TLBO ALGORITHM REPORT

(I) PROBLEM STATEMENT

The objective is to analyze TLBO optimization, which involves finding the optimal global solution to a given objective function which can be linear or nonlinear, continuous or discrete, and single-objective or multi-objective.

Also, it suggests finding optimal solutions and their accurate function value for minimizing or maximizing problems where objective function can represent real-world problems in multiple domains, including engineering, finance, logistics, and healthcare.

(II) APPROACH & LOGIC

TLBO (Teaching-Learning-Based Optimization) is a nature-inspired optimization algorithm based on the teaching and learning process. In TLBO, the optimization process mimics the behaviour of a classroom where there are teachers and students. The teacher represents the best solution found so far, while the students represent the potential solutions to the problem.

The TLBO algorithm follows the below steps to optimize a problem:

1. Initialization: The algorithm initializes the population of students randomly. Each student represents a potential solution to the problem.
2. Evaluation: Each student is evaluated based on a fitness function that measures the quality of the solution.
3. Teacher phase: The best student in the population is selected as the teacher. The teacher then improves the worst students by adjusting their position using the following equation:

$$X_{new} = X_{old} + \alpha * [X_{teacher} - Tf(X_{mean})]$$

[Where, X_{new} = new solution, X_{old} = current solution, Tf = teaching factor either 1 or 2, X_{mean} = mean of solution, $X_{teacher}$ = teacher, α = random number between 0 and 1
Tf = round (1+ rand)]

4. Learner phase: In this phase, each student (except the teacher) improves their position based on the following equation:

$$X_{new} = X_{old} + \alpha * (X_i - X_j)$$

[Where, X_{new} = new solution, X_{old} = current solution, X_i = best solution, X_j = second best solution, α = student factor, a random number between 0 and 1]

5. Termination: The algorithm terminates when the maximum number of iterations is reached or when a satisfactory solution is found.

The logic behind the TLBO algorithm is that the best solution found so far acts as a teacher and guides the other students towards a better solution. The teacher improves the worst students while the other students learn from the best solutions. By doing so, the algorithm balances the exploration and exploitation of the search space, which helps find the global optimum solution.

(III) INPUTS

The inputs required for the TLBO optimization algorithm are:

1. Population (pop): A set of candidate solutions for the optimization problem.
2. Objective function (f_obj): A function that evaluates the quality of a candidate solution. The goal is to minimize or maximize this function.
3. Constraint function (f_cons): A function that evaluates the constraints of a candidate solution. It returns an array of constraint values for each constraint in the problem, either positive or negative, depending on the constraint.
4. Maximum number of iterations (max_iter): The number of iterations the algorithm will run.
5. Number of dimensions (n_dim): The number of dimensions in the problem space.
6. Number of individuals in the population (n_pop): The size of the population.

The algorithm uses these inputs to optimize and finds the best candidate solution that satisfies the problem constraints.

(IV) DATA STRUCTURE SPECIFICATIONS

TLBO does not require specific data structures for its implementation, but it requires basic mathematical operations to represent and manipulate the candidate solutions and their corresponding fitness values. Some commonly used data structures in TLBO include:

1. Population: The population is a collection of candidate solutions, where each key is represented as a vector or an array. The population is stored in a data structure such as an array or a matrix.
2. Fitness Function: The fitness function evaluates the fitness of each candidate solution. It takes as input a candidate solution and outputs a fitness value. The fitness values are stored in a data structure such as an array or a list.
3. Teacher: The teacher guides the learning process in TLBO. The teacher's role is to improve the worst student's solution by combining it with the best student's solution. The teacher's solution can be stored in a data structure such as an array or a vector.

TLBO Algorithm :

