International Journal of Industrial Engineering Computations 3 (2012) 535–560

Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

An elitist teaching-learning-based optimization algorithm for solving complex constrained optimization problems

R. Venkata Rao* and Vivek Patel

Department of Mechanical Engineering, S.V. National Institute of Technology, Ichchanath, Surat, Gujarat – 395 007, India

ARTICLEINFO

Article history:
Received 25 January 2012
Accepted March 10 2012
Available online
15 March 2012
Keywords:
Teaching-learning-based optimization
Elitism
Population size
Number of generations
Constrained optimization problems

ABSTRACT

Nature inspired population based algorithms is a research field which simulates different natural phenomena to solve a wide range of problems. Researchers have proposed several algorithms considering different natural phenomena. Teaching-Learning-based optimization (TLBO) is one of the recently proposed population based algorithms which simulates the teaching-learning process of the class room. This algorithm does not require any algorithm-specific control parameters. In this paper, elitism concept is introduced in the TLBO algorithm and its effect on the performance of the algorithm is investigated. The effects of common controlling parameters such as the population size and the number of generations on the performance of the algorithm are also investigated. The proposed algorithm is tested on 35 constrained benchmark functions with different characteristics and the performance of the algorithm is compared with that of other well known optimization algorithms. The proposed algorithm can be applied to various optimization problems of the industrial environment.

1. Introduction

The difficulties associated with mathematical optimization on large-scale engineering problems have contributed to the development of alternative solutions. Traditional methods like linear programming, dynamic programming etc. often fail (or trapped at local optimum) while solving multimodal problems having large number of variables and non-linear objective functions. To overcome these problems, several modern heuristic algorithms have been developed for searching near-optimum solutions to the problems. These algorithms can be classified into different groups depending on the criteria being considered such as population based, iterative based, stochastic, deterministic, etc. Depending on the nature of phenomenon simulated by the algorithms, the population based heuristic algorithms have two important groups: evolutionary algorithms (EA) and swarm intelligence based algorithms.

Some of the recognized evolutionary algorithms are, Genetic Algorithms (GA), Evolution Strategy (ES), Evolution Programming (EP), Differential Evolution (DE), Bacteria Foraging Optimization

* Corresponding author. Tel.: +91-261-2201661, Fax: +91-261-2201571
E-mail ravipudirao@gmail.com (R. V. Rao)

doi: 10.5267/j.ijiec.2012.03.007

(BFO), Artificial Immune Algorithm (AIA), etc. Among all, GA is a widely used algorithm for various applications. GA works on the principle of the Darwinian theory of the survival of the fittest and the theory of evolution of the living beings (Holland, 1975). ES is based on the hypothesis that during the biological evolution the laws of heredity have been developed for fastest phylogenetic adaptation (Runarsson & Yao, 2000). ES imitates, in contrast to the GA, the effects of genetic procedures on the phenotype. EP also simulates the phenomenon of natural evolution at phenotype level (Fogel et al., 1996). DE is similar to GA with specialized crossover and selection method (Storn & Price, 1997; Price et al., 2005).

BFO is inspired by the social foraging behavior of Escherichia coli (Passino, 2002). AIA works on the immune system of the human being (Farmer et al., 1986). Some of the well known swarm intelligence based algorithms are, Particle Swarm Optimization (PSO) which works on the principle of foraging behavior of the swarm of birds (Kennedy & Eberhart,1995); Shuffled Frog Leaping (SFL) algorithm which works on the principle of communication among the frogs (Eusuff & Lansey, 2003); Ant Colony Optimization (ACO) which works on the principle of foraging behavior of the ant for the food (Dorigo et al.,1991); Artificial Bee Colony (ABC) algorithm which works on the principle of foraging behavior of a honey bee (Karaboga,2005; Basturk & Karaboga, 2006; Karboga & Basturk, 2007; Karaboga & Basturk,2008).

Beside the above mentioned evolutionary and swarm intelligence based algorithms, there are some other algorithms which work on the principles of different natural phenomena. Some of them are: Harmony Search (HS) algorithm which works on the principle of music improvisation in a music player (Geem et al.,2001); Gravitational Search Algorithm (GSA) which works on the principle of gravitational force acting between the bodies (Rashedi et al.,2009); Biogeography-Based Optimization (BBO) which works on the principle of immigration and emigration of the species from one place to the other (Simon,2008); and Grenade Explosion Method (GEM) which works on the principle of explosion of grenade (Ahrari & Atai, 2010).

All the evolutionary and swarm intelligence based algorithms are probabilistic algorithms and require common controlling parameters like population size, number of generations, elite size, etc. In addition to the common control parameters, different algorithm requires its own algorithm specific control parameters. For example, GA uses mutation rate and crossover rate. Similarly PSO uses inertia weight, social and cognitive parameters. The proper tuning of the algorithm specific parameters is very crucial factor, which affect the performance of the above mentioned algorithms. The improper tuning of algorithm-specific parameters either increases the computational effort or yields the local optimal solution. Considering this fact, recently Rao et al. (2011, 2012), Rao & Savsani (2012) and Rao & Patel (2012) introduced the Teaching-Learning-Based Optimization (TLBO) algorithm which does not require any algorithm-specific parameters. TLBO requires only common controlling parameters like population size and number of generations for its working. In this way TLBO can be said as an algorithm-specific parameter-less algorithm.

Elitism is a mechanism to preserve the best individuals from generation to generation. By this way, the system never loses the best individuals found during the optimization process. Elitism can be done by placing one or more of the best individuals directly into the population for the next generation. In the present work, the performance of TLBO algorithm is investigated for different elite sizes, population sizes and number of generations considering various constrained bench mark problems available in the literature.

2. Teaching-learning-based optimization (TLBO)

TLBO is a teaching-learning process inspired algorithm proposed by Rao et al. (2011, 2012), Rao and Savsani (2012) and Rao and Patel (2012) based on the effect of influence of a teacher on the output of learners in a class. The algorithm describes two basic modes of the learning: (i) through teacher (known as teacher phase) and (ii) interacting with the other learners (known as learner phase). In this optimization algorithm a group of learners is considered as population and different subjects offered to the learners are considered as different design variables of the optimization problem and a learner's result is analogous to the 'fitness' value of the optimization problem. The best solution in the entire population is considered as the teacher. The design variables are actually the parameters involved in the objective function of the given optimization problem and the best solution is the best value of the objective function. The working of TLBO is divided into two parts, 'Teacher phase' and 'Learner phase'.

2.1 Teacher phase

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 $M_{j,i}$ be the mean result of the learners in a particular subject 'j' (j=1,2,…,m). The best overall result $X_{total\text{-}kbest,i}$ 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,

$$Difference\_Mean_{j,k,i} = r_i\ (X_{j,kbest,i} - T_F M_{j,i}), \tag{1}$$

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

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

T_F is not a parameter of the TLBO algorithm. The value of T_F is not given as an input to the algorithm and its value is randomly decided by the algorithm using Eq. (2). After conducting a number of experiments on many benchmark functions it is concluded that the algorithm performs better if the value of T_F 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.(2).
Based on the $Difference\_Mean_{j,k,i}$, the existing solution is updated in the teacher phase according to the following expression.

$$X'_{j,k,i} = X_{j,k,i} + Difference\_Mean_{j,k,I}, \tag{3}$$

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. The learner phase depends upon the teacher phase.

2.2 Learner phase

Learners increase their knowledge by interaction among themselves. A learner interacts randomly with other learners for enhancing his or her knowledge. A learner learns new things if the other learner has more knowledge than him or her. Considering a population size of 'n', the learning phenomenon of this phase is expressed below.

Randomly select two learners P and Q such that $X'_{total\text{-}P,i} \neq X'_{total\text{-}Q,i}$ (where, $X'_{total\text{-}P,i}$ and $X'_{total\text{-}Q,i}$ are the updated values of $X_{total\text{-}P,i}$ and $X_{total\text{-}Q,i}$ respectively at the end of teacher phase)

$$X''_{j,P,i} = X'_{j,P,i} + r_i\,(X'_{j,P,i} - X'_{j,Q,i}),\ \text{If } X'_{total\text{-}P,i} < X'_{total\text{-}Q,i} \tag{4a}$$

$$X''_{j,P,i} = X'_{j,P,i} + r_i\,(X'_{j,Q,i} - X'_{j,P,i}),\ \text{If } X'_{total\text{-}Q,I} < X'_{total\text{-}P,i} \tag{4b}$$

Accept $X''_{j,P,i}$ if it gives a better function value.

3. Elitist TLBO algorithm

In the previous work on TLBO algorithm by Rao et al. (2011, 2012), Rao & Savsani (2012) and Rao and Patel (2012), the aspect of 'elitism' was not considered and only two common controlling parameters, i.e. population size and number of generations were used. Moreover, the effects of common controlling parameters such as population size and the number of generations on the performance of the algorithm were not investigated in detail. Hence, in the present work, 'elitism' is introduced in the TLBO algorithm to identify its effect on the exploration and exploitation capacity of the algorithm.

The concept of elitism is utilized in most of the evolutionary and swarm intelligence algorithms where during every generation the worst solutions are replaced by the elite solutions. In the TLBO algorithm, after replacing the worst solutions with elite solutions at the end of learner phase, if the duplicate solutions exist then it is necessary to modify the duplicate solutions in order to avoid trapping in the local optima. In the present work, duplicate solutions are modified by mutation on randomly selected dimensions of the duplicate solutions before executing the next generation. Moreover, in the present work, the effect of the common controlling parameters of the algorithm i.e. population size, number of generations and elite-size on the performance of the algorithm are also investigated by considering different population sizes, number of generations and elite sizes.

At this point, it is important to clarify that in the TLBO algorithm, the solution is updated in the teacher phase as well as in the learner phase. Also, in the duplicate elimination step, if duplicate solutions are present then they are randomly modified. So the total number of function evaluations in the TLBO algorithm is = {(2 × population size × number of generations) + (function evaluations required for duplicate elimination)}. In the entire experimental work of this paper, the above formula is used to count the number of function evaluations while conducting experiments with TLBO algorithm. Since the function evaluations required for duplication removal are not clearly known, experiments are conducted with different population sizes and based on these experiments it is reasonably concluded that the function evaluations required for the duplication removal are 5000, 10000, 15000 and 20000 for population sizes of 25, 50, 75 and 100, respectively.

The flow chart of the Elitist TLBO algorithm is shown in Fig. 1. The next section deals with the experimentation of improved TLBO algorithm on various constrained benchmark functions.

4. Experiments on constrained benchmark functions

In this section, the ability of TLBO algorithm is assessed by implementing it for the parameter optimization of 22 well defined problems of CEC 2006 (Liang et al., 2006). These problems include various forms of objective functions such as linear, nonlinear, quadratic, polynomial and cubic. Each problem has a different number of variables, ranges of constraints, number and types. In the field of optimization, a common platform is required to compare the performance of different algorithms for different benchmark functions. Previously different researchers experimented different algorithms for the considered benchmark functions with 240000 function evaluations.

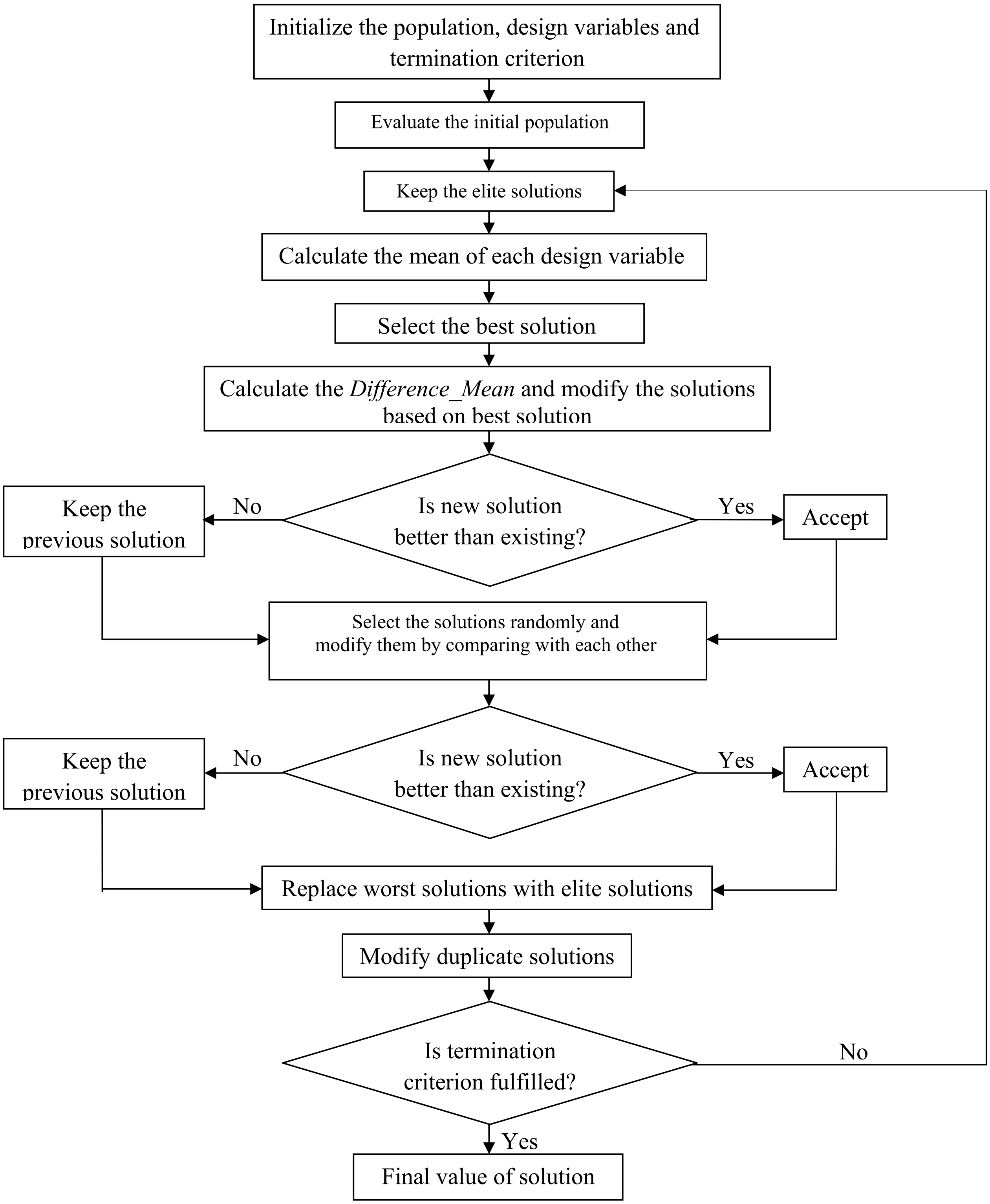


Fig. 1. Flowchart of elitist TLBO algorithm.

