Article:

Yang, Z, Li, K, Guo, Y et al. (2 more authors) (2018) Compact real-valued teaching-learning based optimization with the applications to neural network training. Knowledge-Based Systems, 159. pp. 51-62. ISSN 0950-7051

https://doi.org/10.1016/j.knosys.2018.06.004

Compact Real-valued Teaching-Learning Based Optimization with the Applications to Neural Network Training

Zhile Yang[a], Kang Li[b], Yuanjun Guo*[a], Haiping Ma[c], Min Zheng[d]

[a]*Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, Guangdong, 518055, China*
[b]*School of Electronic and Electrical Engineering, University of Leeds, Leeds, LS2 9JT*
[c]*Department of Electrical Engineering, Shaoxing University, Shaoxing, China*
[d]*Shanghai Key Laboratory of Power Station Automation Technology, School of Mechatronic Engineering and Automation, Shanghai University, Shanghai, China*

Abstract

The majority of embedded systems are designed for specific applications, often associated with limited hardware resources in order to meet various and sometime conflicting requirements such as cost, speed, size and performance. Advanced intelligent heuristic optimization algorithms have been widely used in solving engineering problems. However, they might not be applicable to embedded systems, which often have extremely limited memory size. In this paper, a new compact teaching-learning based optimization method for solving global continuous problems is proposed, particularly aiming for neural network training in portable artificial intelligent (AI) devices. Comprehensive numerical experiments on benchmark problems and the training of two popular neural network systems verify that the new compact algorithm is capable of maintaining the high performance while the memory requirement is significantly reduced. It offers a promising tool for continuous optimization problems including the training of neural networks for intelligent embedded systems with limited memory resources.

1. Introduction

Compact embedded systems have been widely used in many engineering fields, from portable monitoring, autonomous control devices, to battery management systems in electric vehicles. In order to meet various often conflicting requirements such as cost, size, speed, reliability and performance, embedded systems are therefore often implemented with limited hardware resources. Many embedded systems require intelligence for system operation, adding that computational intelligent (CI) techniques are indispensable tools to achieve complex tasks. The majority of them require strong support of sufficient hardware resources. For example, neural network training for robot route planning, proportional-integral-derivative (PID) controllers design for chemical production processes [1], as well as the smart clustering for large scale multiple wireless sensor network [2], all of which require intelligent optimization methods. However, embedded systems using microprocessors like Intel MCS 51 series, one of the most popular micro controllers used in the robotic systems and process control systems, has only 128K on-chip RAM [3]. Such small memory size presents an extremely limited design environment in implementing on-board intelligent optimization algorithms.

Compact algorithms have been an independent cluster relating to the estimation distribution based algorithm (EDA) [4]. They generate the solutions in each generation using a certain distribution information and improve the performance through the evolutionary process. It needs to maintain only a very limited number of particles in the process other than updating a group of particles in traditional meta-heuristic methods. The memory usage is therefore significantly reduced by adopting the compact algorithm structure. The compact algorithms are originated from binary compact genetic algorithms (cGA) [4, 5, 6], and have been extended to solve real-valued optimization problems [7, 8, 9, 10, 11, 12].

Teaching-learning based optimization (TLBO) proposed in 2011 [13, 14] is a popular meta-heuristic optimization approach. A classroom teaching situation is mimicked within the particle learning strategy. The relatively competitive performance of TLBO and its variants have been verified [15, 16, 17, 18] and demonstrated in a number of applications [19, 20, 21, 22, 23]. The merit of this algorithm is claimed to be free of algorithm specific parameters, such as the crossover rate and the mutation rate in GA, and social and cognitive rates in particle swarm optimization (PSO), therefore significantly reduces the parameter tuning effort in algorithm applications.

On the other hand, how to train neural networks (NNs) has been a long intractable problem due to the high dimensional and non-linear characteristics. A significant number of non-linear parameters in the neural networks need to be optimized. Many meta-heuristic methods have been adopted to optimize theses non-linear parameters such as genetic algorithm (GA) [24], PSO [25, 26], biogeography-based optimization (BBO) [27], monarch butterfly optimization (MBO) [28], artificial fish swarm algorithm (AFSA) [29], glowworm swarm optimization (GSO) [30] etc. However, very few publications have utilized the novel and efficient TLBO method for NN parameter optimization. In addition, it is also a new topic to utilize compact algorithms to train NNs used in an increasing number of independent intelligent systems.

Our previous work [31] provided a preliminary study of the compact TLBO but with very limited numerical comparison and no applications. In this paper, the detailed compact teaching learning optimization (cTLBO) is presented, where the TLBO algorithm logic is embedded into the compact structure. One solution particle is generated from an updated Gaussian distribution in each iteration and the population distribution is improved through a competition between the particle and a teacher. Numerical results on 32 well-known benchmarks are conducted. Comprehensive results show that the novel cTLBO method outperforms the other typical meta-heuristics as well as other compact algorithms by significantly reducing the memory storage and improving the optimization performance. In addition, the cTLBO method is adopted to train feedfoward neural network (FNN) and radial basis function (RBF) neural network for approximating various non-linear systems, and again it offers competitive performance in comparison with other counterparts.

2. Compact Optimization

2.1. Compact Binary Optimization

Compact algorithm was first termed by Harik et al. [4, 32]. The original compact genetic algorithm design focuses on the crossover scheme of a binary GA. For each bit in a single gene (i.e. a solution), a probability number in a probability vector (PV) is maintained to represent the likelihood of 0 or 1. The evolution process will generate two new particles and select a winner based on the fitness values, then the winner will be used to update the probability through a bit-to-bit improvement. Ahn et al. [5] proposed two elitism based cGA methods, namely the persistent elitist cGA and nonpersistent elitist cGA. The winner is maintained as the global elitist in a bid to retain the best performer and speed up the convergence. Gallagher et al. [6] further designed a mutation step and a re-sampling step to enhance the algorithm performance.

2.2. Compact Real-valued Optimization

The initial cGAs are specialized for binary optimization problems as the maintained PV corresponds to the probability of the bit in gene only for GA. For the particles, their values have to be converted into or coded in the binary form. It will generally require significant computational resources and huge memory size. On the other hand, float point number has been widely supported by Micro control units (MCUs). Therefore, it is less difficult for the implementation of real-valued methods in the embedded control system. A real-valued cGA (rcGA) is proposed by Mininno et al. [7], where the PVs are replaced by truncated normal distributed probability density showed in (1). The idea of this truncated function is to transfer the original normal distributed variables ranging from $[-\infty,\infty]$ to [-1,1], through which the boundary values of variables [a,b] could be easily linked by linear conversion from [-1,1] as mentioned in [7]. The PDF_j below denotes the probability density function of the i^{th} variable where erf is the error function.

$$PDF_i = \frac{e^{-\frac{(x-\mu[i])^2}{2\sigma[i]^2}}\sqrt{\frac{2}{\pi}}}{\sigma[i]\left(erf\left(\frac{\mu[i]+1}{\sqrt{2}\sigma[i]}\right) - erf\left(\frac{\mu[i]-1}{\sqrt{2}\sigma[i]}\right)\right)} \quad (1)$$

The values in each dimension are generated from the corresponding PDF which are updated through a straightforward elite strategy illustrated in [7]. Another elegant property of the compact real-valued method is that this structure enables the integration of the compact algorithms with numerous meta-heuristic algorithms. The advantages of the small memory size necessity of compact algorithms and the powerful learning capability of conventional heuristic methods will be retained within such a structure.

In addition to the rcGA, some other real-valued optimization methods have been proposed in association with the differential evolution (DE) algorithm [8, 10, 9, 33, 34], particle swarm optimization (PSO) [11, 35], artificial bee colony [36, 12], bat algorithm [37] and flower pollination algorithm [38] respectively. Fig.1 illustrates the process of cDE. In the initialization stage, the mean value μ and standard deviation σ of a Gaussian distribution are defined as the probability vector and valued as 0 and 10 respectively according to the experimental data for the global continuous problem optimization. The reason for choosing an initial value of 10 for the standard deviation σ was explained in [7] that a large number of initial standard deviation could ensure the initial probability of the first generation to be uniformly distributed. A single particle named the elite is generated from the initial PV. Then the procedure proceeds to the mutation step, where 3 new solutions are generated from PV. The difference of two solutions out of three are calculated and added to the third one to formulate a new candidate solution. A crossover step is then conducted where a random crossover rate ranging from 0 to 1 is used to determine whether the new solution is adopted or not in each dimension. The new solution is subsequently competed with its predecessor and the winner will be used to update the probability density vector (i.e. PV) by modifying the mean value μ and standard deviation σ as in [7]. The whole process is inspired from the evolutionary rule of the original DE.

Comparative studies between the compact real-valued methods and conventional state-of-the-art counterparts

counter $t = 0$;
for $i = 1 : n$ **do**
  // **PV** *initialization* //
  initialize $\mu_t[i] = 0$
  initialize $\sigma_t[i] = \lambda = 10$
end for
generate **elite** by means of **PV**
while budget condition **do**
  // *Mutation* //
  generate 3 individuals x_r,x_s and x_t by means of **PV**
  compute $x'_{off} = x_t + F(x_r - x_s)$
  // *Crossover* //
  $x_{off} = x'_{off}$
  for i=1:n **do**
    generate $rand(0, 1)$
    if $rand(0, 1) > Cr$ **then**
      $x_{off}[i] = elite[i]$
    end if
  end for
  // *Elite Selection* //
  $[winner, loser] = compete(x_{off}, elite)$;
  if $x_{off} == winner$ **then**
    $elite = x_{off}$
  end if
  // *PV Update* //
  for $i = 1 : n$ **do**
    $\mu_{t+1}[i] = \mu_t[i] + \frac{1}{Np}(winner[i] - loser[i])$;
    $\sigma_{t+1}[i] = \sqrt{(\sigma_t[i])^2 + (\mu_t[i])^2 - (\mu_{t+1}[i])^2 + \frac{1}{Np}(winner^2[i] - loser^2[i])}$;
  end for
  $t = t + 1$;
end while

Figure 1: Pseudo code of the compact differential evolution

show that the new cDE and cPSO both outperform the original methods on the majority of the test benchmarks. Although cDE and cPSO perform reasonably well as long as the algorithm specific parameters (i.e. the mutation factor for DE, and the cognitive and social learning factors for PSO) are properly tuned, the tuning of these parameters are however often tedious and time consuming, and the tuned settings often can not be generalized to other optimization problems. Therefore, algorithms free from tuning specific parameters are most attractive in compact algorithm design.