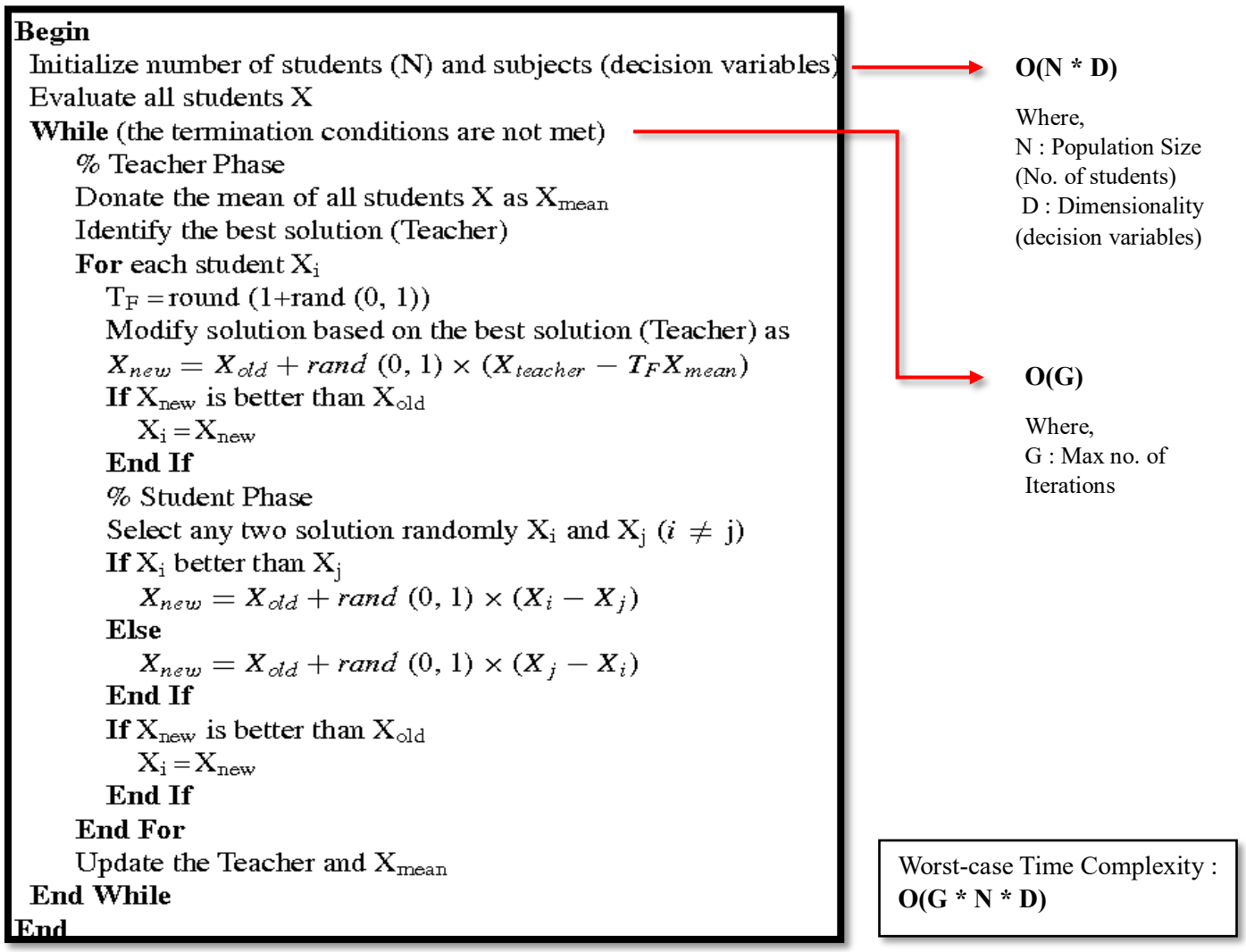

Fig. 1. Pseudocode for TLBO

[Source Code]

Code:

```
# TLBO algorithm
pop = np.random.uniform(0, 1, size=(n_pop, n_dim))
def TLBO(pop, f_obj, f_cons):
    best = pop[np.argmin([f_obj(individual) for individual in pop])]
    for i in range(max_iter):
        # Teacher phase
        teacher = best.copy()
        for j in range(n_dim):
            other_indices = [k for k in range(n_pop) if k != j]
            mean_others = np.mean(pop[other_indices], axis=0)
            if abs(mean_others[j] - teacher[j]) < np.finfo(float).eps:
                delta_j = np.random.uniform(0, 1)
            else:
                delta_j = np.random.uniform(0, 1) * abs(mean_others[j] - teacher[j])
            if mean_others[j] < teacher[j]:
                teacher[j] = teacher[j] - delta_j
            else:
                teacher[j] = teacher[j] + delta_j
        # Learner phase
        for j in range(n_pop):
            learner = pop[j].copy()
            a = np.random.randint(n_pop)
            while a == j:
                a = np.random.randint(n_pop)
            b = np.random.randint(n_pop)
            while b == j or b == a:
                b = np.random.randint(n_pop)
            for k in range(n_dim):
                if np.random.uniform(0, 1) < 0.5:
                    learner[k] = learner[k] + np.random.uniform(0, 1) * (teacher[k] -
np.random.uniform(0, 1) * abs(2 * pop[a][k] - pop[b][k]))
                else:
                    learner[k] = learner[k] - np.random.uniform(0, 1) * (teacher[k] -
np.random.uniform(0, 1) * abs(2 * pop[a][k] - pop[b][k]))
            learner = np.clip(learner, -1, 1)
            # Apply constraints
            c = f_cons(learner)
            if np.any(c > 0):
                for k in range(len(c)):
                    if c[k] > 0:
                        learner[k] = learner[k] - c[k]
                        learner[k] = np.clip(learner[k], -1, 1)
            # Update the population
            if f_obj(learner) < f_obj(pop[j]): # For maximization reverse the sign
                pop[j] = learner.copy()
                if f_obj(learner) < f_obj(best): # For maximization reverse the sign
                    best = learner.copy()
    return best
```

