EAI Endorsed Transactions
on Internet of Things
Review Article EAI.EU

Pseudo Gradient Teaching Learning Based Optimization for Proactive SLA Prediction in 6G Networks

Hilal H. Nuha[1,*], Pratama Azmi Atmajaya[1], Gian Maxmillian Firdaus[1]

[1]School of Computing, Telkom University, Jl. Telekomunikasi No. 1, Bandung 40257, Indonesia

Abstract

Proactive prediction of Service Level Agreement (SLA) satisfaction lets 6G radio access networks act before a link misses its throughput and latency targets. This paper proposes Pseudo Gradient Teaching Learning Based Optimization (PGTLBO), which adds to the teacher step the direction from the worse half to the better half of the ranked population, and uses it to train a multilayer perceptron. All trainers receive the same number of function evaluations, and a random direction control isolates the effect of the direction. On 6G SLA prediction for unseen clients, PGTLBO reaches 0.846 accuracy, tied with standard TLBO (0.844), and outperforms the genetic algorithm, particle swarm optimization, and Harris hawks optimization on all 20 splits. PGTLBO reaches a lower training loss than TLBO in 57 of 80 paired runs and its direction beats a random one, but test accuracy is limited by the data.

Keywords: teaching learning based optimization, pseudo gradient, metaheuristic, neural network training, 6G networks, service level agreement prediction, mobile edge computing

Received on 23 June 2026, accepted on 12 October 2026, published on 12 November 2026

Copyright © 2026 Hilal H. Nuha *et al.*, licensed to EAI. This is an open access article distributed under the terms of the CC BY-NC-SA 4.0, which permits copying, redistributing, remixing, transformation, and building upon the material in any medium so long as the original work is properly cited.

doi: 10.4108/eetiot.13675

1. Introduction

Sixth generation (6G) mobile networks are expected to support stringent and diverse Quality of Service (QoS) requirements, which places a premium on controllers that can anticipate degradation before it occurs. A concrete instance is proactive Service Level Agreement (SLA) prediction, where the controller estimates whether a link will satisfy its throughput and latency targets in the next observation window and acts in advance. Reliable models for such tasks improve resource management and reduce the risk of SLA violations.

The Multilayer Perceptron (MLP) is a compact and expressive model for the tabular telemetry produced by radio access networks. Although gradient training is the default choice, the training objective is a minimization over the network weights, which admits population based metaheuristics as a derivative free alternative. Genetic Algorithm (GA) [1], Particle Swarm Optimization (PSO) [2], and Harris Hawks Optimization (HHO) [3] are representative examples that have been applied to neural network training [4]. Metaheuristic trained or tuned predictors have also been reported for human powered energy generation [5], wireless sensor network lifetime [6], and vertical wind speed extrapolation [7], metaheuristics are routinely used for network control problems such as transmission power optimization [8], and new population based algorithms such as the Komodo Mlipir Algorithm [9] and the Battlefield Optimization Algorithm [10] continue to appear.

Teaching Learning Based Optimization (TLBO) is inspired by the influence of a teacher on a class of learners [11] and has no algorithm specific coefficients beyond the population size and the evaluation budget, which removes the calibration burden that affects GA

*Corresponding author. Email: hilalnuha@ieee.org

and PSO. However, the teacher phase summarizes the population using only the best learner and the arithmetic mean, and therefore ignores the directional information that is implicit in the ranking of the population by fitness. Methods that exploit this ranking are well established, from the rank weighted recombination of CMA-ES [12] and the rank based gradient estimates of natural evolution strategies [13] to ranking based mutation in differential evolution [14] and the outstanding and average groups of Group Teaching Optimization [15].

This paper studies Pseudo Gradient Teaching Learning Based Optimization (PGTLBO), a minimal modification that adds to the TLBO teacher update the difference between the centroid of the superior half and the centroid of the inferior half of the population. The direction points from poor towards good solutions, it averages over many learners, and it adds no continuously tuned coefficient. Because the addition changes both the direction and the length of the teacher step, a careful evaluation must establish whether the *direction* carries useful information, whether this information improves the *optimization* of the training objective, and whether any such improvement survives into the *test* performance of the trained model. It must also ensure that the methods receive the same computational budget. The experiments in this paper are designed around these three questions.

