EAI Endorsed Transactions
on Scalable Information Systems

Research Article EAI.EU

Master-Slave TLBO Algorithm for Constrained Global Optimization Problems

Sandeep U. Mane[1,*], Amol C. Adamuthe[2] and Rajshree R. Omane[3]

[1]Dept. of CSE, Rajarambapu Institute of Technology (affiliated to Shivaji University Kolhapur), Rajaramnagar, Dist. Sangli, MH, India
[2]Dept. of CS&IT, Rajarambapu Institute of Technology (affiliated to Shivaji University Kolhapur), Rajaramnagar, Dist. Sangli, MH, India.
[3]Associate Software Engineer, Amdocs, Pune, MH, India.

Abstract

INTRODUCTION: The teaching-learning based optimization (TLBO) algorithm is a recently developed algorithm. The proposed work presents a design of a master-slave TLBO algorithm.
OBJECTIVES: This research aims to design a master-slave TLBO algorithm to improve its performance and system utilization for CEC2006 single-objective benchmark functions.
METHODS: The proposed approach implemented using OpenMP and CUDA C, a hybrid programming approach to enhance the utilization of the system's computational resources. The device utilization and performance of the proposed approach evaluated using CEC2006 benchmark functions.
RESULTS: The proposed approach obtains best results in significantly reduced time for CEC2006 benchmark functions. The maximum speed-up achieved is 30.14X. The average GPGPU utilization is 90% and the average utilization of logical processors is more than 90%.
CONCLUSION: The master-slave TLBO algorithm improves the utilization of computational resources significantly and obtains the best results for CEC2006 benchmark functions.

Keywords: Master-slave TLBO algorithm, Parallel Evolutionary Algorithms, GPGPU, Constrained benchmark functions, Optimization problems.

Received on 07 June 2020, accepted on 04 September 2020, published on 09 September 2020

doi: 10.4108/eai.26-5-2020.166292

*Corresponding author. Email: manesandip82@gmail.com

1. Introduction

Optimization is defined as "Finding an alternative with the most cost effective or highest achievable performance, by maximizing desired factors and minimizing undesired ones" [1]. Optimization functions are discrete/continuous and constrained/unconstrained types. The optimization problems found in engineering and other domains are constrained and unconstrained in nature. The constrained optimization problems are optimized concerning certain restrictions. The restrictions exist on different things like resources availability, time, etc. The unconstrained optimization problems are free from such restrictions. These problems are optimized with respect to design variables and their range as well as dimensionality. The constrained and unconstrained optimization problems are of single and multi-objective optimizations [2-5].

In literature, different classical methods used to solve the constrained and unconstrained optimization problems. These methods have their own merits and demerits. Researchers have developed nature-inspired approaches to solving complex engineering design and optimization problems [2]. When an algorithm is proposed newly, validation and efficiency of the algorithm need to be evaluated. Therefore, the algorithm is tested on some standard benchmark functions. The test on benchmark functions ensures the suitability of the algorithm to problems with specific properties. The nature of benchmark functions is of different types. The functions are unimodal, multimodal, single and multidimensional. Real word applications belong to these categories and by implementing the proposed algorithm to these benchmark functions, one can determine which kinds of real-world problems the algorithm suits [3-5].

The nature-inspired algorithms are problem-solving approaches inspired by different phenomena that exist in nature. The nature-inspired algorithms are developed by understanding the behaviour of swarms, biological systems, physical and chemical systems, etc. The popular nature-inspired algorithms are Genetic algorithms, Particle swarm optimization, Artificial bee colony algorithm, Ant colony optimization, Intelligent water drop algorithm, Cuckoo search algorithm, Teaching-Learning based optimization algorithm, etc. The literature reveals various applications of nature-inspired algorithms in different domains. Recently, authors have used nature-inspired approaches for virtual machine placement in the cloud. Adhikari and Amgoth used intelligent water drop algorithm for workflow scheduling in cloud data-centre [6]. Abdessamia et al. developed an energy-efficient virtual machine placement using binary gravitational search algorithm [7]. Jangiti et al. used a heuristic approach to perform virtual machine placement in the heterogeneous cloud data centre [8]. Jangiti et al. presented bulk-bin-packing based migration management approach to address the reserved virtual machine request problem in green cloud computing [9]. Tian et al. have used a heuristic approach for scheduling of virtual machine reservations in cloud data centres [10]. Basiri and Kabiri developed a machine learning-based approach for opinion mining, a subfield of data mining [11]. Sharma et al. presented a genetic algorithm (GA) and ontology-based NLP frameworks for online opinion mining [12]. Sharma et al. presented a well-organised study on the use of nature-inspired techniques in agriculture, finance, healthcare, education and engineering domains. Also, authors have presented the publication trends from 2010 to 2019 in selected domains [13]. Aljarah et al. developed a multi-verse optimization algorithm to solve data clustering problems [14]. Sharma and Kaur presented an analysis of nature-inspired meta-heuristic techniques employed in stock prediction, intrusion detection, disease diagnosis, image processing, bioinformatics, agriculture, text mining, robotics, finance, and educational data mining for feature selection [15]. Mafarja et al. presented an extensive literature study about the binary variant of dragonfly algorithm for feature selection [16]. Vinay Kumar et al. used the interactive self-improvement based adaptive particle swarm optimization for optimal floor planning in VLSI circuit design [17]. Saremi et al. presented the application of grasshopper optimization algorithm in hand posture estimation. Authors have also presented literature study about grasshopper optimization algorithm [18]. Mirjalili and Dong introduced nature-inspired algorithms with its importance using TSP problem [19]. The cluster head selection is one of the challenging problem found in the wireless sensor network. John and Rodrigues presented a literature survey on different techniques used for cluster head selection. The nature-inspired optimization techniques such as particle swarm optimization, artificial bee colony, genetic algorithm and harmony search algorithm used to select the cluster head in wireless sensor network [20]. Miranda et al. compared the NSGA-II, SMS-EMOA, and MOEA/D multi-objective optimization algorithms to solve the cluster head selection problem [21]. The self-adaptive mutation factor cross-over probability-based differential evolution algorithm developed by Annepu and Rajesh to solve node localization problem in wireless sensor network [22]. The development in multi and many-objective nature-inspired optimization techniques with its applications is presented in [23].

The nature-inspired algorithms have algorithm-specific parameters. The success of such algorithms largely depends on the efficient tuning of algorithm-specific parameters. Inefficient tuning of algorithm-specific parameters affects the solution of optimization problems. To reduce the impact of algorithm-specific parameters the teaching-learning based optimization (TLBO) algorithm have developed [24]. It requires to tune only common control parameters namely population size and termination criteria. The TLBO algorithm is popular and used by various researchers to solve complex engineering optimization problems. The different variations of TLBO algorithms and their performance comparison presented in [25]. Rao has presented a survey about the applications of TLBO algorithm [26]. Zou et al. presented a TLBO algorithm to solve multi-objective optimization problems [27].