O(n_pop * n_dim)

O(n_pop * n_dim * (f_obj + f_cons))

O(max_iter)

Worst-case Time Complexity :
O(max_iter * n_dim * n_pop * (f_obj + f_cons))

(V) OUTPUTS

The output of TLBO (Teaching Learning Based Optimization) is the optimal solution to the optimization problem defined by the input parameters. More specifically, the results of TLBO include the following:

1. Optimal solutions: The optimal solution is the set of decision variables that produce the best objective function value found by the algorithm. This solution satisfies any constraints defined for the problem.
2. Objective function value: The objective function value is the value of the objective function corresponding to the optimal solution. This represents the best possible value of the objective function that can be achieved using the TLBO algorithm.

Implementation Details:

- *Product Used – Google Colaboratory*
- *GPU – OFF*
- *Memory: 16 GB*
- *Graphics Card: NVIDIA GeForce GTX 1650 Ti*
- *CPU: AMD Ryzen 7 4800H*

Output of benchmark 1: *Quadratic minimization with 13 variables and 9 linear constraints*

```
No. of generation:  500
Population Size:  50
No. of design variables:  13

Best Student found:
[-1.         -1.         -1.         -1.         -1.         -1.
 -1.          0.52602192  1.          1.          1.          0.4126251
  1.        ]

Fitness of best Student: -42.93864702428971


CPU times: user 8.72 s, sys: 249 ms, total: 8.97 s
Wall time: 8.67 s
```

Output of benchmark 2: *Nonlinear maximization with 10 variables and 1 non-linear constraint*

```
No. of generation:  2000
Population Size:  50
No. of design variables:  10

Best Student found:
[-1.  1.  1.  1.  1.  1.  1.  1.  1.  1.]

Fitness of best Student:  -1.0


CPU times: user 24.8 s, sys: 1.38 s, total: 26.2 s
Wall time: 25.5 s
```

Output of benchmark 3: *Nonlinear minimization with 7 variables and 4 nonlinear constraints*

```
No. of generation:  2000
Population Size:  50
No. of design variables:  7

Best Student found:
[ 1.00000000e+00  1.00000000e+00  5.39091887e-06  1.00000000e+00
 -1.73838077e-03  1.00000000e+00  1.00000000e+00]

Fitness of best Student: 972.0


CPU times: user 18.6 s, sys: 1.19 s, total: 19.8 s
Wall time: 18.4 s
```

Output of benchmark 4: *Linear minimization with 8 variables, 3 non-linear and 3 linear constraints*

```
No. of generation:  2000
Population Size:  50
No. of design variables:  8

Best Student found:
[-1.         -1.         -1.          0.75656802  0.02563151 -1.
  1.          0.17692622]

Fitness of best Student: -3.0


CPU times: user 20.9 s, sys: 1.23 s, total: 22.1 s
Wall time: 20.8 s
```

Output of benchmark 5: *Quadratic maximization with three variables and 9^3 non-linear constraints*

```
No. of generation:  1000
Population Size:  50
No. of design variables:  3

Best Student found:
[1. 1. 1.]

Fitness of best Student: 0.52


CPU times: user 6.13 s, sys: 459 ms, total: 6.59 s
Wall time: 6.15 s
```

(VI) ANALYSIS

Time complexity:

The Worst-case Time complexity (Big-Oh) of the implemented TLBO algorithm is –

O(max_iter * n_dim * n_pop * (f_obj + f_cons))

Where,

- The algorithm iterates for a fixed number of iterations 'max_iter', hence it has a time complexity of **O(max_iter)**.
- Within each iteration, the teacher and learner phases iterate over all dimensions 'n_dim' of each individual in the population. Hence, each iteration has a time complexity of **O(n_dim)**.
- The algorithm also calls the objective function 'f_obj' and constraint function 'f_cons' for each individual in the population, giving a total of n_pop calls for each function within each iteration. Hence, each iteration has a time complexity of **O(n_pop)**.

Space complexity:

The Worst-case Space complexity (Big-Oh) of the implemented TLBO algorithm is –

O(n_pop * n_dim).

Where,

- The algorithm uses several variables, including the population 'pop', the best individual 'best', the teacher 'teacher', and the learner 'learner', each of which is an array of size 'n_dim'.
- The algorithm also uses the variables 'mean_others', 'delta_j', 'a', and 'b', each of which is a scalar.

Complexity Analysis (Worst-Case)	**Theoretical Complexity**	**Implemented code Complexity**
Time Complexity	O(max_iter * n_dim * n_pop)	O(max_iter * n_dim * n_pop * (f_obj + f_cons))
Space Complexity	O(n_pop * n_dim)	O(n_pop * n_dim)