Information Sciences 212 (2012) 79–93

Contents lists available at SciVerse ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A note on teaching–learning-based optimization algorithm

Matej Črepinšek [a,*], Shih-Hsi Liu [b], Luka Mernik [c]

[a] University of Maribor, Faculty of Electrical Engineering and Computer Science, Smetanova 17, 2000 Maribor, Slovenia
[b] California State University, Fresno, Department of Computer Science, 2576 E San Ramon Dr, Fresno, CA 93740, USA
[c] California Institute of Technology, 1200 East California Blvd, Pasadena, CA 91125, USA

ARTICLE INFO

Article history:
Received 13 October 2011
Received in revised form 10 May 2012
Accepted 16 May 2012
Available online 28 May 2012

Keywords:
Teaching–learning-based optimization
Constrained optimization problems
Unconstrained optimization problems
Experimental replication

ABSTRACT

Teaching–Learning-Based Optimization (TLBO) seems to be a rising star from amongst a number of metaheuristics with relatively competitive performances. It is reported that it outperforms some of the well-known metaheuristics regarding constrained benchmark functions, constrained mechanical design, and continuous non-linear numerical optimization problems. Such a breakthrough has steered us towards investigating the secrets of TLBO's dominance. This paper reports our findings on TLBO qualitatively and quantitatively through code-reviews and experiments, respectively. Our findings have revealed three important mistakes regarding TLBO: (1) at least one unreported but important step; (2) incorrect formulae on a number of fitness function evaluations; and (3) misconceptions about parameter-less control. Additionally, unfair experimental settings/conditions were used to conduct experimental comparisons (e.g., different stopping criteria). The experimental results for constrained and unconstrained benchmark functions under fairly equal conditions failed to validate its performance supremacy. The ultimate goal of this paper is to provide reminders for metaheuristics' researchers and practitioners in order to avoid similar mistakes regarding both the qualitative and quantitative aspects, and to allow fair comparisons of the TLBO algorithm to be made with other metaheuristic algorithms.

1. Introduction

"*A metaheuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions* [33]." Some fundamental properties of metaheuristics have been identified in [5,6]. For example, metaheuristics range from simple local searches to complex learning processes. They guide the search process strategically, explore search space to find (near-)optimal solutions, and incorporate mechanisms to avoid becoming trapped. Such algorithms are mainly used for solving combinatorial and numerical optimization problems within both academia and industry (e.g., [14,34,31,17,43,19,30,35]). Genetic Algorithms (GAs) [15], Evolutionary Strategies (ESs) [4], Particle Swarm Optimization (PSO) [23], Differential Evolution (DE) [40], Artificial Bee Colony (ABC) [21], Harmony Search (HS) [25], and the Grenade Explosion Method (GEM) [1], are some of the well-known algorithms falling within this category. There have also been many other metaheuristics introduced over recent years (e.g., cuckoo search [41], monkey search [32], and intelligent water drops [39]).

Teaching–Learning-Based Optimization (TLBO) is a kind of newly introduced metaheuristics [36,37]. It is a population-based optimization algorithm inspired by passing on knowledge within a classroom environment, where learners first

* Corresponding author. Tel.: +386 22207429.
E-mail addresses: matej.crepinsek@uni-mb.si (M. Črepinšek), shliu@CSUFresno.edu (S.-H. Liu), lmernik@caltech.edu (L. Mernik).

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2012.05.009

acquire knowledge from a teacher (i.e., Teacher Phase) and then from classmates (i.e., Learner Phase). In many aspects, TLBO resembles Evolutionary Algorithms (EAs) [2,29]. For example, (1) an initial population is randomly generated; (2) moving/learning towards teacher and classmates can be regarded as a special mutation operator; and (3) selection is deterministic (i.e., two solutions are compared and the better one always survives), which is also used often in many other EAs such as ESs [29]. Despite the fact that there are many similarities between conventional EAs and TLBO, remarkable results have been reported about TLBO outperforming ES, PSO, ABC, DE, and GEM on a number of constrained benchmark functions and constrained mechanical design problems [36], as well as on continuous non-linear numerical optimization problems [37]. In particular, TLBO discovers better or equal solutions much faster than ES, PSO, ABC, DE, and GEM. Hence, TLBO seems to have extraordinary convergence rates for solutions. The reported results in [36,37] immediately attracted our attention, due to the existence of the No-Free Lunch (NFL) Theorem as cited by Wolpert and Macready [44]. After obtaining two prototypical implementations from TLBO inventors, we were able to delve more deeply into the algorithm's inner workings.

A primary aim of this paper was to investigate whether TLBO is indeed a rising star of metaheuristics and whether the theories and/or practices that support its dominance are valid. Our investigation was done both quantitatively and qualitatively. For the quantitative arena, the experiments were replicated on the same five constrained benchmark test functions from Ref. [36] and the three experiments on continuous non-linear numerical optimization problems from Ref. [37]. Unfortunately, our experimental results do not confirm the previously reported results published in [36,37]. For the qualitative arena, a code-review was performed on the prototypical implementations of TLBO. Whilst experimenting with TLBO, it was possible to reveal several mistakes or misconceptions that appeared in the original papers [36,37]. Most notably:

- Algorithm comparisons between TLBO and others (i.e., ES, PSO, ABC, DE, and GEM) were unfair, because the experiments were not repeated within the same setting (e.g., the number of fitness evaluations were neither correctly counted, nor were all the benchmark test-functions used. Moreover, the stopping conditions for the TLBO experiments were different from the other compared algorithms). Hence, claims about TLBO's superiority cannot be supported.
- The algorithm descriptions of TLBO in Rao et al. [36,37] (i.e., Fig. 3 of Rao et al. [36,37]) do not match the prototypical implementations, where duplicate elimination was actually integrated within TLBO but not reported.
- TLBO cannot be regarded as a parameter-less algorithm in the sense of Bäck et al. [3], Eiben et al. [11] and Harik and Lobo [16].

TLBO has also outperformed other algorithms on several constrained mechanical design optimization problems (e.g. when designing a pressure vessel, a tension/compression spring, and a welded beam) in [36]. These experiments were not repeated in this paper, because our expertise is outside mechanical design optimization problems. Our replicated experimental results on several constrained and unconstrained benchmark test functions [36,37] showed that TLBO's performance was amongst the average. Conclusively, the investigated results both quantitatively and qualitatively did not support the claims: "*Teaching–Learning-Based Optimization (TLBO) obtain global solutions for continuous non-linear functions with less computational effort and high consistency* [36]" and "*The results show better performance of TLBO method over other natured-inspired optimization methods for the considered benchmark functions. Also, the TLBO method shows better performance with less computational efforts for the large scale problems, i.e. problems with high dimensions* [37]."

This paper is organized as follows. Section 2 provides a short overview of the TLBO algorithm ensuring that this paper is self-contained. The experimental results on constrained and unconstrained benchmark test functions are presented in Section 3, where the presented results confront the previously reported TLBO results [36,37]. In Section 4, short guidelines are provided on how to compare different algorithms. Section 5 concludes the paper.

2. TLBO algorithm

This section first reviews the TLBO algorithm published in [36,37]. Then the source code of two prototypical implementations is reviewed. Algorithm 1, as presented in Section 2.2, is the outcome of the code review. A comparison between the versions of TLBO concludes this section.

2.1. TLBO algorithm from the papers

The main idea behind TLBO is the simulation of a classical school learning process that consists of two stages. During the first stage, called *Teacher Phase*, a teacher imparts knowledge directly to his/her students. The better the teacher, the more knowledge the students obtain. However, the possibility of a teacher's teaching being successful during the Teacher Phase, in practice, is distributed under Gaussian law. There are only very rare students who can understand all the materials presented by the teacher (i.e., the right end of the Gaussian distribution). Most students will partially accept new learning materials (i.e., the mid part of the Gaussian distribution) and, in some cases, the teacher will have almost no direct effect on students' knowledge (i.e., the left end of the Gaussian distribution). However, the possibility for most students to obtain new knowledge is not completely lost. During the second stage, called *Learner Phase*, a student may learn with the help of fellow students. Overall, how much knowledge is transferred to a student does not only depend on his/her teacher but also on interactions amongst students through peer learning.

More specifically, the TLBO algorithm is population-based (P) with a pre-defined size for that population (*pop_size*). An initial population is randomly generated, which resembles many EAs. An individual (X_i) within the population represents a single possible solution to a particular optimization problem. X_i is a real-valued vector with D elements, where D is the dimension of the problem and is used to represent the number of subjects that an individual, either student or teacher, enrolls for/teaches within the TLBO context. The algorithm then tries to improve certain individuals by changing these individuals during the *Teacher* and *Learner Phases*, where an individual is only replaced if his/her new solution is better than his/her previous one. The algorithm will repeat until it reaches the maximum number of generations.