The gap addressed here is therefore not the lack of yet another metaphor based optimizer, of which many exist [9, 10], but the lack of controlled evidence on whether a cheap, coefficient free use of the population ranking helps TLBO train a predictive model. Studies that propose a new or modified metaheuristic for neural network training or for network applications typically report the final accuracy or cost achieved, sometimes under equal iteration counts, which favor methods that evaluate more than once per iteration [16], and seldom with a control that isolates the proposed operator. Compared with those studies, this paper adds three elements: an equal count of objective function evaluations for every trainer, including gradient training, a random direction control that has the same step length as the proposed operator, and a within run counterfactual analysis of each teacher step. These elements make it possible to state precisely what the pseudo gradient contributes and what it does not. The contributions are the following.

(i) We describe PGTLBO and relate its pseudo gradient to rank based search directions in evolution strategies, differential evolution, swarm methods, and TLBO variants (Sections 2 and 4).

(ii) We compare PGTLBO with TLBO, GA, PSO, HHO, and Adam under an identical number of objective function evaluations, with gradient steps charged for their backward pass, and we add a random direction control (TLBO-RD) and an off budget counterfactual diagnostic that isolate the effect of the pseudo gradient direction (Sections 4.3 and 5.3).

(iii) We formulate proactive 6G SLA prediction as a binary classification problem with a client grouped protocol and an explicit persistence baseline, relate it to the published results on the same data under their temporal protocol, and calibrate the attainable accuracy with linear and gradient boosting reference models. We additionally study a Mobile Edge Computing (MEC) latency regression task and a suite of shifted, rotated, and deceptive benchmark functions.

(iv) We report what the pseudo gradient does and does not provide. It is a better than random search direction and it consistently, if modestly, lowers the training loss reached by TLBO under a fixed budget; it does not produce a statistically significant improvement in test performance over TLBO on either learning task (Section 6). Table 8 summarizes the answer to each question together with its supporting evidence.

The remainder of this paper is organized as follows. Section 2 reviews related work and positions PGTLBO against it. Section 3 formulates MLP training as weight optimization and recalls standard TLBO. Section 4 presents PGTLBO and the controls used to isolate its effect. Section 5 describes the datasets, protocols, evaluation budget, and statistical analysis. Section 6 reports and discusses the results, and conclusions are given in Section 7.

2. Related Work

This section reviews the four bodies of work on which PGTLBO builds. The first subsection covers metaheuristic training of neural networks and metaheuristics in network applications, the second TLBO and its variants, the third other methods that use superior and inferior subpopulations, and the fourth previous SLA prediction on the 6G telemetry used here. The last subsection positions the present work against these studies.

2.1. Metaheuristic training of neural networks

Population based metaheuristics have been used to train feed forward networks when gradients are unreliable, when the objective is non differentiable, or when global exploration of the weight space is desired; Ojha et al. [4] review two decades of this literature. GA [1] evolves a population through

selection, crossover, and mutation; PSO [2] moves particles under personal and global attractors; HHO [3] alternates exploration and exploitation phases driven by a decaying escaping energy. These methods expose control parameters (crossover and mutation rates, inertia and acceleration coefficients) that usually require tuning. At the scale of modern networks, evolution strategies have been shown to be competitive with gradient methods in reinforcement learning [17, 18], but for supervised learning with an available gradient, first order methods such as Adam [19] remain the default. A recurrent methodological weakness in this literature is the comparison of optimizers under unequal budgets; Črepinšek et al. [16] showed that TLBO in particular performs two function evaluations per learner and iteration, so iteration matched comparisons with single evaluation methods are biased in its favor. We therefore compare all methods under an identical number of objective function evaluations and charge gradient steps by their forward and backward cost (Section 5.3).

