Engineering Applications of Artificial Intelligence 26 (2013) 1291–1300

Contents lists available at SciVerse ScienceDirect

Engineering Applications of Artificial Intelligence

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

Multi-objective optimization using teaching-learning-based optimization algorithm

Feng Zou [a,b], Lei Wang [a,*], Xinhong Hei [a], Debao Chen [b], Bin Wang [a]

[a] School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China
[b] School of Physics and Electronic Information, Huaibei Normal University, Huaibei 235000, China

ARTICLE INFO

Article history:
Received 8 May 2012
Received in revised form
30 October 2012
Accepted 12 November 2012
Available online 21 December 2012

Keywords:
Teaching-learning-based optimization
Multi-objective optimization
Nondominated sorting
Crowding distance

ABSTRACT

Two major goals in multi-objective optimization are to obtain a set of nondominated solutions as closely as possible to the true Pareto front (PF) and maintain a well-distributed solution set along the Pareto front. In this paper, we propose a teaching-learning-based optimization (TLBO) algorithm for multi-objective optimization problems (MOPs). In our algorithm, we adopt the nondominated sorting concept and the mechanism of crowding distance computation. The teacher of the learners is selected from among current nondominated solutions with the highest crowding distance values and the centroid of the nondominated solutions from current archive is selected as the Mean of the learners. The performance of proposed algorithm is investigated on a set of some benchmark problems and real life application problems and the results show that the proposed algorithm is a challenging method for multi-objective algorithms.

1. Introduction

In many cases, most engineering design problems, such as investment decision, city programming, program management, university timetable, control system design, the objectives present some degree of conflict among them in nature. That is to say, one objective cannot be improved without deterioration of at least another objective. These problems are called multi-objective optimization problems (MOPs), which have a set of several optimal solutions known as Pareto optimal solutions (Deb, 2001). Therefore, multi-objective optimization also differs from single-objective optimization in that the former is composed of two different tasks to solve the problem: a searching task whose goal is to find Pareto optimal solutions, and a decision making task in which a most preferred solution is chosen from the set of Pareto optimal solutions. In other words, two major tasks in multi-objective optimization are to obtain a set of nondominated solutions as closely as possible to the true Pareto front (PF) and maintain a well-distributed solution set along the Pareto front. Hence, the goal of multi-objective optimization methods is to find a set of good trade-off solutions from which the decision maker want to select one.

In order to solve multi-objective problem, V. Pareto offers the most common definition of optimum in multi-objective optimization in 1896. In 1984, Schaffer (Schaffer, 1985) proposed the first actual implementation of evolutionary algorithms to solve multi-objective problems, which it is now called multi-objective evolutionary algorithm (MOEA). Evolutionary computation techniques are suitable for multi-objective optimizations because of the fact that Evolutionary Algorithm (EA) deals with a set of solutions which help in the generation of well distributed Pareto optimal front more quickly and efficiently in comparison to the classical techniques. Since 1984, many researchers have proposed their own multi-objective evolutionary algorithms (MOEAs). Representative multi-objective evolutionary methods, such as NPGA (Horn et al., 1994), NPGA2 (Erickson and Mayer, 2001), NSGA (Srinivas and Deb, 1994), NSGA-II (Deb et al., 2000), SPEA (Zitzler and Thiele, 1999), SPEA2 (Knowles and Corne, 2000), MOPSO (Coello Coello et al., 2004), MODE (Xue and Sanderson, 2003), MOSaDE (Huang et al., 2009), VEDA (Larrañaga and Lozano, 2001), MOHBOA (Pelikan et al., 2005), RM-MEDA (Qingfu, 2008), and MOEA-D (Zhang and Li,. 2007), are utilized to optimize several objectives simultaneously and some efficient results are derived.

In this paper, we propose a teaching-learning-based optimization (TLBO) algorithm based on the nondominated sorting and crowding distance sorting for MOPs. In our algorithm, we adopt the nondominated sorting concept used in NSGA-II, where the entire population is sorted into various non-domination levels. This provides the means for selecting the individuals in the better fronts, hence providing the necessary selection pressure to push the population towards PF. To maintain the diversity of the current best solutions in the external archive, the mechanism of crowding distance computation used in NSGA-II is adopted.

* Corresponding author. Tel.: +86 29 8231 2087.
E-mail address: wangleei@hotmail.com (L. Wang).

0952-1976/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.engappai.2012.11.006

The teacher of the learners is selected from among current nondominated solutions with the highest crowding distance values and the centroid of the nondominated solutions from current archive is selected as the Mean of the learns. The performance of proposed algorithm is investigated on a set of some unconstrained and constrained benchmark problems and the results show that the proposed algorithm is a challenging method for multi-objective algorithms.

The remainder of this paper is organized as follows. The description of teaching-learning-based optimization algorithm is introduced in Sections 2 and 3. describes the proposed algorithm. Comparison and analysis of experimental results of some unconstrained test problems are shown in Section 4. Some constrained optimization examples are shown in Section 5 and some conclusions are given in Section 6.

2. Teaching-learning-based optimization

Rao et al. (2011, 2012) first proposed a novel teaching-learning-based optimization (TLBO) inspired from the philosophy of teaching and learning. TLBO has emerged as one of the simple and efficient techniques for solving single-objective benchmark problems and real life application problems in which it has been empirically shown to perform well on many optimization problems (Rao et al., 2011a, 2011b, 2012; Rao, 2012; Rao and Patel, 2011; Rao and Kalyankar, 2012; Togan, 2012). These are precisely the characteristics of TLBO that make it attractive to extend it to solve MOPs (Rao and Patel, 2012a, 2012b; Niknam and Golestaneh, 2012; Niknam et al., 2012; Satapathy et al., 2012).

The TLBO method is based on the effect of the influence of a teacher on the output of learners in a class which 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. 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, the population is considered as a group of learners or a class of learners. In optimization algorithms, the 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 through the interaction between learners.

2.1. Teaching phase

A good teacher is one who brings his or her learners up to his or her level in terms of knowledge. But in practice this is not possible and a teacher can only move the mean of a class up to some extent depending on the capability of the class. This follows a random process depending on many factors.