During the Teacher Phase, the teaching role is assigned to the best individual ($X_{teacher}$). The algorithm attempts to improve other individuals (X_i) by moving their positions towards the position of the $X_{teacher}$ by taking into account the current mean value of the individuals (X_{mean}). This is constructed using the mean-values for each parameter within the problem space (dimension), and represents the qualities of all students from the current generation. Eq. (1) simulates how student improvement may be influenced by the difference between the teacher's knowledge and the qualities of all students. For stochastic purposes, two randomly-generated parameters are applied within the equation: r ranges between 0 and 1; and T_F is a teaching factor, which can be either 1 or 2, thus emphasizing the importance of student quality.

$$X_{new} = X_i + r \cdot (X_{teacher} - (T_F \cdot X_{mean})) \tag{1}$$

During the Learner Phase, student (X_i) tries to improve his/her knowledge by peer learning from an arbitrary student X_{ii}, where i is unequal to ii. In the case that X_{ii} is better than X_i, X_i is moved towards X_{ii} (Eq. (2)). Otherwise, it is moved away from X_{ii} (Eq. (3)). If student X_{new} performs better by following Eq. (2) or (3), he/she will be accepted into the population. The algorithm will continue its iterations until reaching the maximum number of generations.

$$X_{new} = X_i + r \cdot (X_{ii} - X_i) \tag{2}$$

$$X_{new} = X_i + r \cdot (X_i - X_{ii}) \tag{3}$$

Additionally infeasible individuals must be appropriately handled, to determine whether one individual is better than another, when applied to constrained optimization problems. For comparing two individuals, the TLBO algorithm, according to [36], utilizes the Deb's constrained handling method [9]:

- If both individuals are feasible, the fitter individual (with a better value of fitness function) is preferred.
- If one individual is feasible and the other one is infeasible, the feasible individual is preferred.
- If both individuals are infeasible, the individual having the smaller number of violations (this value is obtained by summing all the normalized constraint violations) is preferred.

2.2. TLBO algorithm via the code review

After analyzing the TLBO code (two TLBO implementations were analyzed) Algorithm 1 was obtained as described as follows (i.e., the qualitative manner).

Algorithm 1. TLBO pseudocode via code review

```
Algorithm TLBO
begin
  g ← 0;
  initialize_population(P, pop_size)
  evaluate(P)
  repeat
    Elite ← select_best(P, elite)
    for i = 1 → pop_size do
      {Teacher Phase}
      T_F = round(1 + r)
      X_mean ← calculate_mean_vector(P)
      X_teacher ← best_solution(P)
      X_new = X_i + r · (X_teacher − (T_F · X_mean))
      evaluate(X_new)
      if X_new better than X_i then
        X_i ← X_new
      end if {End of Teacher Phase}
      {Learner Phase}
      ii ← random(pop_size){ii ≠ i}
```

(*continued on next page*)

```
      if X_i better than X_ii then
        X_i,new = X_i + r · (X_i − X_ii)
      else
        X_i,new = X_i + r · (X_ii − X_i)
      end if
      evaluate(X_i,new)
      if X_i,new better than X_i then
        X_i ← X_i,new
      end if {End of Learner Phase}
    end for
    P← replace_worst_with_elite(P, Elite) {in scenario s4 special elitism algorithm is used}
    P← remove_duplicate_individuals(P)
    g ← g + 1
  until (g ≠ num_gen) {termination_condition}
  print_best_result(P)
end
```

In general, most of the steps in Algorithm 1 are identical to the TLBO description in Section 2 of Rao et al. [36,37]. However, there are some important differences. The first difference appears in both TLBO implementations and applies to both the unconstrained and constrained problems. Namely, TLBO modifies duplicated individuals by mutation on randomly selected dimension(s) at the end of each generation, whilst the second difference only appears for constrained problems where elitism has been used. There are two versions for replacing the worst individuals. Besides the classical elitism approach used in the first TLBO implementation, the following elitist method was also experimented on by the TLBO inventors in the second TLBO implementation: The first-half of the population are compared with the second-half (X_i and $X_{\frac{popsize}{2}-1+i}$). If an individual from the second-half of the population is better than an individual from the first-half then the individual from the second-half replaces the individual from the first-half. In such a manner, duplicates are introduced that will immediately undergo mutation on randomly selected dimension(s). Note, that the first version of elitism used in TLBO requires an additional control parameter (elite_size), whilst there is no need for an additional control parameter in the second version. These steps are neither shown in Fig. 3 of Rao et al. [36,37] nor described in [36,37]. Additionally, there are several minor differences between two of the TLBO implementations. The most important difference is that the first implementation iterates all the population during the Teacher Phase and then iterates all the population during the Learner Phase (see steps 2–4 in Section 3 [37]). However, the second implementation is the same as described in Algorithm 1. Although this change may be more efficient, slightly different results may be generated. Namely, an updated X_i after the Teacher and Learner Phases in Algorithm 1 can immediately influence the *next individuals* to be evaluated and, hence, affect the X_{mean} within the same generation. The TLBO implementation in this paper and the reported results are based on the second implementation with duplicate elimination and two elitisms as options. More detailed comparisons between different TLBO versions are analyzed in the following subsection. Our implementation of both TLBO versions is publicly available at http://lpm.uni-mb.si/evolution/TLBOJavaVersion1_1.zip.

2.3. Comparison

This section details the comparisons between Algorithm 1, which is based on the code review of two prototypical TLBO implementations, and the TLBO algorithm presented in [36,37]. Note that the obtained prototypical TLBO implementations might be slightly different from the TLBO actually used by the TLBO inventors. However, the actual TLBO should not deviate too much from the description in [36,37].

Firstly, Algorithm 1 actually utilizes duplicate elimination and elitism (two different versions have been used for solving constrained problems only) to respectively enhance the exploitation and exploration powers [7,26]. Yet, these two steps were not part of TLBO within the original papers [36,37]. Hence, experimental results are now presented in Section 3 with and without these two additional steps. Moreover, there is another mismatch in both versions of the TLBO algorithm. In Fig. 3 of Rao et al. [36,37], the Learner Phase (also called the Student Phase in Fig. 3 of Rao et al. [36,37]) is performed only if the Teacher Phase produces fitter individuals. However, in TLBO implementations, the Learner Phase is independent of the Teacher Phase. Since the description of the TLBO in Section 2 of Rao et al. [36,37] is closer to the TLBO implementations and Algorithm 1, the authors are convinced that Fig. 3; of Rao et al. [36,37] are slightly inaccurate.

Furthermore, it is clear from Algorithm 1 that the number of fitness function evaluations (*num_eval*) is not the same as the population size (*pop_size*) when multiplying the number of generations (*num_gen*). The formula ($num\_eval = num\_gen \cdot pop\_size$) is used in [36] for comparisons with other EAs. Each member of the population (i.e., solution) is moved towards the teacher during the Teacher Phase by consuming one fitness evaluation, and each member of population is moved towards/from a randomly chosen individual within the Learner Phase, by consuming one fitness evaluation. Moreover, during the duplicate elimination step, new individuals are generated by consuming more fitness evaluations. Hence, a more

accurate estimate for the number of fitness function evaluations in TLBO is $num\_eval \geqslant num\_gen \cdot 2 \cdot pop\_size$. By comparing different EAs, it is extremely risky just to take into consideration the population size and the number of generations, in order to compute the number of fitness evaluations. Firstly, many EAs use and adapt variable population sizes (e.g., [12]), where the number of individuals differs at different stages of an algorithm. Secondly, during an evolution process, individuals might be improved by local search or other exploration/exploitation mechanisms requiring more fitness evaluations. In such cases, it is unfair to neglect those extra fitness evaluations and compare the results by just basing them on the number of generations and population size. More specifically, the Learning Phase in TLBO may be regarded as an additional search-enhancing exploitation/exploration power of the TLBO algorithm, whilst duplicate elimination with modification of the duplicates by the mutation of randomly-selected dimension(s) can again be seen as enhancing the exploratory/exploitation powers of the TLBO algorithm. Both cases require extra fitness evaluations, which must be taken into account when comparing them with other algorithms. Hence, the reported numbers of fitness function evaluations in Tables 1–5 in [36] are inaccurate. Based on these inaccurate numbers for fitness-function evaluations, it was concluded in [36] that TLBO's performance with respect to convergence rates is much better than other EAs. Such a conclusion is simply erroneous. The authors re-ran TLBO regarding the five constrained benchmark test functions [36], and the correct number of fitness function evaluations are reported in Section 3, where it is shown that the TLBO performance is not as good as that reported in [36].