Considering this fact, in the present work also the common platform is maintained by setting the maximum function evaluations as 240000. Thus, the consistency in the comparison is maintained while comparing the performance of TLBO with other optimization algorithms. However, it may be mentioned here that, in general, the algorithm, which requires fewer number of function evaluations to get the same best solution can be considered as better as compared with the other algorithms. If an algorithm gives global optimum solution within certain number of function evaluations, then consideration of more number of function evaluations will go on giving the same best result. Rao et al. (2011, 2012) showed that TLBO requires fewer number of function evaluations as compared with the other optimization algorithms.

Even though certain experiments were not conducted by Rao et al. (2011, 2012) in the same settings, but better test conditions (i.e. comparatively less number of function evaluations) were chosen by them which proved the better performance of TLBO algorithm. There was no need for TLBO algorithm to go to the high settings followed by other researchers who used different number of function evaluations for the considered benchmark functions. The stopping conditions used by Rao et al. (2011, 2012) in certain benchmark functions with 30 runs each time were better than those used by other researchers. However, in this paper, to maintain the consistency in comparison, the number of function evaluations of 240000 is maintained the same for all optimization algorithms including TLBO algorithm for all the benchmark functions considered.

Like other optimization algorithms (e.g. PSO, ABC, ACO, etc.), TLBO algorithm also has not any special mechanism to handle the constraints. So, for the constrained optimization problems it is necessary to incorporate any constraint handling technique with the TLBO algorithm even though the algorithm has its own exploration and exploitation powers. In this experiment, Deb's heuristic constrained handling method (Deb, 2000) is used to handle the constraints with the TLBO algorithm. Deb's method uses a tournament selection operator in which two solutions are selected and compared with each other. The following three heuristic rules are implemented on them for the selection:

- If one solution is feasible and the other infeasible, then the feasible solution is preferred.
- If both the solutions are feasible, then the solution having the better objective function value is preferred.
- If both the solutions are infeasible, then the solution having the least constraint violation is preferred.

These rules are implemented at the end of the teacher phase and the learner phase. Deb's constraint handling rules are used to select new solution based on the above three heuristic rules. For the considered test problems, the TLBO algorithm is run for 30 times for each benchmark function. In each run the maximum function evaluations is considered as 240000 for all the functions and the results obtained using the TLBO algorithm are compared with the results given by other well known optimization algorithms for the same number of function evaluations.

Moreover, in order to identify the effect of population size on the performance of the algorithm, the algorithm is experimented with different population sizes viz. 25, 50, 75 and 100 with number of generations 4700, 2300, 1500 and 1100 respectively so that the function evaluations in each strategy is 240000. Similarly, to identify the effect of elite size on the performance of the algorithm, the algorithm is experimented with different elite sizes, viz. 0, 4, 8, 12 and 16.

Here elite size 0 indicates no elitism consideration. The comparative results of each benchmark function for each strategy are presented in Tables 1-11 in the form of best solution, worst solution, average solution and standard deviation obtained in 30 independent runs on each benchmark function with each strategy. The notations B, W, M, SD and PS in Tables 1-11 denote Best, Worst, Mean, Standard deviation and Population size, respectively. The boldface value given in parenthesis indicates the global optimum value of that function.

Table 1
Comparative results of G01 and G02 for 240000 function evaluations averaged over 30 runs

G01 (-15.00)						**G02 (-0.803619)**					
Elite		PS=25	PS=50	PS=75	PS=100	Elite		PS=25	PS=50	PS=75	PS=100
	B	-15	-15	-15	-15		B	-0.803619	-0.803619	-0.803617	-0.803619
0	W	-10.11	-13	-13	-13	0	W	-0.77322	-0.792556	-0.803613	-0.803619
	M	-13.35	-14.2	-14.6	-14.8		M	-0.800579	-0.802898	-0.803616	-0.803619
	SD	1.58E+00	9.97E-01	8.14E-01	6.10E-01		SD	9.28E-03	2.74E-03	1.19E-06	0.00E+00
	B	-15	-15	-15	-15		B	-0.803611	-0.803606	-0.803602	-0.803619
4	W	-14	-12	-15	-15	4	W	-0.784808	-0.792556	-0.793022	-0.80309
	M	-14.8	-14.7	-15	-15		M	-0.79836	-0.801423	-0.802526	-0.803586
	SD	4.07E-01	9.15E-01	0.00E+00	0.00E+00		SD	6.70E-03	4.39E-03	3.22E-03	1.10E-04
	B	-15	-15	-15	-15		B	-0.803594	-0.803613	-0.803618	-0.8036
8	W	-14	-15	-15	-15	8	W	-0.761609	-0.784813	-0.784817	-0.803089
	M	-14.8	-15	-15	-15		M	-0.797711	-0.800674	-0.801737	-0.80357
	SD	4.07E-01	0.00E+00	0.00E+00	0.00E+00		SD	1.28E-02	6.26E-03	5.74E-03	9.76E-05
	B	-15	-15	-15	-15		B	-0.803555	-0.803606	-0.80361	-0.803612
12	W	-14	-14	-15	-15	12	W	-0.782154	-0.782518	-0.792511	-0.793012
	M	-14.7	-14.9	-15	-15		M	-0.79669	-0.799334	-0.800335	-0.800437
	SD	4.66E-01	3.05E-01	0.00E+00	0.00E+00		SD	7.48E-03	7.16E-03	5.08E-03	4.92E-03
	B	-15	-15	-15	-15		B	-0.803604	-0.803453	-0.803597	-0.803602
16	W	-14	-14	-15	-15	16	W	-0.782506	-0.78233	-0.772091	-0.782499
	M	-14.7	-14.9	-15	-15		M	-0.795926	-0.796302	-0.799385	-0.800373
	SD	4.66E-01	3.05E-01	0.00E+00	0.00E+00		SD	8.22E-03	6.51E-03	9.79E-03	6.92E-03

Table 2
Comparative results of G03 and G04 for 240000 function evaluations averaged over 30 runs

G03 (-1.0005)						**G 04 (-30665.539)**					
Elite		PS=25	PS=50	PS=75	PS=100	Elite		PS=25	PS=50	PS=75	PS=100
	B	-1.0005	-1.0005	-1.0004	-1.0005		B	-30665.539	-30665.539	-30665.539	-30665.539
0	W	-0.994	-0.9871	-0.9975	-1	0	W	-30665.539	-30665.539	-30665.539	-30665.539
	M	-0.9996	-0.9988	-0.9998	-1.0003		M	-30665.539	-30665.539	-30665.539	-30665.539
	SD	1.98E-03	4.16E-03	9.75E-04	1.40E-04		SD	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	B	-1.0004	-1.0004	-1.0004	-1.0004		B	-30665.539	-30665.539	-30665.539	-30665.539
4	W	0	-0.2829	-0.9921	-0.9903	4	W	-30665.539	-30665.539	-30665.539	-30665.539
	M	-0.7124	-0.928	-0.9979	-0.999		M	-30665.539	-30665.539	-30665.539	-30665.539
	SD	4.61E-01	2.27E-01	3.09E-03	3.09E-03		SD	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	B	-0.9996	-1.0004	-1.0005	-1.0004		B	-30665.539	-30665.539	-30665.539	-30665.539
8	W	0	-0.0003	-0.8468	-0.9853	8	W	-30665.539	-30665.539	-30665.539	-30665.539
	M	-0.6336	-0.8952	-0.9825	-0.9987		M	-30665.539	-30665.539	-30665.539	-30665.539
	SD	3.87E-01	3.15E-01	4.81E-02	4.70E-03		SD	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	B	-0.9999	-1.0004	-1.0003	-1.0004		B	-30665.539	-30665.539	-30665.539	-30665.539
12	W	0	0	-0.5508	-0.9257	12	W	-30665.539	-30665.539	-30665.539	-30665.539
	M	-0.5647	-0.8289	-0.9549	-0.9862		M	-30665.539	-30665.539	-30665.539	-30665.539
	SD	4.50E-01	3.22E-01	1.42E-01	2.40E-02		SD	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	B	-0.9647	-1.0005	-1.0002	-1.0004		B	-30665.539	-30665.539	-30665.539	-30665.539
16	W	0	-0.001	-0.154	-0.3626	16	W	-30665.539	-30665.539	-30665.539	-30665.539
	M	-0.417	-0.733	-0.914	-0.9193		M	-30665.539	-30665.539	-30665.539	-30665.539
	SD	3.94E-01	4.31E-01	2.67E-01	1.99E-01		SD	0.00E+00	0.00E+00	0.00E+00	0.00E+00

Table 3
Comparative results of G05 and G06 for 240000 function evaluations averaged over 30 runs

G05 (5126.484)						**G06(-6961.814)**					
Elite		PS=25	PS=50	PS=75	PS=100	Elite		PS=25	PS=50	PS=75	PS=100
	B	5126.584	5126.781	5126.538	5126.991		B	-6961.814	-6961.814	-6961.814	-6961.814
0	W	5382.652	5779.125	5519.789	5608.95	0	W	-6961.814	-6961.814	-6961.814	-6961.814
	M	5251.136	5209.408	5220.174	5260.7		M	-6961.814	-6961.814	-6961.814	-6961.814
	SD	8.66E+01	2.03E+02	1.55E+02	1.62E+02		SD	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	B	5126.633	5126.484	5126.761	5126.589		B	-6961.814	-6961.814	-6961.814	-6961.814
4	W	5261.817	5261.826	5261.805	5356.035	4	W	-6961.814	-6961.814	-6961.814	-6961.814
	M	5189.875	5168.719	5175.632	5192.46		M	-6961.814	-6961.814	-6961.814	-6961.814
	SD	6.34E+01	5.41E+01	5.63E+01	7.80E+01		SD	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	B	5126.863	5128.252	5126.648	5126.859		B	-6961.814	-6961.814	-6961.814	-6961.814
8	W	5261.839	5261.571	5261.571	5331.198	8	W	-6961.813	-6961.814	-6961.814	-6961.814
	M	5185.42	5188.736	5188.838	5228.504		M	-6961.814	-6961.814	-6961.814	-6961.814
	SD	6.31E+01	6.35E+01	6.27E+01	6.13E+01		SD	3.46E-04	0.00E+00	0.00E+00	0.00E+00
	B	5127.247	5128.477	5126.955	5128.473		B	-6961.809	-6961.814	-6961.814	-6961.814
12	W	5328.626	5261.785	5261.829	5261.792	12	W	-6959.81	-6961.814	-6961.814	-6961.814
	M	5202.229	5190.065	5191.267	5231.044		M	-6961.382	-6961.814	-6961.814	-6961.814
	SD	7.24E+01	6.10E+01	6.21E+01	4.96E+01		SD	6.69E-01	0.00E+00	0.00E+00	0.00E+00
	B	5128.481	5126.531	5142.291	5126.555		B	-6961.814	-6961.814	-6961.814	-6961.814
16	W	5261.649	5261.835	5261.799	5461.843	16	W	-6960.577	-6961.814	-6961.814	-6961.814
	M	5209.232	5194.464	5216.64	5277.877		M	-6961.369	-6961.814	-6961.814	-6961.814
	SD	5.19E+01	6.30E+01	5.11E+01	1.12E+02		SD	5.51E-01	0.00E+00	0.00E+00	0.00E+00

Table 4
Comparative results of G 07 and G 08 for 240000 function evaluations averaged over 30 runs

G07 (24.3062)						G08 (-0.095825)					
Elite		PS=25	PS=50	PS=75	PS=100	Elite		PS=25	PS=50	PS=75	PS=100
	B	24.318	24.311	24.309	24.3062		B	-0.095825	-0.095825	-0.095825	-0.095825
0	W	24.9482	24.9578	24.5825	24.322	0	W	-0.095825	-0.095825	-0.095825	-0.095825
	M	24.4926	24.47	24.3978	24.31		M	-0.095825	-0.095825	-0.095825	-0.095825
	SD	0.2451	0.2254	0.1025	0.0071		SD	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	B	24.371	24.3385	24.336	24.3289		B	-0.095825	-0.095825	-0.095825	-0.095825
4	W	25.7564	25.0147	24.9678	24.9735	4	W	-0.095825	-0.095825	-0.095825	-0.095825
	M	25.0117	24.6503	24.6179	24.5273		M	-0.095825	-0.095825	-0.095825	-0.095825
	SD	4.04E-01	2.64E-01	2.40E-01	2.33E-01		SD	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	B	25.0047	24.3442	24.338	24.3313		B	-0.095825	-0.095825	-0.095825	-0.095825
8	W	27.1464	25.1957	25.0057	24.9627	8	W	-0.095825	-0.095825	-0.095825	-0.095825
	M	25.7935	24.7883	24.6865	24.5519		M	-0.095825	-0.095825	-0.095825	-0.095825
	SD	7.01E-01	2.79E-01	2.59E-01	2.41E-01		SD	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	B	25.2597	24.3673	24.358	24.345		B	-0.095825	-0.095825	-0.095825	-0.095825
12	W	36.3906	25.3439	25.0085	25.1646	12	W	-0.095825	-0.095825	-0.095825	-0.095825
	M	29.3526	24.8168	24.7453	24.637		M	-0.095825	-0.095825	-0.095825	-0.095825
	SD	3.74E+00	2.64E-01	2.82E-01	3.41E-01		SD	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	B	26.9248	24.5828	24.463	24.3924		B	-0.095825	-0.095825	-0.095825	-0.095825
16	W	157.7866	25.9771	25.3603	25.1733	16	W	-0.095825	-0.095825	-0.095825	-0.095825
	M	41.381	25.0975	24.7916	24.7334		M	-0.095825	-0.095825	-0.095825	-0.095825
	SD	3.95E+01	3.56E-01	3.01E-01	2.68E-01		SD	0.00E+00	0.00E+00	0.00E+00	0.00E+00

