Teaching Learning based Optimization

K. Satheesh Kumar
M. Tech Student
Department of Computer Applications
SRM University, Chennai, India

R. Harris Samuel
Assistant Professor (O.G)
Department of Mechanical Engineering
SRM University, Chennai, India

Abstract

The main aim of the project is to develop a new efficient optimization method, called ‘Teaching–Learning-Based Optimization (TLBO)’, for the optimization of mechanical design problems. The process of TLBO is divided into two parts. The first part consists of the ‘Teacher Phase’ and the second part consists of the ‘Learner Phase’. ‘Teacher Phase’ means learning from the teacher and ‘Learner Phase’ means learning by the interaction between learners. Teacher tries to reach best harmony on the output of learners in a class, which can be obtained through their grades considered as the output. Output is appraised by means of exam conducted by the teacher. TLBO is a population based algorithm where a group of students (i.e. learners) is considered as population and the different subjects offered to the learners is analogous with the different design variables of the optimization problem. The grades of a learner in each subject represent a possible solution to the optimization problem (value of design variables) and the mean result of a learner considering all subjects corresponds to the quality of the associated solution (fitness value). The best solution in the entire population is considered as the teacher. The method is tested on different unconstrained benchmark test functions. The effectiveness of the TLBO method is compared with Genetic Algorithm based on the best solution. Results show that TLBO is 244% more effective than the Genetic Algorithm for the unconstrained optimization problems considered. This novel optimization method is extended to an engineering design optimization problem. The TLBO method is modified for solving Multi objective optimization problems.

Keywords: optimization, genetic algorithm, teaching, learning, multiobjective

I. INTRODUCTION

Optimization is the act of obtaining the best solution under given circumstances. This technique provides a powerful tool in improving the engineering design in a rational manner and has been proved to be much more efficient than the traditional trial-and-error design process. Today, the optimization tool has become a part of every engineering industry for design improvement. Higher customer expectations and tighter industry standards require more efficient designs. Developments in faster digital computers, sophisticated computing techniques and more frequent use of finite element methods facilitate this to a certain extent, but there arises a need to explore approaches which could use these aids in a better way and solve complex engineering problems.

There are many classical optimization techniques developed and extensively used in the literature. These methods are based on mathematical programming techniques and cover a wide range, including linear, non-linear, geometric and quadratic programming. The more recent methods, like neural networks, simulated annealing and genetic algorithms are called non-classical optimization methods. One of the most recent methods is the use of swarm intelligence techniques in the design optimization of mechanical and structural engineering systems. Present research explores one such technique called Teaching Learning Based Optimization.

II. OPTIMIZATION ALGORITHMS

For different types of optimization problems[8], we often have to use different optimization techniques because some algorithms such as the Newton-Raphson method are more suitable for certain types of optimization than others.

In general, optimization algorithms can be divided into two categories: deterministic algorithms, and stochastic algorithms. Deterministic algorithms follow a rigorous procedure and its path and values of both design variables and the functions are repeatable. For example, hill-climbing is a deterministic algorithm, and for the same starting point, they will follow the same path whether you run the program today or tomorrow. On the other hand, the stochastic algorithms always have some randomness. Genetic algorithms are a good example, the strings or solutions in the population will be different each time you run a program since the algorithms use some pseudorandom numbers, though the final results may be no big difference, but the paths of each individual are not exactly repeatable. Furthermore, there is a third type of algorithm which is a mixture or hybrid of deterministic and stochastic algorithms. For example, hill-climbing with a random restart is a good example. The basic idea is to use the deterministic algorithm, but start with different initial points. This has certain advantages over a simple hill-climbing technique which may be stuck in a local peak. However, since there is a random component in this hybrid algorithm, we often classify it as a type of stochastic algorithm in the optimization literature.
Most conventional or classic algorithms are deterministic. For example, the Simplex method in linear programming is deterministic. Some deterministic optimization algorithms used the gradient information; they are called gradient-based algorithms. For example, the well-known Newton-Raphson algorithm is gradient-based as it uses the function values and their derivatives, and it works extremely well for smooth unimodal problems. However, if there is some discontinuity in the objective function, it does not work well. In this case, a non-gradient algorithm is preferred. Non-gradient-based or gradient-free algorithms do not use any derivative, but only the function values. Hooke-Jeeves pattern search and Nelder-Mead downhill simplex are examples of gradient-free algorithms. For stochastic algorithms, we have in general two types: heuristic and metaheuristic, though their difference is small. Loosely speaking, heuristic means 'to find' or 'to discover by trial and error'. Quality solutions to a tough optimization problem can be found in a reasonable amount of time, but there is no guarantee that optimal solutions are reached. It is expected that these algorithms work most of the time, but not all the time. This is usually good enough when we do not necessarily want the best solutions but rather good solutions which are easily reachable.