Another misconception might come from the claim that '*TLBO does not require any algorithm parameters to be tuned* [36] (page 306)" or "*The strength of TLBO method is that it does not require any parameters setting for the working of the algorithm* [37] (page 9)." In this respect the inventors of TLBO in [36,37] differentiate between common and algorithm-specific control parameters, whilst such a classification is unusual in EAs [3,11,16] and should be more clearly explained. EAs are used to solve many different optimization problems covering a vast number of different fitness landscapes that require unique exploration and exploitation of the search space [7]. Hence, many contemporary EAs (e.g., [26]) employ adaptive or self-adaptive approaches for finding appropriate control parameters [11] rather than eliminating some control parameters by setting fixed values. The proposed TLBO algorithm is no exception in this regard because population size, the maximum number of generations, and the size of the elite (in case the classic elitism is used) must be set experimentally, regardless of whether the parameters are classified as common or algorithm-specific. The burden is still on the end-users to find appropriate settings. Hence, the claim that TLBO does not require any parameter settings for working the algorithm should be considered as incorrect. Even in the case of simple constrained benchmark test functions [36], different values for the number of generations (*num_gen*) (*num_gen* = 500 for benchmark function f_1; *num_gen* = 2000 for benchmark functions f_2, f_3 and f_4; and *num_gen* = 1000 for benchmark function f_5) have been used [36], whilst the population size was fixed (*pop_size* = 50). The presented experiments with TLBO confirm that the control parameter setting is no easier in TLBO than in any other algorithms, nor can TLBO be regarded as a parameter-less algorithm in the sense of Bäck et al. [3], Eiben et al. [11] and Harik and Lobo [16].

3. Experimental results

This section shows the experimental results of five constrained benchmark test functions [36] and three continuous non-linear numerical optimization problems [37]. In other words, we have tried to replicate most of the experiments presented in [36,37]. The results support our findings on the inaccurate claims in [36,37].

3.1. Constrained problems

For experimental purposes, the same constrained benchmark test functions were used as in [36]. Descriptions of these benchmark test functions can be found in [24,36,38]. Since some optimization algorithms terminate when a particular number of fitness evaluations are consumed, the implementation was adjusted in such a way that it could also accept a maximum number of fitness evaluations (*num_eval*) as a termination condition. A termination condition can be enforced during the Teacher or Learner Phases, as well as during the duplicate elimination-step. A better comparison amongst different optimization algorithms can be achieved by using such a termination condition. Of course, it is also possible to compare these experimental results with the original paper [36], the old termination conditions, and the maximum number of generations (*num_gen*).

An attempt was made during these experiments to reproduce the experiments from the original paper [36]. Several different scenarios were constructed because there was a mismatch between TLBO [36] and Algorithm 1, regarding the use of elitism and duplicate elimination. In Scenario 1 ($TLBO_{s1}$), the number of generations (*num_gen*) was set as the algorithm termination condition with the same value as reported in [36]. Elitism and duplicate elimination were omitted (i.e., *elite* = 0). Scenario 2 ($TLBO_{s2}$) was similar to $TLBO_{s1}$ except that the duplicate elimination step was not skipped. Its elitism was omitted by setting *elite* = 0. Scenario 3 ($TLBO_{s3}$) was similar to $TLBO_{s2}$ except that elitism was used where elite size was set to 4, 8, or 12. Scenario 4 ($TLBO_{s4}$) was similar to $TLBO_{s3}$ except that the classic elitism approach was replaced by the elitism method described in Section 2.2.

Scenarios 5–8 ($TLBO_{s5}$, $TLBO_{s6}$, $TLBO_{s7}$, and $TLBO_{s8}$) were respectively similar to $TLBO_{s1}$, $TLBO_{s2}$, $TLBO_{s3}$, and $TLBO_{s4}$ except that their termination conditions were set to the number of evaluations (*num_eval*).

Table 1
Comparisons between the results obtained by different optimization methods on f_1.

Method	Best	Mean ± St. Dev.	Worst	*num_eval* ± St. Dev.	*sr* (%)
M-ES	−15	−15	−15	240,000	100
PESO	−15	−14.710	−13	350,000	100
CDE	−15	−14.999996	−14.999993	100,100	100
CoDE	−15	−15	−15	248,000	100
ABC	−15	−15	−15	240,000	100
TLBO	−15	−15	−15	25,000	100
$TLBO_{s1}$	−15.000	−13.845 ± 2.4	−6.000	50,050.0 ± 0	100
$TLBO_{s2}$	−15.000	−13.864 ± 1.7	−9.000	50,063.9 ± 35.5	100
$TLBO_{s3}$	−15.000	−13.199 ± 1.5	−12.000	53,257.0 ± 283.5	100
$TLBO_{s4}$	−15.000	−13.743 ± 1.9	−9.000	50,269.9 ± 51.4	100

Table 2
Comparisons between the results obtained by different optimization methods on f_2.

Method	Best	Mean ± St. Dev.	Worst	*num_eval* ± St. Dev.	*sr* (%)
M-ES	1	1	1	240,000	100
PESO	0.993930	0.764813	0.464	350,000	100
CDE	0.995413	0.788635	0.639920	100,100	100
ABC	1	1	1	240,000	100
TLBO	1	1	1	100,000	100
$TLBO_{s1}$	1.000	1.000 ± 0.0	1.000	200,050.0 ± 0	100
$TLBO_{s2}$	1.000	1.000 ± 0.0	1.000	200,051.5 ± 2.6	100
$TLBO_{s3}$	1.000	1.000 ± 0.0	1.000	215,610.7 ± 218.3	100
$TLBO_{s4}$	1.000	1.000 ± 0.0	1.000	200,303.3 ± 30.0	100

Table 3
Comparisons between the results obtained by different optimization methods on f_3.

Method	Best	Mean ± St. Dev.	Worst	*num_eval* ± St. Dev.	*sr* (%)
M-ES	680.632	680.643	680.719	240,000	100
PESO	680.630	680.630	680.631	350,000	100
CDE	680.63006	680.63006	680.6301	100,100	100
CoDE	680.771	681.503	685.144	248,000	100
ABC	680.634	680.640	680.653	240,000	100
TLBO	680.630	680.633	680.638	100,000	100
$TLBO_{s1}$	680.631	680.632 ± 0.0	680.635	200,050.0 ± 0	100
$TLBO_{s2}$	680.631	680.632 ± 0.0	680.634	200,050.0 ± 0	100
$TLBO_{s3}$	680.631	680.646 ± 0.0	680.729	215,683.0 ± 111.6	100
$TLBO_{s4}$	680.631	680.632 ± 0.0	680.636	200,250.5 ± 25.1	100

Table 4
Comparisons between the results obtained by different optimization methods on f_4.

Method	Best	Mean ± St. Dev.	Worst	*num_eval* ± St. Dev.	*sr* (%)
M-ES	7051.903	7253.047	7638.366	240,000	100
PESO	7049.38	7205.5	7894.812	350,000	100
CDE	7049.2481	7049.2483	7049.2485	100,100	100
ABC	7053.904	7224.407	7604.132	240,000	100
TLBO	7049.2481	7083.6732	7224.4968	100,000	100
$TLBO_{s1}$	7065.072	7257.002 ± 102.6	7469.046	200,050.0 ± 0	100
$TLBO_{s2}$	7059.632	7244.565 ± 95.8	7469.042	200,050.0 ± 0	100
$TLBO_{s3}$	7105.009	7386.036 ± 209.2	8336.698	212,683.6 ± 1020.8	100
$TLBO_{s4}$	7059.675	7252.795 ± 100.0	7466.724	200,272.3 ± 92.3	100

Tables 1–5 in this paper were the extensions of Tables 1–5 presented in [36]. The first parts of Tables 1–5 are taken from Ref. [36], where TLBO in [36] was compared to Multi-membered Evolutionary Strategy (M-ES) [28], Particle Evolutionary Swarm Optimization (PESO) [45], Cultural Differential Evolution (CDE) [24], Co-evolutionary Differential Evolution (CoDE) [18], and Artificial Bee Colony (ABC) [21]. The second parts of Tables 1–5 present the results for Scenarios 1–4. The experiments during the second part were repeated 30 times, as reported in [24]. However, this information could not be found in [36]. Also, in Scenario 3 ($TLBO_{s3}$) the elite size was set to 8, based on its good results after performing extensive tests that

Table 5
Comparisons between the results obtained by different optimization methods on f_5.