Table 5
Comparative results of G09 and G10 for 240000 function evaluations averaged over 30 runs

G09 (680.63)						G10 (7049.28)					
Elite		PS=25	PS=50	PS=75	PS=100	Elite		PS=25	PS=50	PS=75	PS=100
	B	680.632	680.63	680.63	680.631		B	7218.258	7059.309	7077.486	7129.944
0	W	680.64	680.63	680.63	680.639	0	W	7608.953	7584.887	7331.171	7381.029
	M	680.63588	680.63	680.63	680.632		M	7370.191	7274.506	7201.135	7254.325
	SD	2.67E-03	0.00E+00	0.00E+00	2.35E-03		SD	1.25E+02	1.55E+02	8.49E+01	7.17E+01
	B	680.637	680.63	680.636	680.634		B	7217.398	7072.165	7052.488	7113.42
4	W	680.669	680.63	680.667	680.646	4	W	7613.866	7448.947	7357.629	7285.122
	M	680.646	680.63	680.644	680.638		M	7348.763	7243.093	7143.45	7193.726
	SD	9.15E-03	0.00E+00	8.59E-03	3.45E-03		SD	1.39E+02	1.00E+02	1.13E+02	4.13E+01
	B	680.6396	680.632	680.636	680.636		B	7350.645	7259.272	7166.904	7118.633
8	W	680.7389	680.679	680.651	680.652	8	W	7803.572	7415.447	7407.179	7421.597
	M	680.6583	680.648	680.642	680.641		M	7503.161	7332.667	7263.295	7278.399
	SD	2.85E-02	1.58E-02	4.58E-03	5.04E-03		SD	1.23E+02	5.03E+01	5.90E+01	1.03E+02
	B	680.6367	680.634	680.637	680.638		B	7214.573	7234.14	7222.629	7170.587
12	W	680.8954	680.673	680.662	680.65	12	W	8708.483	7560.957	7454.869	7457.649
	M	680.68686	680.649	680.646	680.643		M	7812.622	7368.696	7292.305	7311.941
	SD	7.47E-02	1.36E-02	7.37E-03	4.38E-03		SD	5.02E+02	9.87E+01	8.51E+01	1.03E+02
	B	680.6765	680.636	680.633	680.906		B	7289.501	7228.79	7185.471	7235.078
16	W	681.3103	680.663	680.683	683.092	16	W	9476.701	7608.954	7457.652	7457.649
	M	680.94622	680.651	680.648	681.623		M	7882.43	7404.467	7325.969	7331.284
	SD	2.35E-01	7.56E-03	1.31E-02	6.11E-01		SD	7.05E+02	1.10E+02	9.80E+01	8.42E+01

Table 6
Comparative results of G11 and G12 for 240000 function evaluations averaged over 30 runs

G11 (0.7499)						G12 (-1.00)					
Elite		PS=25	PS=50	PS=75	PS=100	Elite		PS=25	PS=50	PS=75	PS=100
	B	0.7499	0.7499	0.7499	0.74996		B	-1	-1	-1	-1
0	W	0.78913	0.75588	0.76447	0.7639	0	W	-1	-1	-1	-1
	M	0.75678	0.75058	0.75153	0.75211		M	-1	-1	-1	-1
	SD	1.39E-02	1.80E-03	4.39E-03	4.24E-03		SD	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	B	0.7499	0.7499	0.74991	0.74991		B	-1	-1	-1	-1
4	W	0.93853	0.75417	0.75275	0.76036	4	W	-1	-1	-1	-1
	M	0.78015	0.75036	0.7507	0.7512		M	-1	-1	-1	-1
	SD	6.37E-02	1.29E-03	1.02E-03	3.13E-03		SD	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	B	0.74997	0.7499	0.74991	0.74996		B	-1	-1	-1	-1
8	W	0.77939	0.7501	0.75355	0.7539	8	W	-1	-1	-1	-1
	M	0.75408	0.74998	0.75061	0.7509		M	-1	-1	-1	-1
	SD	8.87E-03	7.06E-05	1.03E-03	1.24E-03		SD	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	B	0.7499	0.7499	0.74991	0.7499		B	-1	-1	-1	-1
12	W	0.93853	0.75587	0.76676	0.76458	12	W	-1	-1	-1	-1
	M	0.78015	0.75124	0.75169	0.75269		M	-1	-1	-1	-1
	SD	6.37E-02	2.13E-03	5.11E-03	5.56E-03		SD	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	B	0.75009	0.74993	0.7499	0.7499		B	-1	-1	-1	-1
16	W	0.99494	0.75782	0.7748	0.77148	16	W	-1	-1	-1	-1
	M	0.8489	0.75162	0.75283	0.7542		M	-1	-1	-1	-1
	SD	7.35E-02	2.30E-03	7.51E-03	7.33E-03		SD	0.00E+00	0.00E+00	0.00E+00	0.00E+00

Table 7
Comparative results of G13 and G14 for 240000 function evaluations averaged over 30 runs

G13 (0.05394)						G14 (-47.764)					
Elite		PS=25	PS=50	PS=75	PS=100	Elite		PS=25	PS=50	PS=75	PS=100
0	B	0.44839	0.39303	0.63431	0.46799	0	B	-46.532	-45.994	-47.41	-46.037
	W	0.99942	0.99979	0.99993	1.00459		W	-39.548	-39.961	-38.809	-39.677
	M	0.82108	0.83851	0.87655	0.88031		M	-41.846	-43.133	-42.857	-42.189
	SD	1.96E-01	2.26E-01	1.32E-01	1.76E-01		SD	2.04E+00	2.30E+00	2.56E+00	2.36E+00
4	B	0.59076	0.57352	0.4941	0.8654	4	B	-46.018	-47.636	-47.138	-46.478
	W	1.01499	0.99983	1.4063	1.07012		W	-37.852	-39.352	-40.638	-39.41
	M	0.86815	0.90849	0.94148	0.97905		M	-41.392	-43.731	-42.995	-42.123
	SD	1.64E-01	1.24E-01	2.35E-01	5.93E-02		SD	2.80E+00	3.22E+00	1.77E+00	2.19E+00
8	B	0.13314	0.35756	0.62085	0.54427	8	B	-46.813	-47.639	-47.46	-44.076
	W	1.17245	0.9997	1.12481	1.5396		W	-36.076	-39.414	-39.16	-40.026
	M	0.91105	0.89156	0.92115	0.93184		M	-42.148	-43.805	-43.433	-42.747
	SD	2.99E-01	1.69E-01	1.50E-01	2.68E-01		SD	2.99E+00	2.32E+00	2.39E+00	1.31E+00
12	B	0.16907	0.55193	0.52184	0.62027	12	B	-47.574	-47.512	-47.401	-47.626
	W	4.91506	0.99983	1.2581	0.99992		W	-38.544	-39.128	-38.47	-38.409
	M	1.17795	0.87997	0.9059	0.93594		M	-42.019	-43.352	-43.013	-42.39
	SD	1.30E+00	1.48E-01	2.02E-01	1.20E-01		SD	3.09E+00	2.62E+00	3.04E+00	3.17E+00
16	B	0.48239	0.5543	0.61935	0.87019	16	B	-43.361	-47.667	-47.59	-45.377
	W	11.24563	0.99952	1.17245	0.99972		W	-33.039	-38.21	-39.623	-37.201
	M	1.94685	0.92613	0.94517	0.94565		M	-40.243	-43.16	-42.575	-41.983
	SD	3.17E+00	1.36E-01	1.43E-01	4.74E-02		SD	2.99E+00	3.80E+00	2.19E+00	2.34E+00

Table 8
Comparative results of G15 and G16 for 240000 function evaluations averaged over 30 runs

G15 (961.715)						G16 (-1.905155)					
Elite		PS=25	PS=50	PS=75	PS=100	Elite		PS=25	PS=50	PS=75	PS=100
0	B	961.715	961.715	961.715	961.715	0	B	-1.905155	-1.905155	-1.905155	-1.905155
	W	966.998	966.955	963.15	962.775		W	-1.905155	-1.905155	-1.905155	-1.905155
	M	963.347	962.576	962.284	962.044		M	-1.905155	-1.905155	-1.905155	-1.905155
	SD	2.06E+00	1.58E+00	6.00E-01	4.39E-01		SD	0.00E+00	0.00E+00	0.00E+00	0.00E+00
4	B	961.862	961.715	961.718	961.718	4	B	-1.905155	-1.905155	-1.905155	-1.905155
	W	972.297	967.406	967.37	964.5		W	-1.905155	-1.905155	-1.905155	-1.905155
	M	963.909	963.555	962.989	962.406		M	-1.905155	-1.905155	-1.905155	-1.905155
	SD	3.17E+00	1.98E+00	1.84E+00	9.74E-01		SD	0.00E+00	0.00E+00	0.00E+00	0.00E+00
8	B	961.715	961.719	961.846	961.72	8	B	-1.905155	-1.905155	-1.905155	-1.905155
	W	971.637	971.194	970.628	971.843		W	-1.905155	-1.905155	-1.905155	-1.905155
	M	964.541	964.761	964.092	963.549		M	-1.905155	-1.905155	-1.905155	-1.905155
	SD	3.41E+00	3.13E+00	2.70E+00	3.11E+00		SD	0.00E+00	0.00E+00	0.00E+00	0.00E+00
12	B	961.721	961.716	961.715	961.715	12	B	-1.905155	-1.905155	-1.905155	-1.905155
	W	972.145	969.744	967.499	966.96		W	-1.905155	-1.905155	-1.905155	-1.905155
	M	965.164	963.835	962.915	962.609		M	-1.905155	-1.905155	-1.905155	-1.905155
	SD	3.30E+00	2.71E+00	1.76E+00	1.39E+00		SD	0.00E+00	0.00E+00	0.00E+00	0.00E+00
16	B	961.823	961.716	961.723	961.762	16	B	-1.905155	-1.905155	-1.905155	-1.905155
	W	969.752	972.293	972.197	967.757		W	-1.905155	-1.905155	-1.905155	-1.905155
	M	965.741	963.99	963.383	962.947		M	-1.905155	-1.905155	-1.905155	-1.905155
	SD	2.54E+00	3.24E+00	3.02E+00	1.78E+00		SD	0.00E+00	0.00E+00	0.00E+00	0.00E+00

Table 9
Comparative results of G17 and G18 for 240000 function evaluations averaged over 30 runs

G17 (8853.5396)						G18 (-0.866)					
Elite		PS=25	PS=50	PS=75	PS=100	Elite		PS=25	PS=50	PS=75	PS=100
0	B	8856.77	8855.501	8855.447	8853.804	0	B	-0.866025	-0.865981	-0.866009	-0.865777
	W	9027.94	9023.582	9024.896	9023.088		W	-0.86457	-0.862273	-0.861352	-0.674812
	M	8981.526	8952.919	8948.2751	8910.0856		M	-0.865755	-0.865371	-0.864972	-0.846037
	SD	6.51E+01	7.96E+01	6.80E+01	6.42E+01		SD	5.09E-04	1.07E-03	1.20E-03	5.81E-02
4	B	8854.392	8861.01	8853.814	8856.052	4	B	-0.866002	-0.865995	-0.865027	-0.866025
	W	9025.256	9025.14	9024.933	9021.472		W	-0.86386	-0.863089	-0.840008	-0.674386
	M	8957.6225	8951.2824	8915.0277	8906.976		M	-0.865381	-0.86527	-0.85823	-0.845979
	SD	7.79E+01	6.90E+01	7.19E+01	7.03E+01		SD	6.04E-04	9.51E-04	7.91E-03	5.82E-02
8	B	8858.566	8855.605	8857.508	8853.81	8	B	-0.866025	-0.866023	-0.865936	-0.866007
	W	9023.442	9023.709	9025.868	9016.279		W	-0.862737	-0.863175	-0.839092	-0.672823
	M	8954.8983	8947.1528	8904.0506	8895.7544		M	-0.865276	-0.864974	-0.852929	-0.845673
	SD	7.54E+01	7.75E+01	6.56E+01	5.14E+01		SD	1.07E-03	1.08E-03	1.05E-02	5.86E-02
12	B	8858.079	8854.553	8857.164	8854.25	12	B	-0.865996	-0.866018	-0.866006	-0.865604
	W	9022.631	9022.731	9024.468	9011.928		W	-0.863496	-0.862728	-0.709067	-0.524783
	M	8986.1785	8953.2101	8931.2958	8899.5362		M	-0.865193	-0.8648	-0.849091	-0.814682
	SD	5.09E+01	6.34E+01	6.83E+01	5.48E+01		SD	8.54E-04	1.14E-03	4.76E-02	1.00E-01
16	B	8827.089	8854.57	8854.21	8855.624	16	B	-0.866023	-0.866014	-0.866012	-0.856991
	W	9827.12	9023.919	9022.432	9012.975		W	-0.863224	-0.852447	-0.674046	-0.515427
	M	9162.0944	8954.4421	8941.8734	8908.7417		M	-0.865084	-0.862779	-0.846048	-0.808539
	SD	3.39E+02	7.44E+01	7.49E+01	6.35E+01		SD	9.29E-04	4.52E-03	5.83E-02	1.00E-01

Table 10

Comparative results of G19 and G21 for 240000 function evaluations averaged over 30 runs