![Classification of Algorithms](image)

**Fig. 1: Classification of Algorithms**

### III. Genetic Algorithm

Many practical optimum design problems are characterized by mixed continuous–discrete variables, and discontinuous and non convex design spaces. If standard nonlinear programming techniques are used for this type of problem they will be inefficient, computationally expensive, and, in most cases, find a relative optimum that is closest to the starting point. Genetic algorithms (GAs) are well suited for solving such problems, and in most cases they can find the global optimum solution with a high probability.

The genetic algorithm (GA), developed by John Holland[2,7] and his collaborators in the 1960s and 1970s, is a model or abstraction of biological evolution based on Charles Darwin's theory of natural selection. Holland was the first to use the crossover and recombination, mutation, and selection in the study of adaptive and artificial systems.

**A. Basic Procedure:**

The essence of genetic algorithms involves the encoding of an optimization function as arrays of bits or character strings to represent the chromosomes, the manipulation operations of strings by genetic operators, and the selection according to their fitness with the aim to find a solution to the problem concerned. This is often done by the following procedure:

**B. Pseudo code for Genetic Algorithm:**

Objective function \( f(x), x = \{x_1, ..., x_n\}^T \)

Encode the solution into chromosomes (binary strings)

Define fitness \( F \) (eg, \( F \alpha f(x) \) for maximization)

Generate the initial population

Initial probabilities of crossover (pc) and mutation (pm)

while \( (t < \text{Max number of generations}) \)

Generate new solution by crossover and mutation

if pc \( > \text{rand}, \) Crossover; end if

if pm \( > \text{rand}, \) Mutate; end if

Accept the new solutions if their fitness increase
Select the current best for new generation (elitism)  
end while  
Decode the results and visualization  

C. Teaching Learning based Optimization:  

Most commonly used evolutionary optimization technique is genetic algorithm (GA). However, GA provides a near optimal solution for a complex problem having large number of variables and constraints. This is mainly due to the difficulty in determining the optimum controlling parameters such as crossover rate and mutation rate[3].  

Therefore, the efforts must be continued to develop an optimization technique which is free from the algorithm parameters, i.e. no algorithm parameters are required for the working of the algorithm. This aspect is considered in the present work. An optimization method, Teaching–Learning-Based Optimization (TLBO), is proposed in this project to obtain global solutions for continuous nonlinear functions with less computational effort and high consistency. The TLBO method works on the philosophy of teaching and learning. The TLBO method is based on the effect of the influence of a teacher on the output of learners in a class. Here, output is considered in terms of results or grades. The teacher is generally considered as a highly learned person who shares his or her knowledge with the learners. The quality of a teacher affects the outcome of learners. It is obvious that a good teacher trains learners such that they can have better results in terms of their marks or grades. Moreover, learners also learn from interaction between themselves, which also helps in their results.  

Teaching–learning process is the heart of education. The fulfilment of the aims and objectives of the education depends on Teaching–learning process. Based on the above fact of teaching–learning process, mathematical model is prepared and it is implemented for the optimization process. Assume two different teachers, T1 and T2, teaching a subject with same content to the same merit level learners in two different classes. The distribution of marks obtained by the learners of two different classes evaluated by the teachers follows some distribution depending on the group of learners. A Normal distribution is assumed for the obtained marks.  

Like other nature-inspired algorithms, TLBO is also a population based method which uses a population of solutions to proceed to the global solution. For TLBO population is considered as a group of learners or a class of learners. In optimization algorithms population consists of different design variables. In TLBO different design variables will be analogous to different subjects offered to learners and the learners’ result is analogous to the ‘fitness’ as in other population-based optimization techniques. The teacher is considered as the best solution obtained so far. The process of working of TLBO is divided into two parts. The first part consists of ‘Teacher Phase’ and the second part consists of ‘Learner Phase’. The ‘Teacher Phase’ means learning from the teacher and the ‘Learner Phase’ means learning due through the interaction between learners.  