Method	Best	Mean ± St. Dev.	Worst	*num_eval* ± St. Dev.	*sr* (%)
M-ES	1	1	1	240,000	100
PESO	1	0.998875	0.994	350,000	100
CDE	1	1	1	100,100	100
CoDE	1	1	1	248,000	100
ABC	1	1	1	240,000	100
TLBO	1	1	1	50,000	100
$TLBO_{s1}$	1.000	1.000 ± 0	1.000	100,050.0 ± 0	100
$TLBO_{s2}$	1.000	1.000 ± 0	1.000	100,050.0 ± 0	100
$TLBO_{s3}$	1.000	1.000 ± 0	1.000	107,936.2 ± 23.3	100
$TLBO_{s4}$	1.000	1.000 ± 0	1.000	100,138.7 ± 12.7	100

varying the elite size. Note that the results for other optimization algorithms (M-ES, PESO, CDE, CoDE, ABC) were not re-computed in [36] but were directly taken from Refs. [18,21,24,28,45] using the same data. This should not have posed any problem if the experiments for TLBO had been performed under the same conditions. Yet, this was not the case in [36]. The authors highlight three critical pitfalls, described as follows:

1. By comparing Tables 1–5 in [36], it was noticed that the number of fitness evaluations (*num_eval*) for M-ES, PESO, CDE, CoDE, and ABC were the same for *all* benchmark test functions f_1 to f_5, whilst for TLBO, *num_eval* was set differently: f_1: *num_eval* = 25,000; f_2, f_3 and f_4: *num_eval* = 100,000; and f_5:*num_eval* = 50,000. Namely, in the results for M-ES, PESO, CDE, CoDE, and ABC, the authors of these papers (i.e., [18,21,24,28,45]) tried to find an appropriate number of fitness evaluations that well-suited *all* 13 constrained functions in the benchmark [38]. For example, the authors of CDE wrote in [24]: "*This parameter setting was chosen to be a good compromise for all the test functions; optimal settings for each problem may exist, but they are not reported here*." However, the inventors of TLBO tried to fine-tune the number of function evaluations for *each* function *separately* (as can be seen in the "TLBO" rows of Tables 1–5). Apparently, TLBO experimented with the benchmark functions under different experimental assumptions/conditions, compared to other experimental settings that the paper [36] cited;
2. There were only five functions selected in [36] out of 13 benchmark functions from Ref. [38];
3. As mentioned previously, the following formula for computing the number of fitness evaluations were used (*num_eval* = *num_gen* · *pop_size*), without counting the extra fitness evaluations during the Learner Phase and the duplicate elimination step. Hence, the reported number of fitness function evaluations in Tables 1–5 (Row *TLBO*) are incorrect (f_1: *pop_size* = 50, *num_gen* = 500, and the number of fitness evaluations 25,000; f_2, f_3 and f_4: *pop_size* = 50, *num_gen* = 2000, and the number of fitness evaluations 100,000; and f_5: *pop_size* = 50, *num_gen* = 1000 and number of fitness evaluations 50,000). Overall, the comparisons done in [36] can hardly be regarded as objective and valid.

The conclusion that TLBO uses less computational effort than other EAs should be reconsidered. The correct number of fitness evaluations were provided in the second part of Tables 1–5, where different versions of TLBO were tested ($TLBO_{s1}$ – TLBO without elitism and duplicate elimination, $TLBO_{s2}$ – TLBO with duplicate elimination, $TLBO_{s3}$ – TLBO with classic elitism and duplicate elimination, and $TLBO_{s4}$ – TLBO with special elitism and duplicate elimination). Due to the importance of success rates (*sr*) for the presentations of results, and noting that solutions may not always be found in all runs, the authors also added a column to the tables representing success rate (*sr*).

From Table 1 (search for minimum of f_1) it was noticed that the number of consumed fitness evaluations in the second part was more than twice that reported in [36] (can also be seen in Row TLBO). Although this value was still almost half the number of fitness evaluations of CDE (the method that needed the fewest number of fitness evaluations amongst the compared optimization algorithms [24]), the results were not of the same quality. Also, the mean-value over 30 runs was worse (−13.743), using $TLBO_{s4}$, than the one reported in [36] (−15). The TLBO versions with elitism ($TLBO_{s3}$ and $TLBO_{s4}$) were slightly worse than the TLBO versions without elitism ($TLBO_{s1}$ and $TLBO_{s2}$).

Table 2 shows the new results for function f_2 (searching for the maximum of f_2). All TLBO versions ($TLBO_{s1}$–$TLBO_{s4}$) generated similar results as reported in [36]. However, the number of fitness evaluations was more than twice that reported in [36]. Hence, its performance regarding the number of fitness evaluations was much worse than CDE.

Table 3 shows the new results for function f_3 (searching for the minimum of f_3). Again, all TLBO versions ($TLBO_{s1}$–$TLBO_{s4}$) generated relatively comparable results, but the number of fitness evaluations was twice as many as needed by CDE.

Table 4 shows the new results for function f_4 (searching for the minimum of f_4). At the specified number of generations, *num_gen* = 2000, TLBO consumed about 200,000 fitness evaluations. This was twice as many as CDE, but the results were not close to those reported in [36]. The TLBO versions $TLBO_{s2}$ and $TLBO_{s4}$ produced slightly better results than the other two TLBO versions.

Table 5 shows the results for function f_5 (searching for the maximum of f_5). The results were the same as those reported in [36], except that the number of fitness evaluations was twice as many as the one originally reported and were now relatively comparable to CDE.

As it was not possible to repeat the results from Ref. [36] on benchmark functions f_1 and f_4 using the same control parameters, TLBO was re-run by raising the number of fitness evaluations to the extent that the reported results in [36] were achieved. The results are reported in Table 6, where it can be noticed that for f_1–f_3 240,000 fitness evaluations were needed, for f_4 350,000 fitness evaluations, and for f_5 50,000 fitness evaluations. If it is necessary to achieve the same experimental settings as in M-ES, PESO, CDE, CoDE, and ABC, the final number of fitness evaluations for *all* five functions in the benchmark [36] should be set at 350,000. This number is the same as in PESO, which needed the most fitness evaluations from amongst all the compared optimization algorithms. This was unsurprising, because TLBO resembles many features from PSO (e.g., moving towards a current global solution). On the other hand, the new number of fitness evaluations for TLBO was only of a lower bound because TLBO was not run on *all* 13 benchmark test functions [38]. The aim was only to show that the reported results in [36] were incorrect, and the conclusions invalid.

Table 7 shows that TLBO is also sensitive to population size, which should be treated as another control parameter of TLBO. This value was fixed at 50 in [36]. The results in Table 7 were obtained by running the TLBO with elitism (*elite* = 8) and duplicate elimination ($TLBO_{s7}$) after 30 runs. The number of fitness evaluations was the same, but the population sizes varied. It can be noticed from Table 7 that *pop_size* = 50 was not the best setting for functions f_1 and f_4. Hence, the experimental results cannot support the claim ([37], page 9) that "*The strength of TLBO method is that it does not require any parameters setting for the working of the algorithm.*"

3.2. Unconstrained problems

TLBO is also introduced in [37], where it was tested on several continuous non-linear numerical optimization problems. The description of the benchmark test functions can be found in [37,42]. This paper replicated the first three experiments from Ref. [37]. The rationale for such a decision was that, after three experiments, enough problems were found so that TLBO's claims cannot be supported. Hence, there was no need to repeat all the experiments in [37]. Scenario 2 was used (*elite* = 0 and duplicates removal) for all unconstrained problems, because such a setting was also used by TLBO inventors for unconstrained problems.

In Experiment 1 [37], TLBO was compared to the experiment in [1], where the newly developed Grenade Explosion Method (GEM) [1] had been compared to GA, the Ant Colony System (ANTS), and the Bee Algorithm (BA). Since GEM outperforms GA, ANTS and BA, only the results on GEM and TLBO are presented in continuation. The stopping criterion was when the difference between the maximum fitness obtained and the global optimum value was less than 0.1% of the optimum value or less than 0.001, whichever was smaller. In the case where the optimum value was 0, the solution was accepted if it

Table 6
Results obtained with the increased number of fitness evaluations averaged over 30 runs.

Problem	Method	Elite	Best	Mean ± St. Dev.	Worst	*eval*	*sr* (%)
f_1	$TLBO_{s5}$	0	−15.000	−13.597 ± 2.3	−6.000	240,000	100
f_1	$TLBO_{s6}$	0	−15.000	−13.321 ± 1.9	−9.000	240,000	100
f_1	$TLBO_{s7}$	4	−15.000	−14.733 ± 0.8	−12.000	240,000	100
f_1	$TLBO_{s7}$	8	−15.000	−15.000 ± 0	−15.000	240,000	100
f_1	$TLBO_{s7}$	12	−15.000	−15.000 ± 0	−15.000	240,000	100
f_1	$TLBO_{s8}$	0	−15.000	−14.233 ± 1.7	−9.000	240,000	100
f_2	$TLBO_{s5}$	0	1.000	1.000 ± 0.0	1.000	240,000	100
f_2	$TLBO_{s6}$	0	1.000	1.000 ± 0.0	1.000	240,000	100
f_2	$TLBO_{s7}$	4	1.000	1.000 ± 0.0	1.000	240,000	100
f_2	$TLBO_{s7}$	8	1.000	1.000 ± 0.0	1.000	240,000	100
f_2	$TLBO_{s7}$	12	1.000	1.000 ± 0.0	1.000	240,000	100
f_2	$TLBO_{s8}$	0	1.000	1.000 ± 0.0	1.000	240,000	100
f_3	$TLBO_{s5}$	0	680.63	680.631 ± 0.0	680.634	240,000	100
f_3	$TLBO_{s6}$	0	680.631	680.632 ± 0.0	680.633	240,000	100
f_3	$TLBO_{s7}$	4	680.63	680.636 ± 0.0	680.649	240,000	100
f_3	$TLBO_{s7}$	8	680.633	680.648 ± 0.0	680.704	240,000	100
f_3	$TLBO_{s7}$	12	680.634	680.664 ± 0.0	680.836	240,000	100
f_3	$TLBO_{s8}$	0	680.63	680.631 ± 0.0	680.634	240,000	100
f_4	$TLBO_{s5}$	0	7066.596	7286.164 ± 122.7	7469.046	350,000	100
f_4	$TLBO_{s6}$	0	7050.497	7214.073 ± 114.7	7440.107	350,000	100
f_4	$TLBO_{s7}$	4	7099.55	7390.221 ± 294.4	8390.143	350,000	100
f_4	$TLBO_{s7}$	8	7059.751	7413.923 ± 329.4	8564.398	350,000	100
f_4	$TLBO_{s7}$	12	7135.984	7348.389 ± 116.4	7541.534	350,000	100
f_4	$TLBO_{s8}$	0	7064.13	7218.174 ± 106.9	7469.046	350,000	100
f_5	$TLBO_{s5}$	0	1.000	1.000 ± 0	1.000	50,000	100
f_5	$TLBO_{s6}$	0	1.000	1.000 ± 0	1.000	50,000	100
f_5	$TLBO_{s7}$	4	1.000	1.000 ± 0	1.000	50,000	100
f_5	$TLBO_{s7}$	8	1.000	1.000 ± 0	1.000	50,000	100
f_5	$TLBO_{s7}$	12	1.000	1.000 ± 0	1.000	50,000	100
f_5	$TLBO_{s8}$	0	1.000	1.000 ± 0	1.000	50,000	100