G19 (32.6555)						G21 (193.274)					
Elite		PS=25	PS=50	PS=75	PS=100	Elite		PS=25	PS=50	PS=75	PS=100
	B	33.3119	33.294	33.2942	33.2944		B	197.426	197.236	197.15	196.122
0	W	50.241	33.5481	34.9013	34.7558	0	W	302.248	289.829	271.237	274.452
	M	35.304	33.3699	33.5474	33.5162		M	239.736	233.383	228.813	224.414
	SD	5.28E+00	7.87E-02	4.82E-01	4.44E-01		SD	5.44E+01	5.14E+01	4.74E+01	5.01E+01
	B	33.6593	33.3008	33.2938	33.2957		B	196.652	195.984	195.481	194.231
4	W	53.0521	34.028	35.1725	34.8166	4	W	273.871	271.831	278.86	241.221
	M	42.8586	33.4554	33.9852	33.7812		M	229.932	219.187	214.344	206.118
	SD	8.09E+00	2.25E-01	8.08E-01	6.84E-01		SD	3.99E+01	3.43E+01	4.00E+01	2.99E+01
	B	34.1188	33.2945	33.2961	33.3041		B	198.922	196.721	196.389	195.776
8	W	51.9498	34.8266	35.0265	35.0266	8	W	279.972	281.218	273.435	264.434
	M	41.5556	33.9212	34.5422	34.2234		M	238.812	232.761	229.983	219.146
	SD	7.35E+00	6.79E-01	6.42E-01	7.64E-01		SD	4.11E+01	4.67E+01	4.86E+01	3.78E+01
	B	34.3523	33.3536	33.3848	33.3		B	197.793	199.982	198.822	197.912
12	W	62.3528	34.7769	46.5817	35.2187	12	W	312.245	281.321	278.118	272.317
	M	47.9449	34.1986	37.9091	34.4617		M	243.417	239.757	233.546	226.672
	SD	9.40E+00	7.06E-01	5.27E+00	7.86E-01		SD	5.80E+01	4.22E+01	3.98E+01	3.25E+01
	B	37.2534	33.2972	33.3099	33.3113		B	201.322	197.531	197.191	196.461
16	W	59.2413	34.899	70.5	37.7744	16	W	283.837	303.345	291.248	284.412
	M	48.1078	34.3361	38.5401	34.9075		M	244.498	241.131	234.642	227.783
	SD	7.81E+00	6.27E-01	1.13E+01	1.12E+00		SD	4.66E+01	5.33E+01	5.12E+01	4.90E+01

Table 11

Comparative results of G23 and G24 for 240000 function evaluations averaged over 30 runs

G23 (-400.055)						G24 (-5.508013)					
Elite		PS=25	PS=50	PS=75	PS=100	Elite		PS=25	PS=50	PS=75	PS=100
	B	-293.872	-336.662	-369.986	-387.716		B	-5.508013	-5.508013	-5.508013	-5.508013
0	W	-213.321	-291.983	-304.425	-321.249	0	W	-5.508013	-5.508013	-5.508013	-5.508013
	M	-244.792	-304.181	-336.644	-352.263		M	-5.508013	-5.508013	-5.508013	-5.508013
	SD	3.91E+01	2.99E+01	3.44E+01	2.33E+01		SD	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	B	-273.166	-327.537	-354.425	-377.431		B	-5.508013	-5.508013	-5.508013	-5.508013
4	W	-209.146	-287.813	-296.692	-309.041	4	W	-5.508013	-5.508013	-5.508013	-5.508013
	M	-231.387	-301.169	-309.923	-324.417		M	-5.508013	-5.508013	-5.508013	-5.508013
	SD	4.39E+01	3.87E+01	4.18E+01	4.23E+01		SD	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	B	-289.235	-283.334	-297.791	-314.417		B	-5.508013	-5.508013	-5.508013	-5.508013
8	W	-218.763	-207.718	-229.875	-234.127	8	W	-5.508013	-5.508013	-5.508013	-5.508013
	M	-243.761	-245.236	-250.083	-297.112		M	-5.508013	-5.508013	-5.508013	-5.508013
	SD	4.00E+01	4.66E+01	4.19E+01	3.67E+01		SD	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	B	-269.951	-299.058	-291.873	-309.082		B	-5.508013	-5.508013	-5.508013	-5.508013
12	W	-211.313	-219.914	-233.346	-231.422	12	W	-5.508013	-5.508013	-5.508013	-5.508013
	M	-229.733	-242.592	-248.881	-273.358		M	-5.508013	-5.508013	-5.508013	-5.508013
	SD	4.06E+01	4.42E+01	3.99E+01	4.92E+01		SD	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	B	-241.764	-273.125	-303.141	-309.912		B	-5.508013	-5.508013	-5.508013	-5.508013
16	W	-198.873	-209.912	-238.435	-243.327	16	W	-5.508013	-5.508013	-5.508013	-5.508013
	M	-213.786	-240.388	-246.647	-259.962		M	-5.508013	-5.508013	-5.508013	-5.508013
	SD	4.39E+01	4.01E+01	4.24E+01	3.99E+01		SD	0.00E+00	0.00E+00	0.00E+00	0.00E+00

It is observed from Tables 1-11 that for functions G02, G03, G07, G15-G17, G21 and G23, strategy with population size of 100 and number of generations of 1100 produced the best result than the other strategies. For functions G05, G09, G11, G13, G14 and G19, strategy with population size of 50 and number of generations of 2300 gives the best results. For functions G10, strategy with population size of 75 and number of generations of 1500 and for function G18 strategy with population size of 25 and number of generations of 4700 produces the best results. While for functions G04, G06, G08, G12 and G24 all the strategies produce the same results and hence there is no effect of population size on these functions to achieve their respective global optimum values with same number of function evaluations. For function G01, strategies with population size of 75 and 100 and number of generations of 1500 and 1100 respectively produce the identical results.

Similarly, it is observed from Tables 1-11 that for functions G02, G03, G07, G13, G15, G16, G18, G19 and G23, strategy with elite size 0, i.e. no elitism produces the best results than the other strategies having different elite sizes. For functions G05, G09, G10 and G21, strategy with elite size of 4 produces the best results. For functions G11, G14, and G17, strategy with elite size of 8 produces the best results. For functions G04, G06, G08, G12 and G24 all the strategies (i.e. strategy without elitism consideration as well as strategies with different elite sizes consideration) produce the same results and hence there is no effect of elitism on these functions. For function G01, strategies with elite size of 4, 8, 12 and 16 with population sizes of 75 and 100 produce the same results. Table 12 shows the optimum results obtained by the TLBO algorithm for all the G functions.

Table 12
Results obtained by TLBO algorithm for 22 benchmark functions over 30 independent runs with 240000 function evaluations

Function	Optimum	Best	Worst	Mean	SD
G01	-15	-15	-15	-15	0.00E+00
G02	-0.803619	-0.803619	-0.803619	-0.803619	0.00E+00
G03	-1.0005	-1.0005	-1	-1.0003	1.40E-04
G04	-30665.539	-30665.539	-30665.539	-30665.539	0.00E+00
G05	5126.484	5126.484	5261.826	5168.7194	5.41E+01
G06	-6961.814	-6961.814	-6961.814	-6961.814	0.00E+00
G07	24.3062	24.3062	24.322	24.31	7.11E-03
G08	-0.095825	-0.095825	-0.095825	-0.095825	0.00E+00
G09	680.63	680.63	680.63	680.63	0.00E+00
G10	7049.28	7052.488	7357.629	7143.45	1.13E+02
G11	0.7499	0.7499	0.7501	0.74998	7.06E-05
G12	-1	-1	-1	-1	0.00E+00
G13	0.05394	0.13314	0.99979	0.83851	2.26E-01
G14	-47.764	-47.639	-39.414	-43.805	2.32E+00
G15	961.715	961.715	962.775	962.044	4.39E-01
G16	-1.905155	-1.905155	-1.905155	-1.905155	0.00E+00
G17	8853.5396	8853.81	9016.279	8895.7544	5.14E+01
G18	-0.866	-0.866025	-0.86457	-0.865755	5.09E-04
G19	32.6555	33.294	33.5481	33.3699	7.87E-02
G21	193.724	194.231	241.221	206.118	2.99E+01
G23	-400.055	-387.716	-321.249	-352.263	2.33E+01
G24	-5.508013	-5.508013	-5.508013	-5.508013	0.00E+00

The performance of TLBO algorithm is compared with the other well known optimization algorithms such as PSO, DE and ABC for G01-G13 functions. The results of PSO, DE and ABC are taken from the previous work of Karaboga and Basturk (2007) where the authors had experimented benchmark functions each with 240000 function evaluations with best setting of algorithm specific parameters.

Table 13
Comparative results of TLBO with other evolutionary algorithms over 30 independent runs

		PSO	DE	ABC	TLBO			PSO	DE	ABC	TLBO
G01	B	-15	-15	-15	-15	G02	B	-0.669158	-0.472	-0.803598	-0.803619
	W	-13	-11.828	-15	-15		W	-0.299426	-0.472	-0.749797	-0.803619
	M	-14.71	-14.555	-15	-15		M	-0.41996	-0.665	-0.792412	-0.803619
G03	B	-1	-0.99393	-1	-1.0005	G04	B	-30665.539	-30665.539	-30665.539	-30665.539
	W	-0.464	-1	-1	-1		W	-30665.539	-30665.539	-30665.539	-30665.539
	M	0.764813	-1	-1	-1.0003		M	-30665.539	-30665.539	-30665.539	-30665.539
G05	B	5126.484	5126.484	5126.484	5126.484	G06	B	-6961.814	-6954.434	-6961.814	-6961.814
	W	5249.825	5534.61	5438.387	5261.826		W	-6961.814	-6954.434	-6961.805	-6961.814
	M	5135.973	5264.27	5185.714	5168.7194		M	-6961.814	-------	-6961.813	-6961.814
G07	B	24.37	24.306	24.33	24.3062	G08	B	-0.095825	-0.095825	-0.095825	-0.095825
	W	56.055	24.33	25.19	24.322		W	-0.095825	-0.095825	-0.095825	-0.095825
	M	32.407	24.31	24.473	24.31		M	-0.095825	-0.095825	-0.095825	-0.095825
G09	B	680.63	680.63	680.634	680.63	G10	B	7049.381	7049.248	7053.904	7052.488
	W	680.631	680.631	680.653	680.63		W	7894.812	9264.886	7604.132	7357.629
	M	680.63	680.63	680.64	680.63		M	7205.5	7147.334	7224.407	7143.45
G11	B	0.749	0.752	0.75	0.7499	G12	B	-1	-1	-1	-1
	W	0.749	1	0.75	0.7501		W	-0.994	-1	-1	-1
	M	0.749	0.901	0.75	0.74998		M	-0.998875	-1	-1	-1
G13	B	0.085655	0.385	0.76	0.39303						
	W	1.793361	0.99	1	0.99979						
	M	0.569358	0.872	0.968	0.83851						

Table 14
Comparative results of H01 and H02 for 240000 function evaluations averaged over 30 runs

H01						H02					
Elite		PS=25	PS=50	PS=75	PS=100	Elite		PS=25	PS=50	PS=75	PS=100
	B	0.00E+00	8.53E-84	2.45E-40	2.83E-32		B	-2.36E+00	-3.17E+00	-3.16E+00	-3.17E+00
0	W	0.00E+00	1.31E-45	6.37E-35	8.68E-29	0	W	-2.23E+00	-2.27E+00	-2.22E+00	-2.27E+00
	M	0.00E+00	1.98E-46	9.45E-36	3.88E-29		M	-2.28E+00	-2.54E+00	-2.58E+00	-2.68E+00
	SD	0.00E+00	4.90E-46	2.39E-35	3.64E-29		SD	3.94E-02	4.27E-01	4.05E-01	4.51E-01
	B	0.00E+00	0.00E+00	4.11E-54	9.98E-35		B	-2.96E+00	-3.17E+00	-3.17E+00	-3.17E+00
4	W	0.00E+00	1.20E-73	6.31E-43	7.33E-24	4	W	-1.81E+00	-3.15E+00	-3.16E+00	-3.16E+00
	M	0.00E+00	1.71E-74	9.96E-44	1.11E-24		M	-2.24E+00	-3.16E+00	-3.16E+00	-3.17E+00
	SD	0.00E+00	4.53E-74	2.36E-43	2.68E-24		SD	3.11E-01	5.78E-03	1.56E-03	8.93E-04
	B	8.30E-70	4.75E-78	3.80E-48	2.76E-30		B	-3.00E+00	-3.16E+00	-3.17E+00	-3.17E+00
8	W	2.30E-58	9.67E-58	3.29E-36	8.43E-18	8	W	-1.14E+00	-3.15E+00	-3.16E+00	-3.16E+00
	M	7.21E-59	9.67E-59	5.08E-37	1.40E-18		M	-1.88E+00	-3.16E+00	-3.16E+00	-3.17E+00
	SD	9.57E-59	2.98E-58	1.20E-36	3.14E-18		SD	6.35E-01	3.16E-03	2.38E-03	8.24E-04
	B	1.19E-58	5.58E-49	5.48E-34	2.15E-58		B	-2.31E+00	-3.16E+00	-3.17E+00	-3.17E+00
12	W	1.08E-50	8.49E-37	2.55E-27	5.86E-27	12	W	-2.50E-01	-3.02E+00	-3.15E+00	-3.16E+00
	M	1.62E-51	1.38E-37	3.85E-28	9.19E-28		M	-1.17E+00	-3.11E+00	-3.16E+00	-3.16E+00
	SD	3.94E-51	3.07E-37	9.34E-28	2.13E-27		SD	7.65E-01	5.21E-02	6.55E-03	1.83E-03
	B	1.82E-50	7.35E-48	2.42E-43	7.62E-33		B	-2.10E+00	-3.11E+00	-3.17E+00	-3.17E+00
16	W	8.02E-40	1.30E-36	9.58E-29	3.18E-19	16	W	-3.80E-01	-2.06E+00	-3.15E+00	-3.16E+00
	M	1.20E-40	1.95E-37	1.44E-29	4.77E-20		M	-1.48E+00	-2.44E+00	-3.16E+00	-3.16E+00
	SD	2.94E-40	4.76E-37	3.51E-29	1.16E-19		SD	5.32E-01	4.18E-01	6.46E-03	1.30E-03