3. Teaching-learning based optimization

Teaching-learning based optimization is a recently proposed meta-heuristic algorithm that mimics a teaching and learning process [13, 14]. In TLBO, there is no algorithm specific parameters that need to be tuned in the optimizing process. This new method and its variants have been well adopted in solving a range of mathematical and engineering optimization problems including multi-objective optimization applications [39], medical diagnoses [40], power systems [19, 20, 41, 42, 43, 44, 45], and chemical industry[46]. The method has also been hybridized with the harmony search [47] and the two phases in TLBO, namely teaching phase and learning phase, are performed along with the evolutionary process.

3.1. Teaching Phase

Teaching phase is similar to the PSO method in which the best solution (named as the teacher) in the population has the overall impact on the whole population of particles (named as the students in the TLBO). A teacher is first selected from the class by sorting the grades (fitness function). Then, the mean values of subject knowledges $Mean_i$ (i.e. values in each dimension) for all the students are calculated. The value difference between the teacher T_i

and the mean value is further calculated and (2) is adopted as the teacher's instruction introduced all students.

$$DM_i = rand_1 \times (T_i - T_F Mean_i) \quad (2)$$

where DM_i is the value difference in the i^{th} iteration. T_F is a teaching factor defined as either 1 or 2 presented as:

$$T_F = round(1 + rand_2(0,1)) \quad (3)$$

Subsequently, the teacher's instruction will be exerted on the students by adding the difference value to all the students:

$$X_{ij}^{new} = X_{ij}^{old} + DM_i \quad (4)$$

X_{ij} denotes the j^{th} student in the class during the i^{th} iteration. X_{ij}^{new} and X_{ij}^{old} are the specific ones before and after the learning phase. The new learners will compete with their predecessors and replace them if a better fitness value is achieved. In the teaching phase, the mutation factor is denoted by two random numbers: $rand_1$ and $rand_2$ for determining the learning step length DM_i.

3.2. Learning Phase

The main purpose of the teaching phase is to guide the students moving towards proper directions, due to which this phase is adopt in global exploration, and however lacks exploitation ability. Learning phase is therefore proposed to complement and enhance the exploitation ability. In the learning phase, each student will learn from a classmate to speed up the convergence of the whole population. The process of learning phase is illustrated as follows [13, 14] :

$$X_{ij}^{new} = \begin{cases} X_{ij}^{old} + rand_3(X_{ik} - X_{ij}) & if \quad f(X_{ik}) < f(X_{ij}) \\ X_{ij}^{old} + rand_3(X_{ij} - X_{ik}) & if \quad f(X_{ij}) < f(X_{ik}) \end{cases} \quad (5)$$

where X_{ik} is the randomly selected k^{th} student to share his/her knowledge with X_{ij}. The learning direction would be determined by the better performed one. In another word, the better student among these two will be subtracted by the worse one. The deviation will be added to the original learning candidate. Similarly, the new solutions will compete with the original ones, and the better one will remain in the population. In this phase, another random number $rand_3$ is used to determine the mutation step in learning step.

It is evident that the both phases in TLBO only utilize random numbers in determining the mutating rate. All the algorithm specific parameters have been eliminated and the whole process is now free of tuning. This advantage has a significant implication on the compact algorithm design.

4. Compact Teaching-learning Algorithm

In order to take the advantages of both the compact algorithm in saving memory storage and TLBO in being free of parameter tuning, a new compact teaching-learning based optimization is proposed in this section. The cTLBO maintains a PV for generating new particle solutions in every single iteration. This PV is formulated by the mean and standard deviation values for each dimension of the solutions. It is updated in every evolutionary generation by new winner solutions in the competition of learning process and represents the whole population distribution. The evolutionary logic of TLBO is integrated with the compact algorithm structure as illustrated in Fig. 2.

4.1. Initialization

In the initialization step, n denotes the dimension number and t refers to the iteration time. A two-column PV is initialized, with the first column $\mu_t[i]$ representing the mean value of each dimension and the second column $\sigma_t[i]$ standing for the standard deviation in t^{th} generation. Similar to the cDE and cPSO [8, 11], they are initialized as 0 and 10 for all dimensions respectively according to the empirical test. A global optimum solution is first generated as the teacher followed by PV assignment.

4.2. Compact Teaching Phase

A compact teaching phase is designed to share the same logic of the original TLBO. Only one new solution is generated from the updated Gaussian distribution represented by PV and is denoted as St_t. The difference between the mean value μ_t and the teacher Tr_t is calculated and added to the student, thus generating a new student St_t^{new}. This new student will compete with the teacher by comparing the fitness value. The winner will update the probability distribution density of the whole population by modifying the mean and standard deviation values in PV. It should be noted that in the equation of probability updating method, Np is the equivalent particle number which is a virtual parameter that represents the impact of each of the solutions on the whole population. This number could also be taken as the particle number in calculating the function evaluations.

4.3. Compact Learning Phase

After being updated through a learning process from the teacher, student interactive learning scheme is also introduced into the compact structure. One more new student is generated from PV represented as St_t^{new2}. This second student competes with the St_t^{new} in the previous phase, sharing knowledges and generating a new student St_t^{new3} similar as in the equation (5). The winner also updates the PV so as to further improve the whole population performance. The winner of the learning phase will be defined as the teacher for the next iteration. The global optimum will be the winner of the final iteration.

It could be observed that the predominant distinction of the compact teaching phase and learning phase is that only two or three new solutions are used in the

counter $t = 0$;
for $i = 1 : n$ **do**
  // **PV** *initialization* //;
  initialize $\mu_t[i] = 0$;
  initialize $\sigma_t[i] = \lambda$;
end for
generate $\mathbf{Tr}_t$ by means of **PV**;
while counter t has not arrived its maximum value **do**
  // **Compact Teaching Phase** //
  // *New student generation* //
  $\mathbf{St}_t = generate(\mathbf{PV})$;
  $DMean_t = rand_1 \times (\mathbf{Tr}_t - round(1 + rand_2(0,1)) \times \mu_t)$;
  $\mathbf{St}_t^{new} = \mathbf{St}_t + DMean_t$;
  // *Better student Selection* //
  $[winner, loser] = compete(\mathbf{St}_t^{new}, \mathbf{Tr}_t)$;
  for $i = 1 : n$ **do**
    $\mu_{t+1}[i] = \mu_t[i] + \frac{1}{Np}(winner[i] - loser[i])$;
    $\sigma_{t+1}[i] = \sqrt{(\sigma_t[i])^2 + (\mu_t[i])^2 - (\mu_{t+1}[i])^2 + \frac{1}{Np}(winner^2[i] - loser^2[i])}$;
  end for
  // **Compact Learning Phase** //
  $\mathbf{St}_t^{new2} = generate(\mathbf{PV})$;

$$\mathbf{St}_t^{new3} = \begin{cases} \mathbf{St}_t^{new} + rand_3(\mathbf{St}_t^{new2} - \mathbf{St}_t^{new}) & if \ \ f(\mathbf{St}_t^{new2}) < f(\mathbf{St}_t^{new}); \\ \mathbf{St}_t^{new} + rand_3(\mathbf{St}_t^{new} - \mathbf{St}_t^{new2}) & if \ \ f(\mathbf{St}_t^{new}) < f(\mathbf{St}_t^{new2}); \end{cases}$$

  $[winner, loser] = compete(\mathbf{St}_t^{new3}, \mathbf{Tr}_t)$;
  for $i = 1 : n$ **do**
    $\mu_{t+1}[i] = \mu_t[i] + \frac{1}{Np}(winner[i] - loser[i])$;
    $\sigma_{t+1}[i] = \sqrt{(\sigma_t[i])^2 + (\mu_t[i])^2 - (\mu_{t+1}[i])^2 + \frac{1}{Np}(winner^2[i] - loser^2[i])}$;
  end for
  $\mathbf{Tr}_{t+1} = winner$;
  $t = t + 1$;
end while
$St_{opt} = \mathbf{Tr}_{tmax}$

Figure 2: Pseudo code of the compact teaching-learning based optimization

evolutionary logic other than a population of Np students in each iteration, which aims to retain the compact structure. In the rest of the paper, the novel cTLBO method is tested in a number of popular benchmark functions and then applied to training feedforward neural network and RBF neural network. The corresponding problems and the results are also discussed and the proposed algorithm is well compared with other meta-heuristic algorithms from all respects.

5. Benchmark Tests

In this section, the proposed cTLBO is tested on 32 well-known benchmark functions with 30 dimensions or 100 dimensions [48, 49, 50]. All benchmark functions are shown in Table 1, where D denotes the dimension of the problems. In order to comprehensively compare the algorithm performance, several well-applied meta-heuristic methods including inertial weighted PSO (wPSO) [51], constriction factor PSO (cfPSO) [52], DE/rand/1 algorithm [53] and a new algorithm moth flame optimization [54], some state-of-the-art TLBO variants including the original TLBO, an elite TLBO (ETLBO) [55], a modified TLBO [18] (mTLBO) and a self-learning TLBO (SL-TLBO) [42], as well as the compact algorithm counterparts rcGA [7], cDE [8] and cPSO [11] are implemented for comparative study. It should be noted that the function evaluations (FES) is significantly different between TLBO variants and other meta-heuristic algorithms. This issue has been discussed in [15, 16]. Therefore, 2 FES are counted in each iteration for original TLBO, ETLBO, mTLBO and cTLBO, while 3 FES are counted for SL-TLBO due to an additional self-learning phase.

In the algorithm tests, the particle numbers Np of each method are set to 30 and FES are 30,000 for $f1$-$f16$ and 60,000 for $f17$-$f32$. The weight of the wPSO inertially decreases from 0.9-0.4 while the two learning coefficients $C1$ and $C2$ are set as 2.05 respectively. Given the same learning coefficients, cfPSO adopts the constrict factor as 0.729. In the DE algorithm, the mutation rate is 0.7 and the cross rate is 0.9. The parameters of MFO are employed the same as in the original paper [54]. In terms of the elite number in ETLBO, an inertial factor is designed such that the elite number increases with the evolution as $Ne = 1 + Iter/50$ where $Iter$ is the iteration number. The weighting factor in self-learning phase in SL-TLBO is set as $w = 3$ based on [42]. In regards to the compact algorithms, the parameters are referred to those defined in the original papers of rcGA [7], cDE [8] and cPSO [11], except for that the learning coefficients $C1$ and $C2$ of cPSO are set as 2.05.

The totally 12 different algorithms are tested on the 32 benchmarks $f1$-$f32$ respectively, all of which are continuous global optimization problems. In order to make fair comparisons, 30 independent runs are conducted to eliminate the randomness impact. The mean values and average standard deviation values of the algorithms are presented in Table 2, Table 3 and Table 4, in which the novel cTLBO are compared with typical heuristic methods, TLBO variants and compact algorithms respectively. The first number in each grid is the average mean best value and the next number is the standard deviations.