Table 7
Influence of *pop_size* on TLBO.

Problem	Method	*pop_size*	Best	Mean ± St. Dev.	Worst	*eval*	*sr* (%)
f_1	$TLBO_{s7}$	25	−15.000	−15.000 ± 0	−15.000	100,100	100
f_1	$TLBO_{s7}$	50	−15.000	−14.267 ± 1.2	−12.000	100,100	100
f_1	$TLBO_{s7}$	75	−15.000	−13.332 ± 1.6	−10.000	100,100	100
f_1	$TLBO_{s7}$	100	−15.000	−12.931 ± 1.7	−9.000	100,100	100
f_2	$TLBO_{s7}$	25	1.000	0.944 ± 0.2	0.149	100,100	100
f_2	$TLBO_{s7}$	50	1.000	0.940 ± 0.2	0.251	100,100	100
f_2	$TLBO_{s7}$	75	1.000	0.904 ± 0.2	0.232	100,100	100
f_2	$TLBO_{s7}$	100	1.000	0.877 ± 0.2	0.205	100,100	100
f_3	$TLBO_{s7}$	25	680.637	680.918 ± 0.2	681.821	100,100	100
f_3	$TLBO_{s7}$	50	680.631	680.653 ± 0.0	680.835	100,100	100
f_3	$TLBO_{s7}$	75	680.632	680.644 ± 0.0	680.667	100,100	100
f_3	$TLBO_{s7}$	100	680.633	680.644 ± 0.0	680.671	100,100	100
f_4	$TLBO_{s7}$	25	7073.214	8611.175 ± 1,259.0	11,827.657	100,100	100
f_4	$TLBO_{s7}$	50	7079.959	7371.161 ± 239.5	8368.779	100,100	100
f_4	$TLBO_{s7}$	75	7066.196	7268.55 ± 92.2	7469.475	100,100	100
f_4	$TLBO_{s7}$	100	7070.271	7302.254 ± 104.8	7471.246	100,100	100
f_5	$TLBO_{s7}$	25	1.000	1.000 ± 0	1.000	50,000	100
f_5	$TLBO_{s7}$	50	1.000	1.000 ± 0	1.000	50,000	100
f_5	$TLBO_{s7}$	75	1.000	1.000 ± 0	1.000	50,000	100
f_5	$TLBO_{s7}$	100	1.000	1.000 ± 0	1.000	50,000	100

differed from the optimum value by less than 0.001. Such a stopping criterion enables a comparison of algorithms regarding the mean number of fitness evaluations. The algorithms were tested over 100 independent runs. A small inconsistency was revealed when comparing the experiments of Ahrari and Atai [1] and Rao et al. [37]. The experiment in [37] was an inexact replication of the experiment in [1], because one optimization problem had been omitted (Branin function). The experiment from Ref. [1] was repeated and the results are presented in Table 8. The results for GEM and the original TLBO (denoted as $TLBO_{orig}$) were taken directly from Ref. [37], whilst $TLBO_{imp}$ is our implementation of TLBO (i.e., $TLBO_{s2}$). The results for $TLBO_{orig}$ and $TLBO_{imp}$ were slightly different. The results (Table 8) show that GEM and TLBO were quite comparable. GEM performed slightly better than $TLBO_{imp}$ on De Jong, Martin and Gaddy, Rosenbrock ($D = 2$) (a), and Hyper Sphere, and $TLBO_{imp}$ performed slightly better on Goldstein and Price, Rosenbrock ($D = 2$) (b), Rosenbrock ($D = 3$), and Branin. A much better performance could only be noticed by one particular algorithm on the Rosenbrock function ($D = 3$), where TLBO was clearly a winner. On the other hand, the success rate (*sr*) on the Goldstein and Price function was not 100%. By replicating this experiment it could not be concluded that TLBO outperformed GEM as claimed in [37]: *"From Table 3, it can be seen that for all the considered benchmark functions, TLBO requires less number of mean function evaluations with very high consistency of 100% success."*

In Experiment 2 [37] TLBO was compared to PSO by the hybridization of PSO using the Nelder-Mead simplex search method (NM-PSO) [13]. After comparing the executions of experiments in [13,37], serious problems were found during the comparison [37]. The biggest problem was that the algorithms' stopping criterion was not the same during both experiments. In the experiment of Fan and Zahara [13] the algorithm stopped when the number of iterations reached 1000*D (D is the dimension of the problem), or when the simplex relative size was smaller than 0.0001. The latter condition was proposed by Dennis and Woods [10] with the main characteristic that stopping was not based on function value information, which is usually the case during EAs comparisons (e.g., see the aforementioned Experiment 1). Instead, the stopping condition was based on vertex information (point in D-dimensional space). On the other hand, the stopping criterion for the experiment in [37] was based on function value information – the algorithm stopped when the difference between the best obtained solution and the global optimum was less than 0.001. It is clear that algorithm comparison under such a different stopping criterion is impossible and any derived conclusion is invalid. A comparison might be possible if the best $D + 1$ individuals (points) in TLBO acted as a D-dimensional simplex and Dennis and Woods' stopping criterion was applied to the $D + 1$ points in TLBO. But, even in this case, this would be a rough approximation since these points undergo local search steps in the Nelder-Mead method, and smaller simplex relative size might be more easily achieved under the Nelder-Mead method.

Table 8
Experiment 1 (Table 3 in [37]).

Function	*GEM*		$TLBO_{orig}$		$TLBO_{imp}$	
	sr	Mean FE	*sr*	Mean FE	*sr*	Mean ± St. Dev. FE
De Jong	100	746	100	676	100	832.2 ± 400.1
Goldstein and Price	100	701	100	649	90	629.0 ± 119.7
Martin in Gaddy	100	258	100	243	100	317.0 ± 97.0
Rosenbrock ($D = 2$) (a)	100	572	100	541	100	694.3 ± 374.0
Rosenbrock ($D = 2$) (b)	100	2289	100	1082	100	1911.0 ± 884.2
Rosenbrock ($D = 3$)	100	82,188	100	2563	100	9992.6 ± 3791.3
Hyper Sphare ($D = 6$)	100	423	100	308	100	750.8 ± 60.2
Branin	100	689	N/A	N/A	100	577.5 ± 220.1

Nevertheless such problems need to be mentioned, and claims of validity must be discussed. Due to the mistakes in experiment replication, the results presented in Table 5 of Rao et al. [37] are non-comparable, and the conclusion that: "*TLBO is better for mean number of function evaluations but the error value is better for NM-PSO. The TLBO method requires approximately 1/10th of function evaluations for the functions 1 and 2, 1/50th of function evaluations for the function 6, and 1/200th of function evaluations for the function 7. Also success rate of TLBO is better than NM- PSO for functions 4, 5, and 7* [37]" cannot be supported. Another problem during the comparison is that the experiment in [13] included 20 benchmark functions, whilst the experiment in [37] reported replication on only 7 benchmark functions. Nevertheless, the experiment in [37] was repeated with the following control parameters: *pop_size* = 20 and *num_gen* = 200, as reported in [37]. The mean number of fitness evaluations and success rate (*sr*) were considered during this experiment. The results are presented in Table 9, but the comparison with NM-PSO was omitted due to the aforementioned problems. Whilst the presented implementation of TLBO ($TLBO_{imp}$) achieved better results than TLBO from Ref. [37] ($TLBO_{orig}$) regarding the mean number of fitness evaluations for 3 benchmark functions (Powell badly scaled, B2, and Griewank ($D = 50$)), the success rate (*sr*) measured over 100 independent runs for 2 benchmark functions, Griewank ($D = 10$) and Rastrigin ($D = 10$), were not 100% as reported in [37]. This means that solutions were not found during all the independent runs, and the claim: "*This experiment shows that TLBO is effective in terms of the computational effort, consistency, and obtaining the optimum solution* [37]" cannot be supported. Note that TLBO performed better on the higher dimensional problems of the Griewank function (compared with the success rates of Griewank ($D = 10$) and ($D = 50$)). This encouraged us to perform some additional tests on the Griewank function regarding different dimensions. The results were presented in the second part of Table 9. These results are in-line with the previous observation that TLBO performs better on higher dimensional problems of the Griewank function. Explanations for this behavior were provided in the next experiment.