Table 15
Comparative results of H03 and H04 for 240000 function evaluations averaged over 30 runs

H03						H04					
Elite		PS=25	PS=50	PS=75	PS=100	Elite		PS=25	PS=50	PS=75	PS=100
	B	0.00E+00	0.00E+00	1.01E-62	4.59E-44		B	3.75E-97	1.52E-89	3.04E-53	6.16E-59
0	W	0.00E+00	1.07E-68	8.80E-54	2.56E-38	0	W	8.58E-93	3.77E-40	4.03E-46	1.36E-47
	M	0.00E+00	1.54E-69	9.46E-55	3.23E-39		M	1.93E-93	5.38E-41	7.41E-47	2.12E-48
	SD	0.00E+00	3.55E-69	2.77E-54	8.08E-39		SD	3.20E-93	1.42E-40	1.51E-46	5.09E-48
	B	2.30E-97	2.60E-85	3.37E-58	3.27E-42		B	4.66E-52	5.65E-48	2.62E-42	1.17E-34
4	W	1.00E-90	9.29E-73	3.04E-53	9.10E-33	4	W	2.02E-40	1.15E-36	6.76E-27	7.41E-28
	M	2.31E-91	9.29E-74	4.40E-54	1.27E-33		M	2.88E-41	1.64E-37	1.01E-27	1.61E-28
	SD	4.09E-91	2.94E-73	9.69E-54	2.89E-33		SD	7.62E-41	4.33E-37	2.53E-27	2.75E-28
	B	1.26E-61	6.00E-65	1.88E-52	7.32E-41		B	5.55E-28	2.16E-41	2.73E-37	2.45E-25
8	W	4.20E-38	7.72E-55	3.95E-46	1.83E-35	8	W	1.00E-26	6.37E-22	1.09E-19	2.07E-20
	M	4.20E-39	7.76E-56	7.27E-47	2.01E-36		M	4.29E-27	2.74E-22	1.96E-20	5.00E-21
	SD	1.33E-38	2.44E-55	1.32E-46	5.73E-36		SD	3.13E-27	2.51E-22	4.07E-20	7.49E-21
	B	4.05E-38	9.94E-54	1.72E-47	4.83E-41		B	1.57E-26	2.80E-30	4.01E-20	1.65E-20
12	W	3.70E-21	6.05E-45	3.74E-44	6.90E-34	12	W	4.31E-25	5.81E-21	8.42E-16	1.97E-18
	M	4.93E-22	6.05E-46	7.17E-45	7.51E-35		M	1.38E-25	1.07E-21	1.25E-16	7.84E-19
	SD	1.16E-21	1.91E-45	1.33E-44	2.16E-34		SD	1.53E-25	2.18E-21	3.16E-16	8.03E-19
	B	2.71E-33	7.98E-43	2.57E-42	4.23E-36		B	1.47E-15	1.07E-17	2.99E-15	1.21E-14
16	W	1.61E-18	1.21E-35	4.03E-36	3.98E-29	16	W	3.50E-12	3.09E-13	4.22E-10	9.97E-13
	M	1.61E-19	3.30E-36	5.13E-37	4.21E-30		M	5.15E-13	4.57E-14	7.66E-11	2.51E-13
	SD	5.08E-19	5.11E-36	1.26E-36	1.25E-29		SD	1.32E-12	1.16E-13	1.54E-10	3.61E-13

Table 16
Comparative results of H05 and H06 for 240000 function evaluations averaged over 30 runs

H05						H06					
Elite		PS=25	PS=50	PS=75	PS=100	Elite		PS=25	PS=50	PS=75	PS=100
	B	-1.90E+01	-2.01E+01	-2.01E+01	-1.99E+01		B	-8.19E+00	-8.38E+00	-8.31E+00	-8.32E+00
0	W	-1.82E+01	-1.74E+01	-1.70E+01	-1.82E+01	0	W	-8.01E+00	-8.32E+00	-8.10E+00	-8.05E+00
	M	-1.86E+01	-1.86E+01	-1.87E+01	-1.89E+01		M	-8.10E+00	-8.48E+00	-8.21E+00	-8.12E+00
	SD	3.72E-01	6.86E-01	7.40E-01	7.12E-01		SD	4.34E-02	2.23E-07	8.94E-04	6.93E-05
	B	-2.01E+01	-2.01E+01	-2.01E+01	-2.01E+01		B	-8.03E+00	-8.35E+00	-8.21E+00	-8.20E+00
4	W	-1.89E+01	-1.79E+01	-1.79E+01	-1.79E+01	4	W	-7.88E+00	-7.99E+00	-7.86E+00	-7.92E+00
	M	-1.93E+01	-1.88E+01	-1.89E+01	-1.90E+01		M	-7.96E+00	-8.05E+00	-7.99E+00	-8.04E+00
	SD	5.27E-01	5.39E-01	5.75E-01	8.55E-01		SD	8.44E-02	4.56E-02	2.43E-01	6.76E-02
	B	-2.01E+01	-2.01E+01	-2.01E+01	-2.01E+01		B	-8.12E+00	-8.32E+00	-8.11E+00	-8.15E+00
8	W	-1.82E+01	-1.79E+01	-1.79E+01	-1.89E+01	8	W	-4.87E+00	-6.25E+00	-5.91E+00	-6.19E+00
	M	-1.93E+01	-1.92E+01	-1.91E+01	-1.91E+01		M	-5.82E+00	-7.13E+00	-6.75E+00	-6.68E+00
	SD	6.41E-01	7.58E-01	7.53E-01	4.04E-01		SD	4.97E-01	7.91E-02	8.31E-01	9.74E-01
	B	-2.01E+01	-2.01E+01	-2.01E+01	-2.01E+01		B	-6.09E+00	-7.05E+00	-7.13E+00	-7.63E+00
12	W	-1.88E+01	-1.79E+01	-1.89E+01	-1.89E+01	12	W	-2.55E+00	-2.55E+00	-2.55E+00	-2.55E+00
	M	-1.98E+01	-1.95E+01	-1.95E+01	-1.95E+01		M	-5.19E+00	-6.77E+00	-6.35E+00	-6.02E+00
	SD	5.10E-01	6.89E-01	5.82E-01	5.96E-01		SD	5.36E-01	3.22E-01	7.34E-01	9.41E-01
	B	-2.01E+01	-2.01E+01	-2.01E+01	-2.01E+01		B	-5.90E+00	-6.67E+00	-7.03E+00	-6.64E+00
16	W	-1.87E+01	-1.79E+01	-1.79E+01	-1.79E+01	16	W	-2.55E+00	-2.55E+00	-2.55E+00	-2.55E+00
	M	-1.95E+01	-1.94E+01	-1.92E+01	-1.90E+01		M	-4.56E+00	-5.23E+00	-5.15E+00	-4.97E+00
	SD	6.05E-01	7.01E-01	7.30E-01	6.06E-01		SD	1.02E+00	8.72E-01	9.74E-01	1.32E+00

Table 17
Comparative results of H07 and H08 for 240000 function evaluations averaged over 30 runs

H07						**H08**					
Elite		PS=25	PS=50	PS=75	PS=100	Elite		PS=25	PS=50	PS=75	PS=100
	B	-7.09E+00	-7.53E+00	-7.36E+00	-7.59E+00		B	-4.79E+02	-4.83E+02	-4.83E+02	-4.84E+02
0	W	-6.46E+00	-7.10E+00	-6.91E+00	-7.13E+00	0	W	-4.69E+02	-4.82E+02	-4.82E+02	-4.84E+02
	M	-6.98E+00	-7.25E+00	-7.09E+00	-7.37E+00		M	-4.73E+02	-4.82E+02	-4.82E+02	-4.84E+02
	SD	2.34E-01	6.57E-02	8.73E-01	1.54E-01		SD	5.69E+00	3.79E-01	8.42E-01	0.00E+00
	B	-7.12E+00	-7.41E+00	-7.59E+00	-7.62E+00		B	-4.74E+02	-4.82E+02	-4.82E+02	-4.83E+02
4	W	-6.41E+00	-7.04E+00	-7.13E+00	-7.62E+00	4	W	-4.64E+02	-4.80E+02	-4.80E+02	-4.80E+02
	M	-6.98E+00	-7.25E+00	-7.33E+00	-7.62E+00		M	-4.67E+02	-4.80E+02	-4.80E+02	-4.81E+02
	SD	8.02E-01	4.32E-02	3.48E-03	2.41E-09		SD	9.44E+00	7.65E-01	8.94E-01	8.14E-01
	B	-7.13E+00	-7.24E+00	-7.38E+00	-7.62E+00		B	-4.70E+02	-4.78E+02	-4.79E+02	-4.80E+02
8	W	-6.26E+00	-6.73E+00	-7.10E+00	-7.62E+00	8	W	-4.59E+02	-4.73E+02	-4.73E+02	-4.76E+02
	M	-6.78E+00	-6.99E+00	-7.18E+00	-7.62E+00		M	-4.64E+02	-4.77E+02	-4.77E+02	-4.78E+02
	SD	6.31E-01	4.51E-01	3.41E-02	4.63E-08		SD	9.94E+00	8.26E-01	8.33E-01	9.11E-01
	B	-7.07E+00	-7.13E+00	-7.20E+00	-7.34E+00		B	-4.69E+02	-4.77E+02	-4.77E+02	-4.80E+02
12	W	-5.99E+00	-6.10E+00	-6.46E+00	-6.56E+00	12	W	-4.53E+02	-4.68E+02	-4.68E+02	-4.71E+02
	M	-6.28E+00	-6.42E+00	-6.71E+00	-6.84E+00		M	-4.58E+02	-4.71E+02	-4.72E+02	-4.74E+02
	SD	7.63E-01	4.21E-01	1.29E-01	9.77E-02		SD	1.89E+01	2.26E+00	1.95E+00	1.46E+00
	B	-6.44E+00	-6.64E+00	-6.69E+00	-6.71E+00		B	-4.68E+02	-4.76E+02	-4.77E+02	-4.78E+02
16	W	-5.35E+00	-5.83E+00	-5.98E+00	-6.11E+00	16	W	-4.50E+02	-4.62E+02	-4.63E+02	-4.67E+02
	M	-6.09E+00	6.16E+00	-6.24E+00	-6.36E+00		M	-4.53E+02	-4.63E+02	-4.68E+02	-4.71E+02
	SD	7.76E-01	4.73E-01	3.22E-01	1.54E-01		SD	1.91E+01	5.32E+00	4.10E+00	2.13E+00

Table 18
Comparative results of H09 and H10 for 240000 function evaluations averaged over 30 runs

H09						**H10**					
Elite		PS=25	PS=50	PS=75	PS=100	Elite		PS=25	PS=50	PS=75	PS=100
	B	-6.84E+01	-6.56E+01	-6.84E+01	-6.84E+01		B	5.68E-06	1.18E-05	1.84E-04	4.67E-04
0	W	-5.96E+01	-6.16E+01	-5.72E+01	-6.39E+01	0	W	2.96E-01	8.59E-02	4.41E-02	2.84E-02
	M	-6.35E+01	-6.32E+01	-6.32E+01	-6.49E+01		M	5.16E-02	1.90E-02	7.14E-03	5.12E-03
	SD	3.43E+00	1.07E+00	2.82E+00	1.36E+00		SD	1.09E-01	3.13E-02	1.63E-02	1.04E-02
	B	-6.56E+01	-6.56E+01	-6.84E+01	-6.84E+01		B	1.55E+00	2.80E-01	1.66E-02	9.58E-03
4	W	-5.71E+01	-6.23E+01	-5.72E+01	-5.72E+01	4	W	2.53E+00	7.75E-01	3.13E-01	2.07E-01
	M	-6.30E+01	-6.37E+01	-6.30E+01	-6.38E+01		M	2.10E+00	5.20E-01	1.23E-01	6.44E-02
	SD	2.51E+00	1.64E+00	2.70E+00	3.24E+00		SD	3.36E-01	1.71E-01	1.02E-01	7.14E-02
	B	-6.84E+01	-6.56E+01	-6.56E+01	-6.83E+01		B	4.05E+00	1.48E+00	2.76E-01	1.78E-01
8	W	-6.06E+01	-6.21E+01	-6.06E+01	-6.23E+01	8	W	5.06E+00	3.14E+00	1.47E+00	7.25E-01
	M	-6.31E+01	-6.35E+01	-6.31E+01	-6.44E+01		M	4.51E+00	2.37E+00	9.64E-01	3.46E-01
	SD	2.18E+00	1.53E+00	1.29E+00	2.01E+00		SD	3.79E-01	5.07E-01	3.94E-01	1.97E-01
	B	-6.56E+01	-6.84E+01	-6.56E+01	-6.84E+01		B	4.84E+00	2.98E+00	1.11E+00	5.35E-01
12	W	-5.96E+01	-6.23E+01	-5.72E+01	-6.23E+01	12	W	6.15E+00	4.01E+00	2.77E+00	2.41E+00
	M	-6.29E+01	-6.35E+01	-6.28E+01	-6.38E+01		M	5.49E+00	3.61E+00	2.13E+00	1.09E+00
	SD	2.07E+00	2.04E+00	2.60E+00	2.10E+00		SD	4.57E-01	4.05E-01	5.67E-01	6.49E-01
	B	-6.56E+01	-6.84E+01	-6.56E+01	-6.84E+01		B	5.25E+00	3.69E+00	7.86E-01	1.20E+00
16	W	-5.69E+01	-6.06E+01	-5.77E+01	-5.96E+01	16	W	6.28E+00	4.90E+00	3.36E+00	1.99E+00
	M	-6.20E+01	-6.31E+01	-6.15E+01	-6.33E+01		M	5.97E+00	4.23E+00	2.56E+00	1.62E+00
	SD	2.34E+00	2.36E+00	2.73E+00	2.85E+00		SD	3.75E-01	3.98E-01	8.51E-01	2.76E-01