From the Table 2, it could be observed that the new cTLBO outperforms the other four typical meta-heuristic algorithms on 24 out of 32 benchmarks, particularly on high dimensional benchmarks. For some problems such as Schwelfel's Problem 1.2 in $f2$, $f18$, and $f28$, it is however outperformed by other typical methods. It should be noted that the FES selected in this paper is fairly small, due to which some of popular methods have not converged yet, whereas the novel cTLBO has successfully achieved relatively well results. Such behaviors have demonstrated that the novel algorithm has competitive performance. The reason for such good performance could be majorly due to the efficient logic of TLBO, which could be found in Table 3. In the comparison among the TLBO variants, Table 3 shows that though cTLBO show reasonable well performance, it only achieves the best results on 15 out of 32 benchmarks and roughly half of these show the equally results with all or some of counterparts. It is also worth to notice that the cTLBO method outperforms all the others in $f1$, $f9$ and $f25$, which demonstrates the strong search ability for the new approach in some unimodal problems. On the other hand, the majority of benchmark tests on other problems show that the cTLBO cannot achieve the original performance of TLBO methods.

The aforementioned benchmark tests for typical methods and TLBO variants have demonstrated the competitive performance of the proposed cTLBO. Moreover, it is also indispensable to compare the novel algorithm with other compact algorithms and investigate the potential for future corresponding applications. According to Table 4, it is clear that the cTLBO shows dominated performance among 32 benchmark function tests, where only 5 of them are outperformed by the other counterparts. In all the beaten tests $f2$, $f6$, $f12$, $f16$ and $f18$, cTLBO ranks the second, while in the function tests $f1$, $f4$, $f5$, $f9$, $f11$, $f17$, $f20$, $f25$ and $f27$, cTLBO method has achieved global optimum with no standard deviations.

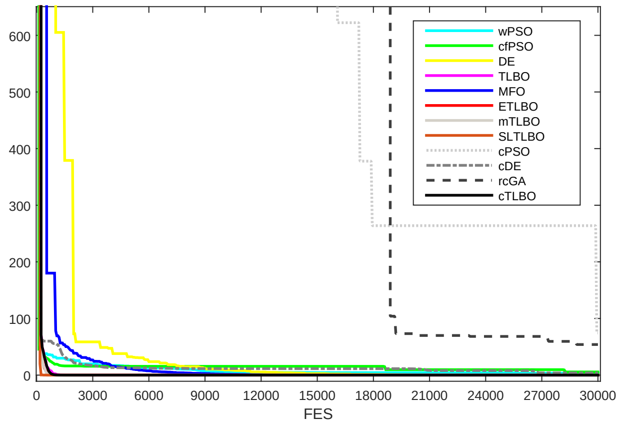

Figure 3: Evolutionary process of algorithms on benchmark $f9$

The typical performance trends of the all 12 algorithms for benchmarks $f9$ and $f19$ are illustrated in Fig 3 and Fig 4. It could be easily observed from the two figures that all the TLBO variants converge faster than the typical methods, generally being able to converge within 1000 FES. This has confirmed the fact that cTLBO method has successfully maintained the remarkable performance of TLBO logic. Among all the five TLBO variants, they are fairly close in terms of the converging speed. Both wPSO and cfPSO methods converge faster than the latest MFO method, however, they both are trapped at local minimum and produce worse results in the final process. The original DE/rand/1/bin method is shown to have better in exploitation performance. It is found to be converge slowly within the first 15000 FES and then speed up afterwards. On the other hand, the compact algorithms show less competitive performance, where both rcGA and cPSO converge fairly slowly. It should also be noted that the method cDE converges faster than other two methods and is only outperformed by cTLBO. In a result, the converging speed comparison of all methods has confirmed that the proposed cTLBO method has better exploration and exploitation capability.

In terms of the memory size reduction, the memory storage of all the employed 12 algorithms are showed in Table 5. It is clear that the original DE needs to maintain Np slots for the optimization process while the memory necessity has to be doubled as $2Np$ for both

Table 1: Test problems adopted in the paper

$f1$	Sphere function from [48] with boundary $[-100, 100]^D$, $D = 30$;
$f2$	Schwefel's problem 1.2 from [48] with boundary $[-100, 100]^D$, $D = 30$;
$f3$	Rosenbrock function from [48] with boundary $[-30, 30]^D$, $D = 30$;
$f4$	Ackley's function from [48] with boundary $[-32, 32]^D$, $D = 30$;
$f5$	Griewank function from [48] with boundary $[-600, 600]^D$, $D = 30$;
$f6$	Rastrigin function from [48] with boundary $[-5.12, 5.12]^D$, $D = 30$;
$f7$	Step function [48] with boundary $[-100, 100]^D$, $D = 30$;
$f8$	Schwefel's problem 2.21 from [48] with boundary $[-100, 100]^D$, $D = 30$;
$f9$	Schwefel's problem 2.22 from [48] with boundary $[-10, 10]^D$, $D = 30$;
$f10$	Quartic function from [48] with boundary $[-1.28, 1.28]^D$, $D = 30$;
$f11$	Shifted Sphere function from [49] with boundary $[-100, 100]^D$, $D = 30$, $f_{bias} = -450$;
$f12$	Shifted Schwelfel's problem 1.2 from [49] with boundary $[-100, 100]^D$, $D = 30$, $f_{bias} = -450$;
$f13$	Shifted Rosenbrock function from [49] with boundary $[-30, 30]^D$, $D = 30$, $f_{bias} = 390$;
$f14$	Shifted Ackley's function from [50] with boundary $[-32, 32]^D$, $D = 30$, $f_{bias} = -450$;
$f15$	Shifted Griewank function from [50] with with boundary $[-600, 600]^D$, $D = 30$, $f_{bias} = -180$;
$f16$	Shifted Rastrigin function from [49] with with boundary $[-5, 5]^D$, $D = 30$, $f_{bias} = -330$;
$f17$	Sphere function from [48] with boundary $[-100, 100]^D$, $D = 100$;
$f18$	Schwefel's problem 1.2 from [48] with boundary $[-100, 100]^D$, $D = 100$;
$f19$	Rosenbrock function from [48] with boundary $[-30, 30]^D$, $D = 100$;
$f20$	Ackley's function from [48] with boundary $[-32, 32]^D$, $D = 100$;
$f21$	Griewank function from [48] with boundary $[-600, 600]^D$, $D = 100$;
$f22$	Rastrigin function from [48] with boundary $[-5.12, 5.12]^D$, $D = 100$;
$f23$	Step function [48] with boundary $[-100, 100]^D$, $D = 100$;
$f24$	Schwefel's problem 2.21 from [48] with boundary $[-100, 100]^D$, $D = 100$;
$f25$	Schwefel's problem 2.22 from [48] with boundary $[-10, 10]^D$, $D = 100$;
$f26$	Quartic function from [48] with boundary $[-1.28, 1.28]^D$, $D = 100$;
$f27$	Shifted Sphere function from [49] with boundary $[-100, 100]^D$, $D = 100$, $f_{bias} = -450$;
$f28$	Shifted Schwelfel's problem 1.2 from [49] with boundary $[-100, 100]^D$, $D = 100$, $f_{bias} = -450$;
$f29$	Shifted Rosenbrock function from [49] with boundary $[-30, 30]^D$, $D = 100$, $f_{bias} = 390$;
$f30$	Shifted Ackley's function from [50] with boundary $[-32, 32]^D$, $D = 100$, $f_{bias} = -450$;
$f31$	Shifted Griewank function from [50] with with boundary $[-600, 600]^D$, $D = 100$, $f_{bias} = -180$;
$f32$	Shifted Rastrigin function from [49] with with boundary $[-5, 5]^D$, $D = 100$, $f_{bias} = -330$;

PSO and TLBO variants as well as the MFO method. The compact algorithms including rcGA, cPSO, cDE and cTLBO needs only 4 or 5 memory slots, where cTLBO only requires the memory storage for 3 new student particles, 1 teacher particle and 1 buffer particle slot in the algorithm process. Therefore, cTLBO has reduced over 90% memory requirement from the original TLBO method if the particle number Np is 30. This is a significant improvement for implementing the optimization methods on memory limited embedded systems. In regards to the computational cost, we have normalized 30 dimension and 100 dimension tests within a single index and utilized DE method as the benchmark time. It could be found that PSO variants and MFO both require over 1.7 folds executive time more than DE, while TLBO variants need roughly half executive time more than DE. Due to that all the particles are generated from the sampling scheme, compact algorithms inevitably require more executive time than typical meta-heuristic algorithms. The proposed cTLBO method ranks in a medium position, requiring over 3.5 fold exective time more than DE, which is slightly longer than rcGA and cPSO and shorter than cDE. Note