In Experiment 3 [37] TLBO was compared to the experiment in [22] where the Artificial Bee Colony (ABC) algorithm [22] was compared to Harmony Search (HS), and the Bee Algorithm (BA). Since ABC outperformed HS and BA, only the results for ABC and TLBO are presented in continuation. The stopping criterion in experiment [22] was 50,000 fitness evaluations for all the three algorithms (ABC, HS, and BA) in order to achieve a fair comparison. The experiment in [37] was again performed slightly differently, since the same number of fitness evaluations as in [22] was used for the Rosenbrock function only (*pop_size* = 50, *num_gen* = 1000), whilst for other test functions 2000 fitness evaluations were used as the stopping-criterion (*pop_size* = 10, *num_gen* = 200). Due to the problems of TLBO when counting the number of fitness evaluations in [36] (see Section 2) it can be reasonably assumed that the same problem appeared here, and the number of fitness evaluations actually doubled (approximately 100,000 fitness evaluations for the Rosenbrock function and 4000 fitness evaluations for the other test functions). On the other hand, the same benchmark functions were experimented, and none were omitted. Hence, Experiment 3 was the closest replication of Rao et al. [37] presented in this paper. Although it is always desirable to replicate an experiment in an exact way, the reason for choosing a smaller number of fitness evaluations in this case might have been to show that, even under a smaller number of fitness evaluations, TLBO performed better than ABC, HS, and BA. This can be justifiable assuming that other compared algorithms (e.g., ABC, HS, and BA) cannot also produce comparable results under a smaller number of fitness evaluations. Such an assumption may not always be valid! The overall aim of this experiment was to show good performance on unimodal and multimodal functions with high dimensions and the criteria for comparison were the mean-solution and the standard deviation over 30 independent runs. The experiment in [22] was repeated using 50,000 evaluations for all test functions. The results are presented in Table 10. Comparison between $TLBO_{orig}$ and $TLBO_{imp}$ did not reveal any major inconsistencies, despite the fact that the results were not exactly the same. By qualitatively analyzing the results, the claim can be mostly supported that TLBO performs better than ABC on high dimensional test functions, although quantitative statistical tests should be performed in [37] to show the statistical significance of the results.

Those readers who carefully peruse Tables 9 and 10 may even notice that TLBO performed better on high dimensional problems than on low dimensional problems (e.g., compare the results for $TLBO_{imp}$ on the Griewank function). At first sight this seamed unusual. But, the explanation was quickly found in the fitness-distance correlation (ρ) measurement [20]. The fitness-distance correlation was actually the Pearson correlation coefficient (Eq. (4)) where the variables X and Y were the fitness value and the distance to global optimum, respectively.

Table 9
Experiment 2 (Table 5 in [37]).

Function	$TLBO_{orig}$		$TLBO_{imp}$	
	sr	Mean FE	*sr*	Mean FE ± St. Dev.
Powell badly scaled	100	2867	100	672.2 ± 198.0
B2	100	1048	100	748.2 ± 70.9
Booth	100	654	100	718.4 ± 81.6
Griewank ($D = 10$)	100	1059	50	5642.6 ± 1820.3
Rastrigin ($D = 10$)	100	1134	0	0.0 ± 0
Sphere ($D = 30$)	100	1543	100	1917.2 ± 69.6
Griewank ($D = 50$)	100	1857	100	1770.4 ± 89.4
Griewank ($D = 3$)	N/A	N/A	0	0.0 ± 0
Griewank ($D = 5$)	N/A	N/A	0	0.0 ± 0
Griewank ($D = 15$)	N/A	N/A	100	1735.3 ± 509.5

Table 10
Experiment 3 (Table 7 in [37]): *num_eval* = 50,000.

Function	D	ABC		$TLBO_{orig}$		$TLBO_{imp}$	
		Mean	St. Dev.	Mean	St. Dev.	Mean	± St. Dev.
Sphere	5	4.30E−17	1.07E−17	5.03E−33	1.27E−32	0.00E0	0.00E0
	10	7.36E−17	4.43E−17	2.26E−29	3.61E−29	0.00E0	0.00E0
	30	4.69E−16	1.07E−16	6.90E−26	3.18E−25	0.00E0	0.00E0
	50	1.19E−15	4.68E−16	8.71E−26	1.86E−25	0.00E0	0.00E0
	100	1.99E−06	2.26E−06	9.42E−26	3.70E−25	0.00E0	0.00E0
Rosenbrock	5	2.33E−01	2.24E−01	1.80E−01	8.04E−02	3.55E−2	3.00E−1
	10	4.62E−01	5.44E−01	5.58E0	6.18E−01	9.38E−2	1.54E−1
	30	9.98E−01	1.52E0	2.71E1	1.14E0	2.16E1	9.23E−1
	50	4.33E0	5.48E0	4.78E1	1.01E0	4.33E1	7.43E−1
	100	1.12E2	6.92E1	9.81E1	3.61E−01	9.47E1	1.04E0
Ackley	5	9.64E−17	5.24E−17	0.00E0	0.00E0	1.81E−15	1.57E−15
	10	3.51E−16	6.13E−17	0.00E0	0.00E0	4.23E−15	8.48E−16
	30	3.86E−15	3.16E−15	7.11E−16	1.82E−15	4.48E−15	3.55E−16
	50	4.38E−08	4.65E−08	1.24E−15	1.95E−15	2.75E−1	1.95E0
	100	1.32E−02	1.30E−02	2.13E−15	1.19E−15	1.42E0	4.57E0
Griewank	5	4.04E−17	1.12E−17	0.00E0	0.00E0	1.95E−2	1.79E−2
	10	6.96E−17	4.06E−17	0.00E0	0.00E0	9.94E−3	2.35E−2
	30	5.82E−06	3.13E−05	0.00E0	0.00E0	0.00E0	0.00E0
	50	5.72E−01	9.22E−01	0.00E0	0.00E0	0.00E0	0.00E0
	100	1.31E1	6.30E0	0.00E0	0.00E0	0.00E0	0.00E0
Rastrigin	5	4.34E−17	110E−17	0.00E0	0.00E0	1.67E−1	6.94E−1
	10	5.77E−17	2.98E−17	0.00E0	0.00E0	3.71E0	3.54E0
	30	4.80E−05	243E−04	0.00E0	0.00E0	1.60E1	1.56E1
	50	4.72E−01	4.92E−01	0.00E0	0.00E0	1.48E1	2.48E1
	100	1.46E1	4.18E0	0.00E0	0.00E0	0.00E0	0.00E0

$$\rho(X,Y) = \frac{Cov(X,Y)}{\sigma(X)\sigma(Y)} \tag{4}$$

Fitness-distance correlation is a measurement of how the distance to global optimum correlated with the fitness values. A positive correlation means that the smaller the distance to the global optimum, the smaller the fitness value (for minimization problems). Conversely, the bigger the distance from the global optimum, the bigger the fitness value is. Fitness-distance correlation can be used to predict the performance of an EA on problems with known global optima [20]. The guidelines for the interpretation of a correlation coefficient for minimization problems (positive correlation) are as follows:

- uncorrelated: $0 \leqslant \rho \leqslant 0.09$,
- small correlation: $0.1 \leqslant \rho \leqslant 0.3$,
- medium correlation: $0.3 < \rho \leqslant 0.5$,
- strong correlation: $0.5 < \rho \leqslant 1.0$.