In Internet of Things and network applications, metaheuristics are used both to train or tune predictors and to optimize network decisions directly. Nuha et al. [5] trained an MLP with the Clustered Packet Optimization metaheuristic to predict human powered energy generation, Fauzan et al. used an enhanced Grey Wolf Optimization (EGWO) to optimize a neural network for wireless sensor network lifetime [6] and to optimize transmission power [8], and Al-Shaikhi et al. [7] combined a long short term memory network with PSO for wind speed extrapolation. In EAI Endorsed Transactions, Hartono et al. [20] reviewed metaheuristics for energy optimization in wireless sensor networks and found ant colony optimization and PSO to be the most common choices, with a trend towards hybrids and limited validation beyond simulation; Sabat et al. [21] combined GA operators with water wave optimization for quality of service aware task scheduling with fewer SLA violations in cloud environments; Yang et al. [22] paired a convolutional and support vector regression load forecaster with a two layer PSO scheduler for power distribution; and Tran et al. [23] surveyed quantum machine learning for 6G network optimization. These works optimize routing, scheduling, power, or the parameters of a separate predictor, and they evaluate a new or hybrid search procedure as a whole. The present paper differs in that the metaheuristic directly trains the weights of the SLA predictor, and in that a single added operator is isolated with a matched control.

2.2. TLBO and its variants

TLBO [11] mimics the influence of a teacher on a class and has no algorithm specific coefficients other than population size and iteration budget. Its many variants [24, 25] include elitist TLBO [26], which replaces the worst learners by elite copies; the modified TLBO of Satapathy and Naik [27], which adds a tutorial term to the learner phase; and the dynamic group strategy of Zou et al. [28], in which the teacher phase uses the mean of a randomly regrouped subpopulation rather than of the whole class. Group Teaching Optimization (GTOA) [15] is the variant most closely related to the present work: it splits the class by fitness into an outstanding group and an average group and applies a different teaching rule to each group. GTOA, however, uses the split to assign *different update rules* to the two groups, whereas PGTLBO uses the two groups only to compute a *single shared direction* that is added to the unchanged TLBO teacher update. Jaya [29] likewise moves every solution towards the best and away from the worst solution, which is a two point (best and worst) version of the superior minus inferior direction used here.

2.3. Rank based use of superior and inferior subpopulations

Exploiting the fitness ranking of a population to construct a search direction is an established idea outside TLBO, and PGTLBO should be read in this context rather than as an entirely new principle.

- *Evolution strategies and estimation of distribution algorithms.* In CMA-ES [12] the new mean is a rank weighted recombination of the best μ of λ samples, so the mean shift is a weighted superior centroid minus the old mean; the active covariance update of Jastrebski and Arnold [30] additionally uses the inferior samples with negative weights. The cross entropy method [31] and estimation of distribution algorithms [32] refit the sampling distribution to an elite fraction. Natural evolution strategies [13, 17] estimate the gradient of a Gaussian smoothed objective as $\sum_k u_k(\mathbf{x}_k - \mathbf{m})$ with rank based utilities u_k. With a two level utility, $u_k = +1$ on the better half and $u_k = -1$ on the worse half, this estimator is proportional to the pseudo gradient $\mathbf{d}$ of Equation (4). PGTLBO can therefore be interpreted as injecting a rank based search gradient estimate into the TLBO teacher phase, with the important difference that the learners are not an isotropic Gaussian sample around $\mathbf{m}$ but the result of the greedy TLBO dynamics, so that $\mathbf{d}$ is a population scale (secant) direction rather than an unbiased estimate of the local gradient. Section 6.5 examines this distinction empirically.
- *Differential evolution.* The DE/current-to-$p$best mutation of JADE [33] and SHADE [34] pulls each

vector towards a randomly chosen member of the best p fraction, and ranking based DE [14] selects the base and terminal vectors of the difference vector with probabilities proportional to their rank, so that difference vectors tend to point from worse to better solutions. Like DE [35], PGTLBO uses a difference of population points, but it averages over two halves of the population instead of using a single pair, which lowers its variance.

- *Swarm methods with winners and losers.* The competitive swarm optimizer [36] lets the loser of each pairwise competition learn from the winner, and social learning PSO [37] lets each particle learn from randomly selected better particles. In Grey Wolf Optimization [38] the population is guided by its three best members. The gradient based optimizer of Ahmadianfar et al. [39] builds a Newton inspired search rule from the best and worst solutions.