Table 2: The comparison of cTLBO against typical optimization methods

TP	wPSO	cfPSO	DE	MFO	cTLBO	Rank
$f1$	2.337e02 ± 5.562e02	1.889e03 ± 4.361e03	8.701e-03 ± 04.611e-02	1.458e-04 ± 1.601e-03	0 ± 0	1
$f2$	5.854e-04 ± 8.901e-03	7.711e-03 ± 1.687e-01	7.743e-02 ± **7**.420e-01	6.799e-27 ± 7.136e-26	2.623e-02 ± 2.212e-01	4
$f3$	1.212e04 ± 5.307e04	3.690e05 ± 1.371e06	5.058e01 ± 1.828e02	1.637e02 ± 9.099e02	4.182 e02 ± 1.396e00	1
$f4$	6.914e00 ± 6.644e00	1.001e01 ± 8.077e00	3.372e-02 ± 8.312e-02	1.085 e00 ± 6.389e00	8.882e-16 ± 0	1
$f5$	3.063e00 ± 6.781e00	1.845e01 ± 4.029e01	3.051e-02 ± 2.220e-01	1.642e-02 ± 1.065e-01	0 ± 0	1
$f6$	**7**.029e01 ± 8.891e01	1.097e02 ± 1.071e02	1.673e02 ± 2.184e02	6.600e02 ± 7.743e02	1.097e02 ± 6.244e01	2
$f7$	2.152e02 ± 5.061e02	1.690e03 ± 3.739e03	7.753e-02 ± 2.871e-02	1.357e-04 ± 1.121e-03	3.146e00 ± 1.489e00	3
$f8$	1.953e01 ± 2.155e01	2.331e01 ± 2.843e01	8.4775e00 ± 1.776e01	4.226e01 ± 4.065e01	1.467e-15 ± 6.560e-15	1
$f9$	7.502e00 ± 1.462e01	1.551e01 ± 2.856e01	6.795e-11 ± 2.240e-01	2.314e-04 ± 2.308e-03	0 ± 0	1
$f10$	1.548e01 ± 8.918e00	1.595e01 ± 7.993e00	1.336e01 ± 6.350e00	1.798e00 ± 5.459e00	8.822e00 ± 9.628e-01	2
$f11$	-2.348e02 ± 4.900e02	1.189e03 ± 4.112e03	-4.499e02 ± 3.312e-02	-4.499e02 ±5.484e-04	-4.500e02 ± 0	1
$f12$	-4.499e02 ± 9.640e-02	-4.499e02 ± 4.127e-01	-4.499e02 ± 6.871e-01	-4.500e02 ± 0	-4.499e02 ± 2.868e-02	5
$f13$	1.752e04 ± 5.308e04	3.723e05 ± 1.371e06	4.303e02 ± 1.828e02	5.625e02 ± 9.099e02	2.846e01 ± 1.450e00	1
$f14$	-4.427e02 ± 7.148e00	-4.397e02 ± 6.967e00	-4.499e02 ± 1.107e-01	-4.481e02 ± 1.025e01	-4.500e02 ± 1.271e-13	1
$f15$	-1.770e02 ± 5.454e01	-1.654e02 ± 3.360e01	-1.799e02 ± 4.676e-01	-1.799e02 ± 8.674e-02	-1.799e02 ± 1.233e-02	1
$f16$	-2.599e02 ± 9.519e01	-2.162e02 ± 1.274e02	-1.518e02 ± 1.620e02	-2.688e02 ± 9.165e02	-2.304e-02 ± 4.707e01	3
$f17$	8.389e03 ± 2.999e03	2.344e04 ± 3.891e03	6.573e01 ± 1.401e02	4.364e02 ± 1.092e03	0 ± 0	1
$f18$	5.921e-02 ± 1.468e-01	1.059e-02 ± 4.652e-02	1.890e-01 ± 3.726e-01	1.959 e-25 ± 4.589e-25	1.937e-01 ± 5.676e-01	5
$f19$	3.972e06 ± 2.675e06	1.146e07 ± 4.446e06	1.300e04 ± 2.307e04	6.340e05 ± 1.453e06	9.822e01 ± 1.394e00	1
$f20$	1.335e01 ± 2.011e00	1.443e01 ± 1.376e00	3.019e00 ± 1.022e00	9.320e00 ± 4328e00	8.882e-16 ± 0	1
$f21$	-1.079e02 ± 8.586e01	3.344e01 ± 1.728e02	-1.783e02 ± 3.935e00	-1.740e02 ± 3.008e01	-1.799e02 ± 3.231e-02	1
$f22$	1.763e02 ± 2.685e02	3.285e02 ± 4.761e02	1.989e02±5.534 e02	-4.172e01 ± 1.860e02	-1.277e02 ± 1.373e03	1
$f23$	9.053e03 ± 5.074e03	2.126e04 ± 5.748e03	1.455e02 ± 4.337e02	3.528e02 ± 1.060e03	1.524e01 ± 2.856e00	1
$f24$	3.977e01 ± 5.806e00	4.064e01 ± 6.631e00	3.829e01 ± 4.131e00	7.454e01 ± 7.855e00	8.677e-15 ± 2.378e-14	1
$f25$	1.430e03 ± 2.460e02	1.317e03 ± 2.826e02	8.673e01 ± 1.016e02	8.252e01 ± 5.827e01	0 ± 0	1
$f26$	5.690e01 ± 9.592e00	7.038e01 ± 1.264e01	4.996e01 ± 6.345e00	7.696e01 ± 2.028e01	3.770e01 ± 1.264e00	1
$f27$	8.173e03 ± 2.441e03	1.895e04 ± 1.228e04	-3.609e02 ± 1.607e02	4.474e01 ± 1.199e03	-4.500e02 ± 0	1
$f28$	-4.499e02± 7.472e-02	-4.499e02± 1.167e-01	-4.495e02± 1.103e00	-4.500e02± 0	-4.498e02 ± 9.287e-01	4
$f29$	3.075e06 ± 2.732e06	1.654e07 ± 9.318e06	1.028e04 ± 9.046e03	9.133e04 ± 1.580e05	4.884e02± 7.055e-01	1
$f30$	-4.371e02 ± 1.668e00	-4.367e02 ± 6.205e00	-4.470e02 ± 2.314e00	-4.402e02 ± 4.503e00	-4.500± 1.271e-13	1
$f31$	-1.079e02 ± 8.586e01	3.344e01 ± 1.728e02	-1.783e02 ± 3.935e00	-1.740e02 ± 3.008e01	-1.800e02 ± 3.231e-02	1
$f32$	1.763e02 ± 2.685e02	3.285e02 ± 4.761e02	1.989e02 ± 5.534e02	-4.173e01 ± 1.860e02	-1.277e02 ± 1.373e03	1

Table 3: The comparison of cTLBO against Other TLBO variants

TP	TLBO	ETLBO	mTLBO	SLTLBO	cTLBO	Rank
$f1$	1.247e-125 ± 3.095e-124	1.337e-170 ± 0	7.125-239 ± 0	1.376e-290 ± 0	0 ± 0	1
$f2$	0 ± 0	4.207e-31 ± 1.241e-29	3.717e-33 ± 1.091e-31	1.039e-208 ± 0	2.623e-02 ± 2.212e-01	5
$f3$	2.893e01 ± 1.986e-01	2.895e01 ± 1.562e-01	2.895e01 ± 1.436e-01	2891.e01 ± 1.636e-01	4.182 e02 ± 1.396e00	5
$f4$	4.086e-15 ± 5.838e-15	3.494e-15 ± 8.605e-15	3.494e-15 ± 8.605e-15	8.882 e-16 ± 0	8.882e-16 ± 0	1
$f5$	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	1
$f6$	0 ± 0	0 ± 0	0 ± 0	0 ± 0	1.097e02 ± 6.244e01	5
$f7$	5.460e00 ± 4.072e00	6.213e00 ± 3.111e00	5.931e00 ± 4.136e00	4.986e00 ± 4.303e00	3.146e00 ± 1.489e00	1
$f8$	4.572e-61 ± 6.343e-60	1.060e-83 ± 9.934e-83	2.209e-117 ± 4.566e-116	3.626e-147 ± 7.247e-146	1.467e-15 ± 6.560e-15	5
$f9$	1.593e-63 ± 1.109e-62	2.490e-85 ± 3.211e-84	2.688e-120 ± 4.562e-119	5.843e-148 ± 1.085e-146	0 ± 0	1
$f10$	8.981e00 ± 2.915e00	8.938e00 ± 2.490e00	9.010e00 ± 2.338e00	9.064e00 ± 2.612e00	8.822e00 ± 9.628e-01	1
$f11$	-4.500e02 ± 0	-4.500e02 ± 0	-4.500e02 ± 0	-4.500e02 ± 0	-4.500e02 ± 0	1
$f12$	-4.500e02 ± 0	-4.500e02 ± 0	-4.500e02 ± 0	-4.500e02 ± 0	-4.499e02 ± 2.868e-02	5
$f13$	4.189e02 ± 1.768e-01	4.189e02 ± 1.855e-01	4.189e02 ± 1.569e-01	4.189e02 ± 2.100e-01	2.846e01 ± 1.450e00	1
$f14$	-4.500e02 ± 2.127e-13	-4.500e02 ± 1.392e-13	-4.500e02 ± 1.504e-13	-4.500e02 ± 0	-4.500e02 ± 1.271e-13	2
$f15$	-1.80e02 ± 0	-1.80e02 ± 0	-1.80e02 ± 0	-1.80e02 ± 0	-1.799e02 ± 1.233e-02	5
$f16$	-3.300e02 ± 0	-3.300e02 ± 0	-3.300e02 ± 0	-3.300e02 ± 0	-2.304e-02 ± 4.707e01	5
$f17$	5.771e-258 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	1
$f18$	9.861e-33 ± 4.401e-32	3.852e-35 ± 1.723e-34	8.875e-32 ± 3.969e-31	0 ± 0	1.937e-01 ± 5.676e-01	5
$f19$	9.893e01 ± 7.222e-02	9.891e01 ± 8.790e-02	9.895e01 ± 2.268e-02	9.890e01 ± 4.243e-02	9.822e01 ± 1.394e00	1
$f20$	3.730e-15 ± 3.178e-15	3.020e-15 ± 3.892e-15	3.730e-15 ± 3.178e-15	8.882e-16 ± 0	8.882e-16 ± 0	1
$f21$	-1.800e02 ± 0	-1.800e02 ± 0	-1.800e02 ± 0	-1.800e02 ± 0	-1.799e02 ± 3.231e-02	5
$f22$	-3.300e02 ± 0	-3.300e02 ± 0	-3.300e02 ± 0	-3.300e02 ± 0	-1.277e02 ± 1.373e03	5
$f23$	2.318e01 ± 1.142e00	2.371e01 ± 1.415e00	2.324e01 ± 1.670e00	2.258e01 ± 1.113e00	1.524e01 ± 2.856e00	1
$f24$	4.834e-126 ± 1376e-125	2.828e-171 ± 0	8.890e-240 ± 0	7.977e-299 ± 0	8.677e-15 ± 2.378e-14	5
$f25$	2.531e-128 ± 5.933e-128	5.523e-172 ± 0	1.149e-240 ± 0	2.884e-299 ± 0	0 ± 0	1
$f26$	3.837e01 ± 1.943e00	3.832e01 ± 1.406e00	3.868e01 ± 1.139e00	3.824e01 ± 1.239e01	3.770e01 ± 1.264e00	1
$f27$	-4.500e02 ± 0	-4.500e02 ± 0	-4.500e02 ± 0	-4.500e02 ± 0	-4.500e02 ± 0	1
$f28$	-4.500e02 ± 0	-4.500e02 ± 0	-4.500e02 ± 0	-4.500e02 ± 0	-4.498e02 ± 9.287e-01	5
$f29$	4.889e02 ± 1.098e-01	4.890e02 ± 5.026e-02	4.889e02 ± 5.132e-02	4.889e02 ± 6.301e-02	4.884e02± 7.055e-01	5
$f30$	-4.500e02 ± 9.846e-14	-4.500 e02 ± 8.039e-14	-4.500 e02 ± 0	-4.500 e02 ± 0	-4.500± 1.271e-13	5
$f31$	-1.800e02 ± 0	-1.800e02 ± 0	-1.800e02 ± 0	-1.800e02 ± 0	-1.800e02 ± 3.231e-02	5
$f32$	-3.300e02 ± 0	-3.300e02 ± 0	-3.300e02 ± 0	-3.300e02 ± 0	-1.277e02 ± 1.373e03	5