Table 19
Comparative results of H11 and H12 for 240000 function evaluations averaged over 30 runs

H11						**H12**					
Elite		PS=25	PS=50	PS=75	PS=100	Elite		PS=25	PS=50	PS=75	PS=100
	B	5.83E+02	5.81E+02	5.81E+02	5.81E+02		B	1.03E-26	4.16E-08	4.51E-06	1.19E-24
0	W	5.91E+02	5.83E+02	5.81E+02	5.81E+02	0	W	2.72E-01	1.19E-01	1.64E-02	1.73E-08
	M	5.87E+02	5.82E+02	5.81E+02	5.81E+02		M	8.68E-02	2.73E-02	3.74E-03	2.47E-09
	SD	1.35E+00	4.26E-01	4.23E-03	3.43E-09		SD	1.17E-01	4.83E-02	6.03E-03	6.52E-09
	B	5.84E+02	5.83E+02	5.82E+02	5.81E+02		B	2.85E-12	2.98E-13	1.80E-14	5.88E-31
4	W	5.93E+02	5.85E+02	5.85E+02	5.83E+02	4	W	1.04E-06	4.40E-08	6.42E-11	3.31E-18
	M	5.91E+02	5.84E+02	5.84E+02	5.82E+02		M	1.57E-07	9.79E-09	1.06E-11	5.02E-19
	SD	3.98E+00	9.54E-01	8.79E-01	6.74E-02		SD	3.91E-07	1.76E-08	2.37E-11	1.24E-18
	B	5.90E+02	5.83E+02	5.83E+02	5.83E+02		B	4.30E-09	2.46E-04	1.38E-07	1.36E-09
8	W	6.01E+02	5.91E+02	5.91E+02	5.88E+02	8	W	4.73E-01	2.63E-03	2.27E-05	7.79E-06
	M	5.96E+02	5.87E+02	5.87E+02	5.86E+02		M	1.59E-01	8.73E-04	6.08E-06	1.45E-06
	SD	5.34E+00	1.10E+00	1.12E+00	1.05E+00		SD	1.64E-01	8.92E-04	8.22E-06	2.88E-06
	B	5.95E+02	5.88E+02	5.87E+02	5.85E+02		B	1.96E-08	1.51E-03	2.71E-06	7.52E-08
12	W	6.09E+02	5.97E+02	5.96E+02	5.93E+02	12	W	7.04E-01	1.97E-02	1.83E-04	3.53E-05
	M	6.00E+02	5.93E+02	5.92E+02	5.90E+02		M	3.24E-01	1.12E-02	4.96E-05	5.97E-06
	SD	5.15E+00	1.60E+00	1.62E+00	1.15E+00		SD	2.60E-01	6.77E-03	6.16E-05	1.30E-05
	B	5.96E+02	5.88E+02	5.87E+02	5.86E+02		B	4.82E-01	1.67E-02	2.32E-04	9.20E-07
16	W	6.14E+02	6.02E+02	6.01E+02	5.98E+02	16	W	1.02E+00	5.47E-02	1.30E-02	3.99E-05
	M	6.11E+02	6.01E+02	5.97E+02	5.94E+02		M	7.91E-01	3.33E-02	3.50E-03	1.43E-05
	SD	6.56E+00	4.87E+00	2.87E+00	1.48E+00		SD	2.26E-01	1.34E-02	4.62E-03	1.47E-05

Table 20

Comparative results of H 13 for 240000 function evaluations averaged over 30 runs **H13**

Elite		PS=25	PS=50	PS=75	PS=100
0	B	-4.64E+01	-4.64E+01	-4.64E+01	-4.64E+01
	W	-4.13E+01	-4.36E+01	-4.13E+01	-4.13E+01
	M	-4.27E+01	-4.60E+01	-4.53E+01	-4.45E+01
	SD	1.93E+00	1.02E+00	2.02E+00	2.38E+00
4	B	-4.64E+01	-4.64E+01	-4.64E+01	-4.64E+01
	W	-4.63E+01	-4.64E+01	-4.64E+01	-4.64E+01
	M	-4.64E+01	-4.64E+01	-4.64E+01	-4.64E+01
	SD	3.14E-02	7.67E-15	7.67E-15	7.67E-15
8	B	-4.60E+01	-4.64E+01	-4.64E+01	-4.64E+01
	W	-2.58E+01	-4.63E+01	-4.62E+01	-4.61E+01
	M	-3.63E+01	-4.64E+01	-4.63E+01	-4.63E+01
	SD	6.70E+00	2.83E-02	7.56E-02	1.13E-01
12	B	-4.44E+01	-4.64E+01	-4.64E+01	-4.64E+01
	W	-1.19E+01	-4.60E+01	-4.55E+01	-4.57E+01
	M	-3.58E+01	-4.63E+01	-4.62E+01	-4.62E+01
	SD	1.13E+01	1.55E-01	3.50E-01	2.70E-01
16	B	-3.85E+01	-4.68E+01	-4.64E+01	-4.64E+01
	W	-1.79E+01	-4.17E+01	-4.32E+01	-4.00E+01
	M	-2.71E+01	-4.54E+01	-4.59E+01	-4.46E+01
	SD	9.74E+00	2.19E+00	1.19E+00	3.01E+00

Table 13 shows the comparative results of the considered algorithm in the form of the best solution, the worst solution and the mean solution. It is observed from Table 13 that TLBO algorithm outperforms the PSO, DE and ABC algorithms for function G02 in every aspect of comparison criteria. For function G01 and G03, the performance of the TLBO and ABC are alike and TLBO outperforms the PSO and DE algorithms. For function G07, the performances of the TLBO and DE are alike and TLBO produces better results than PSO and ABC.

For function G12, the performances of TLBO, ABC and DE are alike and these algorithms produce better results than PSO. For function G10, performance of TLBO is better than the rest of the considered algorithms in terms of mean solution obtained by the algorithms while for function G11 the performance of PSO, ABC and TLBO is similar. For functions G04, G06, G08 and G09, the performance of all the considered algorithms is identical and these algorithms produce equally good results. For functions G05 and G13, the results obtained using PSO are better than the rest of the considered algorithms though the TLBO results are better than DE and ABC algorithms in terms of mean solution. The graphical comparison of TLBO, DE, ABC and PSO algorithms in searching the best and the mean solution is shown in Fig. 2.

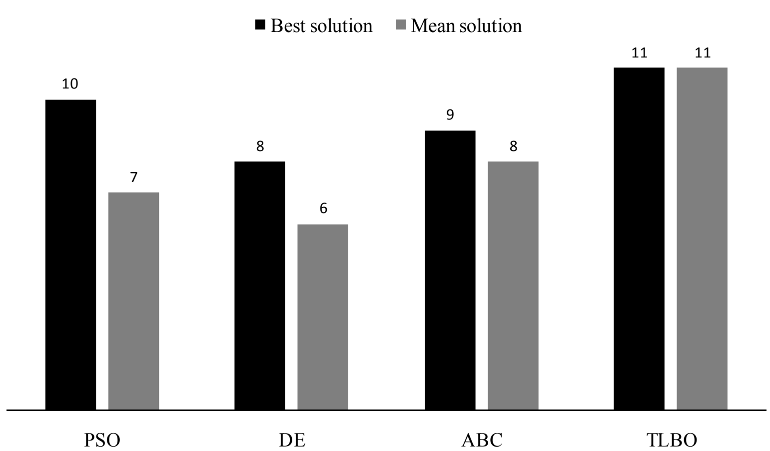
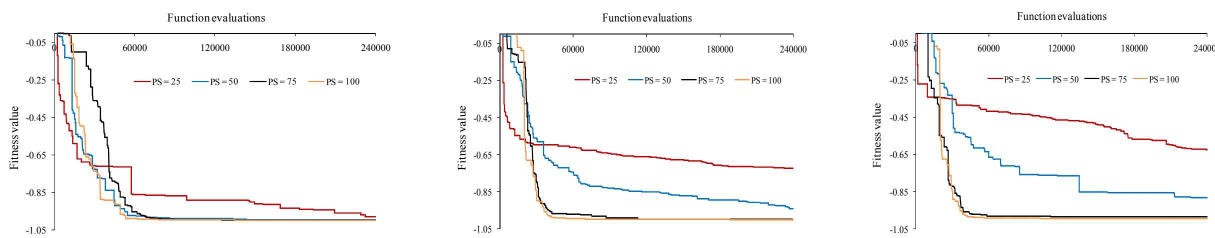


Fig. 2. Comparison of TLBO with other optimization algorithm for the 13 constrained benchmark functions (G 01-G 13) in searching the best and the mean solutions.

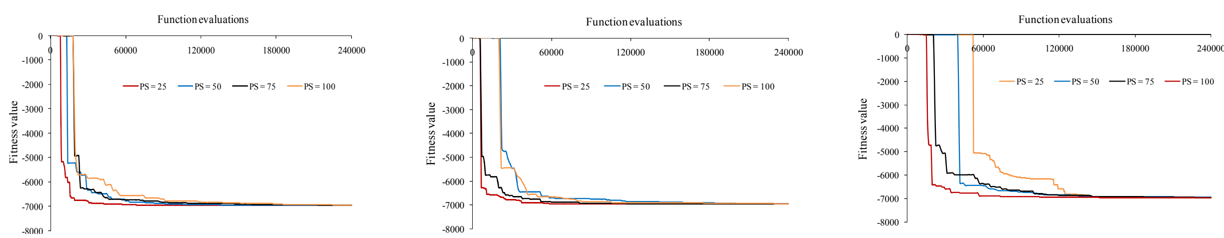
The ability of an algorithm for finding the global optimum value is indicated by black column. The number above the column indicates the total number of functions for which the algorithm is able to find global optimum. Similarly, the grey column indicates the ability of an algorithm in finding the better mean solution. Here also, the number above the column indicates the total number of function for which the mean result obtained by the algorithm is better or comparable to the other considered algorithms.

To identify the effect of population size, number of generations and elite size on the convergence rate of the TLBO algorithm, five benchmark functions (G03, G06, G10, G18 and G19) are considered. The considered benchmark function possess different forms of the objective function (i.e. G03 is polynomial, G06 is cubical, G10 is linear, G18 is quadratic and G19 is non linear) and having different number of variables. The TLBO algorithm is implemented on the considered functions with 240000 function evaluations. Graph is plotted between the fitness value (i.e function value) and function evaluations. Function value taken is the average of function value for 10 different independent runs. Figs. 3-7 show the convergence graphs for different benchmark problems. It is observed from Fig. 3 that for function G03, the convergence rate of algorithm increases with the increase in as population size. The convergence rate is almost similar as the population size increases from 75 to 100. Also, as the elite size increases from 0, the convergence rate of the algorithm reduces.



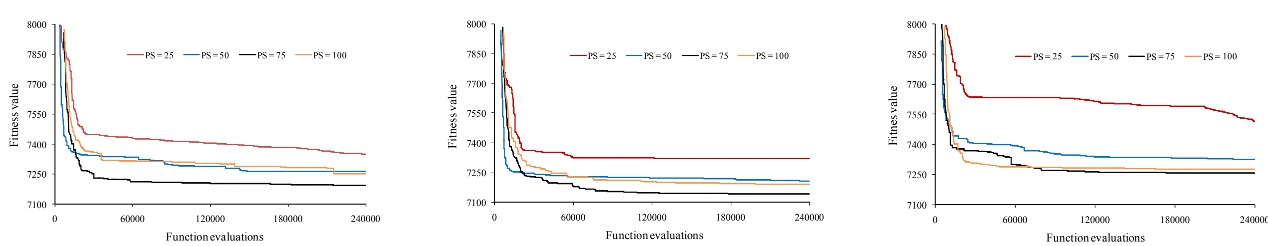
(a) (b) (c)

Fig. 3. Convergence of TLBO for polynomial function (G 03) for 240000 function evaluations averaged over 10 runs, (a) elite size = 0 (b) elite size = 4 and (c) elite size = 8



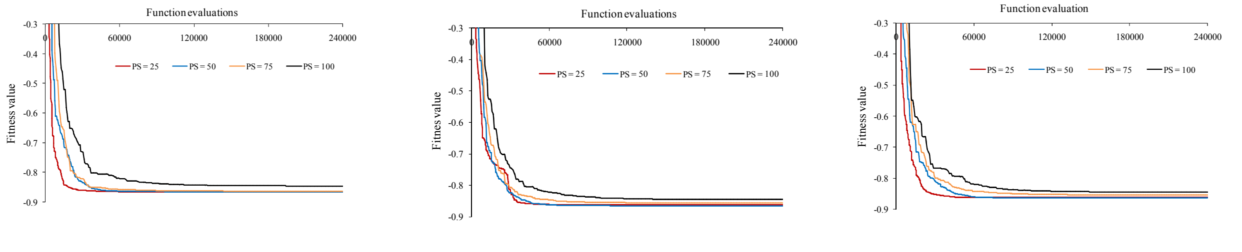
(a) (b) (c)

Fig. 4. Convergence of TLBO for cubic function (G 06) for 240000 function evaluations averaged over 10 runs, (a) elite size = 0 (b) elite size = 4 and (c) elite size = 8



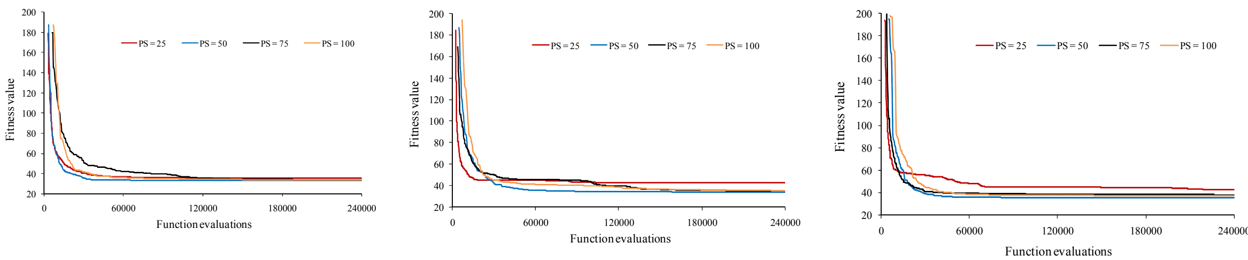
(a) (b) (c)

Fig. 5. Convergence of TLBO for linear function (G 10) for 240000 function evaluations averaged over 10 runs, (a) elite size = 0 (b) elite size = 4 and (c) elite size = 8



(a) (b) (c)

Fig. 6. Convergence of TLBO for quadratic function (G 18) for 240000 function evaluations averaged over 10 runs, (a) elite size = 0 (b) elite size = 4 and (c) elite size = 8



(a) (b) (c)

Fig. 7. Convergence of TLBO for non-linear function (G 19) for 240000 function evaluations averaged over 10 runs, (a) elite size=0 (b) elite size=4 and (c) elite size=8

For function G06, population size of 25 and number of generations of 4700 produce better convergence rate as shown in Fig. 4. For function G10, strategy with population size of 75 and elite size of 4 produces better convergence rate than any other strategy as shown in Fig. 5. For functions G18 and G19, strategy with population size 25 and 50 produces the better convergence rate respectively as shown in Figs. 6 and 7. It is observed from Figs. 3-7 that for any given population size, with increase in the number of generations (i.e. increase in the function evaluations) the performance of the algorithm is improved.

