard deviation values of residuals

Residuals	LSA		WDE	
	μ (m)	σ	μ (m)	σ
v_x	− 0.0000669	0.0093955	0.0000000	0.0085625
v_y	0.0017000	0.0051921	0.0000000	0.0051045
v_z	0.0000746	0.0113934	0.0000000	0.0096674

Table 9 m_0 values of the residual values obtained by using LSA and WDE

Statistics	Method	
	LSA (m)	WDE (m)
$m_0 = \sqrt{\frac{v^T v}{27}}$	0.00833	0.00735

$$\begin{aligned} c_1(x) &= -x_1 + 0.0193 \cdot x_3 \le 0 \\ c_2(x) &= -x_2 + 0.0954 \cdot x_3 \le 0 \\ c_3(x) &= -\pi \cdot x_3^2 \cdot x_4 - \frac{4}{3} \cdot \pi \cdot x_3^3 + 1,296,000 \le 0 \\ c_4(x) &= x_4 - 240 \le 0 \end{aligned} \tag{19}$$

where the bounds are,

$$0.0625 \le x_1, x_2 \le 6.1875 \tag{20}$$

and

$$10 \le x_3, x_4 \le 200 \tag{21}$$

4.5 Speed-reducer design problem: F30

This problem aims the design of the speed reducer shown in Fig. 3.

The mathematical model of this problem is defined in Eq. 22 [2, 31].

$$\begin{aligned} \underset{x}{\text{argmin}}\ f(x) = 0.7854 \cdot x_1 \cdot x_2^2 \cdot (3.3333 \cdot x_3^2 + 14.9334 \\ \cdot x_3 - 43.0934) \\ - 1.508 \cdot x_1 \cdot (x_6^2 + x_7^2) + 7.4777 \cdot (x_6^2 + x_7^2) \\ + 0.78054 \cdot (x_4 \cdot x_6^2 + x_5 \cdot x_7^2) \end{aligned} \tag{22}$$

Subject to

Table 10 The mean-solution (Ave) and standard deviation of mean-solution (Std) values obtained by the relevant algorithms for the F29–F32

Algorithm	F	Ave	Std
PSO2011	F29	6235.36961742	2.51e+02
	F30	3108.37805916	1.53e+02
	F31	0.01275660	1.12e−04
	F32	1.86762054	1.16e−01
CPI-JDE	F29	6310.75991379	2.23e+02
	F30	3004.18039468	4.25e+01
	F31	0.01368178	1.00e−03
	F32	2.55281872	4.32e−01
A+	F29	**5885.33277360**	1.70e−13
	F30	**2994.92524435**	6.00e−13
	F31	0.01267052	1.27e−05
	F32	**1.72485231**	0.00e+00
CS	F29	5910.20912757	1.45e+01
	F30	**2994.92524435**	4.90e−13
	F31	0.01272432	2.68e−05
	F32	1.77712735	2.10e−02
ABC	F29	6920.90919830	4.30e+02
	F30	3089.68804116	5.21e+01
	F31	0.01342275	4.39e−04
	F32	1.89243016	1.05e−01
JADE	F29	**5885.33277360**	0.00e+00
	F30	**2994.92524435**	4.60e−13
	F31	**0.01266523**	0.00e+00
	F32	**1.72485231**	0.00e+00
BSA	F29	**5885.33277360**	0.00e+00
	F30	**2994.92524435**	4.60e−13
	F31	**0.01266523**	0.00e+00
	F32	**1.72485231**	0.00e+00
WDE (proposed)	F29	**5885.33277360**	0.00e+00
	F30	**2994.92524435**	4.60e−13
	F31	**0.01266523**	0.00e+00
	F32	**1.72485231**	0.00e+00