Table 4: The comparison of cTLBO against Other compact alrotighms

TP	rcGA	cDE	cPSO	cTLBO	Rank
$f1$	2.423e04 ± 3.321e03	5.112e01 ± 2.604e02	2.013e04 ± 5.818e04	0 ± 0	1
$f2$	2.375e-01 ± 2.011e00	1.802e-02 ± 4.441e-01	5.400e06 ± 1.931e05	2.623e-02 ± 2.212e-01	2
$f3$	3.371e07 ± 3.961e07	9.408e04± 3.114e05	1.816e07 ± 3.961e07	4.182 e02 ± 1.396e00	1
$f4$	1.815e01 ± 7.816e-01	1.031e01 ± 4.060e00	1.626 e01 ± 5.332e00	8.882e-16 ± 0	1
$f5$	1.756e02 ± 6.088e01	1.448e00 ± 1.096e00	2.161e02 ±1.443 e02	0 ± 0	1
$f6$	2.887e02 ± 7.141e02	7.740e01± 1.861e01	1.937e02 ± 2.503e02	1.097e02 ± 6.244e01	2
$f7$	2.111e04 ± 8.915e03	3.412e01 ± 6.920e01	1.510e04 ± 2.918e04	3.146e00 ± 1.489e00	1
$f8$	6.831e01 ± 4.283e00	4.204e01± 8.772e00	5.636e01 ± 3.745e01	1.467e-15 ± 6.560e-15	1
$f9$	1.247e03 ± 3.037e02	3.593e00 ± 6.965e01	1.289e03 ± 5.511e03	0 ± 0	1
$f10$	3.299e01 ± 1.403e01	1.862e01 ± 4.037e00	2.885e01 ± 1.698e01	8.822e00 ± 9.628e-01	1
$f11$	2.364e03 ± 4.438e03	-2.646e02± 2.902 e02	1.821e04 ± 2.488e04	-4.500e02 ± 0	1
$f12$	-4.498e02 ± 5.071e-01	-4.500e02 ± 0	6.040e03 ± 2.903e04	-4.499e02 ± 2.868e-02	2
$f13$	3.621e07 ± 4.591e07	2.185e05± 5.383e05	2.446e07 ± 3.675e07	2.846e01 ± 1.450e00	1
$f14$	-4.314e02 ± 1.258e00	-4.403e02± 4.873e00	-4.330e02 ± 9.824e00	-4.500e02 ± 1.271e-13	1
$f15$	2.271e01 ± 6.438e01	-1.779e02± 3.724e00	6.419e01 ± 1.363e02	-1.799e02 ± 1.233e-02	1
$f16$	-3.271e01 ± 4.753e01	-2.667e02± 1.573e01	-9.349e01 ± 2.698e02	-2.304e-02 ± 4.707e01	2
$f17$	9.500e04 ± 2.654e04	3.225e04 ± 1.634e04	1.335e05 ± 1.865e05	0 ± 0	1
$f18$	1.854e-01 ± 3.580e-01	1.073e-01± 4.092e-01	4.578e-01 ± 1.794e00	1.937e-01 ± 5.676e-01	2
$f19$	2.330e08 ± 7.062e07	6.096e07± 6.977e07	1.232e08 ± 3.546e08	9.822e01 ± 1.394e00	1
$f20$	1.905e01 ± 5.951e-01	1.809e01 ± 9.075e-01	1.670 e01 ± 1.443e01	8.882e-16 ± 0	1
$f21$	7.079e02 ± 5.815e02	1.690e02± 3.370e02	6.134e02 ± 3.367e03	-1.799e02 ± 3.231e-02	1
$f22$	7.802e02 ± 2.637e02	2.982e02 ± 3.590e02	7.626e02 ± 1.593e03	-1.277e02 ± 1.373e03	1
$f23$	1.014e05 ± 1.064e04	4.300e04± 2.069e04	7.146e04 ± 1.578e05	1.524e01 ± 2.856e00	1
$f24$	8.443e01 ± 6.303e00	7.456e01± 9.660e00	6.129e01 ± 7.187e01	8.677e-15 ± 2.378e-14	1
$f25$	4.759e113 ± 2.128e114	1.587e03± 2.413e02	4.008e55 ± 1.793e56	0 ± 0	1
$f26$	3.741e02 ± 2.073e02	3.204e02± 5.659e01	5.697e02 ± 8.174e02	3.770e01 ± 1.264e00	1
$f27$	9.318e04 ± 2.164e04	4.342e04 ± 1.560e04	9.489e-04 ± 1.315e05	-4.500e02 ± 0	1
$f28$	-4.428e02 ± 2.273e01	-4.494e02± 1.322e00	-4.497e02 ± 7.466e-01	-4.498e02 ± 9.287e-01	1
$f29$	2.442e08 ± 6.148e07	4.350e07 ± 2.354e07	5.151e08 ± 6.268e08	4.884e02± 7.055e-01	1
$f30$	-4.307e02 ± 9.504e-01	-4.322e02± 8.504e-01	-4.307e02± 2.045e00	-4.500± 1.271e-13	1
$f31$	7.079e02 ± 5.815e02	1.690e02±3.370e02	6.134e02 ± 3.370e03	-1.800e02 ± 3.231e-02	1
$f32$	7.802e02 ± 2.637e02	2.982e02± 3.590e02	7.626e02 ± 1.593e03	-1.277e02 ± 1.373e03	1

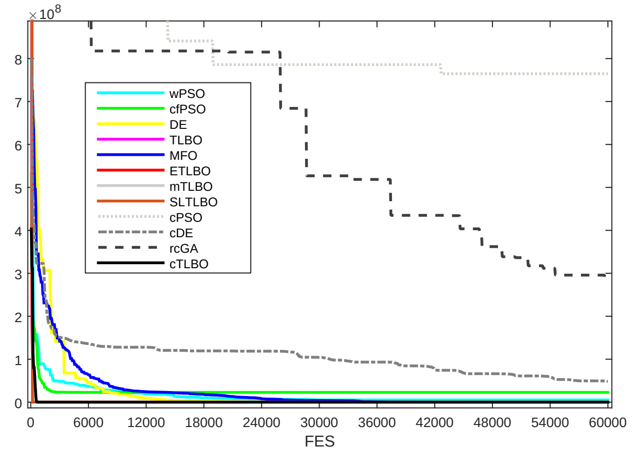

Figure 4: Evolutionary process of algorithms on benchmark $f19$

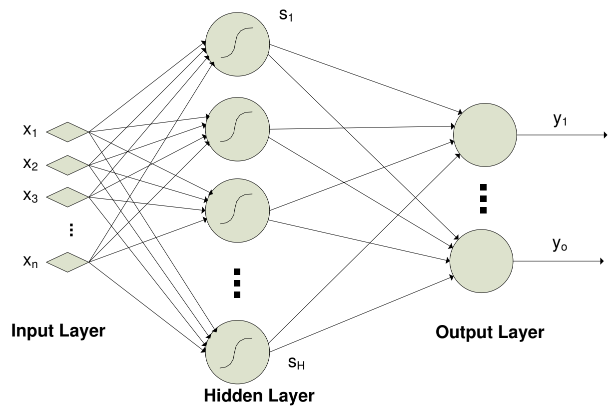

Figure 5: Feedforward neural network structure

that according to the previous study [11], the relative time scale is strongly determined by the dimension and problems. We therefore could conclude that the novel cTLBO method does not require more execution time or memory spaces than normal compact algorithms. It is worth to note that in typical practical implementations [7], the optimization task is successfully solved within micro second scale and faster than binary converted based algorithm. The computational time for compact algorithms are acceptable for on-line design of controller parameter training.

Through comprehensive benchmark tests, the novel cTLBO method has demonstrated competitive performance. On one hand, compared with other compact algorithms, the new algorithm improves the overall exploration and exploitation ability without adding any storage burdens. On the other hand, compared with conventional non-compact algorithms, the new algorithm significantly reduces the memory storage resources and maintains the computational performance. It is therefore a promising tool for compact optimization tasks in particular for energy and storage limited applications. On the other hand, neural networks are frequently adopted approaches in path planning and model prediction for compact independent systems, while the key task to train neural network is the determination of non-linear parameters in the basis functions. In the next section, we adopt the novel cTLBO methods to train feedforward and radial basis function neural networks and investigate the training and validation results.

6. Neural Network Training Tests

In this paper, we adopt two typical types of neural networks including FNN and RBF neural network to illustrate the performance of proposed cTLBO in training the non-linear NN models. Both of the models are feed forward neural networks with three layers, whereas the model structures and non-linear transfer functions differentiate them.

6.1. Feedforward Neural Network Training

Feedforward neural network is one of most popular neural network structures due to the simple typology and strong approximation ability. The structure of FNN [56] is shown in Fig.5, where a three layers FNN is adopted including an input layer, a hidden layer and an output layer. Equations (6)-(10) denote the relationship of input and output variables. The well adopted sigmoid function is employed as the activation function in hidden node as shown in (6), where n, h and m denote the numbers of input, hidden and output nodes respectively. The weights between the inputs x_i and hidden nodes are denoted as w_{ih}, and θ_j is the threshold of hidden nodes. Note that the output of the hidden layer, e.g. the input of output layer s_j, is calculated as $s_j = \sum_{i=1}^{n} w_{ih} \cdot x_i - \theta_j$.

$$f(s_j) = 1/(1 + exp(-(\sum_{i=1}^{n} w_{ih} \cdot x_i - \theta_j))), j = 1, 2, ..., H, \tag{6}$$

where the activation function output from hidden nodes is denoted as $f(s_j)$. Consequently, the output variables y_k are denoted as below,

$$y_k = \sum_{j=1}^{H} w_{ho} \cdot f(s_j) - \theta_k, k = 1, 2, ..., O, \tag{7}$$

where H is the number of hidden nodes. Moreover, θ_k denotes the threshold of output and w_{ho} represents the weights between the hidden nodes and output nodes. In this regard, the error Err_k between the actual output and the desired output of the k^{th} is presented as below,

$$Err_k = \sum_{i=1}^{O} (y_i^k - C_i^k)^2 \tag{8}$$

Table 5: The comparison memory slots and executive time for different algorithms

Algorithm	Structure	Particles in Memory	Memory slots	Executive time scale
DE	DE based	Np particles	Np	1.00
wPSO	PSO based	Np particles, Np velocity	$2Np$	1.71
cfPSO	PSO based	Np particles, Np velocity	$2Np$	1.71
MFO	MFO based	Np moths, Np flames	$2Np$	1.74
TLBO	TLBO based	Np students, Np new students	$2Np$	1.53
ETLBO	TLBO based	Np students, Np new students, elites	$2Np$+elites	1.58
mTLBO	TLBO based	Np students, Np new students	$2Np$	1.57
SLTLBO	TLBO based	Np students, Np new students	$2Np$	1.54
rcGA	GA based	1 sample, persistent elites	4	3.368
cDE	DE based	3 samples, 1 crossover backup	4	4.125
cPSO	PSO based	2 samples, 2 best particles	5	3.202
cTLBO	TLBO based	3 students, 1 teacher, 1 deviation	5	3.596

where C_i^k is the desired output. To accumulate the sectional error Err_k, a final accounted error Err is shown as in (9).

$$Err = \sum_{k=1}^{q} Err_k/(q \cdot O) \tag{9}$$

Finally, the fitness function for the FNN training task is denoted as in (10)

$$min\ fitness(X_i) = Err(X_i) \tag{10}$$

In meta-heuristic optimization training process, the variables are encoded in a particle and updated in the evolutionary process. The encoding scheme in this paper employs the method in [56]. Assume an 1-5-1 structure FNN, the variable coding details is shown in equation (11).

$$\begin{aligned} particle(i) = [&w_{12}, w_{13}, w_{14}, w_{15}, w_{16}, \\ &w_{27}, w_{37}, w_{47}, w_{57}, w_{67}, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7] \end{aligned} \tag{11}$$

It is worth to note that the input node is numbered as 1, followed by the hidden nodes numbered as node 2-6 and output node as number 7. The weights w_{12} to w_{16} belong to the w_{ih} while w_{27} to w_{67} represent the weights w_{ho}.