With the era of increasing processing speeds, computer architects are exploring new ways to increase throughput. One of the most promising technique is to exploit parallelism. If your application use parallelism, resources are used more efficiently and performance is increased. The main advantage is that the CPU overhead is minimized. Some of the applications require more time to solve the problem, as it contains a large number of tasks so distributing those tasks in a balanced way across available resources improves the performance. This is the basic need for parallelism. Due to advancement in computer systems processors, the utilization of such systems is the need of the hour. As evolutionary algorithms are inherent in parallel nature, the parallel development of optimization algorithms will take benefit of it [28-30]. Gong et al. presented a survey on the parallel implementation of evolutionary algorithms [31]. Authors have categorised the parallelization strategy adopted to develop parallel versions of evolutionary algorithms. The future research direction highlights the importance of the parallel development of evolutionary algorithms [31]. The GPGPU based development of the population and swarm-based implementation on GPU is discussed with real-world problems from different domains such as data mining, bioinformatics, drug discovery, crystallography, artificial chemistries, and Sudoku [32].

The proposed work presents the design and implementation of a master-slave TLBO algorithm to solve

CEC2006 single-objective constrained optimization problems. The motivation behind this work is to improve the execution time, enhance device utilization and the performance of an algorithm. The novelty of this work is that the Teacher phase and Learner phase of TLBO algorithm is executed on GPGPU. Generally, in sequential program execution, the single-core (logical processor) of the multi-core CPU is utilized. In the proposed approach, to enhance the CPU utilization OpenMP programming model is used. The Odd-Even sorting is implemented to determine the best value from obtained solutions. The proposed algorithm tested using thirteen single-objective constrained benchmark functions. The master-slave strategy is adopted to develop GPGPU based TLBO algorithm.

The main contributions of this paper are as follows. The Master-Slave TLBO algorithm is presented to improve the CPU and GPGPU utilization. The proposed approach implemented using OpenMP-CUDA C, a hybrid parallel programming approach. The proposed algorithm's performance is evaluated using well-known CEC2006 single-objective constrained benchmark functions. The device utilization, speed-up computed and performance of the algorithm is measured using a statistical test.

The remaining paper is organized as follows: Section 2 discusses the related work. Section 3 presents the proposed methodology. The single-objective benchmark functions used in the proposed work are discussed in section 4. The obtained results and discussion are presented in section 5. Section 6 outlines the conclusion and future research directions.

2. Related work

This section presents the literature review of different evolutionary algorithms developed on GPGPU to solve benchmark problems, the variations of TLBO algorithms and parallel TLBO implementation.

Johannes Hofmann et al. in [33] studied genetic algorithm on the graphics card. In this paper, the author tried to find out which steps of the genetic algorithm (GA) can be done on the graphics processing unit using profiler so that algorithm can effectively work in parallel. The algorithm is tested on the Weierstrass function. The first phase of generating population randomly is done on the GPU where the time required to generate a random number on each card is considered. In the second phase, crossover operation is performed in which each thread can operate on two offspring. The third phase of mutation is done parallel in which, each thread will operate on single offspring. In [34], the author implemented a genetic algorithm for three benchmark functions on GPGPU using CUDA. In [35], the GA on multi-core and many-core systems implemented using different approaches like master-slave, coarse-grained, fine-grained approaches. The multi-core and many-core architecture make use of thread-level parallelism to improve the performance. Luca Mussi et al. in [36] discussed possible approaches of parallelizing PSO on graphics hardware. The main attention was given to minimize the data transfer, minimize the use of global memory using one CUDA kernel per swarm. Jitendra Kumar et al. also implemented PSO on GPGPU. Initial population generated on GPU minimizes the time required for copying of data [37]. The Bees algorithm, artificial bee colony and multi-hive artificial bee colony algorithm are implemented on GPU using CUDA to address the benchmark optimization problems [38-40]. The ant colony optimization algorithm is parallelized using CUDA on GPU [41-42].

In [43], the improved version of TLBO based on the orthogonal design, with a new selection strategy to decrease the number of generations is proposed. The classes of learners are divided into some vectors where each of them acts as a factor in the orthogonal design. In [44], the TLBO algorithm modified by adding the concept of tutorial class. To make a stochastic variation in the available solution, the random scale factor added to the learner solution. It helps to maintain diversity and a better value obtained in the multimodal surface. Rao and Patel proposed improved TLBO adding the number of teachers, self-adapting factor, and tutorial-based learning [45]. The TLBO tested for continuous non-linear large-scale benchmark functions [46]. Zou et al. presented a survey of TLBO. Authors discussed the working of basic TLBO algorithm and presented a survey of its variations and applications developed using TLBO. The analysis of TLBO also presented [47].

The researchers developed the parallel TLBO algorithm and implemented on GPU. Rico-Garcia et al. implemented TLBO on GPU and compared with Jaya on GPU. Authors used unconstrained benchmark functions to test the proposed approach. Authors also analysed the utilization of GPUs by each approach [48]. García-Monzó et al. developed a shared memory-based and message-passing based parallel TLBO algorithm. Authors have used thirty unconstrained benchmark functions to evaluate the performance of proposed approach [49]. Other parallel implementation of TLBO found in [2, 50] to solve unconstrained benchmark functions.

The findings of the literature study are, the TLBO algorithm is an algorithm-specific parameter-less approach developed to solve various optimization problems. There exist different variations of TLBO. The evolutionary algorithms successfully implemented to solve standard benchmark problems on GPGPU. The few researchers have developed a multi-core or many-core system based TLBO algorithm. The parallel TLBO algorithms are tested mostly for unconstrained optimization problems. These findings motivate the author of this paper to develop a master-slave TLBO algorithm to solve constrained benchmark optimization problems.

3. Methodology

This section presents the proposed master-slave TLBO algorithm with a flowchart.

3.1. Teaching-Learning based optimization algorithm

The Teaching-Learning Based Optimization (TLBO) algorithm proposed by Rao et al. for solving various types of optimization problems. The TLBO algorithm is inspired by the teaching-learning process. The special feature of TLBO algorithm is that authors have removed the algorithm-specific parameter tuning. The detailed working with the flowchart of the TLBO algorithm and demonstration with manually solved constrained and unconstrained optimization problems as well as its applications can be found in [26].

3.2. Proposed approach

The proposed master-slave TLBO algorithm is described in this section. The proposed approach described with the flowchart. The master-slave based TLBO algorithm's execution steps are also discussed.

3.2.1 Master-slave model

In the literature, there are different parallelization models exists to implement evolutionary algorithms in a parallel fashion. The widely used are master-slave, island, cellular, hierarchical, pool, coevolution and multi-agent models [51]. There are different levels of parallelism found in each model. The operation level parallelism is used in a master-slave approach, where compute-intensive operations are executed on slaves. The limitation of the master-slave model is communication cost. It is due to the slave communicates frequently with the master to exchange results and data [51]. The development of GPGPU eases the implementation of master-slave evolutionary algorithms. The CPU acts as master whereas GPGPU works as a slave.

3.2.2 Master-slave TLBO algorithm

This section presents the master-slave TLBO algorithm.

The GPGPU based parallel TLBO algorithm is developed based on the master-slave model to solve constrained benchmark problems. The proposed approach designates CPU as a master while GPGPU works as slaves. The compute-intensive or the steps which require more time are executed on GPGPU. These include generation of the initial population, teacher phase, learner phase and fitness function evaluation. The step which compares the final solution at the end of the algorithm and identifying best value is only performed on CPU (master). This approach overcomes the limitation of the master-slave model, i.e. time required to perform frequent communication between master and slave. Figure 1 presents the flowchart of the proposed algorithm.