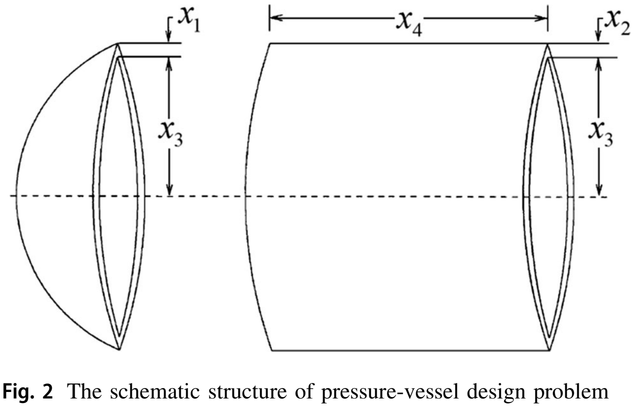

Fig. 2 The schematic structure of pressure-vessel design problem

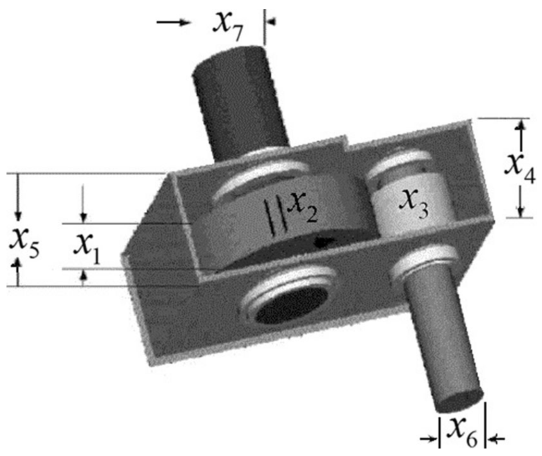

Fig. 3 Speed-reducer problem

$$
\begin{aligned}
c_1(x) &= \frac{27}{x_1 \cdot x_2^2 \cdot x_3} - 1 \le 0 \\
c_2(x) &= \frac{397.5}{x_1 \cdot x_2^2 \cdot x_3^2} - 1 \le 0 \\
c_3(x) &= \frac{1.93 \cdot x_4^3}{x_2 \cdot x_3 \cdot x_6^4} - 1 \le 0 \\
c_4(x) &= \frac{1.93 \cdot x_5^3}{x_2 \cdot x_3 \cdot x_7^4} - 1 \le 0 \\
c_5(x) &= \frac{1}{110 \cdot x_6^3} \sqrt{\left(\frac{750.0 \cdot x_4}{x_2 \cdot x_3}\right)^2 + 16.9 \cdot 10^6} - 1 \le 0 \\
c_6(x) &= \frac{1}{85 \cdot x_7^3} \sqrt{\left(\frac{750.0 \cdot x_5}{x_2 \cdot x_3}\right)^2 + 157.5 \cdot 10^6} - 1 \le 0 \\
c_7(x) &= \frac{x_2 \cdot x_3}{40} - 1 \le 0 \\
c_8(x) &= \frac{5 \cdot x_2}{x_1} - 1 \le 0 \\
c_9(x) &= \frac{x_1}{12 \cdot x_2} - 1 \le 0 \\
c_{10}(x) &= \frac{1.5 \cdot x_6 + 1.9}{x_4} - 1 \le 0 \\
c_{11}(x) &= \frac{1.5 \cdot x_7 + 1.9}{x_5} - 1 \le 0
\end{aligned}
\tag{23}
$$

where the bounds are,

$$
\begin{aligned}
&2.6 \le x_1 \le 3.6 \\
&0.7 \le x_2 \le 0.8 \\
&17 \le x_3 \le 28 \\
&7.3 \le x_4 \le 8.3 \\
&7.8 \le x_5 \le 8.3 \\
&2.9 \le x_6 \le 3.9 \\
&5.0 \le x_7 \le 5.5
\end{aligned}
\tag{24}
$$

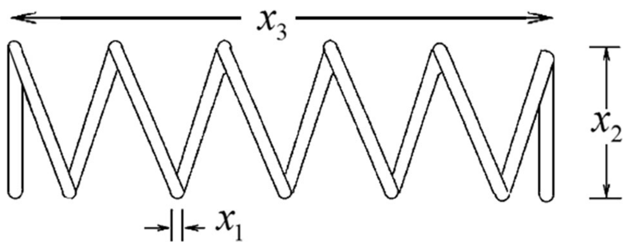

Fig. 4 Tension/compression spring problem

4.6 Tension/compression spring design problem: F31

The aim of this problem is to determine the values of the relevant parameters minimizing the weight of a tension/compression spring under various constraints [2, 31, 46, 47]. The schematic structure of a tension/compression spring is shown in Fig. 4.