In order to test the performance of the proposed cTLBO method on FNN training, we adopt a non-linear function $f = sin(4x)$ as the approximation target and utilize regular structure wPSO, TLBO and as compact counterparts rcGA, cPSO and cDE to compare the performance. All the algorithm specific parameter configurations are the same with those in benchmark tests as in section 5. To fairly compare the algorithm performance, a consistent FES 10,000 is adopted in the training process, and the initial values of the weight variables are randomly generated within (0,1). The upper and lower boundaries are set as (-10,10), and the input section is selected as (-4π, 4π) with 0.05 intervals. We adopt 70% of the input data for training and 30% data for validation, and 30 different tests are conducted to eliminate the randomness. The mean and standard deviation values of training and validation results are shown in Table. 6.

We employ 3 to 7 hidden nodes for the training comparisons. It could be observed from the Table. 6 that the proposed cTLBO method achieves the best training and validation results in the majority of scenarios. Among the six competitors, wPSO and TLBO see similar performances, where TLBO outperforms wPSO in 4 and 5 hidden nodes scenarios and is slightly outperformed in 3 and 7 hidden nodes tests. Comparing with all the other compact based algorithms, the cTLBO significantly outperforms all the counterparts including rcGA, cPSO and cDE. It is worth to note that cDE sees relatively inferior performance probably due to the improper algorithm specific parameter settings such as less tuned crossover and mutation rates, which also shows the advantage of the freedom of parameter tuning for proposed cTLBO algorithm.

6.2. Radial Basis Function Neural Network Training

The sigmoid based FNN neural network may not be sufficient to cover the strong non-linear behaviours of specific datasets. To further investigate the training performance of cTLBO, RBF neural network is also employed in this section. Other than using basic sigmoid function, the activation functions in RBF are equipped with the Gaussian functions. The RBF neural network is also a typical feed forward neural network including three layers, namely input layer, hidden layer and output layer respectively as shown in Fig.6. Consider a multi-input and single-output (MISO) RBF network, the mathematical output is formulated as

$$y(t) = \sum_{i=1}^{n} w_i \cdot \phi_i(X) \tag{12}$$

where $y(t)$ is the output at sample time t, and w_i denotes the linear output weight for the i^{th} node in the hidden layer. The radial basis function ϕ_i of input vector X is chosen as Gaussian function defined below:

$$\phi_i(X) = exp(-\frac{1}{2\sigma_i^2}\|X - c_i\|^2), i = 1, 2, ..., n \tag{13}$$

Table 6: Training and validation results of different algorithms for FNN in approximating $f = sin(4x)$

Hidden Node	Algorithm	Training Err	Training STD	Validation Err	Validation STD
3	wPSO-FNN	3.707E-02	1.769E-04	5.318E-02	9.514E-04
	TLBO-FNN	3.711E-02	1.659E-04	5.249E-02	1.329E-03
	rcGA-FNN	3.377E-02	7.211E-04	5.533E-02	6.911E-03
	cPSO-FNN	3.368E-02	1.871E-03	5.575E-02	5.667E-03
	cDE-FNN	3.459E-01	4.149E-01	3.045E-01	4.846E-01
	cTLBO-FNN	3.311E-02	1.641E-04	5.119E-02	7.855E-04
4	wPSO-FNN	3.310E-02	8.329E-04	5.269E-02	1.503E-03
	TLBO-FNN	3.292E-02	6.099E-04	5.300E-02	2.162E-03
	rcGA-FNN	3.299E-02	1.019E-03	5.332E-02	6.526E-03
	cPSO-FNN	3.297E-02	1.989E-03	5.278E-02	6.235E-03
	cDE-FNN	2.778E-01	4.064E-01	4.163E-01	6.579E-01
	cTLBO-FNN	3.188E-02	5.622E-04	5.104E-02	1.063E-03
5	wPSO-FNN	3.015E-02	7.240E-05	5.226E-02	6.121E-04
	TLBO-FNN	3.012E-02	1.482E-05	5.217E-02	9.231E-05
	rcGA-FNN	3.251E-02	7.818E-04	5.758E-02	4.533E-02
	cPSO-FNN	6.388E-02	2.973E-01	1.044E-01	4.438E-01
	cDE-FNN	2.787E-01	4.237E-01	2.578E-01	6.922E-01
	cTLBO-FNN	2.917E-02	1.966E-05	4.801E-02	1.201E-04
6	wPSO-FNN	2.980E-02	1.613E-05	5.362E-02	1.278E-02
	TLBO-FNN	2.980E-02	1.359E-05	5.230E-02	2.905E-04
	rcGA-FNN	3.182E-02	9.857E-04	6.041E-02	6.275E-02
	cPSO-FNN	3.191E-02	1.113E-03	5.580E-02	4.741E-02
	cDE-FNN	2.749E-01	4.195E-01	1.699E-01	2.943E-01
	cTLBO-FNN	2.808E-02	1.239E-05	5.129E-02	3.670E-04
7	wPSO-FNN	2.713E-02	1.913E-05	5.292E-02	8.612E-03
	TLBO-FNN	2.716E-02	5.774E-05	5.130E-02	1.512E-02
	rcGA-FNN	3.154E-02	1.104E-03	6.124E-02	6.151E-02
	cPSO-FNN	3.136E-02	1.602E-03	7.784E-02	2.409E-01
	cDE-FNN	2.789E-01	4.224E-01	3.146E-01	7.430E-01
	cTLBO-FNN	2.554E-02	3.473E-05	3.970E-02	1.156E-03

where σ_i is the Gaussian distributed width and c_i denotes the Gaussian center of the i^{th} hidden node. n denotes the total number of hidden node.

In order to properly train the RBF network, the root mean squared error (RMSE) of the NN prediction is employed to be the objective function in the training and it is denoted as follows:

$$min\ f = \sqrt{\frac{1}{N_m} \cdot \sum_{i=1}^{N_m} (\hat{y} - y_m)^2} \tag{14}$$

where $\hat{y}$ is the prediction value and y_m is the measured data set. Note that the formulation and all the parameters should be pre-set or determined before calculating the model output $\hat{y}$, which is denoted in equation

$$\hat{y}(t) = \sum_{i=1}^{n_h} w_i \cdot exp(-\frac{1}{2\sigma_i^2}\|X - c_i\|^2), i = 1, 2, ..., n. \tag{15}$$

We utilize heuristic based optimization methods to determine c_i, σ_i and w_i in the RBF-NN model to approximate a non-linear system. In regards to the encoding scheme for

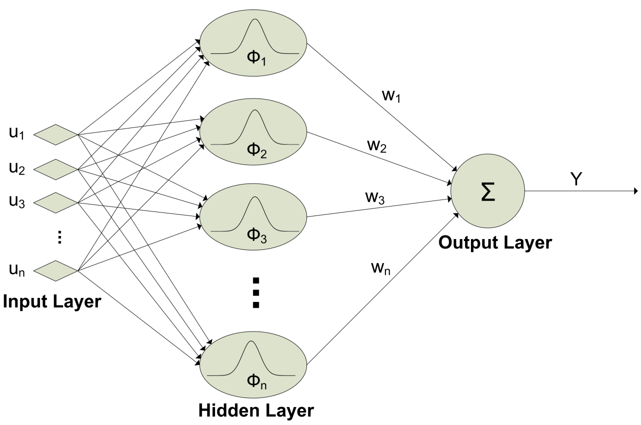

Figure 6: RBF network structure

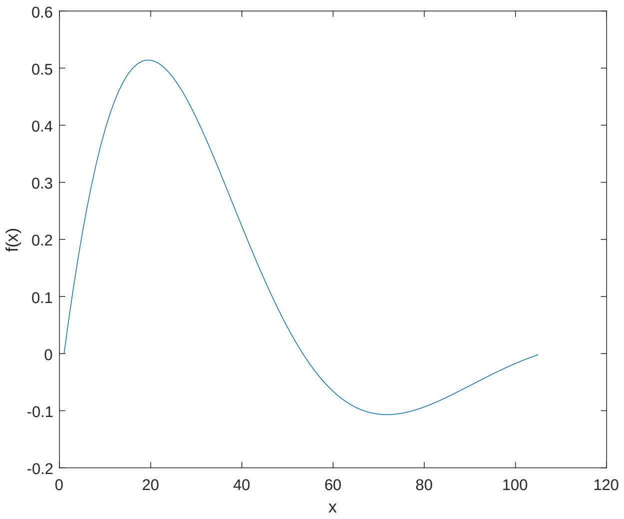

Figure 7: Data distribution of test system 1

RBF optimization variables, we assume an 2-5-1 structure with 2 input nodes, 5 hidden nodes and 1 output node for illustration. Each hidden node has a set of c_i, σ_i and w_i where the dimension of mean vector c_i should be consistent with the input number. The encoding scheme is denoted as below equation (16). Again we assume that the input nodes are number 1 and 2, and hidden nodes are 3-7 followed by that node 8 denotes the output node.

$$\begin{aligned} particle(i) = [&c_{13}, c_{23}, \sigma_3, w_{38}, c_{14}, c_{24}, \sigma_4, w_{48}, \\ &c_{15}, c_{25}, \sigma_5, w_{58}, c_{16}, c_{26}, \sigma_6, w_{68}, c_{17}, c_{27}, \sigma_7, w_{78}] \end{aligned} \tag{16}$$

In this case study, we select two typical non-linear systems for algorithm training: a smooth system and a highly non-linear system respectively. Training system 1 is a smooth non-linear system $f = sin(2x)e^{-x}$ from [57], which is shown in Fig. 7. In the training process for test system 1, we adopt the dataset $(0, \pi)$ with 0.03 interval as the model input. 60% dataset are employed as the training data while 40% data are adopted for validation. To compare the impact of the hidden nodes number on the model training performance, an 1-n-1 RBF model structure with n=3 to 9 nodes are tested respectively, where $x(t)$ and $f(x)$ are the input and output vectors. The FES are also set as 10,000, and 10 independent runs are conducted for all the six algorithms again including wPSO, TLBO, rcGA, cPSO, cDE and proposed cTLBO. All the initial values of the variables are among (0,1) and the particle updating is free of any boundary settings. The best obtained results among the 10 tests are listed in Fig. 8, where the 3-9 hidden nodes results are shown respectively.