The algorithms execution begins with memory allocation on CPU and GPGPU. The required input for the algorithm is transferred on GPGPU. The initial population is generated on GPGPU. As the TLBO algorithm consists of two phases, the teacher phase executes first and then learner phase. Both these phases are executed on the GPGPU. The GPGPU computes the mean of each design variable in teacher phase and transfers to CPU to identify the best individual (teacher). The odd-even sorting scheme implemented on CPU to find the best teacher. This step is developed using OpenMP – a multi-core approach to improve the utilization of all CPU cores in the multi-core CPU environment. The identified best value is transferred to the GPGPU to update the solution. The learner phase updates the solution based on identified best teacher on GPGPU. The final solution is better than earlier and/or if the termination criteria meet then algorithm exits. The initial population generation, teacher phase and learner phase implemented using CUDA framework, developed by Nvidia for implementation on Nvidia's GPU.

In minimization type of constrained single objective benchmark optimization functions, the teacher with null or minimum constraint violation is preferred and in case of a tie, the solution with minimum objective value is selected. In the case of maximization type of problem, the solution with maximum objective value is selected. The constraint violation checking rule is the same in both types of problems

The pseudo-code of the master-slave TLBO algorithm is presented below.

Algorithm 1: Pseudo-code of Master-Slave TLBO Algorithm

Input: No. of subjects (design variables), No. of learners (population size), max_iteration number

Output: Selected benchmark function's optimal value

1. Allocate memory on CPU and GPGPU for input data and results

2. Copy input from CPU to GPGPU.

3. Initialize CUDA blocks ← Number of design variable

4. Initialize CUDA Threads ← population size

5. Launch "kernel 1" to generate initial population

6. While i ≤ max_iteration number **do**

//Teacher phase

7. To compute mean for each design variable: Launch "kernel 2"

8. __global__ void cal(double *a,double *mean)

9. {

10. for(int j=0;j<N;j++)

{

11. mean[blockIdx.x]=mean[blockIdx.x]+a[blockIdx.x*N+j];

12. }

13. __syncthreads();

14. }

15. To compute the difference between existing and new mean, update value of each teacher and update the teacher: Launch "kernel 3"

16. diff[j]=(r/11)*(s[bestpos*D+j]-tf*m[j]);

17. $diff - mean = ri(M_{new} - TF.M_i)$

18. Update the Teacher's value

// End of Teacher Phase

// Learner Phase

19. To improve the value of learner based on best teacher: Launch "kernel 4"

20. X''j,P,i= X'j,P,i+ ri(X'j,P,i- X'j,Q,i), If X'total-P,i<X'total-Q,i

21. X''j,P,i= X'j,P,i+ ri(X'j,Q,i - X'j,P,i), If X'total-Q,i < X'total-P,i

22. Update the Learner's value

//End Learner Phase

23. End While

24. Copy the obtained values of the objective function from GPGPU to CPU

25. Perform odd-even sort to find best value

The proposed master-slave TLBO algorithm generates the initial solution on GPGPU to reduce the communication time required for data transfer from CPU to GPGPU. It is implemented in parallel on GPGPU using kernel-1. Objective function computation and evaluation is a compute-intensive task in any optimization problem. If the number of design variables and the population size is large then it increases the computational time for objective function evaluation. The kernel-2 and kernel-3 execute the teacher phase. The mean value of each design variable, the difference between the result of class and individual learners mean is computed and objective value is updated. The teacher phase ends and the obtained results are used for the next phase. The learner phase begins at the end of teacher phase. The learner phase updates the value of each learner using the updated function value from teacher phase. The kernel-4 is learner phase. It selects any two learners and improves the result of each learner according to best value among them. It results in improvement of the class mean. The obtained values of the objective function are copied to the host (CPU). The odd-even sorting is implemented on CPU using OpenMP to obtain the best value among all the individuals. The algorithm exits when it reaches the predefined termination criteria.

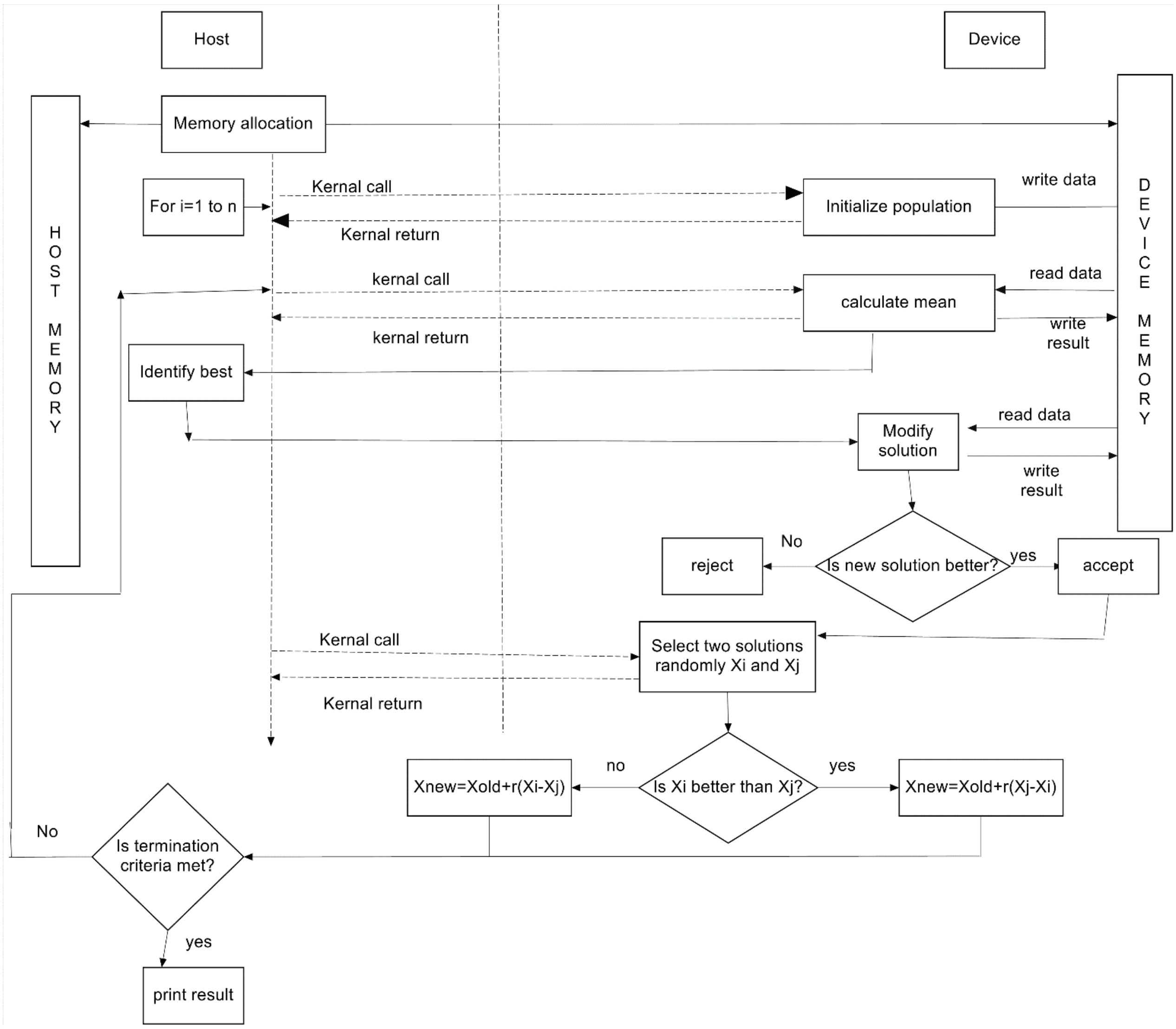

Figure 1. The flow of Proposed GPGPU based Master-Slave TLBO Algorithm

3.2.3 Merits and demerits of master-slave TLBO algorithm

The proposed approach has advantages and limitations. These are briefly explained here. The proposed approach enhanced the system's computational resources utilization. The proposed approach preserves the characteristics of basic TLBO algorithm. The common controlling parameters are only required to tune. The execution time required is reduced very largely. The limitation of the proposed approach is, it requires more function evaluations. The data transfer time required to and from CPU to GPGPU affects the execution time. It can be overcome by using advanced CUDA libraries. The merits and limitation of the proposed master-slave TLBO algorithm verified after performing experimentation.