Let M_i be the mean and T_i be the teacher at any iteration i. T_i will try to move mean M_i towards its own level, so now the new mean will be T_i designated as M_{new}. The solution is updated according to the difference between the existing and the new mean given by

$$\text{Difference\_Mean}_i = r_i(M_{new} - T_F M_i) \quad (1)$$

where T_F is a teaching factor that decides the value of mean to be changed, and r_i is a random number in the range [0, 1]. The value of T_F can be either 1 or 2, which is again a heuristic step and decided randomly with equal probability as

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

This difference modifies the existing solution according to the following expression

$$X_{new,i} = X_i + \text{Difference\_Mean}_i \quad (3)$$

Learner modification is expressed as (*Pn* is the number of learners),

$T_F = \text{round}[1 + \text{rand}(0,1)]$
for p = 1:P_n
  $\text{Difference\_Mean}_i = r_i \times (M_{new} - T_F \times Mi)$
  $X_{new,p} = X_p + \text{Difference\_Mean}_i$
endfor
Accept X_{new} if it gives a better function value

2.2. Learning phase

Learners increase their knowledge by two different means: one through input from the teacher and 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 (*Pn* is the number of learners),

for I = 1:P_n
  Randomly select one learner X_j, such that $i \neq j$
  if $f(X_i) < f(X_j)$
    $X_{new,i} = X_{old,i} + r_i \times (X_i - X_j)$
  else
    $X_{new,i} = X_{old,i} + r_i \times (X_j - X_i)$
  endif
endfor
Accept X_{new} if it gives a better functions value

2.3. The sketch of TLBO algorithm

As explained above, the step-wise procedure for the implementation of TLBO can be summarized as follows.

Step 1: Define the optimization problem and initialize the optimization parameters.
Step 2: Initialize the population.
Step 3: Teacher phase. Learners is learning from the teacher.
Step 4: Learner phase. Learners increase their knowledge with the help of their mutual interaction.
Step 5: Termination criterion. Stop if the maximum generation number is achieved; otherwise repeat from Step 3.

3. Description of the proposed algorithm

In the current study, we have concentrated our work on teaching-learning-based optimization (TLBO) for solving MOPs. In this paper, we proposed an improved TLBO called multi-objective teaching-learning-based optimization (MOTLBO). In our MOTLBO, we use an external archive to keep the best solutions obtained so far. We adopt the nondominated sorting concept used in NSGA-II

(Deb et al., 2000) to select the individuals in the better fronts in order to push the population towards PF. At the same time, to maintain the diversity of the current best solutions in the external archive, the mechanism of crowding distance computation used in NSGA-II (Deb et al., 2000) is adopted. The following sections describe these methods.

3.1. External archive

We use an external archive to keep the best solutions generated so far by the MOTLBO algorithm, that is to say, we incorporates the nondominated sorting concept and the mechanism of crowding distance computation into the algorithm specifically on Teacher selection and in the deletion method of population including an external archive of the current best solutions.

At the beginning of the process of MOTLBO, NP (the number of individuals of initial population) solutions are sorted on the basis of non-domination rank and crowding distance rank and added to the external archive. As the evolution progresses, MOTLBO applies the TLBO to create NP new solutions. It then combines the two (current & external archive) populations. Note that the total size of the set after combination becomes 2NP. After that, NP solutions are selected on the basis of non-domination rank and crowding distance rank for next generation from 2NP solutions. Those solutions which are in the most crowded areas are most likely to be selected so that this method promotes diversity among the stored solutions in the archive.

3.2. Selection operator

In single-objective optimization, it is easy to decide which one is better between two individuals. But in MOPs, the decision is not so straightforward. We could use the concept of dominance that the candidate replaces the parent only if the former one dominates the latter one.

The selection of the teacher of the learners is a crucial step in a MOTLBO algorithm. It affects both the convergence capability of the algorithm as well as maintaining a good spread of nondominated solutions. In MOTLBO, a bounded external archive stores nondominated solutions found in previous iteration. We note that any of the nondominated solutions in the external archive can be used as the teacher of the learners. But we want to ensure that the learners in the population move towards the sparse regions of the search space. So, the teacher of the learners is selected from among those nondominated solutions with the highest crowding distance values. Selecting different teachers for each learner in a specified top part of the external archive based on a decreasing crowding distance allows the learners in the primary population to move towards those nondominated solutions in the external archive which are in the least crowded area in the objective space. At the same time, we select the centroid of the nondominated solutions from current archive as the Mean of the learns.

In the process of adopting the teaching phase and the learning phase typical of TLBO thus reproducing its search logic, we adopt the following operation. The candidate replaces the parent if the candidate dominates the parent, the candidate is discarded if the parent dominates the candidate, otherwise, when the candidate and parent are nondominated with regard to each other, we randomly select one to add to the population.

3.3. Nondominated sorting

In this approach (Deb et al., 2000), each solution must be compared with every other solution in the population to find if it is dominated in order to sort a population according to the level of non-domination. For each solution i of a solutions set, two entities are calculated: n_i, the number of solutions which dominate the solution i, and S_i, a set of solutions that the solution i dominates. At the end of this procedure, all solutions in the first nondominated front F_1 have their domination count $n_i=0$. Now, for each solution i with $n_i=0$, it visits each member j of its set S_i and reduces its domination count by one. While doing so, if for any member j the domination count becomes zero then it is put in a separate list P. These members belong to the second nondominated front F_2. The above procedure is continued with each member of P and the third front F_3 is identified. This process continues until all fronts are identified.

3.4. Crowding distance sorting

Crowding distance (Deb et al., 2000) is used to get an estimate of the density of solutions surrounding a particular solution i in the population. The crowding degree estimation method is invoked in two situations. First, when target vector and trial vector do not dominate each other, we evaluate the crowding degree of the target vector and trial vector with respect to the nondominated solutions in the external archive. The less crowded one is chosen as the new target vector of the next generation. Secondly, when the external archive exceeds the prespecified size, the solutions located at the most crowded place should be detected and eliminated.

The crowding distance computation requires sorting the population according to each objective function value in ascending order of magnitude. Thereafter, for each objective function, the boundary solutions (solutions with smallest and largest function values) are assigned an infinite distance value. All other intermediate solutions are assigned a distance value equal to the absolute normalized difference in the function values of two adjacent solutions. This calculation is continued with other objective functions. The overall crowding-distance value is calculated as the sum of individual distance values corresponding to each objective. Each objective function is normalized before calculating the crowding distance.

3.5. Pseudocode for MOTLBO

As previously analyzed, the pseudocode of MOTLBO is summarized as follows.

3.6. Computational complexity analysis

Consider the complexity of one iteration of the entire algorithm. We define complexity here as the total number of function value comparisons, M and NP represent the number of objective functions and population size respectively. Basic operations and their worst case complexities are as follows:

1. Selection of NP solutions out of 2NP solutions (NP Archive solutions and NP new solutions) for next generation using nondominated sorting: $O(M \cdot (2NP)^2)$.
2. Selection of NP solutions out of 2NP solutions (NP Archive solutions and NP new solutions) for next generation using crowding distance sorting: $O(M \cdot (2NP) \cdot \log(2NP))$.
3. Procedure to check the domination status of new solution with target solution in teaching phase for one iteration: $O(M \cdot NP)$
4. Procedure to check the domination status of new solution with target solution in learning phase for one iteration: $O(M \cdot NP)$

Therefore, the overall computational complexity of the MOTLBO is less than or equal to $O(M \cdot (2NP)^2)$, which is in well agreement with the overall computational complexity of NSGA-II (Deb et al., 2000).