Relative to these methods, PGTLBO makes a deliberately minimal change: it keeps the TLBO teacher and learner phases and the greedy acceptance intact, it uses the fixed half split (the only structural choice it introduces, and not a continuously tuned coefficient), and it adds one population statistic per iteration. The contribution claimed in this paper is therefore not a new search principle but a controlled study of whether this statistic helps TLBO, including a random direction control that separates the effect of the *direction* of $\mathbf{d}$ from the effect of its *magnitude*.

2.4. SLA prediction on the 6G telemetry data

Learning based methods have been used for SLA prediction, resource management, and slice aware analytics in 5G and 6G systems, and edge computing studies have considered latency prediction and offloading [40]. The 6G RAN telemetry collection adopted here was introduced by Maitlo et al. [41], who studied communication and fairness trade offs of federated learning across 200 non identically distributed clients. Their centralized baseline, an MLP with two hidden layers of 512 and 256 units trained with Adam, reaches an accuracy of 0.8505 (mean over five seeds) under a temporal split in which the first 80% of windows of every client are used for training and the last 20% for testing; federated variants reach 0.849 to 0.850. No persistence baseline is reported. In a companion study, Maitlo et al. [42] predict whether the *next* window meets the same SLA definition (downlink throughput of at least 40 Mbps and 95th percentile latency of at most 50 ms) with a compact federated MLP over eight clients and a per client stratified 80/20 split, and report an accuracy of 0.914 and a macro F1 of 0.850 after twenty rounds. To our knowledge these are the only published uses of this telemetry for SLA prediction. Two differences to the present study should be noted. First, both works use splits in which windows of the same client appear in training and testing, whereas our primary protocol holds out entire clients, which is harder and tests generalization to unseen sites. Second, because the SLA state is strongly autocorrelated (the next label equals the current label in 81.3% of all windows), accuracy alone overstates what a model learns, so we report the lift over a persistence baseline. For a direct comparison with Maitlo et al. [41] we additionally evaluate all MLP trainers under their temporal protocol (Section 6.2).

2.5. Positioning of this work

Table 1 places the present study among the closest related works. Three differences stand out. First, unlike TLBO variants that change the update rules of whole groups (GTOA) or replace the teacher by best and worst solutions (Jaya), PGTLBO keeps both TLBO phases unchanged and adds a single shared direction. Second, unlike recent metaphor based optimizers and the metaheuristic studies in network applications, including those published in EAI Endorsed Transactions, which optimize schedules, routes, power levels, or a predictor as a whole, the present study isolates one operator with a random direction control of equal step length. Third, unlike earlier work on the same 6G telemetry, which trained MLPs with Adam on splits that share clients between training and testing, the present study holds out entire clients and reports the lift over a persistence baseline.

3. Problem Formulation and Background

This section formulates MLP training as a bounded weight optimization problem shared by all trainers and then recalls the two phases of standard TLBO.

3.1. Learning as weight optimization

Let $\{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$ be a training set. We use a single hidden layer MLP with a rectified linear hidden activation. For classification the output passes through a logistic function and the objective is the binary cross entropy (BCE); for regression the output is linear and the objective is the mean squared error (MSE). In both cases the weights and biases are concatenated into a vector $\mathbf{w} \in \mathbb{R}^D$ and training solves

$$\min_{\mathbf{w}\in\Omega} f(\mathbf{w}), \qquad \Omega = \{\mathbf{w} : |w_j| \le B\}, \tag{1}$$

where f is the full batch training loss. Equation (1) is shared by every optimizer, so differences are attributable to the optimizer rather than to the model or the loss.