D. Teacher Phase:  

It is the first part of the algorithm where learners learn through the teacher. During this phase a teacher tries to increase the mean result of the class in the subject taught by him or her depending on his or her capability. At any iteration i, assume that there are m number of subjects (i.e. design variables), n number of learners (i.e. population size, k = 1, 2, ..., n) and Mj,i be the mean result of the learners in a particular subject j (j = 1, 2, ..., m). The best overall result Xtotal-kbest,i considering all the subjects together obtained in the entire population of learners can be considered as the result of best learner kbest. However, as the teacher is usually considered as a highly learned person who trains learners so that they can have better results, the best learner identified is considered by the algorithm as the teacher. The difference between the existing mean result of each subject and the corresponding result of the teacher for each subject is given by Eq. 1.1 as,  

\[ \text{Difference}_\text{Mean}_j,k,i = \text{ri}(X_{j,k\text{best},i} - TFM_{j,i}) \]  

where, \( X_{j,k\text{best},i} \) is the result of the best learner (i.e. teacher) in subject j. TF is the teaching factor which decides the value of mean to be changed and \( \text{ri} \) is the random number in the range [0, 1]. Value of TF can be either 1 or 2. The value of TF is decided randomly with equal probability as,  

\[ \text{TF} = \text{round}[1 + \text{rand}(0, 1)(2 - 1)] \]  

TF is not a parameter of the TLBO algorithm. The value of TF is not given as an input to the algorithm and its value is randomly decided by the algorithm using Eq. 1.1. After conducting a number of experiments on many benchmark functions it is concluded that the algorithm performs better if the value of TF is between 1 and 2. However, the algorithm is found to perform much better if the value of TF is either 1 or 2 and hence to simplify the algorithm, the teaching factor is suggested to take either 1 or 2 depending on the rounding up criteria given by Eq. 1.2. Based on the Difference_Mean_j,k,i, the existing solution is updated in the teacher phase according to the following expression.  

\[ X'_{j,k,i} = X_{j,k,i} + \text{Difference}_\text{Mean}_j,k,i \]  

where \( X'_{j,k,i} \) is the updated value of \( X_{j,k,i} \). Accept \( X'_{j,k,i} \) if it gives better function value. All the accepted function values at the end of the teacher phase are maintained and these values become the input to the learner phase.
E. Learner Phase:

Learners increase their knowledge by two different means: one through input from the teacher and the other through interaction between themselves. A learner interacts randomly with other learners with the help of group discussions, presentations, formal communications, etc. A learner learns something new if the other learner has more knowledge than him or her. Learner modification is expressed as

For $i = 1 : Pn$
Randomly select two learners $X_i$ and $X_j$, where $i \neq j$
If $f(X_i) < f(X_j)$
   $X_{new,i} = X_{old,i} + r_i(X_i - X_j)$
Else
   $X_{new,i} = X_{old,i} + r_i(X_j - X_i)$
End If
End For
Accept $X_{new}$ if it gives a better function value.

IV. IMPLEMENTATION OF GENETIC ALGORITHM

The computational procedure involved in maximizing the fitness function $F(x_1, x_2, x_3, \ldots, x_n)$ in the genetic algorithm can be described by the following steps.

1) Choose a suitable string length $l = nq$ to represent the $n$ design variables of the design vector $X$. Assume suitable values for the following parameters: population size $m$, crossover probability $p_c$, mutation probability $p_m$, permissible value of standard deviation of fitness values of the population $(sf)_{max}$ to use as a convergence criterion, and maximum number of generations $(i_{max})$ to be used as a second convergence criterion.
2) Generate a random population of size $m$, each consisting of a string of length $l = nq$. Evaluate the fitness values $F_i$, $i = 1, 2, \ldots, m$, of the $m$ strings.
3) Carry out the reproduction process.
4) Carry out the crossover operation using the crossover probability $p_c$.
5) Carry out the mutation operation using the mutation probability $p_m$ to find the new generation of $m$ strings.
6) Evaluate the fitness values $F_i$, $i = 1, 2, \ldots, m$, of the $m$ strings of the new population. Find the standard deviation of the $m$ fitness values.
7) Test for the convergence of the algorithm or process. If $sf \leq (sf)_{max}$, the convergence criterion is satisfied and hence the process may be stopped. Otherwise, go to step 8.
8) Test for the generation number. If $i \geq i_{max}$, the computations have been performed for the maximum permissible number of generations and hence the process may be stopped. Otherwise, set the generation number as $i = i + 1$ and go to step 3.