4. Simulation experiments

4.1. Performance measures

To test the performance of MOTLBO, some optimization problems are used in the experiments. Standard performance measures of multi-objective evolutionary algorithms have been used to evaluate the performance of the proposed algorithm. They represent both quantitative and qualitative comparisons with MOEAs. For these metrics we need to know the true Pareto front for a problem. In our experiments we use 1000 uniformly spaced Pareto optimal solutions as the approximation of the true Pareto front. The performance measures are described briefly as follows.

4.1.1. Generational distance (GD)

The metric called generational distance (GD) which is proposed by Van Veldhuizen and Lamont (1998) indicates the closeness of the Pareto optimal solutions obtained to the true Pareto optimal solutions. The mathematical description of GD are described as follows.

Let Q is a Pareto optimal solution set which obtained by a multi-objective evolutionary algorithm, the closeness of the Pareto optimal solution set Q obtained to the true Pareto optimal solution set (i.e. the Pareto front) PF is evaluated by GD defined by Eq. (4):

$$GD = \frac{\sqrt{\sum_{i=1}^{|Q|} d_i^2}}{|Q|} \tag{4}$$

where $|Q|$ is the number of the Pareto optimal solution set Q, d_i is the Euclidean distance between each solution and the nearest member of the true Pareto optimal solution set (i.e. the Pareto front) PF. d_i is defined as shown in Eq. (5):

$$d_i = \min\left\{ \sqrt{\sum_{j=1}^{M} \left(f_j(i) - f_j^*(k)\right)} \quad k = 1,2,\ldots,|PF| \right\} \tag{5}$$

where M is the number of the goals, $|\mathrm{PF}|$ is the number of reference vector solution set, $f_j^*(k)$ is the jth objective function value of kth member of the true Pareto optimal solution set (i.e. the Pareto front) PF, $f_j(i)$ is the jth objective function value of the ith member of the Pareto optimal solution set Q.

When the result is zero, it indicates that the Pareto optimal solution set Q obtained by the algorithm is same as the true Pareto optimal solution set (i.e. the Pareto front) PF, any other value indicates the Pareto optimal solution set Q obtained by the algorithm deviates from the true Pareto optimal solution set (i.e. the Pareto front PF).

4.1.2. Spacing metric (SP)

The spacing metric (SP) which is proposed by Schott (1995) aims at assessing the spread (distribution) of the Pareto optimal solution set Q obtained by the algorithm. This metric is measured by evaluating the variance of the nearest distance between neighbor solutions obtained by the algorithm and it is defined by Eq. (6):

$$SP = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} \left(\overline{d} - d_i\right)^2} \tag{6}$$

where n is the number of solutions in the Pareto optimal solution set Q obtained by the algorithm, d_i is the distance between each solution and the nearest member of the Pareto optimal solution set obtained by the algorithm, $\overline{d}$ is the average value for all d_i. d_i and $\overline{d}$ are defined by Eqs. (7) and (8), respectively.

$$d_i = \min_j \left(\sum_{m=1}^{M} f_i^m - f_j^m \quad i,j = 1,2,\ldots,n \quad i \neq j \right) \tag{7}$$

$$\overline{d} = \frac{1}{n-1} \sum_{i=1}^{n-1} d_i \tag{8}$$

where n is the number of solutions in the Pareto optimal solution set Q obtained by the algorithm, M is the number of the goals, f_i^m is the mth objective function value of the ith member of the Pareto optimal solution set Q, f_j^m is the mth objective function value of the jth member of the Pareto optimal solution set Q.

Lower value of SP indicates a more even spread of the Pareto optimal solution set Q obtained by the algorithm, and a value of zero suggests that the Pareto optimal solution set Q obtained by the algorithm are evenly spread out.

4.2. Test problems

The performance of the proposed algorithm is tested on a set of 6 unconstrained benchmark problems generally used to validate the performance of different MOEAs. Here GD, SP and Time (the computation time of the algorithm) have been used to evaluate the performance of the proposed algorithm and 6 benchmark problems which we have taken are SCH, DEB, FON, ZDT1, ZDT3 and ZDT6. All these problems have two objective functions. We describe these problems in Table 1. also shows the number of variables, their bounds, the Pareto-optimal solutions, and the nature of the Pareto-optimal front for each problem.

4.3. Comparison of the results

In order to illustrate the efficiency of our approach, it is compared to NSGA-II, MOPSO-CD and RM-MEDA. The codes of these algorithms were programmed by the authors of the paper according to the algorithms introduced in the references. All algorithms were compiled in MATLAB 7.9 and are executed on Intel Pentium 4(R) 3.00 GHz PC with 512MB RAM,.

In this paper, NP of all algorithms are set to 100 respectively. Maximum number of function evaluations is set to 10,000 for SCH, DEB, FON, ZDT1, ZDT3 and 50,000 for ZDT6. NSGA-II was run using a crossover probability of 0.9, tournament selection, a mutation rate of 1/n (where n is the number of decision variables), and distribution indexes for crossover and mutation operators as $\eta_c=20$ and $\eta_m=20$, respectively. In MOPSO-CD, $c_1=1.0$, $c_2=0.5$, r_1 and r_2 are random values between 0 and 1, inertial weight w might a constant value. In RM-MEDA, the number of clusters of Local PCA algorithm is set to be 5. All the experiments are run 20 times for each problem. The values of the two metrics for each algorithm are presented in Table 2. To display the difference of optimal Pareto front between the given method and the real one clearly, the best optimal Pareto fronts of NSGA-II, MOPSO-CD, RM-MEDA and MOTLBO are shown in Table 3.

For SCH with convex Pareto front, it can be seen from in Table 3 that except RM-MEDA, which is based on the regularity property that the Pareto set of a continuous multi-objective optimization problem is a piecewise continuous (m-1)-D manifold but the dimension of SCH is one, all other algorithms are able to find solutions near the global Pareto front. Considering all of the metrics from Table 2, the average performance of our algorithm is the best with respect to GD and Time. MOPSO-CD is the best with respect to SP but the computation time required for this algorithm is more that the same for our algorithm.

For DEB with disconnected Pareto front, it can be seen from in Table 3 that all other algorithms are able to find solutions near the global Pareto front. It can be seen from Table 2 that the average performance of our algorithm ranks second with respect to SP (MOPSO-CD has the best one) and our algorithm has the worst average performance with respect to GD (MOPSO-CD has the

Table 1
Test problems.