Table 1. Position of this work relative to closely related studies

Study	Method	Role of the method	Application
Rao et al. (2011)	TLBO	A general optimizer with a "teacher" phase and a "learner" phase	Mechanical design
Zhang and Jin (2020)	GTOA	A variant of TLBO with different rules for outstanding and average groups	Benchmark and engineering problems
Rao (2016)	Jaya	Moves solutions towards the best one and away from the worst one	Constrained benchmark problems
Suyanto et al. (2022)	Komodo Mlipir	A new general optimizer based on a nature metaphor	General optimization
Setiawan et al. (2025)	Battlefield Optimization	A new general optimizer based on a battlefield metaphor	General optimization
Nuha et al. (2026)	CPO	Optimizes an MLP (neural network) predictor	Human-powered energy generation
Fauzan et al. (2026)	EGWO	Optimizes a neural network	Wireless sensor network lifetime
Fauzan et al. (2025)	EGWO	Optimizes transmission power	Wireless sensor networks
Al-Shaikhi et al. (2022)	PSO with LSTM	Optimizes an LSTM predictor	Wind speed extrapolation
Hartono et al. (2026) (EAI)	Review	A survey of ACO, PSO, and hybrid methods for routing and clustering	Wireless sensor network energy
Sabat et al. (2026) (EAI)	GA with water wave	Task scheduling and load balancing	Cloud computing, QoS and SLA
Yang et al. (2026) (EAI)	Two-layer PSO	Charging and load scheduling, with a separate forecaster	Power distribution
Maitlo et al. (2026)	Adam (federated)	Trains the MLP weights; clients are shared between training and testing	6G SLA prediction (same data)
This work	PGTLBO	Trains MLP weights; isolates one ranked population operator and compares it with a matched control, under an equal evaluation budget	6G SLA prediction, MEC latency regression, benchmarks

3.2. Standard TLBO

TLBO maintains N_P learners $\{\mathbf{X}_i\}$. Let $\mathbf{X}_{\text{teacher}}$ be the best learner and $\mathbf{M}$ the population mean. The teacher phase proposes

$$\mathbf{X}_i^{\text{new}} = \mathbf{X}_i + \mathbf{r} \odot (\mathbf{X}_{\text{teacher}} - T_F\,\mathbf{M}), \tag{2}$$

with $\mathbf{r}$ a vector of independent $U(0,1)$ variables and the teaching factor $T_F = \text{round}(1+\rho) \in \{1,2\}$, $\rho \sim U(0,1)$. The learner phase moves each learner relative to a random peer $j \neq i$,

$$\mathbf{X}_i^{\text{new}} = \begin{cases} \mathbf{X}_i + \mathbf{r} \odot (\mathbf{X}_i - \mathbf{X}_j), & f(\mathbf{X}_i) < f(\mathbf{X}_j), \\ \mathbf{X}_i + \mathbf{r} \odot (\mathbf{X}_j - \mathbf{X}_i), & \text{otherwise.} \end{cases} \tag{3}$$

A candidate is clipped to Ω and accepted only if it improves the fitness. Each iteration therefore costs $2N_P$ function evaluations, which matters when TLBO is compared with methods that evaluate each individual once per iteration [16]. Figure 1 illustrates the two phases.

4. Proposed Method: PGTLBO

This section defines the pseudo gradient, presents the modified teacher phase of PGTLBO and its cost, and describes the controls that isolate the effect of the added direction.

4.1. Pseudo gradient

The teacher update represents the population by the best learner and the mean. The mean is a position rather than a direction, and the best learner is a single point. The ranked population, however, defines a direction from the inferior region towards the superior region. We partition the ranked population into a superior half S and an inferior half I and define

$$\mathbf{d} = \frac{1}{|S|}\sum_{k\in S}\mathbf{X}_k - \frac{1}{|I|}\sum_{k\in I}\mathbf{X}_k. \tag{4}$$

We call $\mathbf{d}$ a *pseudo* gradient because it is not an estimate of the local gradient $-\nabla f$ at any particular point. As discussed in Section 2, if the learners were an isotropic Gaussian sample around a center $\mathbf{m}$, then $\mathbf{d}$ would be proportional to the natural evolution strategy estimate of the descent direction of the Gaussian smoothed objective with a two level rank utility [13]. In TLBO the learners are shaped by greedy selection, so $\mathbf{d}$ is better described as a population scale secant direction that summarizes where the better half of the class sits relative to the worse half. Because it averages over N_P learners, its variance is lower than that of a single difference vector as used in DE [35].