Now the computational complexity of the TLBO algorithm is calculated as per the guidelines given in CEC 2006 (Liang, 2006). G1-G24 functions are considered for calculating the computation complexity. The complexity of the algorithm is given in the form $(T_2 - T_1) / T_1$, where T_1 is the average computing time of 10000 function evaluations for each optimization problem and T_2 is the average of the complete computing time for the algorithm with 10000 evaluations for each optimization problem. The computational time T_1 = 8.6352 s, T_2=10.8934 s and $(T_2 - T_1) / T_1$= 0.2615.

The TLBO is coded in MATLAB 7 and implemented on a laptop having Intel Pentium 2 GHz processor with 1 GB RAM. The code of the TLBO algorithm is given in Appendix of this paper.

5. Experiments on complex constrained optimization problems

In this experiment, the TLBO algorithm is implemented on 13 specifically designed constrained optimization problems. These problems were designed by Mallipeddi and Suganthan (2010) and the details of the problems are available in their work. The capability of the algorithm to find global solution for the constrained problem depends on the constraint handling technique also. In this experiment, ensemble of four different constrained handling techniques, suggested by Mallipeddi ans Suganthan (2010) is used to handle different constraints. An ensemble of constraint handling techniques (ECHT) includes four different constraint handling techniques, viz. superiority of feasible solutions, self-adaptive penalty, ε-constraint and stochastic ranking. The details of ECHT is available in the previous work of Mallipeddi and Suganthan (2010).

Mallipeddi and Suganthan (2010) used DE and EP algorithms along with ECHT and set the maximum number of function evaluations as 240000 for all the functions. In order to maintain the consistence in the comparison, TLBO is also implemented with the 240000 maximum function evaluations. Here also to identify the effects of population size and elite size on the performance of the algorithm, the TLBO algorithm is experimented with different strategies mentioned in the previous experiment. All the functions are experimented 30 times for each strategy with the TLBO algorithm and the comparative results for each strategy are shown in Tables 14-20. Here the comparison criteria are the best solution, worst solution, mean solution and standard deviation obtained from the different independent runs with specified maximum function evaluations.

It is observed from Tables 14-20 that for functions H02 and H07-H12, strategy with population size of 100 produced best results than the other strategies. For functions H06 and H13, strategy with population size of 50 produced the best results. For the rest of the functions (i.e H01, H03-H05), strategy with population size of 25 produced the best results. Similarly, it is observed from Tables 14-20 that for functions H03, H04, H06 and H08-H11, strategy without elitism consideration (i.e. elite size of 0) produced best results than elitism consideration. For functions H02, H07, H12 and H13, strategy with elite size of 4 produced best results. For function H05, strategy with elite size of 12 produced the best results. For function H01, strategy without elitism consideration as well elite size of 4 produced equally good results. Table 21 shows the optimum results obtained by TLBO algorithm for all the H functions.

The performance of TLBO is compared in this experiment with the DE and EP for all the H functions. The results of DE and EP are taken from the previous work of Mallipeddi and Suganthan (2010).

Table 21
Results obtained by TLBO algorithm for 13 benchmark functions over 30 independent runs with 240000 function evaluations

Function	Best	Worst	Mean	SD
H01	0.00E+00	0.00E+00	0.00E+00	0.00E+00
H02	-3.1662	-3.1645	-3.1653	8.93E-04
H03	0.00E+00	0.00E+00	0.00E+00	0.00E+00
H04	3.75E-97	8.58E-93	1.93E-93	3.20E-93
H05	-20.078	-18.8439	-19.7771	5.10E-01
H06	-8.3826	-8.3246	-8.4761	2.23E-07
H07	-7.6159	-7.6159	-7.6159	2.41E-09
H08	-483.6106	-483.6106	-483.6106	0.00E+00
H09	-68.4294	-63.9172	-64.9266	1.36E+00
H10	4.67E-04	2.84E-02	5.12E-03	1.04E-02
H11	580.7304	580.7304	580.7304	3.43E-09
H12	5.88E-31	3.31E-18	5.02E-19	1.24E-18
H13	-46.3756	-46.3756	-46.3756	7.67E-15

Table 22 shows the comparative results of the considered algorithm in the form of best solution, worst solution and mean solution. It is observed from Table 22 that TLBO algorithm outperforms the DE and EP algorithms for functions H01-H04 in every aspect of comparison criteria. For function H10, TLBO outperforms the rest of the algorithms in terms of mean solution. For functions H07, H08 H11 and H13, performances of TLBO, DE and EP are almost identical and produced equally good results. For functions H05, H09 and H12, the results obtained using DE are better than the TLBO results.

Table 22
Comparative results of TLBO with other evolutionary algorithms over 30 independent runs

		DE	EP	TLBO			DE	EP	TLBO
H01	B	8.29E-83	5.58E-13	0.00E+00	H02	B	1.01E−92	1.89E−11	1.93E-93
	W	7.41E-77	1.04E-10	0.00E+00		W	1.85E−92	5.40E−11	3.20E-93
	M	2.66E-78	3.02E-11	0.00E+00		M	−8.3826	−8.8326	-8.3826
H03	B	1.35E-77	3.19E-11	0.00E+00	H04	B	−8.3826	−8.8327	-8.3246
	W	1.19E−83	1.00E−15	0.00E+00		W	−8.3826	−8.8328	-8.4761
	M	3.68E−80	2.38E−09	0.00E+00		M	3.76E−15	1.77E-05	2.23E-07
H05	B	6.90E−81	3.26E−10	0.00E+00	H06	B	−483.6106	−483.6106	-483.6106
	W	1.12E−80	7.43E−10	0.00E+00		W	−483.6107	−483.6107	-483.6106
	M	−20.0780	−20.0780	-20.078		M	−483.6108	−483.6108	-483.6106
H07	B	−20.0599	−18.0109	-18.8439	H08	B	0.00E+00	1.00E+00	0.00E+00
	W	−20.0774	−19.3877	-19.7771		W	0.00E+00	0.00E+00	4.67E-04
	M	3.30E-03	6.00E-01	5.10E-01		M	8.99E+00	8.99E+00	2.84E-02
H09	B	−7.6159	−7.6159	-7.6159	H10	B	5.99E-01	3.60E+00	5.12E-03
	W	−7.6159	−7.6159	-7.6159		W	2.28E+00	4.64E+00	1.04E-02
	M	−7.6159	−7.6159	-7.6159		M	1.54E−32	5.00E-07	5.88E-31
H11	B	4.26E−10	3.18E-09	2.41E-09	H12	B	1.75E−30	1.06E−05	3.31E-18
	W	−68.4294	−68.4294	-68.4294		W	4.55E−31	1.95E−06	5.02E-19
	M	−63.5175	−63.5174	-63.9172		M	4.61E−31	3.06E−06	1.24E-18
H13	B	−67.9231	−64.9120	-64.9266					
	W	1.09E+00	2.04E+00	1.36E+00					
	M	580.7301	580.7301	580.7304					

For function H06, the results obtained using EP are better than the TLBO results. The graphical comparison of TLBO, DE and EP in searching the best and the mean solutions is shown in Fig. 8. The black and grey columns of Fig. 8 indicate the ability of the algorithm to find global optimum and better mean solution respectively.

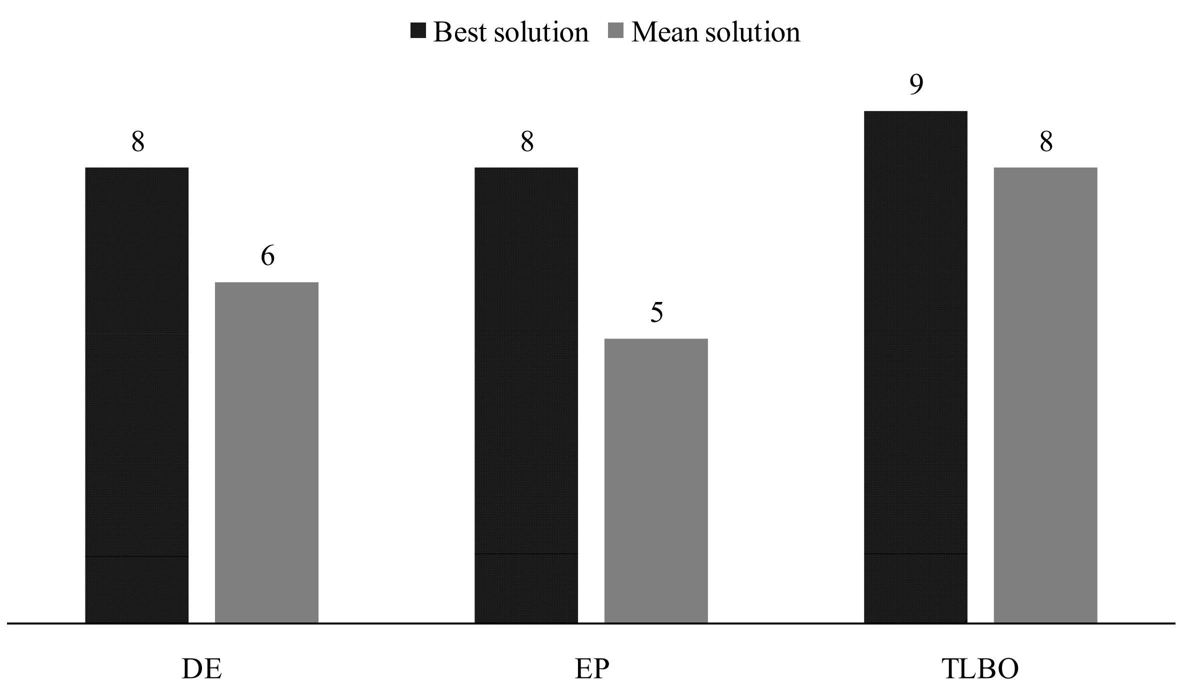


Fig. 8. Comparison of TLBO with other optimization algorithm for the 13 constrained benchmark functions (H 01-H 13) in searching the best and the mean solutions.

It is observed from both the experiments that out of 35 constrained benchmark functions, TLBO algorithm without elitism consideration has given better results in the case of 15 functions. For rest of the functions, different elite sizes have produced better results. Thus, it may be said that the concept of elitism enhances the performance of the TLBO algorithm for the constrained optimization problems. Similarly, it is observed from both the experiments that for majority of the problems the strategy with higher population size produced the better results. Smaller population size required more number of iterations to achieve the global optimum value. For some class of problems the strategy with smaller population size produced the promising results than higher population size. Thus, similar to the other evolutionary or swarm intelligence based algorithms, the TLBO algorithm requires proper tuning of the common controlling parameters (i.e. population size, number of generations and elite size) before applying it to any problem. However, TLBO does not require any algorithm-specific control parameters.

6. Conclusion

All the evolutionary and swarm intelligence based algorithms require proper tuning of algorithm-specific parameters in addition to tuning of common controlling parameters. A change in the tuning of the algorithm specific parameters influences the effectiveness of the algorithm. The recently proposed TLBO algorithm does not require any algorithm-specific parameters. It only requires the tuning of the common controlling parameters of the algorithm for its working. In the present work, the concept of elitism is introduced in the TLBO algorithm and its effect on the performance of the algorithm for the constrained optimization problems is investigated. Moreover, the effect of common controlling parameters (i.e population size, elite size and number of generations) on the performance of TLBO algorithm is also investigated by considering different combinations of common controlling parameters. The proposed algorithm is implemented on 35 well defined constrained optimization problems having different characteristics to identify the effect of elitism and common controlling parameters. The results show that for many functions the strategy with elitism consideration produces better results than that without elitism consideration. Also, in general, the strategy with higher population size has produced better results than that with smaller population size for same number of function evaluations. The results obtained by using TLBO algorithm are compared with the other optimization algorithms available in the literature for the considered benchmark problems. Results have shown the satisfactory performance of TLBO algorithm for the constrained optimization problems.

The proposed algorithm can be easily applied to various optimization problems of the industrial environment such as job shop scheduling, flow shop scheduling, FMS scheduling, design of cellular manufacturing systems, project scheduling; design of facility location networks; portfolio optimization; determination of optimal ordering and pricing policies; supplier selection and order lot sizing; assembly line balancing; inventory control; production planning and control; locating distribution centers and allocating customers demands in supply chains; vehicle-routing problems in transportation, etc. In general, the proposed algorithm may be easily customized to suit the optimization of any system involving large number of variables and objectives.

References

Ahrari, A. & Atai A. A. (2010). Grenade explosion method - A novel tool for optimization of multimodal functions. *Applied Soft Computing*, 10, 1132-1140.