Problem	Function (minimization)	D	Variable bounds	Optimal solutions	Character of PF
SCH	$f_1(x)=x^2$ $f_2(x)=(x-2)^2$	1	$x_i\in[-10^3,10^3]$	$x\in[0,2]$	Convex
DEB	$f_1(x)=x_1$ $f_2(x)=(1+10x_2)[1-(x_1/1+10x_2)^2-(x_1/1+10x_2)\sin(8\pi x_1)]$	2	$x_i\in[0,1]$	$x_1\in[0,1]$ $x_i=0(i\neq 1)$	Convex
FON	$f_1(x)=1-\exp\left(-\sum_{i=1}^{3}x_i-1/\sqrt{3}\right)^2$ $f_2(x)=1-\exp\left(-\sum_{i=1}^{3}x_i+1/\sqrt{3}\right)^2$	3	$x_i\in[-4,4]$	$x_1=x_2=x_3$ $\in\left[-1/\sqrt{3},1/\sqrt{3}\right]$	Nonconvex
ZDT1	$f_1(x)=x_1$ $f_2(x)=g(x)\left(1-\sqrt{f_1(x)/g(x)}\right)$ $g\left(x\right)=1+9\sum_{i=2}^{n}x_i/(n-1)$	30	$x_i\in[0,1]$	$x_1\in[0,1]$ $x_i=0(i\neq 1)$	Convex
ZDT3	$f_1(x)=x_1$ $f_2(x)=g(x)\left(1-\sqrt{f_1(x)/g(x)}-(f_1(x)/g(x))\sin(10\pi x_1)\right)$ $g\left(x\right)=1+9\sum_{i=2}^{n}x_i/(n-1)$	30	$x_i\in[0,1]$	$x_1\in[0,1]$ $x_i=0(i\neq 1)$	Convex
ZDT6	$f_1(x)=1-\exp\left(-4x_1\right)\sin^6(6\pi x_1)$ $f_2(x)=g(x)\left(1-(f_1(x)/g(x))^2\right)$ $g(x)=1+9\left[\sum_{i=2}^{n}x_i/n-1\right]^{0.25}$	10	$x_i\in[0,1]$	$x_1\in[0,1]$ $x_i=0(i\neq 1)$	Nonconvex

Table 2
Statistics of the results on the test problems.

Fun	Metric	NSGA-II	MOPSO-CD	RM-MEDA	MOTLBO
SCH	GD	0.000378 ± 0.000014	0.000389 ± 0.000024	–	$\mathbf{0.000358\pm0.000014}$
	SP	0.030155 ± 0.004266	$\mathbf{0.027766\pm0.004302}$	–	0.027800 ± 0.003712
	Time	9.115113 ± 0.2766890	4.059689 ± 0.125918	–	$\mathbf{3.047368\pm0.030585}$
DEB	GD	0.002110 ± 0.003857	$\mathbf{0.000384\pm0.000032}$	0.000401 ± 0.000115	0.000399 ± 0.000021
	SP	0.007297 ± 0.000441	$\mathbf{0.005691\pm0.000596}$	0.009392 ± 0.002597	0.010104 ± 0.00249
	Time	9.544951 ± 0.322534	5.133636 ± 0.191751	17.882322 ± 0.071690	$\mathbf{3.498872\pm0.006000}$
FON	GD	0.002854 ± 0.000617	0.002820 ± 0.000160	0.002921 ± 0.000288	$\mathbf{0.002691\pm0.000100}$
	SP	0.007401 ± 0.000435	$\mathbf{0.005545\pm0.000873}$	0.009379 ± 0.000882	0.005598 ± 0.000692
	Time	10.524940 ± 0.845527	4.374676 ± 0.138059	20.714957 ± 0.331591	$\mathbf{3.185973\pm0.095462}$
ZDT1	GD	0.168330 ± 0.013318	$\mathbf{0.000274\pm0.000034}$	0.002867 ± 0.000180	0.000625 ± 0.000106
	SP	0.011166 ± 0.002099	0.006334 ± 0.000340	0.011011 ± 0.001581	$\mathbf{0.006252\pm0.000868}$
	Time	11.175467 ± 0.669782	4.883601 ± 0.372220	20.661726 ± 0.217293	$\mathbf{3.091229\pm0.099618}$
ZDT3	GD	0.165879 ± 0.010088	0.001724 ± 0.000353	0.002173 ± 0.000280	$\mathbf{0.000962\pm0.000482}$
	SP	0.011793 ± 0.001047	$\mathbf{0.007426\pm0.001191}$	0.011890 ± 0.001691	0.013946 ± 0.004921
	Time	10.988647 ± 0.727347	4.459787 ± 0.321976	20.426158 ± 0.684494	$\mathbf{3.197631\pm0.160441}$
ZDT6	GD	0.183056 ± 0.035606	0.063544 ± 0.013625	0.019598 ± 0.003118	$\mathbf{0.006778\pm0.012506}$
	SP	0.088437 ± 0.099184	0.178535 ± 0.103466	$\mathbf{0.008927\pm0.002258}$	0.014462 ± 0.066016
	Time	59.831565 ± 3.393011	17.615634 ± 2.455597	99.511384 ± 9.800831	$\mathbf{16.786678\pm0.387123}$

The bold values in Table 2 represent the best results among the algorithms in terms of the mean values and average deviations.

best one), but it has the best average performance with respect to Time.