4.2. Modified teacher phase and complexity

PGTLBO adds the pseudo gradient to the teacher update,

$$\mathbf{X}_i^{\text{new}} = \mathbf{X}_i + \mathbf{r}_1 \odot (\mathbf{X}_{\text{teacher}} - T_F\,\mathbf{M}) + \mathbf{r}_2 \odot \mathbf{d}, \tag{5}$$

with $\mathbf{r}_1, \mathbf{r}_2$ independent uniform random vectors. The learner phase and the greedy acceptance are unchanged. PGTLBO introduces no continuously tuned coefficient; the only structural choice is the half split of the ranked population, which we fix at $|S| = \lfloor N_P/2 \rfloor$

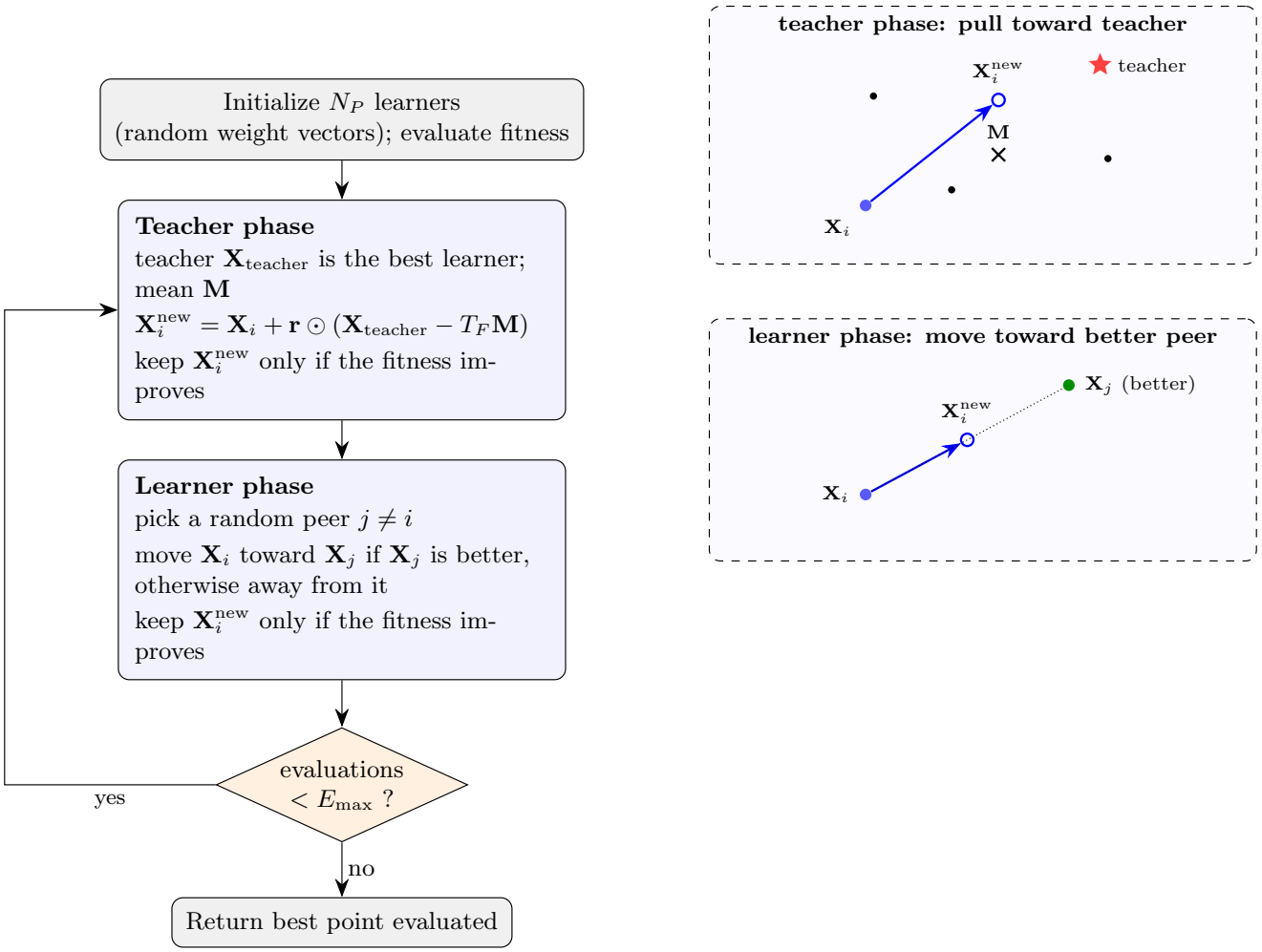


Figure 1. Standard TLBO
The flow on the left alternates a teacher phase and a learner phase under greedy acceptance until the evaluation budget is exhausted. The panels on the right illustrate the geometry of each phase: the teacher phase moves a learner $\mathbf{X}_i$ toward the teacher and away from a scaled mean $\mathbf{M}$, and the learner phase moves $\mathbf{X}_i$ toward a randomly chosen peer $\mathbf{X}_j$ when the peer is better (and away from it otherwise).

throughout and do not tune. Each iteration performs $2N_P$ evaluations, exactly as TLBO; the extra cost of ranking and two centroids is $\mathcal{O}(N_P \log N_P + N_P D)$ and is negligible relative to the evaluations. Algorithm 1 summarizes the method.

4.3. Controls used to isolate the effect of the pseudo gradient

Adding $\mathbf{r}_2 \odot \mathbf{d}$ changes two things at once: the teacher step acquires a new *direction* and a larger *magnitude*. To separate the two effects we evaluate two additional variants that differ from PGTLBO in a single component.

- **TLBO-RD** (random direction control) replaces $\mathbf{d}$ in Equation (5) by $\mathbf{u} = \|\mathbf{d}\|\,\mathbf{z}/\|\mathbf{z}\|$ with $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ drawn once per iteration. TLBO-RD has the same step magnitude as PGTLBO but no directional information, so any advantage of PGTLBO over TLBO-RD is attributable to the direction of $\mathbf{d}$.