The mathematical model of this problem is defined in Eq. 25 [2, 32].

$$\underset{x}{\operatorname{argmin}}\ f(x) = (x_3 + 2) \cdot x_1^2 \cdot x_2 \tag{25}$$

Subject to

$$\begin{aligned}
c_1(x) &= 1 - \frac{x_2^3 \cdot x_3}{71785 \cdot x_1^4} \le 0\\
c_2(x) &= \frac{4x_2^2 - x_1 \cdot x_2}{12566 \cdot x_1^3 \cdot x_2 - x_1^4} + \frac{1}{5108 \cdot x_1} - 1 \le 0\\
c_3(x) &= 1 - \frac{140.45 \cdot x_1}{x_2^2 \cdot x_3} \le 0\\
c_4(x) &= \frac{x_1 + x_2}{1.50} - 1 \le 0
\end{aligned} \tag{26}$$

where the bounds are

$$\begin{aligned}
0.05 &\le x_1 \le 2.0\\
0.25 &\le x_2 \le 1.3\\
2.0 &\le x_3 \le 15.0
\end{aligned} \tag{27}$$

4.7 Welded-beam design problem: F32

This problem aims dimensioning of the welded steel beam [2, 31], the structure of which is shown in Fig. 5, and determining the welding length value.

The mathematical model of this problem is given in Eq. 28;

$$\underset{x}{\operatorname{argmin}}\ f(x) = 1.10471 \cdot x_1^2 \cdot x_2 + 0.04811 \cdot x_3 \cdot x_4 \cdot (14 + x_2) \tag{28}$$

Subject to

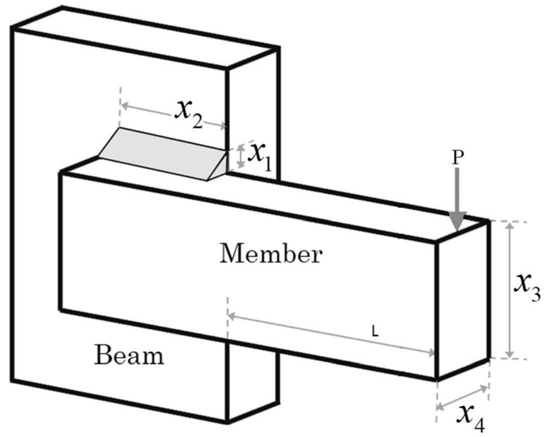

Fig. 5 The schematic structure of welded-beam design problem

$$\begin{aligned}
c_1(x) &= t(x) - t_{\max} \le 0\\
c_2(x) &= \sigma(x) - \sigma_{\max} \le 0\\
c_3(x) &= x_1 - x_4 \le 0\\
c_4(x) &= 0.10471 \cdot x_1^2 + 0.04811 \cdot x_3 \cdot x_4 \cdot (14 + x_2)\\
&\quad - 5 \le 0\\
c_5(x) &= 0.125 - x_1 \le 0\\
c_6(x) &= \delta(x) - \delta_{\max} \le 0\\
c_7(x) &= P - P_c(x) \le 0
\end{aligned} \tag{29}$$

where

$$\begin{aligned}
t(x) &= \sqrt{t_1^2 + \frac{2 \cdot t_1 \cdot t_2 \cdot x_2}{2 \cdot R} + t_2^2}\\
t_1 &= \frac{P}{\sqrt{2} \cdot x_1 \cdot x_2}\\
t_2 &= \frac{M \cdot R}{J}\\
M &= P \cdot \left(L + \frac{x_2}{2}\right)\\
R &= \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}\\
J &= 2 \cdot \left\{\sqrt{2} \cdot x_1 \cdot x_2 \cdot \left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\}\\
\sigma(x) &= \frac{6 \cdot P \cdot L}{x_4 \cdot x_3^2}\\
\delta(x) &= \frac{4 \cdot P \cdot L^3}{E \cdot x^4 \cdot x_3^3}\\
P_c(x) &= \frac{4.013 \cdot \sqrt{E \cdot G \cdot x_3^2 \cdot x_4^6/36}}{L^2}\left(1 - \frac{x^3}{2 \cdot L} \cdot \sqrt{\frac{E}{4 \cdot G}}\right)
\end{aligned} \tag{30}$$

The material properties and constraint values used above are given as follows:

$$\begin{aligned} P &= 6000 \\ L &= 14 \\ \delta_{\max} &= 0.25 \\ E &= 30 \cdot 10^6 \\ G &= 12 \cdot 10^6 \\ t_{\max} &= 13{,}600 \\ \sigma_{\max} &= 30{,}000 \end{aligned} \quad (31)$$

where the bounds are,

$$\begin{aligned} &0.1 \le x_1 \le 2 \\ &0.1 \le x_2 \le 10 \\ &0.1 \le x_3 \le 10 \\ &0.1 \le x_4 \le 2 \end{aligned} \quad (32)$$