4. Constrained benchmark functions

The various properties of optimization applications vary from one problem to another, so the testing of strength and weakness of the new or modified optimization algorithm becomes difficult. The new or modified optimization algorithms performance vary from one problem to another. The benchmark functions enable to test the hypothetical performance of optimization algorithm in practically. The benchmark functions are used to test any newly developed or modified optimization algorithm in an unbiased manner. The benchmark functions help researchers to understand the behaviour of the optimization algorithm. These benchmark problems used by researchers to test evolutionary algorithms. The IEEE Congress on Evolutionary Computation (CEC) competitions have announced the constrained benchmark optimization problems in 2006. One of the objectives of the proposed work is to test the performance of the proposed master-slave TLBO algorithm for the single-objective constrained optimization problem. Hence, the proposed work uses the CEC2006 single-objective constrained benchmark functions. These functions are widely used in literature to test the newly developed single objective optimization algorithms. The selected benchmark functions have linear, nonlinear, cubic, and polynomial objective functions. The number of design variables varies from 2 to 20, depends on the type of benchmark function. The selected benchmark functions provide the common platform to evaluate the performance of optimization algorithms. The CEC2006 benchmark problems have characteristics similar to single-objective real-time optimization problems. The "Appendix A" presents the definition of selected CEC2006 constrained real-parameter optimization benchmark functions [52].

The constrained benchmark functions with the number of design variables, function type, number of constraints with the nature of constraints, and known best solution are presented in Table 1. The known best solution indicates the best results obtained after solving the benchmark functions recently by any optimization algorithm. It will help new researchers to measure the quality of the solution by his/her proposed approach.

Table 1. Benchmark functions with its properties

Function	No. of Design Variables	Function Type	Number and Type of constraints	Known Best Solution
G1	13	Quadratic	9 Linear Inequality	-1.50E+01
G2	20	Nonlinear	2 Nonlinear Inequality	-8.04E-01
G3	10	Polynomial	1 Nonlinear Equality	-1.00E+00
G4	5	Quadratic	6 Nonlinear Inequality	-3.07E+04
G5	4	Cubic	2 Linear Inequality and 3 Nonlinear Equality	5.13E+03
G6	2	Cubic	2 Nonlinear Inequality	-6.96E+03
G7	10	Quadratic	3 Linear Inequality 5 Nonlinear Inequality	2.43E+01
G8	2	Nonlinear	2 Nonlinear Inequality	-9.58E-02
G9	7	Polynomial	4 Nonlinear Inequality	6.81E+02
G10	8	Linear	3 Linear Inequality and 3 Nonlinear Inequality	-7.05E+03
G11	2	Quadratic	1 Nonlinear Equality	-7.50E-01
G12	3	Quadratic	1 Nonlinear Inequality	-1.00E+00
G13	5	Nonlinear	3 Nonlinear Equality	-5.39E-02

5. Results and discussion

The master-slave TLBO algorithm developed to test the improvement in the optimal solution, speed-up and device utilization. The obtained solution is evaluated using statistical tests such as best, mean, and standard deviation (SD). The experimental setup, parameter settings, results obtained, and its analysis is presented in this section.

To perform experimentation the GeForce GTX 680 Nvidia's GPGPU used, which has 8 streaming multiprocessors with 1536 CUDA cores. The global memory is 2GB and its speed is 6.0 Gbps. The device has 3.0 compute capability. The Intel's i7 processor with 2.40 GHz processing speed and 8 GB RAM is used. It has 4 logical processors. The proposed approach is implemented on Ubuntu 16.04 using CUDA Toolkit 7.0 with the Thrust library and OpenMP 5.0. The G1 to G13 constrained benchmark functions from CEC2006 dataset used to evaluate the performance of proposed Master-slave TLBO algorithm.

The basic TLBO algorithm is a parameter-less algorithm, in the proposed parallel version, the same behaviour is preserved. The common controlling parameter only initialized at the beginning. It is presented in Table 2. The execution parameters need to select cautiously to obtain good performance from the proposed algorithm. The parameter settings for the proposed master-slave TLBO algorithm are determined after performing extensive experimentation.

Table 2. Common Controlling Parameter Settings

Parameters	Values	Functions
Population size	512	G1 to G6 and G9
Maximum iteration number	500	
CUDA Blocks	Number of design variables of selected benchmark function	
CUDA Threads	512	
Population size	512	G7, G8, and G10 to G13
Maximum iteration number	1000	
CUDA Blocks	Number of design variables of selected benchmark function	
CUDA Threads	512	

Table 3 presents the results obtained by the master-slave TLBO algorithm for CEC2006 single-objective constrained benchmark functions. The results are presented in the form of obtained best value, obtained worst value, standard deviation (SD), and mean. The execution time is presented in a millisecond. The proposed master-slave TLBO algorithm is executed for 30 times for each function and best, mean and standard deviation values obtained. The mean and standard deviation tests performed to measure the quality of the obtained results. The standard deviation is used to interpret the spread of solution from the mean value. The low value indicates that the best value obtained in each run is close to mean value while high value indicates results obtained are away from the mean. The mean value obtained is used to interpret where the obtained best values clustered.

Table 3. Results obtained using the master-slave TLBO algorithm for 13 benchmark functions

Function	Known Best	Obtained Best	Obtained Worst	SD	Mean	Time (ms)	No. of Iterations
G1	-1.50E+01	**-1.50E+01**	-1.47E+01	9.63E-02	-1.50E+01	35.8224	500
G2	-8.04E-01	-7.99E-01	-7.80E-01	5.33E-03	-7.95E-01	49.0347	500
G3	-1.00E+00	**-1.00E+00**	-1.48E-01	3.01E-01	-9.98E-01	32.5706	500
G4	-3.07E+04	**-3.07E+04**	-3.02E+04	1.41E+02	-3.06E+04	24.8815	500
G5	5.13E+03	**5.13E+03**	5.13E+03	4.85E-01	5.13E+03	1.5902	500
G6	-6.96E+03	**-6.96E+03**	-6.96E+03	4.08E-01	-6.96E+03	2.4098	500
G7	2.43E+01	**2.43E+01**	2.65E+01	7.98E-01	2.45E+01	3.9212	1000
G8	-9.58E-02	**-9.58E-02**	9.61E-02	9.70E-05	-9.59E-02	2.8407	1000
G9	6.81E+02	**6.81E+02**	6.84E+02	1.16E+00	6.81E+02	1.8099	500

Function	Known Best	Obtained Best	Obtained Worst	SD	Mean	Time (ms)	No. of Iterations
G10	-7.05E+03	**-7.05E+03**	-7.04E+03	6.77E-01	-7.05E+03	3.3763	1000
G11	-7.50E-01	**-7.50E-01**	-7.35E-01	9.50E-05	-7.50E-01	8.0081	1000
G12	-1.00E+00	-9.98E-01	1.00E+00	8.40E-01	-9.96E-01	4.0277	1000
G13	-5.39E-02	-5.37E-02	8.15E-02	9.34E-03	-5.35E-02	3.3492	1000

The values in boldface indicate the best results

The master-slave TLBO obtains know best results for G1, G3, G4, G5, G6, G7, G8, G9, G10, and G11 benchmark functions. The results obtained for G2, G12 and G13 functions are close to the known best value. The population size used is 256. The parallel execution time recorded is from 1.59 milliseconds to 49.034 milliseconds. From the obtained results concluded that master-slave TLBO performs same as sequential TLBO. The master-slave TLBO algorithm is a promising approach to solve real-time single-objective constrained optimization problems from different domains.