It could be observed from the Fig. 8 that the proposed cTLBO outperforms all the counterparts in the training scenarios from 3 to 9 hidden nodes. The best training results could be found at the 3 hidden node scenario, with the least RSME is less than 9.4×10^{-4} obtained by cTLBO. Moreover, the other algorithms results are not stable and cDE again performs the worst. It could be generally concluded that for test system 1, with the increase of hidden nodes, the training error increases. Therefore, it is sufficient for a small number hidden node RBF neural network structure to model the smooth non-linear system.

In addition to the training system 1, a more challenging task training system 2 is also employed for further case study. It is a highly non-linear system original from [58, 59] shown as below:

$$\begin{aligned} y(t) =& 0.5y(t-1) + 0.8u(t-2) + u(t-1)^2 \\ &- 0.05y(t-2)^2 + 0.5 + \xi(t), \\ &\xi(\cdot) \sim N(0, 0.05), \end{aligned} \tag{17}$$

where t, u and y denotes time series, system input and output. The adopted system is a non-linear autoregressive exogenous (NARX) model associated with a Gaussian system noise $N(0, 0.05)$. By simulating the input u with uniform distributed range [-1,1], 500 data are sampled as shown in Fig. 9, where 350 of them are used for model training and 150 data samples are used as model validation. To compare the algorithm performance, 5 algorithms including wPSO, TLBO, and the other three compact algorithms e.g. rcGA, cPSO and cDE are employed to compare with the proposed cTLBO. All the parameters settings of the algorithms are the same with aforementioned benchmark test. The number of particles is set as 30 and FES is adopted as 3,000, while 30 independent runs are implemented for fair comparison. Consider the system non-linear behaviours, we conduct three experiments by selecting 10, 15 and 20 hidden nodes respectively. We select $u(t-1)$, $u(t-2)$, $y(t-1)$, $y(t-2)$ and 1 as the RBF neural model inputs. The training and validation results of all algorithms are shown in Table 7.

It could be observed in Table 7 that the RBF neural network with 15 hidden nodes gives the best training and validation results, achieving the RMSE by $4.691e-02$ and $1.585e-02$ within 3000 FES. Among all the algorithms, cTLBO outperforms the other competitors in both training and validation results. The RBF neural network training results again confirm the superior capacity of the proposed cTLBO in solving highly non-linear problems.

In a result, the proposed cTLBO shows competitive performance in continuous optimization and neural networks for hardware limited systems. The structure of both NN test systems are fairly simple and more deep neural networks are not considered. This is due to that deep neural networks often require significant computation resources and particular remarkable memory storages, which may not be suitable for the applications of compact algorithms. We therefore focus on simple and less layers neural network applications for embedded system rather than the deep ones. Due to the space of the paper and topic focus, system implementation for the algorithms is

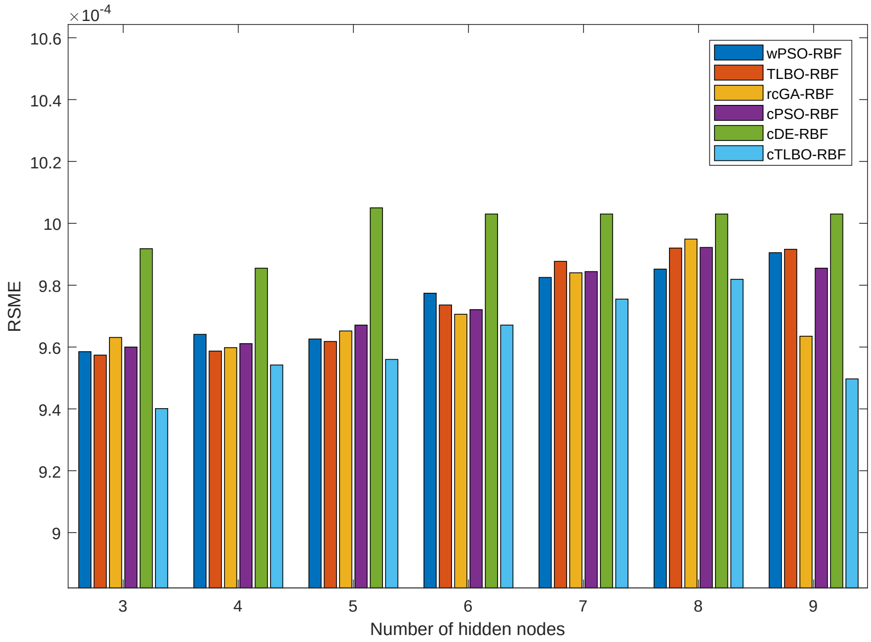

Figure 8: The comparison of the best results of RBF network training errors for test system 1

Table 7: RBF network training results of test system 2

Hidden Node	Algorithm	Training RMSE	Training STD	Validation RMSE	Validation STD
10	wPSO-RBF	9.077e-02	1.436e-02	3.749e-02	4.690e-03
	TLBO-RBF	8.873e-02	1.389e-02	3.548e-02	6.556e-03
	rcGA-RBF	9.312e-02	1.559e-02	3.475e-02	4.712e-03
	cPSO-RBF	1.611e-01	1.144e-01	6.261e-02	5.328e-02
	cDE-RBF	8.080e-01	9.770e-01	1.844e-01	6.574e-02
	cTLBO-RBF	8.579e-02	8.918e-03	3.307e-02	5.523e-03
15	wPSO-RBF	4.834e-02	1.763e-03	1.659e-02	5.526e-03
	TLBO-RBF	4.915e-02	2.436e-03	1.919e-02	5.677e-03
	rcGA-RBF	4.957e-02	1.190e-03	1.961e-02	4.366e-03
	cPSO-RBF	5.137e-02	2.527e-03	1.785e-02	5.928e-03
	cDE-RBF	1.404e-01	8.001e-02	7.694e-02	4.512e-02
	cTLBO-RBF	4.691e-02	7.631e-04	1.585e-02	3.530e-03
20	wPSO-RBF	7.714e-02	2.335e-04	1.961e-02	5.264e-03
	TLBO-RBF	7.677e-02	6.664e-04	2.013e-02	5.950-e03
	rcGA-RBF	7.597e-02	1.298e-03	2.068e-02	2.588e-03
	cPSO-RBF	7.583e-02	1.165e-03	2.237e-02	5.412e-03
	cDE-RBF	8.898e-02	6.499e-03	3.286e-02	2.499e-02
	cTLBO-RBF	7.495e-02	1.731e-03	1.877e-02	3.291e-03

not included and will be conducted in our future work.

7. Conclusion and Future Work

The stringent requirement on the limited computational resource and memory size has long been a challenging problem in implementing advanced intelligent optimization algorithms in real-time embedded applications. In this paper, a new compact teaching-learning based optimization method has been proposed to reduce the algorithm memory size requirement. The teaching-learning based optimization is integrated within a compact algorithm structure, and the new cTLBO has been compared with some typical meta-heuristic algorithms and the latest variants of TLBO on 32 benchmark problems. In addition, the proposed method is also employed to train a RBF neural network and to investigate the potential use of the technique for embedded systems. The comparative study results show that the cTLBO outperforms the other

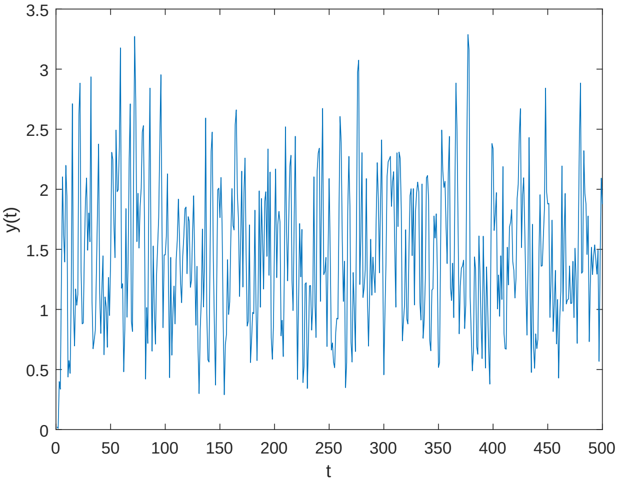

Figure 9: Data distribution of test system 2

typical algorithms and compact variants on the majority of benchmarks, while maintain the competitive performance of TLBO variants. Similar results could also be found in its application to two typical neural network trainings. On the other hand, this new method is able to significantly reduce the memory size requirement, paving a way for its wide real-time embedded applications.

In the new era of artificial intelligence, learning methods such as neural network are expected to be adopted in various compact systems with limited energy and storage resources. The novel cTLBO provides a powerful tool for continuous optimization problems, in particular training the simple structure neural networks in intelligent systems. The implementation on embedded system for the proposed algorithm will be conducted in the future.

Acknowledgments

This research is financially supported by China NSFC under grants 51607177, 61433012, U1435215, China Post-doctoral Science Foundation (2018M631005), and UK EPSRC grant under the Optimising Energy Management in Industry - OPTEMIN project EP/P004636/1.