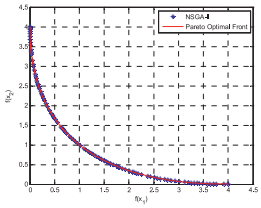
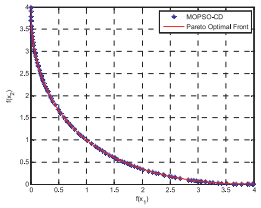
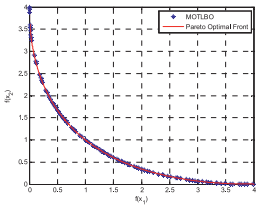
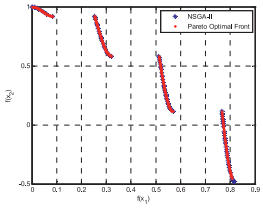
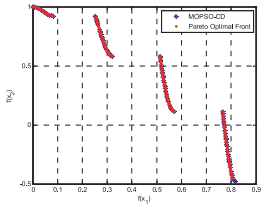
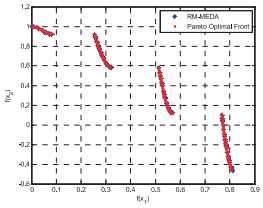
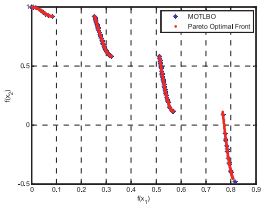
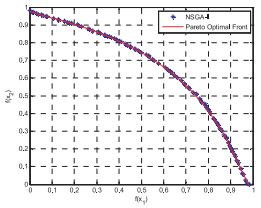
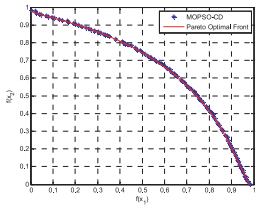
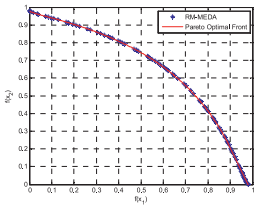
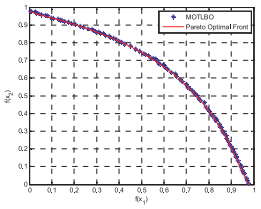
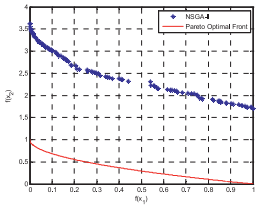
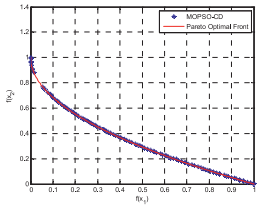
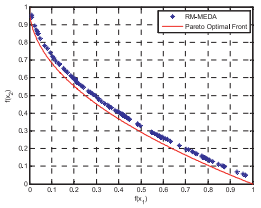
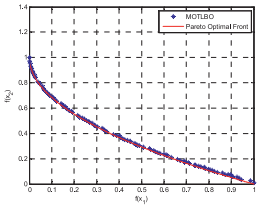
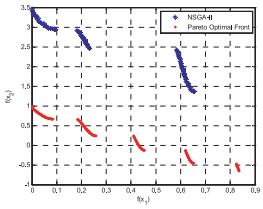
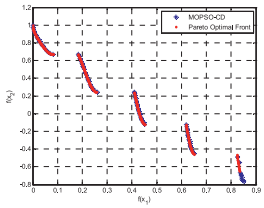
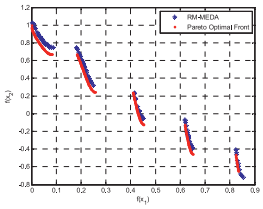
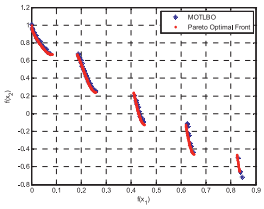
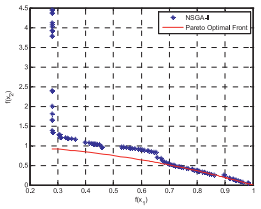
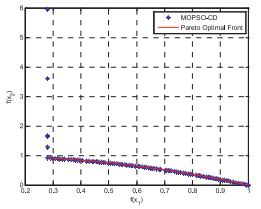
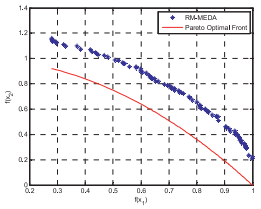
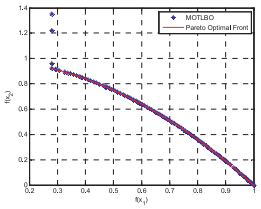
For FON with non-convex Pareto front, it can be seen from in Table 3 that all other algorithms are able to find solutions near the global Pareto front. Our algorithm has the best average performance with respect to GD and Time, but the average performance of our algorithm ranks second with respect to SP (MOPSO-CD has the best one).

For ZDT1 with convex Pareto front, it can be seen from in Table 3 that both NSGA-II and RM-MEDA still failed to find its true Pareto front or its approximation within 10,000 evaluations. MOPSO-CD has the beat performance with respect to GD and our algorithm has the best average performances with respect to SP and Time.

For ZDT3 with disconnected Pareto front, it can be seen from in Table 3 that both NSGA-II and RM-MEDA still failed to find its true Pareto front or its approximation within 10,000 evaluations. Our algorithm has the best average performances with respect to GD and Time but but it has the worst average performance with respect to SP (MOPSO-CD has the best one).

For ZDT6 with non-convex Pareto front, RM-MEDA still failed to find its true Pareto front or its approximation within 50,000 evaluations. Our algorithm has the best average performances with respect to GD and Time but but the average performance of

Table 3
The results between the true Pareto front and the Pareto front.

Fun	NSGA-II	MOPSO-CD	RM-MEDA	MOTLBO
SCH				
DEB				
FON				
ZDT1				
ZDT3				
ZDT6				

our algorithm ranks third with respect to SP (RM-MEDA has the best one). In addition, it can be seen from in Table 3 that NSGA-II have the most solutions which located far from the true Pareto front and our algorithm have the least.

5. Constrained optimization examples

To deeply test the performance of the given algorithm, two typical constrained optimization problems are used in experiments. All other parameters are similar to unconstrained optimization problems. The number of iterations is 100 for the four methods.

5.1. Two-Bar truss design

This problem was originally studied using the ε-constraint method (Pallli et al., 1999). The truss in Figs. 1 and 2 has to carry a certain load without elastic failure. Thus, in addition to the objective of designing the truss for minimum volume (which is equivalent to designing for minimum cost of fabrication), there are additional objectives of minimizing stresses in each of the two members AC and BC.We construct the following two-objective optimization problem for three variables x_1 (length of AC in m), x_2(length of BC in m) and x_3(vertical distance between B and C in m).

- Initialize the values of NP, FES(number of function evaluations)=0, MaxFES(maximum FES)
- Input lower and upper bounds on decision variables xmin[n] and xmax[n].
- Generate NP random solutions using uniform distribution.
- Evaluate function values at these NP solutions.
- Adopt nondominated and crowding distance sorting to these NP solutions and store them in current archive.
- While (FES<maxFES) // MODEA main loop starts here.
  - Select the centroid of the nondominated solutions from current archive as the Mean
  - for i=1:Np //Teaching phase
    - Select randomly the nondominated best from current archive as Teacher
    - Generate a trial individual Ui:Ui=Xi+rand*(Teacher-(1+rand*Mean))
    - Evaluate function value; FES++;
    - Nondomination checking of trial individual Ui with target individual Xi.
    - If (Ui dominates Xi) Replace Xi by Ui
    - elseif(Ui and Xi non-dominates each other) Select randomly a individual and replace Xi; endif
  - endfor
  - for i=1:Np //Learning phase
    - Select randomly a individual individual Vi different from target individual Xi
    - If (Ui dominates Xi) Ui=Xi+(Xi-Vi); Elseif Ui=Xi+(Vi-Xi); Endif
    - Evaluate function value; FES++;
    - Nondomination checking of trial individual Ui with target individual Xi.
    - If (Ui dominates Xi) Replace Xi by Ui;
    - elseif (Ui and Xi non-dominates each other) Select randomly a individual and replace Xi;
    - endif
  - endfor
  - Select NP fittest solutions using nondominated and crowding distance sorting from these 2NP solutions in current population and archive and store them in current archive.
- endwhile // MODEA main loop ends here.

Fig. 1. Pseudocode of MOTLBO.

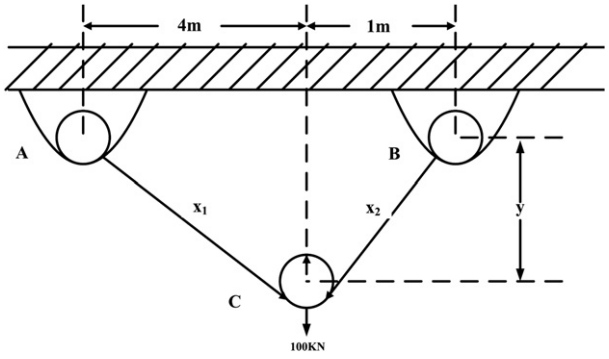


Fig. 2. Two-bar truss design.