Fig. 2 presents the number of iterations and function evaluation of master-slave TLBO algorithm. The number of iterations used is 500 (G1 to G6 and G9 benchmark functions) and 1000 (G7, G8, and G10 to G13 benchmark functions). The TLBO algorithm updates the solution in both the phases i.e. Teacher phase and Learner phase. The function evaluation is performed as discussed in [5]. The function evaluation varies based on population size and the number of iterations. The maximum function evaluation performed are 1024000 and minimum are 512000.

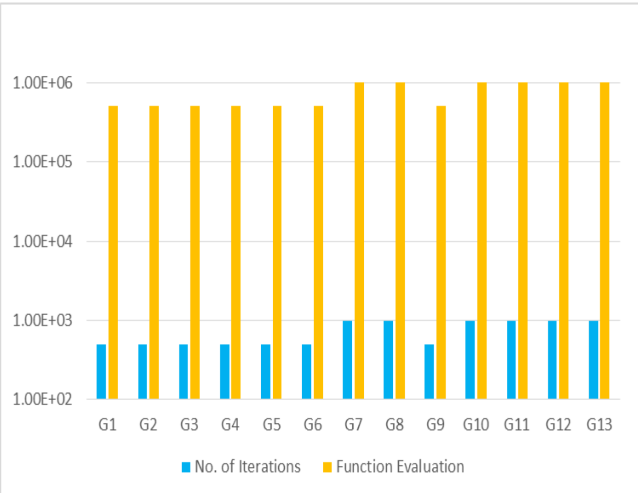

Figure 2. No. of iterations and function evaluation of master-slave TLBO algorithm

Table 4 shows the execution time taken by sequential TLBO algorithm and Master-slave TLBO algorithm. Using Amdahl's law, it is found that, the proposed approach is 11.8X to 30.14X faster than the sequential TLBO algorithm. The less speed-up is obtained for G12 function while G1 has highest speed-up. The G1 and G12 functions are quadratic but the G1 has 9 linear inequality constraint while G12 has 1 non-linear inequality constraint. As compare to G12, the G1 function in sequential execution taken more execution time. The G5 function has taken less execution time in both the versions.

Table 4: Comparison of execution time and speed-up for sequential TLBO and master-slave TLBO algorithm

Function	Time (in ms)		Speed-up
	Sequential TLBO Algorithm	Proposed Master-slave TLBO Algorithm	
G1	1079.6891	35.8224	30.14
G2	1204.7845	49.0347	24.57
G3	502.2398	32.5706	15.42
G4	292.1090	24.8815	11.74
G5	18.6223	1.5902	11.71
G6	30.6291	2.4098	12.71
G7	87.7570	3.9212	22.38
G8	53.2633	2.8407	18.75
G9	44.6155	1.8099	24.65
G10	79.9191	3.3763	23.67
G11	93.3756	8.0081	11.66
G12	47.5273	4.0277	11.8
G13	74.2874	3.3492	22.18

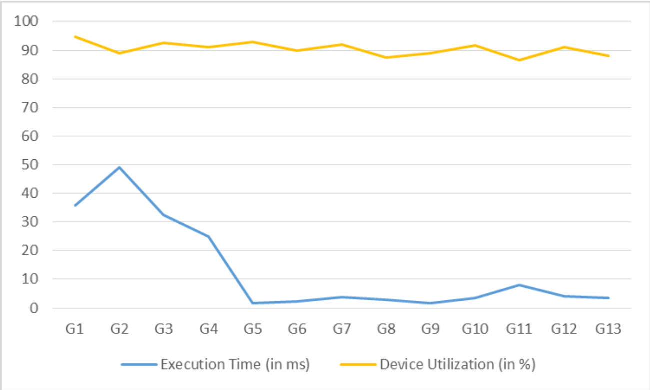

Figure 3. Execution time and Device utilization of Master-slave TLBO algorithm

The speed-up and device utilization are the other two important metrics used to measure the performance of a parallel implementation of the proposed approach. Fig. 3 presents the execution time and device (GPGPU) utilization of master-slave TLBO algorithm. The GPGPU utilization is measured using Nvidia's profiler. The CPU utilization is computed using Ubuntu's "system monitor" application. The speed-up is computed using Amdahl's law. The execution time and device utilization vary with characteristics of selected benchmark functions. The average GPGPU utilization is 90% and it varies from 86% to 94%. For G1 function the device utilization is maximum and it is 94.83%. The minimum device utilization observed for G11 test function, which is 86.57%. The device utilization affected by complexity and nature of the selected problem, the number of ideal threads in execution and divergence among thread execution. The execution time taken by the master-slave TLBO algorithm is from 1.590 milliseconds to 49.034 milliseconds. The G5 function requires minimum time while G2 requires maximum time to execute.