[1] Miller III, W.T.: Real-time neural network control of a biped walking robot. Control Systems, IEEE **14**(1) (1994) 41–48

[2] Kulkarni, R.V., Venayagamoorthy, G.K.: Particle swarm optimization in wireless-sensor networks: A brief survey. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **41**(2) (2011) 262–267

[3] Intel: Mobile intel celeron processors product order codes for mobile intel celeron processors. "`http://www.intel.com/support/processors/mobile/celeron/sb/cs-007472.htm`" (2003)

[4] Harik, G.R., Lobo, F.G., Goldberg, D.E.: The compact genetic algorithm. Evolutionary Computation, IEEE Transactions on **3**(4) (1999) 287–297

[5] Ahn, C.W., Ramakrishna, R.S.: Elitism-based compact genetic algorithms. Evolutionary Computation, IEEE Transactions on **7**(4) (2003) 367–385

[6] Gallagher, J.C., Vigraham, S., Kramer, G.: A family of compact genetic algorithms for intrinsic evolvable hardware. Evolutionary Computation, IEEE Transactions on **8**(2) (2004) 111–126

[7] Mininno, E., Cupertino, F., Naso, D.: Real-valued compact genetic algorithms for embedded microcontroller optimization. Evolutionary Computation, IEEE Transactions on **12**(2) (2008) 203–219

[8] Neri, F., Mininno, E.: Memetic compact differential evolution for cartesian robot control. Computational Intelligence Magazine, IEEE **5**(2) (2010) 54–65

[9] Neri, F., Iacca, G., Mininno, E.: Disturbed exploitation compact differential evolution for limited memory optimization problems. Information Sciences **181**(12) (2011) 2469–2487

[10] Mininno, E., Neri, F., Cupertino, F., Naso, D.: Compact differential evolution. Evolutionary Computation, IEEE Transactions on **15**(1) (2011) 32–54

[11] Neri, F., Mininno, E., Iacca, G.: Compact particle swarm optimization. Information Sciences **239** (2013) 96–121

[12] Banitalebi, A., Aziz, M.I.A., Bahar, A., Aziz, Z.A.: Enhanced compact artificial bee colony. Information Sciences **298** (2015) 491–511

[13] Rao, R., Savsani, V., Vakharia, D.: Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. Computer-Aided Design **43**(3) (2011) 303–315

[14] Rao, R., Savsani, V., Vakharia, D.: Teaching–learning-based optimization: an optimization method for continuous non-linear large scale problems. Information Sciences **183**(1) (2012) 1–15

[15] Crepinsek, M., Liu, S.H., Mernik, L.: A note on teaching–learning-based optimization algorithm. Information Sciences **212** (2012) 79–93

[16] Waghmare, G.: Comments on a note on teaching–learning-based optimization algorithm. Information Sciences **229** (2013) 159–169

[17] Wang, Z., Lu, R., Chen, D., Zou, F.: An experience information teaching–learning-based optimization for global optimization. IEEE Transactions on Systems, Man, and Cybernetics: Systems **46**(9) (2016) 1202–1214

[18] Satapathy, S.C., Naik, A.: Modified teaching–learning-based optimization algorithm for global numerical optimizationa comparative study. Swarm and Evolutionary Computation **16** (2014) 28–37

[19] Niknam, T., Azizipanah-Abarghooee, R., Aghaei, J.: A new modified teaching-learning algorithm for reserve constrained dynamic economic dispatch. Power Systems, IEEE Transactions on **28**(2) (2013) 749–763

[20] Niu, Q., Zhang, H., Li, K.: An improved tlbo with elite strategy for parameters identification of pem fuel cell and solar cell models. International Journal of Hydrogen Energy (2014)

[21] Guo, Y., Li, K., Yang, Z., Deng, J., Laverty, D.M.: A novel radial basis function neural network principal component analysis scheme for pmu-based wide-area power system monitoring. Electric Power Systems Research **127** (2015) 197–205

[22] Sleesongsom, S., Bureerat, S.: Four-bar linkage path generation through self-adaptive population size teaching-learning based optimization. Knowledge-Based Systems **135** (2017) 180–191

[23] Shao, W., Pi, D., Shao, Z.: A hybrid discrete optimization algorithm based on teaching–probabilistic learning mechanism for no-wait flow shop scheduling. Knowledge-Based Systems **107** (2016) 219–234

[24] Amin, A.: A novel classification model for cotton yarn quality based on trained neural network using genetic algorithm. Knowledge-Based Systems **39** (2013) 124–132

[25] Gudise, V.G., Venayagamoorthy, G.K.: Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. In: Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE, IEEE (2003) 110–117

[26] Yu, J., Wang, S., Xi, L.: Evolving artificial neural networks using an improved pso and dpso. Neurocomputing **71**(4) (2008)

1054–1060

[27] Mirjalili, S., Mirjalili, S.M., Lewis, A.: Let a biogeography-based optimizer train your multi-layer perceptron. Information Sciences **269** (2014) 188–209

[28] Faris, H., Aljarah, I., Mirjalili, S.: Improved monarch butterfly optimization for unconstrained global search and neural network training. Applied Intelligence (2017) 1–20

[29] Shen, W., Guo, X., Wu, C., Wu, D.: Forecasting stock indices using radial basis function neural networks optimized by artificial fish swarm algorithm. Knowledge-Based Systems **24**(3) (2011) 378–385

[30] Cui, H., Feng, J., Guo, J., Wang, T.: A novel single multiplicative neuron model trained by an improved glowworm swarm optimization algorithm for time series prediction. Knowledge-Based Systems **88** (2015) 195–209

[31] Yang, Z., Li, K., Guo, Y.: A new compact teaching-learning-based optimization method. In: International Conference on Intelligent Computing, Springer (2014) 717–726

[32] Harik, G., Cantú-Paz, E., Goldberg, D.E., Miller, B.L.: The gambler's ruin problem, genetic algorithms, and the sizing of populations. Evolutionary Computation **7**(3) (1999) 231–253

[33] Iacca, G., Mallipeddi, R., Mininno, E., Neri, F., Suganthan, P.N.: Super-fit and population size reduction in compact differential evolution. In: Memetic Computing (MC), 2011 IEEE Workshop on, IEEE (2011) 1–8

[34] Jewajinda, Y.: Covariance matrix compact differential evolution for embedded intelligence. In: Region 10 Symposium (TENSYMP), 2016 IEEE, IEEE (2016) 349–354

[35] Pan, J.S., Dao, T.K., Pan, T.S., et al.: Compact particle swarm optimization for optimal location of base station in wireless sensor network. In: International Conference on Genetic and Evolutionary Computing, Springer (2016) 54–62

[36] Dao, T.K., Chu, S.C., Shieh, C.S., Horng, M.F., et al.: Compact artificial bee colony. In: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Springer (2014) 96–105

[37] Dao, T.K., Pan, J.S., Chu, S.C., Shieh, C.S., et al.: Compact bat algorithm. In: Intelligent Data analysis and its Applications, Volume II. Springer (2014) 57–68

[38] Dao, T.K., Pan, T.S., Nguyen, T.T., Chu, S.C., Pan, J.S.: A compact flower pollination algorithm optimization. In: Computing Measurement Control and Sensor Network (CMCSN), 2016 Third International Conference on, IEEE (2016) 76–79

[39] Patel, V.K., Savsani, V.J.: A multi-objective improved teaching–learning based optimization algorithm (mo-itlbo). Information Sciences **357** (2016) 182–200

[40] Rajinikanth, V., Satapathy, S.C., Fernandes, S.L., Nachiappan, S.: Entropy based segmentation of tumor from brain mr images–a study with teaching learning based optimization. Pattern Recognition Letters (2017)

[41] Shabanpour-Haghighi, A., Seifi, A.R., Niknam, T.: A modified teaching–learning based optimization for multi-objective optimal power flow problem. Energy Conversion and Management **77** (2014) 597–607

[42] Yang, Z., Li, K., Niu, Q., Xue, Y., Foley, A.: A self-learning tlbo based dynamic economic/environmental dispatch considering multiple plug-in electric vehicle loads. Journal of Modern Power Systems and Clean Energy (2014) 1–10

[43] Yu, K., Chen, X., Wang, X., Wang, Z.: Parameters identification of photovoltaic models using self-adaptive teaching-learning-based optimization. Energy Conversion and Management **145** (2017) 233–246

[44] Sahu, R.K., Panda, S., Rout, U.K., Sahoo, D.K.: Teaching learning based optimization algorithm for automatic generation control of power system using 2-dof pid controller. International Journal of Electrical Power & Energy Systems **77** (2016) 287–301

[45] Niknam, T., Zare, M., Aghaei, J.: Scenario-based multi-objective volt/var control in distribution networks including renewable energy sources. IEEE Transactions on Power Delivery **27**(4) (2012) 2004–2019

[46] Yu, K., While, L., Reynolds, M., Wang, X., Wang, Z.: Cyclic scheduling for an ethylene cracking furnace system using diversity learning teaching-learning-based optimization. Computers & Chemical Engineering **99** (2017) 314–324

[47] Tuo, S., Yong, L., Li, Y., Lin, Y., Lu, Q., et al.: Hstlbo: A hybrid algorithm based on harmony search and teaching-learning-based optimization for complex high-dimensional optimization problems. PloS one **12**(4) (2017) e0175114

[48] Yao, X., Liu, Y., Lin, G.: Evolutionary programming made faster. Evolutionary Computation, IEEE Transactions on **3**(2) (1999) 82–102

[49] Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.P., Auger, A., Tiwari, S.: Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. KanGAL report **2005005** (2005) 2005

[50] Qin, A., Huang, V., Suganthan, P.: Differential evolution algorithm with strategy adaptation for global numerical optimization. Evolutionary Computation, IEEE Transactions on **13**(2) (2009) 398–417

[51] Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the sixth international symposium on micro machine and human science. Volume 1., New York, NY (1995) 39–43

[52] Clerc, M., Kennedy, J.: The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE transactions on Evolutionary Computation **6**(1) (2002) 58–73

[53] Das, S., Suganthan, P.: Differential evolution: A survey of the state-of-the-art. Evolutionary Computation, IEEE Transactions on **15**(1) (2011) 4–31

[54] Mirjalili, S.: Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. Knowledge-Based Systems **89** (2015) 228–249

[55] Liu, Y., Chen, Z., Yang, Z., Li, K., Tan, J.: An inline surface measurement method for membrane mirror fabrication using two-stage trained zernike polynomials and elitist teaching–learning-based optimization. Measurement Science and Technology **27**(12) (2016) 124005

[56] Zhang, J.R., Zhang, J., Lok, T.M., Lyu, M.R.: A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training. Applied mathematics and computation **185**(2) (2007) 1026–1037

[57] Mirjalili, S., Hashim, S.Z.M., Sardroudi, H.M.: Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm. Applied Mathematics and Computation **218**(22) (2012) 11125–11137

[58] Piroddi, L., Spinelli, W.: An identification algorithm for polynomial narx models based on simulation error minimization. International Journal of Control **76**(17) (2003) 1767–1781

[59] Li, K., Peng, J.X.: Neural input selectiona fast model-based approach. Neurocomputing **70**(4) (2007) 762–769