The mathematical description of two bar truss design problem can be expressed by Eq. (9):

$$\begin{aligned} &Minimize: f_1(x) = x_1\sqrt{16+x_3^2}+x_2\sqrt{1+x_3^2} \\ &\qquad f_2(x) = \max(\sigma_{AC},\sigma_{BC}) \\ &subject\ \ to: \max(\sigma_{AC},\sigma_{BC}) \le 1\left(10^5\right) \\ &\qquad x1 \ge 0 \\ &\qquad x2 \ge 0 \\ &\qquad 1 \le x_3 \le 3 \\ &where: \sigma_{AC} = \frac{20\sqrt{16+x_3^2}}{x_1x_3},\ \ \sigma_{BC} = \frac{80\sqrt{1+x_3^2}}{x_2x_3} \end{aligned} \tag{9}$$

The original study reported only five solutions with the following spread: (0.004445 m^3, 89,983 kPa) and (0.004833 m^3, 83,268 kPa). Fig. 3 shows the optimized fronts obtained using NSGAII, MOPSO-CD, RM-MEDA and MOTLBO methods after 100 iterations.

It can be seen from Table 4 that the four methods have a wide variety of alternatives. If minimum volume is desired, MOPSO-CD gives a value as low as 0.004214 m^3, and MOTLBO gives a value as low as 0.004174 m^3. If minimization of stress is important, MOTLBO finds a solution with stress as low as 8431.376521 kPa. MOTLBO has good performance in variable domain of stress and MOPSO-CD has good performance in variable domain of volume. The MOTLBO solutions are very competitive with MOPSO-CD solutions in terms of both closeness to the true optimum front and their spread, and these two methods are all better than NSGAII and RM-MEDA.

5.2. 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):

$$\begin{aligned} &Minimize: f_1(x) = 2x_2x_4+x_3(x_1-2x_4) \\ &\qquad f_2(x) = \frac{PL^2}{48EI} \\ &subject\ to: g(x) = \frac{M_y}{Z_z}+\frac{M_z}{Z_z}-\sigma_\alpha \le 0 \\ &\qquad 10 \le x_1 \le 80, 10 \le x_2 \le 50, 1 \le x_3 \le 3, 0.9 \le x_4 \le 5 \end{aligned}$$

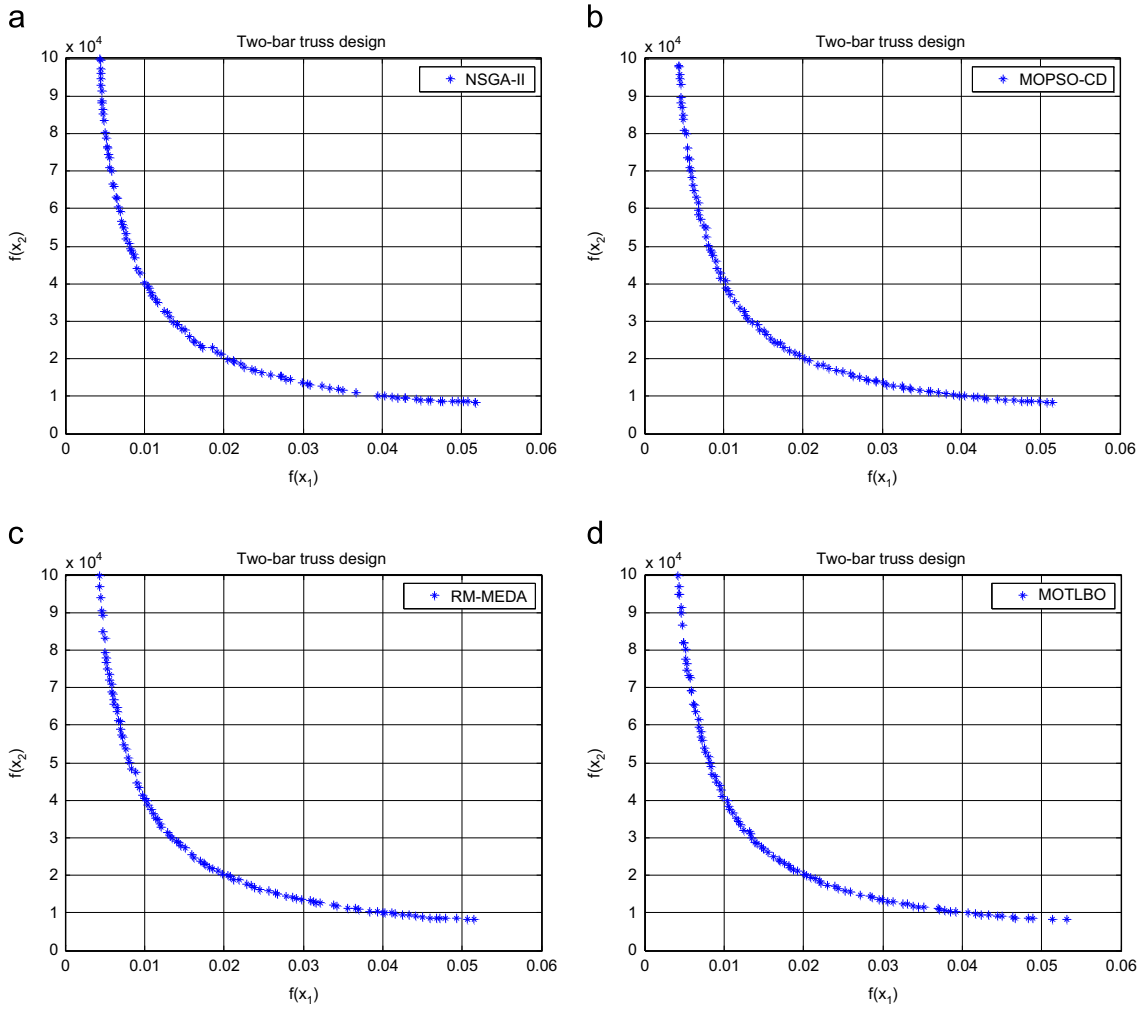


Fig. 3. Two-bar truss design problem. (a) NSGAII, (b) MOPSO-CD, (c) RM-MEDA and (d) MOTLBO. (a) NSGAII, (b) MOPSO-CD, (c) RM-MEDA and (d) MOTLBO.

Table 4
Comparison of the best solution found.