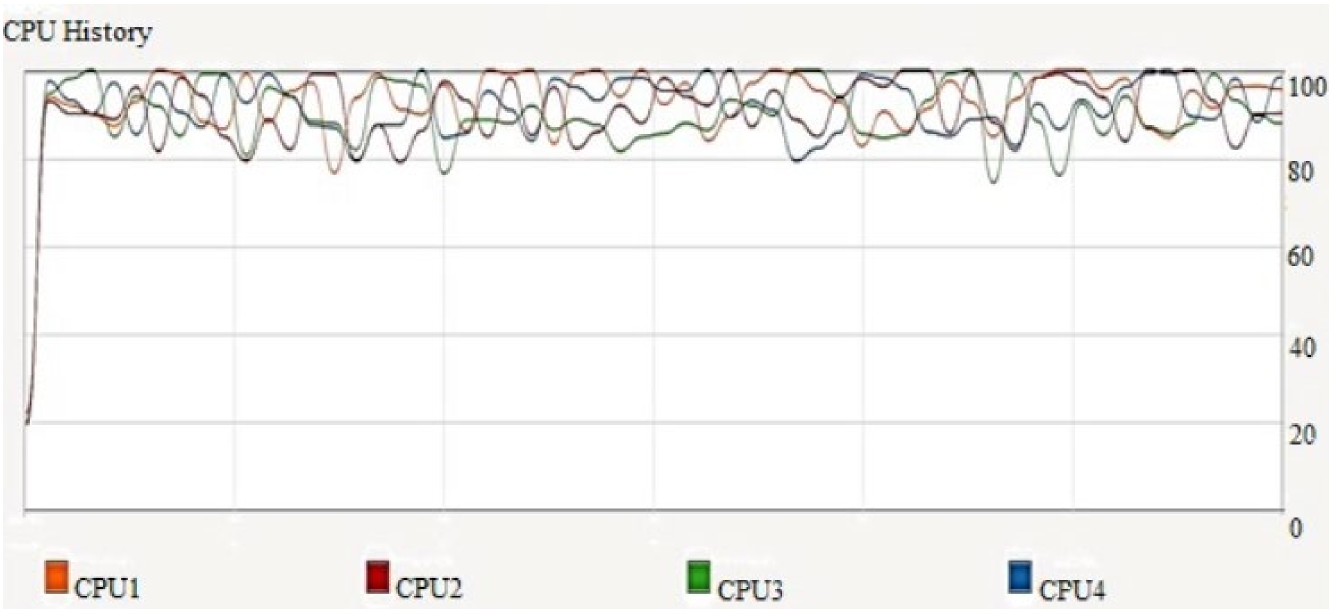

Figure 4. Logical processor (CPU) utilization by Master-Slave TLBO Algorithm

The proposed master-slave TLBO algorithm implemented using a hybrid programming model, viz. OpenMP-CUDA. The CUDA toolkit used to implement kernels (the functions to be executed on GPGPU) and OpenMP used to implement the code-block, to be executed on CPU. One of the objectives of the proposed work is to improve system utilization. The Teacher phase and Learner phase is executed on GPGPU, due to which the GPGPU utilized from 86% to 94% (presented in Fig. 3). The OpenMP facilitates the parallel implementation on a multi-core system, which results in the utilization of all logical processors instead of utilizing a single logical processor during execution. Fig. 4 presents the average percentage (%) utilization of logical processors available on the system, which was used for experimentation in 60 seconds. The significance of Fig. 4 is that it validates the utilization of all the cores of CPU by master-slave TLBO algorithm. The master-slave TLBO algorithm is implemented truly in a parallel fashion. The master phase of proposed approach utilizes all the logical processors (CPU1 to CPU4). Each logical processor's average utilization is more than 90% in 60 seconds.

6. Conclusion and Future Work

The design and implementation of a master-slave TLBO algorithm to solve CEC2006 single-objective constrained benchmark optimization is presented in this paper. The master-slave TLBO algorithm is developed to improve the computational resources of available systems. The best, worst, mean, and standard deviation, statistical tools used to measure the efficiency of the proposed algorithm. The proposed master-slave TLBO algorithm gives best results for G1, G3 to G9, G11 test functions. The results obtained for G2, G10, G12 and G13 are close to best-known results from the literature. The standard deviation confirms that the results obtained for each test function are close to mean results of that function. The Amdahl's law is used to compute speed-up obtained by master-slave TLBO algorithm. The proposed algorithm is 11.66X to 30.14X faster than sequential TLBO algorithm. The parallel execution time to exit the algorithm varies from 1.80 milliseconds to 49.03 milliseconds based on the characteristics of selected benchmark functions. One of the motives behind the parallel implementation is to improve the utilization of the system's computational resources. The average GPGPU device utilization is 90%. The maximum device utilized is 94.83% for G1 function and minimum device utilized is 86.57% for G11 function. The average CPU utilization also recorded; it is more than 90% for each logical processor. The function evaluation computed for proposed approach; it is from 5,12,000 to 10,24,000. The function evaluation is more because the TLBO algorithm has two phases (Teacher phase and Learner phase). From obtained results, it is concluded that the master-slave TLBO algorithm is one of the promising approaches to solve CEC2006 single-objective constrained optimization problems by improving system utilization.

In the future, the master-slave TLBO algorithm can be implemented to solve large-scale and complex constrained optimization problems. The advanced feature of recent CUDA toolkit and various CUDA libraries can be used to implement the proposed master-slave TLBO algorithm. The real-world single-objective constrained optimization problems can be solved using the proposed master-slave TLBO algorithm to test its efficiency and system utilization. As future work, other parallelization strategy found in the literature can be used to develop a GPGPU based parallel TLBO algorithm. The system utilization with other parallel strategies can be tested in future, for the other benchmark problems.

References

[1] Rao SS. Engineering optimization: theory and practice. John Wiley & Sons; 2019 Nov 12.

[2] Mane SU, Omane R, Pawar A. GPGPU based teaching learning based optimization and artificial bee colony algorithm for unconstrained optimization problems. In2015 IEEE International Advance Computing Conference (IACC). IEEE; 2015 Jun 12. pp. 1056-1061.

[3] Opara K, Arabas J. Benchmarking procedures for continuous optimization algorithms. Journal of Telecommunications and Information Technology. 2011; 73-80.

[4] Jamil M, Yang XS. A literature survey of benchmark functions for global optimisation problems. International Journal of Mathematical Modelling and Numerical Optimisation. 2013 Jan 1; 4(2):150-94.

[5] Rao, R. Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. International Journal of Industrial Engineering Computations. 2016; 7(1): 19-34.

[6] Adhikari M, Amgoth T. An intelligent water drops-based workflow scheduling for IaaS cloud. Applied Soft Computing. 2019 Apr 1; 77: 547-66.

[7] Abdessamia F, Zhang WZ, Tian YC. Energy-efficiency virtual machine placement based on binary gravitational search algorithm. Cluster Computing. 2019 Nov 28:1-12.

[8] Jangiti S, Sriram E, Jayaraman R, Ramprasad H, Sriram VS. Resource ratio based virtual machine placement in heterogeneous cloud data centres. Sādhanā. 2019 Dec 1; 44(12):236.

[9] Jangiti S, Subramaniyaswamy V, Shankar VS. Bulk-bin-packing based migration management of reserved virtual machine requests for green cloud computing. EAI Endorsed Transactions on Energy Web. 2019 Oct 1; 6(24).

[10] Tian W, He M, Guo W, Huang W, Shi X, Shang M, Toosi AN, Buyya R. On minimizing total energy consumption in the scheduling of virtual machine reservations. Journal of Network and Computer Applications. 2018 Jul 1; 113:64-74.

[11] Basiri ME, Kabiri A. HOMPer: A new hybrid system for opinion mining in the Persian language. Journal of Information Science. 2020 Feb; 46(1):101-17.

[12] Sharma M, Singh G, Singh R. Design of GA and Ontology based NLP Frameworks for Online Opinion Mining. Recent Patents on Engineering. 2019 Jun 1; 13(2):159-65.

[13] Sharma S, Singh G, Singh D. Role and Performance of Different Traditional Classification and Nature-Inspired Computing Techniques in Major Research Areas. EAI Endorsed Transactions on Scalable Information Systems. 2019 Jun 1; 6(21).

[14] Aljarah I, Mafarja M, Heidari AA, Faris H, Mirjalili S. Multi-verse optimizer: theory, literature review, and application in data clustering. In Nature-Inspired Optimizers 2020 (pp. 123-141). Springer, Cham.

[15] Sharma M, Kaur P. A Comprehensive Analysis of Nature-Inspired Meta-Heuristic Techniques for Feature Selection Problem. Archives of Computational Methods in Engineering. 2020 Feb 20:1-25.

[16] Mafarja M, Heidari AA, Faris H, Mirjalili S, Aljarah I. Dragonfly algorithm: theory, literature review, and application in feature selection. In Nature-Inspired Optimizers 2020 (pp. 47-67). Springer, Cham.