5 Conclusions and possible research directions

The determination of most feasible evolution direction and evolutionary step size is the most important problem that is encountered in evolutionary computation. Determination of the most feasible evolutionary direction depends on the type of the problem to a high extent. WDE, introduced in this paper, uses fully randomized control parameters, hence it does not have any control parameters in practice. WDE uses *swarmmized* sub-pattern matrices in order to generate unique direction patterns. This property of WDE facilitates its solving problems of different types to a high extent. The simple strategy that WDE uses for evolutionary step size generation brings in a quite feasible local and global search capability. Functioning of WDE consists of classic genetic processes (i.e., selection, mutation, crossover). WDE is a bijective, non-recursive, swarm-based global search algorithm. Performed tests show that WDE is statistically very successful in the solution of numerical function optimization problems. WDE is also quite successful in the solution of real-world design problems. As demonstrated by the statistical results obtained by the extensive tests that have been carried out, WDE is shown to be quite successful in the solution of complex numerical problems.

In this paper, the general structure of WDE and the problem-solving success are examined. In order to further develop WDE in the future, the possibilities for using different chaotic maps can be explored. In future studies, the performance of WDE in solution of large-scale optimization problems can be examined.

As a consequence, WDE has a potential to be used in the solution of various engineering design problems, signal processing applications, and various industrial design problems.

Compliance with ethical standards

Conflict of interest The authors declare that there is no conflict of interest regarding the publication of this paper.

Funding The authors would like to thank the editor and the referees for their contribution in enhancing the technical quality of this paper. Some parts of this paper were supported by the below mentioned projects: Erciyes University BAP FDA-2013-4530, FBA-10-3067, FBA-9-1131, FBA-2013-4525, 06-AY-15 and Tubitak 115Y235.

References

1. Karaboga D, Akay B (2009) A comparative study of artificial bee colony algorithm. Appl Math Comput 214(12):108–132
2. Civicioglu P (2013) Artificial cooperative search algorithm for numerical optimization problems. Inform Syst 229:58–76
3. Yang XS, Deb S (2009) Cuckoo search via levy flights. World congress on nature and biologically inspired computing-Nabic'2009. Coimbatore, India, vol 4, pp 210–214
4. Simon D (2008) Biogeography-based optimization. IEEE Trans Evol Comput 12(6):702–713
5. Qin AK, Huang VL, Suganthan PN (2009) Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evol Comput 13(2):398–417
6. Yong W, Han-Xion L, Tingwen H, Long L (2014) Differential evolution based on covariance matrix learning and bimodal distribution parameter setting. Appl Soft Comput 218:232–247
7. Civicioglu P (2013) Backtracking search optimization algorithm for numerical optimization problems. Appl Math Comput 219:8121–8144
8. Civicioglu P, Beşdok E (2013) A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms. Artif Intell Rev 39(4):315–346
9. Civicioglu P (2012) Transforming geocentric cartesian coordinates to geodetic coordinates by using differential search algorithm. Comput Geosci 46:229–247
10. Bratton D, Kennedy J (2007) Defining a standard for particle swarm optimization. In: IEEE swarm intelligence symposium, Honolulu 1-4244-0708-7
11. Liang JJ, Qin AK, Suganthan PN, Baskar S (2006) Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans Evol Comput 10:281–295
12. Clerc M, Kennedy J (2002) The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Trans Evol Comput 6(1):58–73
13. Omran MGH, Clerc M (2015) http://www.particleswarm.info/Programs.html. Accessed 20 Feb 2018
14. Hansen N, Ostermeier A (2001) Completely derandomized self-adaptation in evolution strategies. Evol Comput 9(2):159–195
15. Price KV, Storn R, Lampinen J (2005) Differential evolution: a practical approach to global optimization. Springer, Berlin
16. Salimi H (2015) Stochastic fractal search: a powerful metaheuristic algorithm. Knowl-Based Syst 75:1–18
17. Cheng MY, Prayogo D (2014) Symbiotic organisms search: a new metaheuristic optimization algorithm. Comput Struct 139:98–112