Algorithm	min($f1$)	max($f2$)	max($f1$)	min($f2$)
NSGA-II	0.004254	99825.169387	0.052143	8448.256317
MOPSO-CD	0.004214	99911.764943	0.051937	8432.740427
RM-MEDA	0.004295	98148.392335	0.052462	8451.563276
MOTLBO	0.004384	96976.046902	0.051733	8431.376521

$$
\begin{aligned}
&where:\ I=\frac{1}{12}\left\{x_3(x_1-2x_4)^3+2x_2x_4\left[4x_4^2+3x_1(x_1-2x_4)\right]\right\}\\
&M_y=\frac{PL}{4},\ M_z=\frac{QL}{4}\\
&Z_y=\frac{1}{6x_1}\left\{x_3(x_1-2x_4)^3+2x_2x_4[4x_4^2+3x_1(x_1-2x_4)]\right\}\\
&Z_z=\frac{1}{6x_2}\left[(x_1-x_4)x_3^3+2x_4x_2^3\right]\\
&E=2\times10^4\ \text{kN/cm}^2,\sigma_\alpha=16\ \text{kN/cm}^2\\
&P=600\ \text{kN},Q=50\ \text{kN},L=200\ \text{cm}
\end{aligned}
\tag{10}
$$

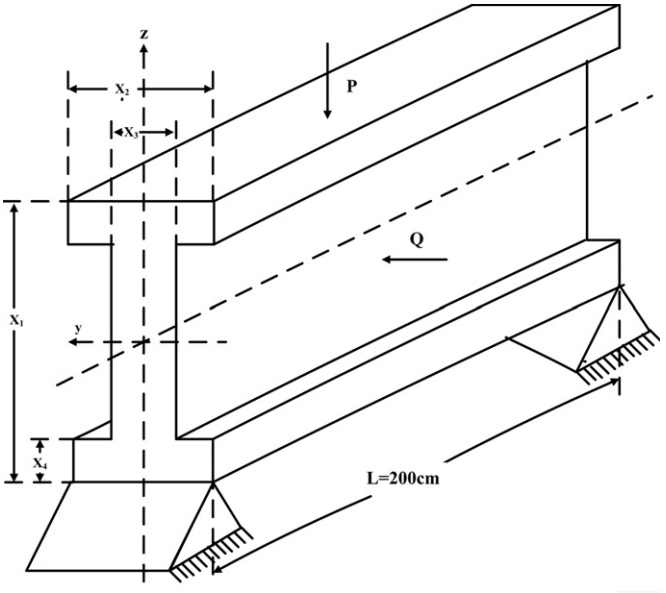


Fig. 4. *I*-beam design.

The Pareto optimal solutions obtained using NSGAII, MOPSO-CD, RM-MEDA and MOTLBO methods are shown in Fig. 5 after 100 iterations. The results from Table 5 indicate that the minimal cross sectional area of MOTLBO is the smallest among the four methods, and the minimal deflections of four algorithms are equal. Based on these points we can say that the MOTLBO are

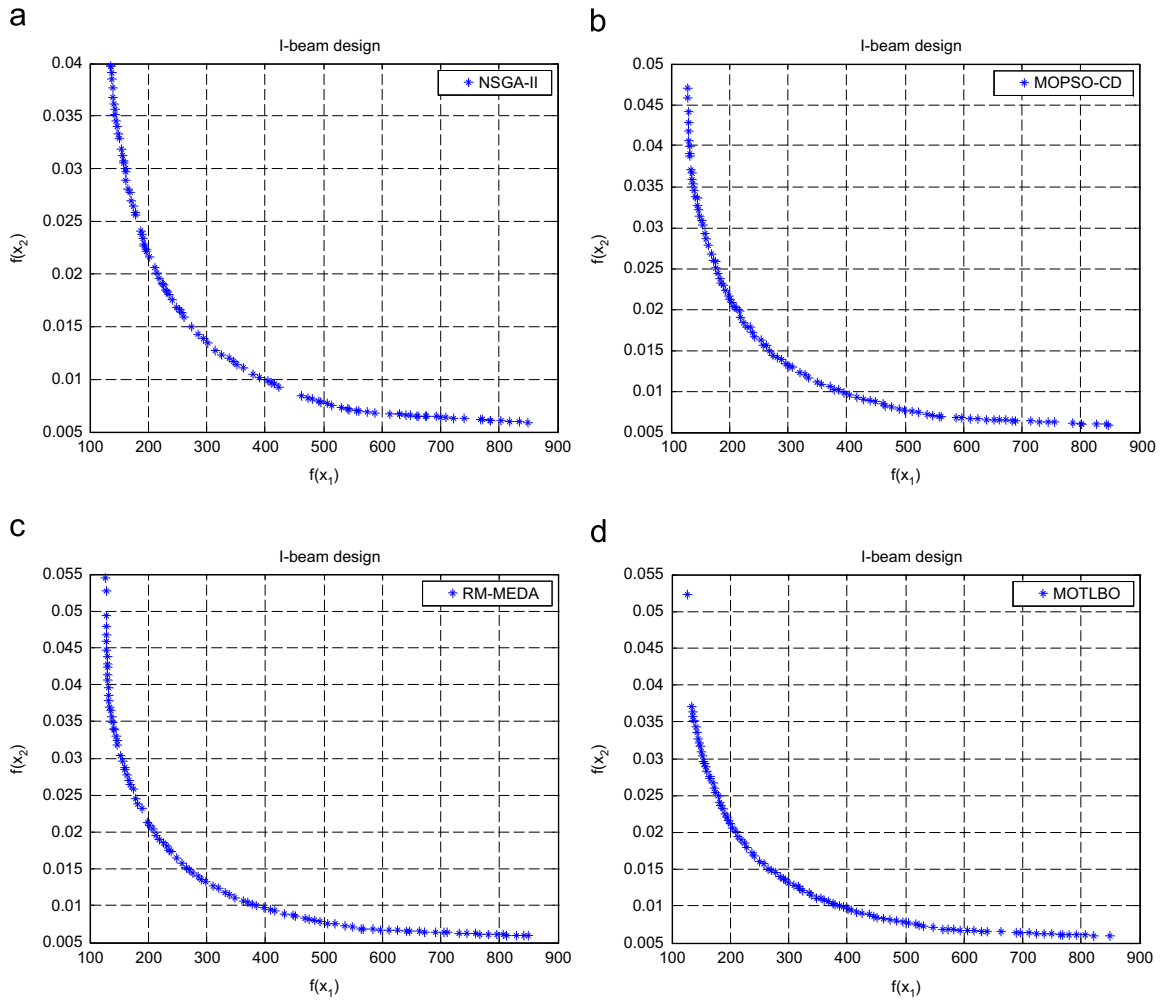


Fig. 5. *I*-beam design problem. (a) NSGAII, (b) MOPSO-CD, (c) RM-MEDA and (d) MOTLBO.

Table 5
Comparison of the best solution found.

Algorithm	min($f1$)	max($f2$)	max($f1$)	min($f2$)
NSGA-II	135.555108	0.039744	850.000000	0.005903
MOPSO-CD	128.170895	0.049401	850.000000	0.005903
RM-MEDA	127.390955	0.054075	850.000000	0.005903
MOTLBO	126.705114	0.052304	850.000000	0.005903

very competitive with other three algorithms in terms of cross-sectional area of the beam.