[17] Kumar SV, Rao PV, Singh MK. Optimal floor planning in VLSI using improved adaptive particle swarm optimization. Evolutionary Intelligence. 2019 Jul 9:1-14.

[18] Saremi S, Mirjalili S, Mirjalili S, Dong JS. Grasshopper optimization algorithm: theory, literature review, and application in hand posture estimation. In Nature-Inspired Optimizers 2020 (pp. 107-122). Springer, Cham.

[19] Mirjalili S, Dong JS. Introduction to nature-inspired algorithms. In Nature-Inspired Optimizers 2020 (pp. 1-5). Springer, Cham.

[20] John J, Rodrigues P. A survey of energy-aware cluster head selection techniques in wireless sensor network. Evolutionary Intelligence. 2019 Nov 27:1-13.

[21] Miranda K, Zapotecas-Martínez S, López-Jaimes A, García-Nájera A. A comparison of bio-inspired approaches for the cluster-head selection problem in WSN. In Advances in Nature-Inspired Computing and Applications 2019 (pp. 165-187). Springer, Cham.

[22] Annepu V, Rajesh A. Implementation of self adaptive mutation factor and cross-over probability based differential evolution algorithm for node localization in wireless sensor networks. Evolutionary Intelligence. 2019 Sep 1; 12(3):469-78.

[23] Mane S, Rao MN. Many-objective optimization: Problems and evolutionary algorithms–a short review. International Journal of Applied Engineering Research. 2017; 12(20):9774-93.

[24] Rao RV, Savsani VJ, Vakharia DP. Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. Computer-Aided Design. 2011 Mar 1; 43(3):303-15.

[25] Kumar MS, Gayathri GV. A short survey on teaching learning based optimization. In Emerging ICT for Bridging the Future-Proceedings of the 49th Annual Convention of the Computer Society of India CSI Volume 2 2015 (pp. 173-182). Springer, Cham.

[26] Rao R. Review of applications of TLBO algorithm and a tutorial for beginners to solve the unconstrained and constrained optimization problems. Decision science letters. 2016; 5(1):1-30.

[27] Zou F, Wang L, Hei X, Chen D, Wang B. Multi-objective optimization using teaching-learning-based optimization algorithm. Engineering Applications of Artificial Intelligence. 2013 Apr 1; 26(4):1291-300.

[28] Jesshope CR. Computational physics and the need for parallelism. Computer Physics Communications. 1986 Aug 1; 41(2-3):363-75.

[29] Alba E, Tomassini M. Parallelism and evolutionary algorithms. IEEE transactions on evolutionary computation. 2002 Dec 10; 6(5):443-62.

[30] Maitre O, Krüger F, Querry S, Lachiche N, Collet P. EASEA: specification and execution of evolutionary algorithms on GPGPU. Soft Computing. 2012 Feb 1; 16(2):261-79.

[31] Gong YJ, Chen WN, Zhan ZH, Zhang J, Li Y, Zhang Q, Li JJ. Distributed evolutionary algorithms and their models: A survey of the state-of-the-art. Applied Soft Computing. 2015 Sep 1; 34:286-300.

[32] Collet P, Krüger F, Maitre O. Automatic parallelization of EC on GPGPUs and clusters of GPGPU machines with EASEA and EASEA-CLOUD. In Massively Parallel Evolutionary Computation on GPGPUs 2013 (pp. 35-59). Springer, Berlin, Heidelberg.

[33] Hofmann J, Limmer S, Fey D. Performance investigations of genetic algorithms on graphics cards. Swarm and Evolutionary Computation. 2013 Oct 1; 12:33-47.

[34] Pospichal P, Jaros J, Schwarz J. Parallel genetic algorithm on the CUDA architecture. In European conference on the applications of evolutionary computation 2010 Apr 7 (pp. 442-451). Springer, Berlin, Heidelberg.

[35] Zheng L, Lu Y, Guo M, Guo S, Xu CZ. Architecture-based design and optimization of genetic algorithms

on multi-and many-core systems. Future Generation Computer Systems. 2014 Sep 1; 38:75-91.

[36] Mussi L, Daolio F, Cagnoni S. Evaluation of parallel particle swarm optimization algorithms within the CUDA™ architecture. Information Sciences. 2011 Oct 15; 181(20):4642-57.

[37] Kumar J, Singh L, Paul S. GPU based parallel cooperative particle swarm optimization using C-CUDA: a case study. In2013 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE) 2013 Jul 7 (pp. 1-8). IEEE.

[38] Luo GH, Huang SK, Chang YS, Yuan SM. A parallel Bees Algorithm implementation on GPU. Journal of Systems Architecture. 2014 Mar 1; 60(3):271-9.

[39] Basturk A, Akay R. Performance analysis of the coarse-grained parallel model of the artificial bee colony algorithm. Information Sciences. 2013 Dec 20; 253:34-55.

[40] Mane SU, Adamuthe AC, Pawar AS. GPGPU based Multi-hive ABC Algorithm for Constrained Global Optimization Problems. EAI Endorsed Transactions on Energy Web, 2020 Feb 14; 7(28).

[41] Cecilia JM, García JM, Nisbet A, Amos M, Ujaldón M. Enhancing data parallelism for ant colony optimization on GPUs. Journal of Parallel and Distributed Computing. 2013 Jan 1; 73(1):42-51.

[42] Delévacq A, Delisle P, Gravel M, Krajecki M. Parallel ant colony optimization on graphics processing units. Journal of Parallel and Distributed Computing. 2013 Jan 1; 73(1):52-61.

[43] Satapathy SC, Naik A, Parvathi K. A teaching learning based optimization based on orthogonal design for solving global optimization problems. SpringerPlus. 2013 Dec 1; 2(1):130.

[44] Satapathy SC, Naik A. Modified Teaching–Learning-Based Optimization algorithm for global numerical optimization—A comparative study. Swarm and Evolutionary Computation. 2014 Jun 1; 16:28-37.

[45] Rao RV, Patel V. An improved teaching-learning-based optimization algorithm for solving unconstrained optimization problems. Scientia Iranica. 2013 Jun 1; 20(3):710-20.

[46] Rao RV, Savsani VJ, Vakharia DP. Teaching–learning-based optimization: an optimization method for continuous non-linear large scale problems. Information sciences. 2012 Jan 15; 183(1):1-5.

[47] Zou, F., Chen, D., & Xu, Q. (2019). A survey of teaching–learning-based optimization. Neurocomputing, 335, 366-383.

[48] Rico-Garcia H, Sanchez-Romero JL, Jimeno-Morenilla A, Migallon-Gomis H, Mora-Mora H, Rao RV. Comparison of High Performance Parallel Implementations of TLBO and Jaya Optimization Methods on Many core GPU. IEEE Access. 2019 Sep 12; 7:133822-31.

[49] García-Monzó A, Migallón H, Jimeno-Morenilla A, Sánchez-Romero JL, Rico H, Rao RV. Efficient Subpopulation Based Parallel TLBO Optimization Algorithms. Electronics. 2019 Jan; 8(1):19.

[50] Balande U, Shrimankar D, Funde N. MTLBO-MS: Modified teaching learning based optimization on multicore system. In2018 4th International Conference on Recent Advances in Information Technology (RAIT) 2018 Mar 15 (pp. 1-5). IEEE.