18. Rashedi E, Nezamabadi-pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci 13:2232–2248
19. Sadollah A, Bahreininejad A, Eskandar H, Hamdi M (2013) Mine blast algorithm: a new population based algorithm for solving constrained engineering optimization problems. Appl Soft Comput 13:2592–2612
20. Wang D, Wua Z, Fei Y, Zhang W (2014) Structural design employing a sequential approximation optimization approach. Comput Struct 134:75–87
21. Maheri MR, Narimani MM (2014) An enhanced harmony search algorithm for optimum design of side sway steel frames. Comput Struct 136:78–89
22. Civicioglu P, Alcı M (2004) Edge detection of highly distorted images suffering from impulsive noise. AEU Int J Electron C 58(6):413–419
23. Wu X, Yang Z (2013) Nonlinear speech coding model based on genetic programming. Appl Soft Comput 13(7):3314–3323
24. Yoon Y, Kim YH (2013) An efficient genetic algorithm for maximum coverage deployment in wireless sensor networks. IEEE Trans Cybern 43(5):1473–1483
25. Civicioglu P, Alcı M, Besdok E (2004) Using an exact radial basis function artificial neural network for impulsive noise suppression from highly distorted image databases. LNCS 3261:383–391
26. Chauhan RS, Arya SK (2013) An optimal design of IIR digital filter using particle swarm optimization. Appl Artif Intell 27(6):429–440
27. Yan Y, He Y, Hu Y, et al (2014) Video superresolution via parameter-optimized particle swarm optimization. Math Probl Eng 373425
28. Moezi SA, Zakeri E, Zare A, Nedaei M (2015) On the application of modified cuckoo optimization algorithm to the crack detection problem of cantilever Euler–Bernoulli beam. Comput Struct 157:42–50
29. Wang GG, Gandomi AH, Alavi AH, Deb S (2016) A hybrid method based on krill herd and quantum-behaved particle swarm optimization. Neural Comput Appl 4(27):989–1006
30. Heidari AA, Abbaspour RA, Jordehi AR (2017) An efficient chaotic water cycle algorithm for optimization tasks. Neural Comput Appl 1(28):57–85
31. Faris H, Aljarah I, Azmi Al-Betar M, Mirjalili S (2017) Grey wolf optimizer: a review of recent variants and applications. Neural Comput Appl 30:413–435
32. Aljarah I, Faris H, Mirjalili S, Al-Madi N (2018) Training radial basis function networks using biogeography-based optimizer. Neural Comput Appl 7(29):529–553
33. Wang GG, Gandomi AH, Alavi AH, Hao GS (2014) Hybrid krill herd algorithm with differential evolution for global numerical optimization. Neural Comput Appl 2(25):297–308
34. Mirjalili S (2016) Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput Appl 4(27):1053–1073
35. Alweshah M (2018) Construction biogeography-based optimization algorithm for solving classification problems. Neural Comput Appl. https://doi.org/10.1007/s00521-018-3402-8
36. Liang JJ, Qu BY, Suganthan PN, Hernandez-Diaz AG (2013) Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization. Technical report 201212, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, January 2013
37. Zhang J, Sanderson AC (2009) JADE: adaptive differential evolution with optional external archive. IEEE Trans Evol Comput 13(5):945–958
38. Wang Y, Liu ZZ, Li J et al (2016) Utilizing cumulative population distribution information in differential evolution. Appl Soft Comput 48:329–346
39. Civicioglu P, Besdok E (2018) A+ Evolutionary search algorithm and QR decomposition based rotation invariant crossover operator. Expert Syst Appl 103:49–62
40. https://www.mathworks.com/matlabcentral/fileexchange/68370-weighted-differential-evolution-algorithm-wde. Accessed 20 Feb 2018
41. Ghilani CD, Wolf PR (2006) Adjustment computations, spatial data analysis, Forth edn. Wiley, New Jersey
42. Yetkin M, Berber M (2014) Implementation of robust estimation in GPS networks using the Artificial Bee Colony algorithm. Earth Sci Inform 7:39–46. https://doi.org/10.1007/s12145-013-0131-5
43. Yetkin M (2018) Application of robust estimation in geodesy using the harmony search algorithm. J Spat Sci 63(1):63–73. https://doi.org/10.1080/14498596.2017.1341856
44. Derrac J, Garca S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol Comput 1:3–18
45. Matsumoto M, Nishimura T (1998) Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Trans Model Comput Sci 8(1):3–30
46. Mezura-Montesa E, Coellob CAC (2011) Constraint-handling in nature-inspired numerical optimization: past, present and future. Swarm Evol Comput 1(4):173–194
47. Deb K (2000) An efficient constraint handling method for genetic algorithms. Comput Methods Appl Mech Engrg 186:311–338