V. IMPLEMENTATION OF TLBO

The step-wise procedure for the implementation of TLBO is given in this section.

1) Step 1: Define the optimization problem and initialize the optimization parameters.
Initialize the population size ($P_n$), number of generations ($G_n$), number of design variables ($D_n$), and limits of design variables ($UL, LL$).
Define the optimization problem as: Minimize $f(X)$.
   Subject to $X_i \in x_i = 1, 2, \ldots, D_n$
where $f(X)$ is the objective function, $X$ is a vector for design variables such that
   $LL,i \leq x,i \leq UL,i$.

2) Step 2: Initialize the population.
Generate a random population according to the population size and number of design variables. For TLBO, the population size indicates the number of learners and the design variables indicate the subjects (i.e. courses) offered. This population is expressed as

3) Step 3: Teacher phase.
Calculate the mean of the population column-wise, which will give the mean for the particular subject as
   $M,D = [m_1,m_2, \ldots, m_D]$ (3.1)
The best solution will act as a teacher for that iteration
   $X_{teacher} = X_f (X) = \min$ (3.2)
The teacher will try to shift the mean from $M,D$ towards $X_{teacher}$, which will act as a new mean for the iteration. So,
   $M_{new,D} = X_{teacher,D}$ (2.7)
The difference between two means is expressed as
Difference, \( D = r (M_{\text{new}}, D - TFM_{\text{D}}) \). (3.3)

The value of TF is selected as 1 or 2. The obtained difference is added to the current solution to update its values using \( X_{\text{new}, \text{D}} = X_{\text{old}, \text{D}} + \text{Difference,}_{\text{D}} \). (3.4)

Accept \( X_{\text{new}} \) if it gives better function value.

4) Step 4: Learner phase.

As explained above, learners increase their knowledge with the help of their mutual interaction.

5) Step 5: Termination criterion.

Stop if the maximum generation number is achieved. Otherwise repeat from Step 3.

Unconstrained benchmark implementation of functions

1) Sphere Function:

\[
 f(x) = \sum_{i=1}^{n} x_i^2 \\
-100 \leq x_i \leq 100 \\
\min(f) = f(0,0 \ldots 0) = 0
\]

2) Schwefel 2.22 Function:

\[
 f(x) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i| \\
-100 \leq x_i \leq 100 \\
\min(f) = f(0,0 \ldots 0) = 0
\]

3) Schwefel 1.2 Function:

\[
 nf(x) = \sum_{i=1}^{n} \left( \sum_{j=1}^{i} x_j \right)^2 \\
-100 \leq x_i \leq 100 \\
\min(f) = f(0,0 \ldots 0) = 0
\]

4) Schwefel 2.21 Function:

\[
 gnf(x) = \max_{i} \{|x_i|, 1 \leq i \leq n\} \\
-100 \leq x_i \leq 100 \\
\min(f) = f(0,0 \ldots 0) = 0
\]

5) Rosenbrock Function:

\[
 f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \\
-30 \leq x_i \leq 30 \\
\min(f) = f(1,1 \ldots 1) = 0
\]

6) Step Function:

\[
 f(x) = \sum_{i=1}^{n} [x_i + 0.5]^2 \\
-100 \leq x_i \leq 100 \\
\min(f) = f(0,0 \ldots 0) = 0
\]

7) Quartic Function:

\[
 nf(x) = \sum_{i=1}^{n} \lfloor x_i^4 \rfloor \\
-1.28 \leq x_i \leq 1.28 \\
\min(f) = f(0,0 \ldots 0) = 0
\]

8) Schwefel 2.26 Function:

\[
 f(x) = -\sum_{i=1}^{30} x_i \sin \left( \sqrt{|x_i|} \right)
\]
9) Rastrigin Function:

\[ f(x) = \sum_{i=1}^{n} \left[ x_i^2 - 10 \cos(2\pi x_i) + 10 \right] \]

\[ f(x) = \sum_{i=1}^{n} -20 \exp \left( -0.2 \sqrt{\frac{1}{30} \sum_{i=1}^{30} x_i^2} \right) - \exp \left( \frac{1}{30} \sum_{i=1}^{30} \cos(2\pi x_i) \right) \]

\[-500 \leq x_i \leq 500 \]
\[ \min(f) = f(0,0 \ldots 0) = 0 \]