[51] Gong YJ, Chen WN, Zhan ZH, Zhang J, Li Y, Zhang Q, Li JJ. Distributed evolutionary algorithms and their models: A survey of the state-of-the-art. Applied Soft Computing. 2015 Sep 1; 34:286-300.

[52] Liang JJ, Runarsson TP, Mezura-Montes E, Clerc M, Suganthan PN, Coello CC, Deb K. Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization. Journal of Applied Mechanics. 2006 Sep 18; 41(8):8-31.

Appendix A. CEC2006 benchmark functions

Function G1:

$$\min f(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$$

subject to

$$g1(x) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \le 0$$
$$g2(x) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \le 0$$
$$g3(x) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \le 0$$
$$g4(x) = -8x_1 + x_{10} \le 0$$
$$g5(x) = -8x_2 + x_{11} \le 0$$
$$g6(x) = -8x_3 + x_{12} \le 0$$
$$g7(x) = -2x_4 - x_5 + x_{10} \le 0$$
$$g8(x) = -2x_6 - x_7 + x_{11} \le 0$$
$$g9(x) = -2x_8 - x_9 + x_{12} \le 0$$
$$x_i \ge 0, i = 1,....,13$$
$$x_i \le 1, i = 1,....,13$$

Number of design variables: 13

Search space: $0 \le x_i \le u_i,\ i = 1, 2, ...n$

$u = (1,1,..1,100,100,100,1)$

Best known at $x^* = (1,1,1,1,1,1,1,1,1,3,3,3,1)$,

$f(x^*) = -15$

Function G2:

$$\min f(x) = -\left| \frac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} x_i^2}} \right|$$

subject to

$$g1(x) = 0.75 - \prod_{i=1}^{n} x_i \le 0$$

$$g2(x) = \sum_{i=1}^{n} x_i - 7.5n \le 0$$

Number of design variables: 20

Search space: $0 \le x_i \le 10,\ i = 1,2,\ldots n$

Best known $f(x^*) = 0.8036$

Function G3:

$$\min f(x) = -\left(\sqrt{n}\right)^n \prod_{i=1}^{n} x_i$$

subject to

$$h_1(x) = \sum_{i=1}^{n} x_i^2 - 1 = 0$$

Number of design variables: 10

Search space: $0 \le x_i \le 1,\ i = 1,2,\ldots n$

Best known $f(x^*) = -1.0005$

Function G4:

$\min f(x) = 5.3578547x_3^2 + 0.8356891x_1x_5$

$+37.293239x_1 - 40792.141$

subject to

$g1(x) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4$

$-0.0022053x_3x_5 - 92 \le 0$

$g2(x) = -85.334407 - 0.0056858x_2x_5$

$-0.0006262x_1x_4 + 0.0022053x_3x_5 \le 0$

$g3(x) = 80.51249 + 0.0071317x_2x_5$

$+0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \le 0$

$g4(x) = -80.51249 - 0.0071317x_2x_5 -$

$0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \le 0$

$g5(x) = 9.300961 + 0.0047026x_3x_5$

$+0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \le 0$

$g6(x) = -9.300961 - 0.0047026x_3x_5$

$-0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \le 0 \le 0$

$78 \le x_1 \le 102,\ 33 \le x_2 \le 45$ and $27 \le x_i \le 45\ (i = 3,4,5)$

Number of design variables: 5

Best known $f(x^*) = -30665.539$

Function G5:

$\min f(x) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3$

subject to

$g1(x) = -x_4 + x_3 - 0.55 \le 0$

$g2(x) = -x_3 + x_4 - 0.55 \le 0$

$h3(x) = 1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0$

$h4(x) = 1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$

$h5(x) = 1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 = 0$

$0 \le x_1 \le 1200,\ 0 \le x_2 \le 1200,\ -0.55 \le x_3 \le 0.55$ and

$-0.55 \le x_4 \le 0.55$

Number of design variables: 5

Best known $f(x^*) = 5126.4967$

Function G6:

$\min f(x) = (x_1 - 10)^3 + (x_2 - 20)^3$

subject to

$g1(x) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \le 0$

$g2(x) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \le 0$

$13 \le x_1 \le 100$ and $0 \le x_2 \le 100$

Number of design variables: 2

Best known $f(x^*) = -6961.813875$

Function G7:

$\min f(x) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2$

$+4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2$

$+2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$

subject to

$g1(x) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \le 0$

$g2(x) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \le 0$

$g3(x) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \le 0$

$g4(x) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \le 0$

$g5(x) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \le 0$

$g6(x) = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \le 0$

$g7(x) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \le 0$

$g8(x) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \le 0$

$-10 \le x_i \le 10\ \ (i = 1,\ldots,10)$

Number of design variables: 10

Best known $f(x^*) = 24.306$

Function G8:

$$\min f(x) = -\frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1+x_1)}$$

subject to

$g1(x) = x_1^2 - x_2 + 1 \le 0$

$g2(x) = 1 - x_1 + (x_2 - 4)^2 \le 0$

$0 \le x_1 \le 10$ and $0 \le x_2 \le 10$

Number of design variables: 2

Best known $f(x^*) = -0.095825$

Function G9:

$\min f(x) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2$
$+10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$

subject to

$g1(x) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \le 0$

$g2(x) = -282 + 7x_1 + 3x_2 + 10x_3^2 - x_5 \le 0$

$g3(x) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 - 12 \le 0$

$g4(x) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \le 0$

$-10 \le x_i \le 10 \quad (i = 1,\ldots,7)$

Number of design variables: 7

Best known $f(x^*) = 680.630$

Function G10:

$\min f(x) = x_1 + x_2 + x_3$

subject to

$g1(x) = -1 + 0.0025(x_4 + x_6) \le 0$

$g2(x) = -1 + 0.0025(x_5 + x_7 - x_4) \le 0$

$g3(x) = -1 + 0.01(x_8 + x_5) \le 0$

$g4(x) = -x_1x_6 + 833.33252x_4 + 100x_1 - 83333.333 \le 0$

$g5(x) = -x_2x_7 + 1250x_5 + x_2x_4 + 1250x_4 \le 0$

$g6(x) = -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \le 0$

$100 \le x_i \le 10000,\ 1000 \le x_i \le 10000\ (i = 2,3)$ and
$10 \le x_i \le 1000\ (i = 4,\ldots.,8)$

Number of design variables: 8

Best known $f(x^*) = 7049.2480$

Function G11:

$\min f(x) = x_1^2 + (x_2 - 1)^2$

subject to

$h1(x) = x_2 - x_1^2 = 0$

$-1 \le x_1 \le 1$ and $-1 \le x_2 \le 1$

Number of design variables: 2

Best known $f(x^*) = 0.7499$

Function G12:

$\min f(x) = -(100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2)/100$

subject to

$g1(x) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \le 0$

$0 \le x_i \le 10\ (i = 1,2,3)$ and $p,q,r = 1,2,\ldots,9.$

Number of design variables: 3

Best known $f(x^*) = -1$

Function G13:

$\min f(x) = e^{x_1x_2x_3x_4x_5}$

subject to

$h1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0$

$h2(x) = x_2x_3 - 5x_4x_5 = 0$

$h3(x) = x_1^3 + x_2^3 + 1 = 0$

$-2.3 \le x_i \le 2.3\ (i = 1,2)$ and $-3.2 \le x_i \le 3.2\ (i = 3,4,5)$

Number of design variables: 5

Best known $f(x^*) = 0.05394$