Merz [27] showed that in uncorrelated or low correlated fitness landscapes, moves in random directions obtained by mutation/crossover are more effective than moves towards other solutions with high fitness values. TLBO is a typical algorithm that moves towards the best solution (see Section 2), and as such must suffer from problems where fitness-distance correlation does not exist or is small. The authors computed ρ for the different optimization problems used in Experiment 3 [37] on different dimensions (Fig. 1). The results showed that, in many cases, the fitness-distance correlation increased with the dimensions, and hence such problems were actually easier to solve. For example, for Griewank ($D = 2,[-50,50]$) the fitness-distance correlation was $\rho = 0.45$, whilst for ($D = 50$) it was $\rho = 0.99$ (Fig. 1). This explained the results on the Griewank functions under higher dimensions in Table 9. From amongst the tested functions, only the Schwefel ridge function (Eq. (5)) exhibited a property that the fitness-distance correlation was decreasing with higher dimensions. The fitness-distance scattered plot for the Schwefel ridge function ($D = 50$, $[-64,64]$), is shown in Fig. 2. The authors also ran TLBO on the Schwefel ridge function which has very low ρ, and the results confirmed that this was a very difficult problem for TLBO. Solutions were not found due to non-correlation between fitness and distance to global optimum (Table 11). This experiment again, did not support the too-general claim: "*This experiment shows that TLBO is effective in finding the optimum solution with increase in dimensions* [37]."

$$f(\vec{x}) = \sum_{i=1}^{D}\left(\sum_{j=1}^{i} x_j\right)^2 \quad \text{where} \quad -64 \leqslant x_j \leqslant 64 \tag{5}$$

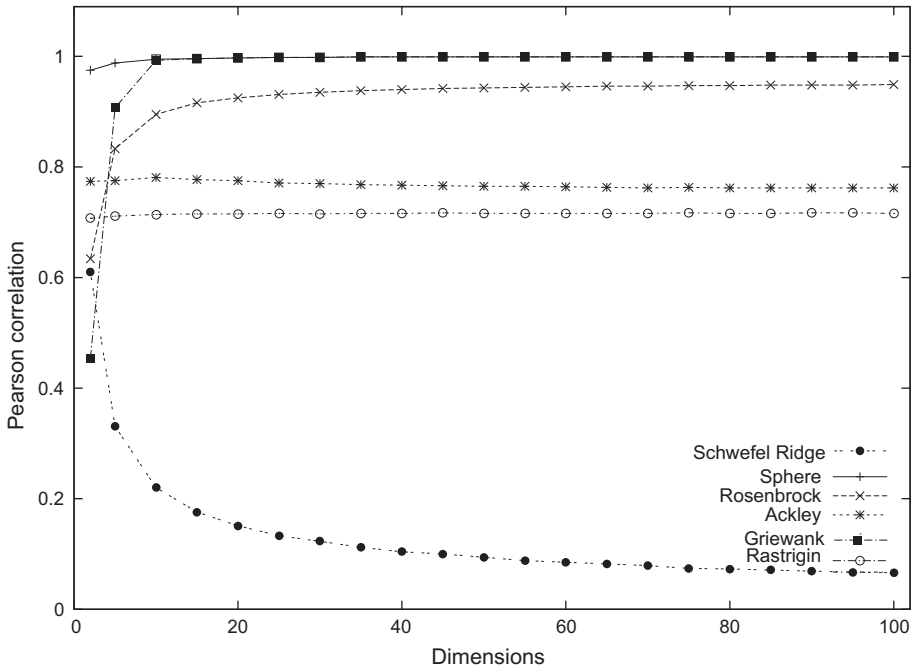

Fig. 1. Fitness-distance correlation for functions from Experiment 3.

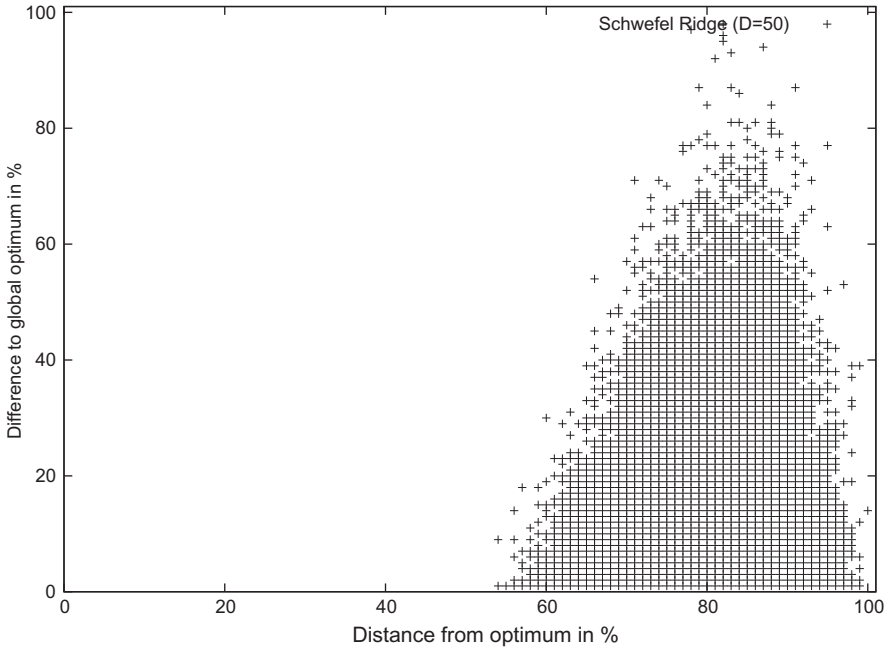

Fig. 2. Fitness-distance correlation plot for Schwefel ridge ($D = 50$); $\rho = 0.093$.

4. Discussion

Replication of experiments and comparisons amongst different evolutionary algorithms must be done with sufficient care. The authors provide the following simple guidelines for prospective researchers in order to avoid mistakes and uncertainties when replicating experiments:

Table 11
Experiment 3 for Schwefel ridge function; *Elite* = 0, *num_eval* = 50,000.

Function	*D*	$TLBO_{imp}$	
		Mean	± St. Dev.
Schwefel ridge	5	8.20E−111	8.17E−110
	10	5.68E−13	2.72E−12
	30	1.73E1	3.30E1
	50	8.94E3	7.45E3
	100	1.71E5	1.22E5

1. *Don't be afraid to acknowledge that your algorithm is not the best*. According to the NFL theorem [44], no such algorithm exists. Claiming that your algorithm is the best means that you have either not compared your algorithm with the best available algorithms or you have omitted some problems for which your algorithm is unsuitable. Reviewers should raise red flags for both cases.
2. *Carefully examine those problems where your algorithm performs well*. Try to find as many different families of problems as possible on which your algorithm performs well, as well as discussing the related problems where your algorithm might not perform as well.
3. *Qualitatively and quantitatively evaluate the results*. Try to find good explanations as to why your algorithm works well for particular kinds of problems and, conversely, why your algorithm has difficulties finding optimal solutions for other problems. Such qualitative descriptions should be followed by quantitative analyses showing the statistical significances of the results.
4. *Try to replicate experiments under the same conditions*. If the same conditions cannot be achieved, you should clearly explain why the same conditions were not achieved and what the threats to the validity of the experiment are. The same conditions include, firstly, all benchmark functions and the same stopping criteria. The number of independent runs should be at least as high as in the original experiment. Any deviations from the original experiments must be carefully documented.
5. *Don't be too general in your conclusions*. Often conclusions are invalid, as they are not specific enough. Great care needs to be devoted to proper conclusions. Good experimental parts and the evaluation of results are crucial for proper conclusions.
6. *Prefer an equal number of fitness evaluations rather than an equal number of generations*. Many EAs have extra steps where more fitness evaluations are required or variable population sizes are employed. In such cases, comparing algorithms over an equal number of generations is unfair.
7. *Describe the optimization problem exactly*. Often many mistakes in the descriptions of benchmark functions were found (e.g., wrong upper and lower bounds, missing constraints, wrong constants in equations). Such mistakes can be easily propagated into other experiments making comparisons useless.
8. *Don't forget to mention all the parameters for an experiment*. Often experiments were ill-described and/or missing important information (e.g., number of independent runs, number of fitness evaluations). In such cases experiment replication is impossible.
9. *Carefully consider which results are important for the algorithm's performance*. Often the results would be more meaningfully presented by measuring proper variables. Standard deviation is often omitted, as well as the success rate.
10. *Make your code publicly available*. Description in papers is often too abstract and lacks sufficient detail. With your code available to the public, researchers and reviewers can easily replicate your experiment and validate your results. Confidence in your results will be increased.
11. *Implement your algorithm in a comparable programming language*. Different programming languages and their compilers and runtime environments have different optimization techniques, which may result in different execution efficiencies. It is also important to understand a platform's default floating point arithmetic (float, double, 32bit vs. 64bit). The quality and type of the default random generator can also have a big impact on algorithm performance. Best practice when comparing algorithms would be to have all of them implemented on the same standard platform.
12. *Implement your algorithm with the same precision*. When individuals are represented as binary strings in some EAs, the string-length matters for each variable of an individual. Smaller or insufficient string lengths may result in imprecise real value conversion. Unfair or imprecise comparisons may then occur regardless of whether EAs are implemented under the same standard platform or not. Michalewicz [29] (pages 33–34) provides a general guideline on how to compute the minimum length requirement of a binary string in order to represent a real value.

5. Conclusion

The goal of this paper was to experiment with the recently developed Teaching–Learning-Based Optimization (TLBO) algorithm. It was reported in [36,37] that TLBO finds better or equal solutions much faster than other EAs, that TLBO has

no parameters and, as such, can be more easily applied by practitioners for solving difficult engineering design problems. These claims were re-tested during our experiments and both were found to be invalid.

On the same constrained optimization functions and using the same control parameters as reported in [36], we were unable to achieve the same results. Neither better nor equal solutions were found by TLBO or the number of fitness function evaluations were much higher than reported in [36]. In order to achieve the same results as in [36], we had to tune the control parameters (population size, maximum number of generations, and size of the elite), which was a very tedious process and unfortunately just comparable to control parameter tuning in other EAs. Hence, TLBO cannot be regarded as a parameter-less algorithm in the sense of Bäck et al. [3], Eiben et al. [11] and Harik and Lobo [16].