10) Ackley Function:

\[-32 \leq x_i \leq 32 \]
\[ \min(f) = f(0,0 \ldots 0) = 0 \]

VI. ENGINEERING DESIGN OPTIMIZATION PROBLEM

Problem Statement for an Engineering Design Optimization Problem

I-beam design

The goal of the problem is to find the dimensions of the beam presented in Fig. 4 which satisfy the dimensions of the geometric and strength constraints, and at the same time minimize two objectives: cross-sectional area of the beam and static deflection of the beam under the force P (Yang et al., 2002).

The mathematical description of I-beam design problem can be expressed by Eq. (10):

- Minimize : Cross sectional area \( f_1(x) = 22.22x_2 + 87.78x_3 \)
- And deflection \( f_2(x) = PL^3/48EI \)

where \( I = 80589.02x^3 + 1.8516x^2 + 2455.094 \)
\( E = 6000 \text{ N/mm}^2 \)
\( P = 64.134 \text{ N} \)
\( L = 1000 \text{ mm} \)

VII. RESULTS AND DISCUSSION

The results for both TLBO and GA are listed in the table 1. The results for the Mechanical Design Optimization problem are listed in table 2. The main objective of the mechanical design optimization problem is to minimize the cross sectional area and deflection of an I beam. It is solved using the modified TLBO method for multi objective optimization problems.

In this paper, a simple discussion about the comparison for genetic algorithm (GA), and TLBO is given. The comparison is conducted based on 10 benchmark evaluation functions as employed in section 3.3. Table 4.1 summarizes the comparative results of the best solutions of the function evaluations for GA and TLBO. For all the benchmark functions TLBO and GA are evaluated with a population size of 20 and the number of generations as 100. The average error observed in TLBO is 1.19199E-11 and in GA it is observed to be 2.4432. The percentage error between the two methods is given by the equation

\[ \%\text{error} = 100 \times (\text{value obtained} - \text{value expected}) \]

From the above equation we get the percentage error as 244.32%.

Table 1:

<table>
<thead>
<tr>
<th>SI</th>
<th>FUNCTION / METHOD</th>
<th>TLBO RESULT</th>
<th>GA RESULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sphere</td>
<td>9.43E-21</td>
<td>3.764648438</td>
</tr>
<tr>
<td>2</td>
<td>Schwefel 2.22</td>
<td>1.21E-21</td>
<td>6.399902344</td>
</tr>
<tr>
<td>3</td>
<td>Schwefel 1.2</td>
<td>9.43E-21</td>
<td>3.764648438</td>
</tr>
<tr>
<td>4</td>
<td>Schwefel 2.11</td>
<td>1.21E-21</td>
<td>6.399902344</td>
</tr>
<tr>
<td>5</td>
<td>Rosenbrock</td>
<td>1.39E-10</td>
<td>-1.599975586</td>
</tr>
<tr>
<td>6</td>
<td>Step</td>
<td>-1.50E-11</td>
<td>6.399902344</td>
</tr>
<tr>
<td>7</td>
<td>Quartic</td>
<td>-2.08E-12</td>
<td>0.058822632</td>
</tr>
<tr>
<td>8</td>
<td>Schwefel 2.26</td>
<td>6.59E-13</td>
<td>-1.247109375</td>
</tr>
<tr>
<td>9</td>
<td>Rastrigin</td>
<td>-1.30E-12</td>
<td>0.439735413</td>
</tr>
<tr>
<td>10</td>
<td>Ackley</td>
<td>-2.08E-12</td>
<td>0.058822632</td>
</tr>
</tbody>
</table>
Table -2:
Result and Discussion

<table>
<thead>
<tr>
<th></th>
<th>Optimum flange width</th>
<th>x1final</th>
<th>53.9559</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Optimum web thickness</td>
<td>x2final</td>
<td>6.4003</td>
</tr>
<tr>
<td>3</td>
<td>Cross sectional area</td>
<td>objfun1final</td>
<td>1.76e03</td>
</tr>
<tr>
<td>4</td>
<td>Deflection</td>
<td>objfun2final</td>
<td>0.0742</td>
</tr>
</tbody>
</table>

REFERENCES