Basturk, B & Karaboga, D. (2006). An artificial bee colony (ABC) algorithm for numeric function optimization, in: *IEEE Swarm Intelligence Symposium*, Indianapolis, Indiana, USA.

Deb, K. (2000). An efficient constraint handling method for genetic algorithm. *Computer Methods in Applied Mechanics and Engineering*, 186, 311–338.

Dorigo, M., Maniezzo V. & Colorni A. (1991). Positive feedback as a search strategy, *Technical Report 91-016*. Politecnico di Milano, Italy.

Eusuff, M. & Lansey, E. (2003). Optimization of water distribution network design using the shuffled frog leaping algorithm. *Journal of Water Resources Planning and Management, 2*9, 210-225.

Farmer, J. D., Packard, N. & Perelson, A. (1986).The immune system, adaptation and machine learning, *Physica D,* 22,187-204.

Fogel, L. J, Owens, A. J. & Walsh, M.J. (1966). *Artificial intelligence through simulated evolution*. John Wiley, New York.

Geem, Z. W., Kim, J.H. & Loganathan G.V. (2001). A new heuristic optimization algorithm: harmony search. *Simulation*, 76, 60-70.

Holland, J. (1975). *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor.

Karaboga, D. & Basturk, B. (2008). On the performance of artificial bee colony (ABC) algorithm. *Applied Soft Computing*, 8 (1), 687–697.

Karaboga, D. & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, 39 (3), 459–471.

Karaboga, D. & Basturk, B. (2007). Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems. *LNCS: Advances in Soft Computing: Foundations of Fuzzy Logic and Soft Computing,* 4529, 789-798.

Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization, *Technical Report-TR06*, Computer Engineering Department. Erciyes University, Turkey.

Kennedy, J. & Eberhart, R. C. (1995). Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks,* IEEE Press, Piscataway, 1942-1948.

Liang, J.J., Runarsson, T.P., Mezura-Montes, E., Clerc, M., Suganthan, P.N., Coello, C.A.C, Deb, K. (2006). Problem definitions and evaluation criteria for the CEC special session on constrained real-parameter optimization, *Technical Report, Nanyang Technological University*. Singapore, http://www.ntu.edu.sg/home/EPNSugan.

Mallipeddi, R. & Suganthan, P.N. (2010). Ensemble of constraint handling techniques. *IEEE Transactions on Evolutionary Computation*, 14 (4), 561-579.

Passino, K.M. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine,* 22, 52–67.

Price K., Storn, R, & Lampinen, A. (2005). Differential evolution - a practical approach to global optimization, *Springer Natural Computing Series*.

Rao, R.V. & Savsani, V.J. (2012). *Mechanical design optimization using advanced optimization techniques*. Springer-Verlag, London.

Rao, R.V., Savsani, V.J. & Vakharia, D.P. (2012). Teaching-learning-based optimization: A novel optimization method for continuous non-linear large scale problems. *Information Science*s, 183 (1), 1-15.

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

Rao, R.V. & Patel, V.K. (2012). Multi-objective optimization of combined Brayton abd reverse Brayton cycles using advanced optimization algorithms. Engineering Optimization, DOI: 10.1080/0305215X.2011.624183.

Rashedi, E., Nezamabadi-pour, H. & Saryazdi, S. (2009). GSA: A gravitational search algorithm, *Information Sciences*, 179, 2232-2248.

Runarsson, T.P. &Yao X. (2000) Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation,* 4 (3), 284-294.

Simon, D. (2008) Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation,* 12, 702–713.

Storn, R. & Price, K. (1997). Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization,* 11, 341-359.

Appendix

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%% TLBO %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function TLBO(obj_fun, note1, note2)
format long;
global ll
if ~exist('note1', 'var')
   note1 = true;
end
if ~exist('note2', 'var')
   note2 = true;
end
[Students, select, upper_limit, lower_limit, ini_fun, min_result, avg_result, result_fun, opti_fun,
result_fun_new, opti_fun_new] = Initialize(note1, obj_fun);
elite=0;
for COMP = 1 : select.itration
   for i = 1 : elite
      markelite(i,:) = Students(i).mark;
      resultelite(i) = Students(i).result;
   end
   for i=1:length(Students)
   cs(i,:)=Students(i).mark;
   cs_result(i)=Students(i).result;
   end
   cs;
   cs_result;
 for i = 1 : length(Students)
   mean_result=mean(cs);
   TF=round(1+rand*(1));
   [r1 r2]=sort(cs_result);
   best=cs(r2(1),:);
   for k = 1 : select.var_num
       cs_new(i,k)=cs(i,k)+((best(1,k)-TF*mean_result(k))*rand);
   end
      cs_new(i,:) = opti_fun_new(select, cs_new(i,:));
      cs_new_result(i) = result_fun_new(select, cs_new(i,:));
      if cs_new_result(i)<Students(i).result
          Students(i).mark =cs_new(i,:);
          cs(i,:)=cs_new(i,:);
          Students(i).result=cs_new_result(i);
      end
      hh=ceil(length(Students)*rand);
      while hh==i
            hh=ceil(length(Students)*rand);
      end
     if Students(i).result<Students(hh).result
       for k = 1 : select.var_num
        cs_new(i,k)= Students(i).mark(k) + ((Students(i).mark(k) - Students(hh).mark(k))*rand);
       end
     else
       for k = 1 : select.var_num
        cs_new(i,k)= Students(i).mark(k) + ((Students(hh).mark(k) - Students(i).mark(k))*rand);
```

```
         end
     end
        cs_new(i,:) = opti_fun_new(select, cs_new(i,:));
        cs_new_result(i) = result_fun_new(select, cs_new(i,:));
     if cs_new_result(i)<Students(i).result
          Students(i).mark =cs_new(i,:);
          cs(i,:)=cs_new(i,:);
          Students(i).result=cs_new_result(i);
      end
 end
   n = length(Students);
   Students = opti_fun(select, Students);
   Students = result_fun(select, Students);
   Students = sortstudents(Students);
   for i = 1 : elite
       Students(n-(i-1)).mark = markelite(i,:);
       Students(n-(i-1)).result = resultelite(i);
   end
   if rand<1
   Students = remove_duplicate(Students, upper_limit, lower_limit);
   end
   Students = sortstudents(Students);
   [average_result, within_bound] = result_avg(Students);
   min_result = [min_result Students(1).result];
   avg_result = [avg_result average_result];
   Mark = (Students(1).mark);
   if note1
              disp([num2str(min_result(end))]);
              disp([num2str(Mark)]);
   end
end
fprintf('\n %e',min_result(end));
fprintf('\n %6.10f',Mark);
out_put(note1, select, Students, within_bound, min_result);

%%%%%%%%%%%%%%%%%%%%%% %%%  IMPLEMENT  %%%%%%%%%%%%%%%%%%%%%%
function [ini_fun, result_fun, result_fun_new, opti_fun, opti_fun_new] = implement
format long;
ini_fun = @implementInitialize ;
result_fun = @implementresult;
result_fun_new = @implementresult_new;
opti_fun = @implementopti;
opti_fun_new = @implementopti_new;
return;
function [upper_limit, lower_limit, Students, select] = implementInitialize(select)
global lower_limit upper_limit ll ul
Granularity = 1;
lower_limit = ll;
upper_limit = ul;
ll=[78 33 27 27 27];
ul=[102 45 45 45 45];
lower_limit = ll;
```

```
upper_limit = ul;
for popindex = 1 : select.classsize
    for k = 1 : select.var_num
        mark(k) =(ll(k))+ ((ul(k) - ll(k)) * rand);
     end
        Students(popindex).mark = mark;
end
select.OrderDependent = true;
return;
function [Students] = implementresult(select, Students)
global lower_limit upper_limit
classsize = select.classsize;
for popindex = 1 : classsize
    for k = 1 : select.var_num
        x(k) = Students(popindex).mark(k);
    end
    Students(popindex).result = objective(x);
end
return
function [Studentss] = implementresult_new(select, Students)
global lower_limit upper_limit
classsize = select.classsize;
for popindex = 1 : size(Students,1)
    for k = 1 : select.var_num
        x(k) = Students(popindex,k);
    end
    Studentss(popindex) = objective(x);
end
return
function [Students] = implementopti(select, Students)
global lower_limit upper_limit ll ul
for i = 1 : select.classsize
    for k = 1 : select.var_num
        Students(i).mark(k) = max(Students(i).mark(k), ll(k));
        Students(i).mark(k) = min(Students(i).mark(k), upper_limit(k));
    end
end
return;
function [Students] = implementopti_new(select, Students)
global lower_limit upper_limit ll ul
for i = 1 : size(Students,1)
    for k = 1 : select.var_num
        Students(i,k)= max(Students(i,k), ll(k));
        Students(i,k) = min(Students(i,k), upper_limit(k));
    end
end
return;
%%%%%%%%%%%%%%%%%%%%%%%%   INITIALIZATION %%%%%%%%%%%%%%%%%%
function [Students, select, upper_limit, lower_limit, ini_fun, min_result, avg_result, result_fun,
opti_fun, result_fun_new, opti_fun_new] = Initialize(note1, obj_fun, RandSeed)
format long;
select.classsize =100;
```

```
select.var_num = 5;
select.itration = 300;
if ~exist('RandSeed', 'var')
    rand_gen = round(sum(100*clock));
end
rand('state', rand_gen);
[ini_fun, result_fun, result_fun_new, opti_fun, opti_fun_new,] = obj_fun();
[upper_limit, lower_limit, Students, select] = ini_fun(select);
Students = remove_duplicate(Students, upper_limit, lower_limit);
Students = result_fun(select, Students);
Students = sortstudents(Students);
average_result = result_avg(Students);
min_result = [Students(1).result];
avg_result = [average_result];
return;
%%%%%%%%%%%%%%%%%%%%%%%% OBJECTIVE %%%%%%%%%%%%%%%%%%%%%
function yy=objective(x)
format long;
 p1=x(1);
 p2=x(2);
 p3=x(3);
 p4=x(4);
 p5=x(5);
ZZ=(5.3578547*(p3^2))+(0.8356891*p1*p5)+(37.293239*p1)-(40792.141);
t1=85.334407+(0.0056858*p2*p5)+(0.0006262*p1*p4)-(0.0022053*p3*p5)-92;
t2=-85.334407-(0.0056858*p2*p5)-(0.0006262*p1*p4)+(0.0022053*p3*p5);
t3=80.51249+(0.007137*p2*p5)+(0.0029955*p1*p2)+(0.0021813*(p3^2))-110;
t4=-80.51249-(0.007137*p2*p5)-(0.0029955*p1*p2)-(0.0021813*(p3^2))+90;
t5=9.300961+(0.0047026*p3*p5)+(0.0012547*p1*p3)+(0.0019085*p3*p4)-25;
t6=-9.300961-(0.0047026*p3*p5)-(0.0012547*p1*p3)-(0.0019085*p3*p4)+20;
nc=6;
            g1(1)=t1;
            g1(2)=t2;
            g1(3)=t3;
            g1(4)=t4;
            g1(5)=t5;
            g1(6)=t6;
            fun=0;
            cov=0;
            for io=1:nc
                if g1(io)>0
                    fun=fun+g1(io)^2;
                    cov=cov+1;
                end
            end
yy=(ZZ)+(1e20*fun)+(1e15*cov);
%%%%%%%%%%%%%%%%%%%%%%%% SORTSTUDENTS %%%%%%%%%%%%%%%%%%%%
function [Students, indices] = sortstudents(Students)
classsize = length(Students);
Result = zeros(1, classsize);
indices = zeros(1, classsize);
for i = 1 : classsize
```

```
    Result(i) = Students(i).result;
end
[Result, indices] = sort(Result, 2, 'ascend');
Marks = zeros(classsize, length(Students(1).mark));
for i = 1 : classsize
    Marks(i, :) = Students(indices(i)).mark;
end
for i = 1 : classsize
    Students(i).mark = Marks(i,:);
    Students(i).result = Result(i);
End
%%%%%%%%%%%%%%%%%%%%%%%%%% OUTPUT %%%%%%%%%%%%%%%%%%%%%%%%
function out_put(note1, select, Students, within_bound, min_result)
format long;
if note1
    duplicate_no = 0;
    for i = 1 : select.classsize
        Mark_1 = sort(Students(i).mark);
        for k = i+1 : select.classsize
            Mark_2 = sort(Students(k).mark);
            if isequal(Mark_1, Mark_2)
                duplicate_no = duplicate_no + 1;
            end
        end
    end
    Mark = sort(Students(1).mark);
end
return;
 %%%%%%%%%%%%%%%%%%%%%%%%% AVG_RESULT %%%%%%%%%%%%%%%%%%%%%%%
function [result_av, within_bound] = result_avg(Students)
format long;
Result = [];
within_bound = 0;
for i = 1 : length(Students)
    if Students(i).result < inf
        Result = [Result Students(i).result];
        within_bound = within_bound + 1;
    end
end
result_av = mean(Result);
return;
%%%%%%%%%%%%%%%%%%%%%%% REMOVE DUPLICATE %%%%%%%%%%%%%%%%%%%%
function [Students] = remove_duplicate(Students, upper_limit, lower_limit)
format long;
global  ll ul
for i = 1 : length(Students)
    Mark_1 = sort(Students(i).mark);
    for k = i+1 : length(Students)
        Mark_2 = sort(Students(k).mark);
        if isequal(Mark_1, Mark_2)
            m_new = floor(1+(length(Students(k).mark)-1)*(rand));
            if length(upper_limit)==1
```

```
            Students(k).mark(m_new) = (lower_limit + (upper_limit - lower_limit) * rand);
            else
            Students(k).mark(m_new) = (ll(m_new) + (upper_limit(m_new) - ll(m_new)) * rand);
            end
        end
    end
end
return;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% RUNTLBO %%%%%%%%%%%%%%%%%%%%%%%%%
function run_tlbo()
clc;
run=1;
format long;
for i=1:run
    TLBO(@implement);
end
```