On the unconstrained problems, TLBO performed much better [37] and our implementations were mostly consistent with the reported results, but the replications of experiments in [37] were done poorly. It is worth noticing that there were some mismatches regarding success rate for some optimization problems (see Table 9). Moreover, good results were only achieved for those optimization problems where the fitness-distance correlation was high.

Despite the fact that the TLBO algorithm is interesting and worth exploring, the experiments have been incorrectly replicated. The results from our quantitative and qualitative examinations cannot support the conclusions that TLBO uses less computational effort than other EAs for solving constrained and unconstrained problems. This paper also revealed some mistakes when comparing TLBO with other metaheuristics [36,37]. Metaheuristic researchers and practitioners should avoid such mistakes in the future when making algorithm comparisons.

In the future we would like to measure TLBO's exploration and exploitation powers [8] using our recently developed metrics [7], and compare them with other EAs. By identifying the true exploration and exploitation powers of TLBO, more insights about the algorithm's inner workings and a better explanation of the presented results may be given.

Acknowledgements

The third author's work was partly sponsored by the Slovene Human Resources Development and Scholarship Fund. The authors would also like to thank Marjan Mernik for fruitful discussions and guidelines.

References

[1] A. Ahrari, A.A. Atai, Grenade explosion method – a novel tool for optimization of multimodal functions, Applied Soft Computing 10 (4) (2010) 1132–1140.
[2] T. Bäck, D.B. Fogel, Z. Michalewicz, Handbook of Evolutionary Computations, Oxford University Press, 1996.
[3] T. Bäck, A.E. Eiben, N.A.L. van der Vaart, An empirical study on gas without parameters, in: Proceedings of the 6th International Conference on Parallel Problem Solving from Nature, 2000, pp. 315–324.
[4] H.-G. Beyer, H.-P. Schwefel, Evolution strategies: a comprehensive introduction, Journal Natural Computing 1 (1) (2002) 3–52.
[5] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: overview and conceptual comparison, ACM Computing Surveys 35 (3) (2003) 268–308.
[6] C. Blum, J. Puchinger, G.A. Raidl, A. Roli, Hybrid metaheuristics in combinatorial optimization: a survey, Applied Soft Computing 11 (6) (2011) 4135–4151.
[7] M. Črepinšek, M. Mernik, S.H. Liu, Analysis of exploration and exploitation in evolutionary algorithms by ancestry trees, International Journal of Innovative Computing and Applications 3 (1) (2011) 11–19.
[8] M. Črepinšek, S.H. Liu, M. Mernik, Exploration and exploitation in evolutionary algorithms: a survey, ACM Computing Surveys, in press.
[9] K. Deb, An efficient constraint handling method for genetic algorithms, Computer Methods in Applied Mechanics and Engineering 186 (2–4) (2000) 311–338.
[10] J.E. Dennis, D.J. Woods, Optimization on Microcomputers: The Nelder-Mead Simplex Algorithm, Technical Report 85-9, Rice University, 1985.
[11] A.E. Eiben, R. Hinterding, Z. Michalewicz, Parameter control in evolutionary algorithms, IEEE Transactions on Evolutionary Computation 3 (2) (1999) 124–141.
[12] A.E. Eiben, E. Marchiori, V.A. Valkó, Evolutionary algorithms with on-the-fly population size adjustment, Parallel Problem Solving from Nature VIII (2004) 41–50.
[13] S.-K. Fan, E. Zahara, A hybrid simplex search and particle swarm optimization for unconstrained optimization, European Journal of Operational Research 181 (2) (2007) 527–548.
[14] I. Fister, M. Mernik, B. Filipič, A hybrid self-adaptive evolutionary algorithm for marker optimization in the clothing industry, Applied Soft Computing 10 (2) (2010) 527–548.
[15] D. Goldberg, Genetic Algorithms in Search Optimization, and Machine Learning, Addison-Wesley, 1989.
[16] G.R. Harik, F. Lobo, A parameter-less genetic algorithm, Technical Report, University of Illinois at Urbana-Champaign, 1999.
[17] D. Hrnčič, M. Mernik, B.R. Bryant, F. Javed, A memetic grammar inference algorithm for language learning, Applied Soft Computing 12 (3) (2012) 1006–1020.
[18] F.Z. Huang, L. Wang, Q. He, An effective co-evolutionary differential evolution for constrained optimization, Applied Mathematics and Computation 186 (1) (2007) 340–356.
[19] G. Iacca, F. Neri, E. Mininno, Y.-S. Ong, M.-H. Lim, Ockham's Razor in memetic computing: three stage optimal memetic exploration, Information Sciences 188 (2012) 17–43.
[20] T. Jones, S. Forrest, Fitness distance correlation as a measure of problem difficulty for genetic algorithms, in: Proceedings of the 6th International Conference on Genetic Algorithms, 1995, pp. 184–192.
[21] D. Karaboga, B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Applied Soft Computing 8 (1) (2008) 687–697.
[22] D. Karaboga, B. Akay, Artificial bee colony (ABC), harmony search and bees algorithms on numerical optimization, in: Proceeding of IPROMS-2009 on Innovative Production Machines and Systems, 2009.
[23] J. Kennedy, R.C. Eberhart, Y. Shi, Swarm Intelligence, Morgan Kaufmann Publishers, 2001.
[24] R. Landa Becerra, C.A. Coello Coello, Cultured differential evolution for constrained optimization, Computer Methods in Applied Mechanics and Engineering 195 (33–36) (2006) 4303–4322.
[25] K.S. Lee, Z.W. Geem, A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice, Computer Methods in Applied Mechanics and Engineering 194 (36–38) (2005) 3902–3933.
[26] S.-H. Liu, M. Mernik, B.R. Bryant, To explore or to exploit: an entropy-driven approach for evolutionary algorithms, International Journal of Knowledge-based and Intelligent Engineering Systems 13 (3–4) (2009) 185–206.

[27] P. Merz, Advanced fitness landscapes analysis and the performance of memetic algorithms, Evolutionary Computation 12 (3) (2004) 303–325.
[28] E. Mezura-Montes, C.A. Coello Coello, A simple multi-membered evolution strategy to solve constrained optimization problems, IEEE Transactions on Evolutionary Computation 9 (1) (2005) 1–17.
[29] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, third ed., Springer-Verlag, 1996.
[30] A.W. Mohamed, H.Z. Sabry, Constrained optimization based on modified differential evolution algorithm, Information Sciences 194 (2012) 171–208.
[31] D. Mongus, B. Repnik, M. Mernik, B. Žalik, A hybrid evolutionary algorithm for tuning a cloth-simulation model, Applied Soft Computing 12 (1) (2012) 266–273.
[32] A. Mucherino, L. Liberti, C. Lavor, N. Maculan, Comparisons between an exact and a metaheuristic algorithm for the molecular distance geometry problem, in: Proceedings of the 11th Conference on Genetic and Evolutionary Computation, 2009, pp. 333–340.
[33] I.H. Osman, G. Laporte, Metaheuristics: a bibliography, Annals of Operations Research 63 (1996) 513–623.
[34] G. Palubeckis, D. Rubliauskas, A. Targamadze, Metaheuristic approaches for the quadratic minimum spanning tree problem, Information Technology and Control 39 (4) (2010) 257–268.
[35] B.Y. Qu, J.J. Liang, P.N. Suganthan, Niching particle swarm optimization with local search for multi-modal optimization, Information Sciences 197 (2012) 131–143.
[36] R.V. Rao, V.J. Savsani, D.P. Vakharia, Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems, Computer-Aided Design 43 (3) (2011) 303–315.
[37] R.V. Rao, V.J. Savsani, D.P. Vakharia, Teaching–learning-based optimization: an optimization method for continuous non-linear large scale problems, Information Sciences 183 (1) (2012) 1–15.
[38] T.P. Runarsson, X. Yao, Stochastic ranking for constrained evolutionary optimization, IEEE Transactions on Evolutionary Computation 4 (3) (2000) 284–294.
[39] H. Shah-Hosseini, The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm, International Journal of Bio-Inspired Computation 1 (1–2) (2009) 71–79.
[40] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (4) (1997) 341–359.
[41] X. Yang, S. Deb, Cuckoo search via Lévy flights, World Congress on Nature and Biologically Inspired Computing (2009) 210–214.
[42] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Transactions on Evolutionary Computation 3 (2) (1999) 82–102.
[43] Y. Wang, Z. Cai, Q. Zhang, Enhancing the search ability of differential evolution through orthogonal crossover, Information Sciences 185 (1) (2012) 153–177.
[44] D. Wolpert, W. Macready, No free lunch theorems for optimization, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 67–82.
[45] A. Zavala, A. Aguirre, E. Diharce, Constrained optimization via evolutionary swarm optimization algorithm (PESO), in: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, 2005, pp. 209–216.