6. Conclusions

In this paper, we propose a teaching-learning-based optimization algorithm for multi-objective optimization problems. Our MOTLBO adopt the nondominated sorting concept and the mechanism of crowding distance computation. The Pareto fronts of the solutions are guided by the teacher which is the best learner and the mean of learners achieved so far. The efficiency and effectiveness of the proposed MOTLBO are evaluated using 6 unconstrained benchmark test problems with convex and non-convex objective functions and 2 constrained real-word multi-objective problems. The experimental results show that the average Running Time of MOTLBO is the least for all 6 unconstrained benchmark test problems, the average Generational Distance (GD) of MOTLBO is the least for SCH, FON, ZDT3 and ZDT6 and the average Spacing metric (SP) of MOTLBO is the least for ZDT1. It should be noted that the distribution of the Pareto optimal solution set obtained by MOTLBO need to be improved. In summary, the proposed MOTLBO algorithm is a challenging method for multi-objective optimization problems.

In the near future, we also plan to improve the distribution of the Pareto optimal solution set obtained by MOTLBO algorithm and apply it for solving dynamic multi-objective optimization problems.

Acknowledgement

This research was partially supported by National Natural Science Foundation of China (61272283, 61073091, 61100173), Social Science Foundation of Hebei Province (HB11JY006), Scientific Research Plan Foundation in Higher Educational Institutes of Hebei (SZ2011334).

References

Coello Coello, C.A., Pulido, G.T., Lechuga, M.S., 2004. Handling multiple objectives with particle swarm optimization. IEEE Trans. Evol. Computat. 8 (3), 256–279.

Deb, K., Pratab, A., Agrawal, S., Meyarivan., T., 2000. A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6 (2), 182–197.

Deb, K., 2001. Multi-objective Optimization using Evolutionary Algorithms. John Wiley & Sons Ltd., England.

David A. Van Veldhuizen, Gary B.Lamont. Evolutionary computation and convergence to a pareto front. In: John, R. Koza (editor) Late Breaking Papers at the Genetic Programming 1998 Conference, Stanford University, California, 1998, July 1998, pp. 221–228.

Huang, V.L., Zhao, S.Z., Mallipeddi, R., Suganthan, P.N. Multi-objective optimization using self adaptive differential evolution algorithm. In Proceedings of the Congress on Evolutionary Computation 2009 (CEC'2009), 190-194(2009).

J. Horn, N. Nafpliotis, and D.E. Goldberg A niched pareto genetic algorithm for multi-objective optimization. Proceedings of the First IEEE Conference on Evolutionary Computation. 1: 82-87(1994).

Knowles, J., Corne, D., 2000. Approximating the nondominated front using the pareto archived evolution strategy. Evol. Comput. 8 (2), 149–172.

Larrañaga, P., Lozano, J.A. (Eds.), 2001. Kluwer, Norwell, MA.

Mark Erickson, Alex Mayer, Jeffrey Horn. The Niched Pareto Genetic Algorithm 2 Applied to the Design of Groundwater Remediation Systems. First International Conference on Evolutionary Multi-Criterion Optimization, Lecture Notes in Computer Science, No. 1993, 681-695(2001).

Niknam, Taher, Golestaneh, Faranak, Sadeghi, Mokhtar Sha, 2012. θ-Multi-objective teaching–learning-based optimization for dynamic economic emission dispatch. IEEE Syst. J. 6 (2), 341–352.

Niknam, T., et al., 2012. A new multi objective optimization approach based on TLBO for location of automatic voltage regulators in distribution systems. Eng. Appl. Artif. Intel..

Pallli, N., Azram, S., McCluskey, P., et al., 1999. An interactive multistage ε-inequality constraint method for multiple objectives decision making. ASME J. Mech. Des. 120 (4), 678–686.

Pelikan, M., Sastry, K., Goldberg., D., 2005. Multi-objective HBOA, clustering, and scalability. Illinois Genetic Algorithms Laboratory (IlliGAL). Tech. Rep., 2005005.

Rao, R.V., Savsani, V.J., Vakharia, D.P., 2011a. Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput. Aid. Des. 43 (3), 303–315.

Rao, R.V., Savsani, V.J., Vakharia, D.P., 2012. Teaching-learning-based optimization: an optimization method for continuous non-linear large scale problems. Inf. Sci. 183 (1), 1–15.

Rao, R.Venkata, Patel, Vivek, 2012. An elitist teaching-learning-ba sed optimization algorithm for solving complex constrained optimization problems. Int. J. Indus. Eng. Comput. 3, 535–560.

Rao, R.Venkata, Patel, Vivek, 2011. Multi-objective optimization of combined Brayton and inverse Brayton cycles using advanced optimization algorithms. Eng. Optimiz.

Rao, R.V., Savsania, V.J., Balic, J., 2011b. Teaching-learning-based optimization algorithm for unconstrained and constrained real-parameter optimization problems. Eng. Optimiz.

Rao, R.V., Kalyankar, V.D., 2012. Parameter optimization of modern machining processes using teaching-learning-based optimization algorithm. Eng. Appl. Artif. Intel.

Rao, R.V., Patel, V., 2012a. Multi-objective optimization of heat exchangers using a modified teaching-learning-based optimization algorithm. Appl. Math. Modell..

Rao, R.V., Patel, V., 2012b. Multi-objective optimization of two stage thermoelectric cooler using a modified teaching–learning-based optimization algorithm. Eng. Appl. Artif. Intel.

Srinivas, N., Deb, K., 1994. Multi-objective optimization using nondominated sorting in genetic algorithms. Evol. Comput. 2 (3), 221–248.

Schaffer, J.D., 1985. Multiple objective optimization with vector evaluated genetic algorithm. Proceedings of the First International Conference on Genetic Algorithms, 93–100.

Satapathy, S.C., et al., 2012. High dimensional real parameter optimization with teaching learning based optimization. Int. J. Indus. Eng. Comput.

Schott, J.R. Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization. Master's Thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts, May 1995.

Togan, V., 2012. Design of planar steel frames using teaching-learning based optimization. Eng. Struct. 34, 225–232.

Xue, F., Sanderson, A.C., Graves, R.J., 2003. Pareto-based multi-objective differential evolution. Proceedings of the Congress on Evolutionary Computation 2003 (CEC'2003) 2, 862–869.

Yang, B.S., Yeun, Y.S., Ruy, W.S., 2002. Managing approximation models in multi-objective optimization. Struct. Multidiscip. Optim. 24, 141–156.

Zhang, Qingfu, Zhou, Aimin, Jin, Yaochu, 2008. RM-MEDA: a regularity model-based multi-objective estimation of distribution algorithm. IEEE Trans. Evol. Computat. 12 (1), 182–197.

Zhang, Q., Li, H., 2007. MOEA/D: a multi-objective evolutionary algorithm based on decomposition. IEEE Trans. Evol. Computat. 11 (6), 712–731.

Zitzler, E., Thiele, L., 1999. Multi-objective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Trans. Evol. Comput. 3 (4), 257–271.