- **PG-only** drops the teacher term and uses $\mathbf{X}_i + \mathbf{r}_2 \odot \mathbf{d}$, which tests whether $\mathbf{d}$ is useful on its own (benchmark suite only).

In addition, a *counterfactual* diagnostic is computed during every PGTLBO run: for each teacher phase candidate, the plain TLBO candidate and the random direction candidate are formed with the *same* $\mathbf{r}_1$, $\mathbf{r}_2$, and T_F, and both are evaluated off budget without influencing the search. This yields a paired, within run comparison of the three teacher steps from identical

Algorithm 1: Pseudo Gradient Teaching Learn-ing Based Optimization (PGTLBO)

Input: population size N_P, evaluation budget $E_{\max}$, bounds Ω, objective f
Initialize $\{\mathbf{X}_i\}_{i=1}^{N_P}$ uniformly in Ω and evaluate f;
while *fewer than* $E_{\max}$ *evaluations have been used* **do**
  Rank learners; set $\mathbf{X}_{\text{teacher}}$, $\mathbf{M}$, the halves S and I;
  Compute $\mathbf{d}$ by Equation (4);
  foreach *learner i (teacher phase)* **do**
    $T_F \leftarrow \text{round}(1+\rho)$; candidate by Equation (5); clip to Ω; greedy accept;
  foreach *learner i (learner phase)* **do**
    candidate by Equation (3); clip to Ω; greedy accept;
return best point evaluated;

states, which is far more sensitive than comparing final test metrics across runs.

5. Experimental Setup

This section describes the two datasets, the evaluation protocols, the common evaluation budget and the cost charged for a gradient step, the optimizer settings and benchmark functions, and the statistical analysis.

5.1. Datasets

Primary dataset: 6G RAN SLA prediction. The primary evaluation uses the public windowed 6G radio access network (RAN) telemetry collection introduced by Maitlo et al. [41], which we adopt without modification. Figure 2 depicts its structure.

The recording is organized as a panel: two hundred clients, each representing an edge device, a base station, or a comparable entity that observes its own local traffic, are tracked over one hundred and fifty consecutive observation windows of ninety seconds, which yields 30,000 client by window samples. Every window aggregates the radio and load conditions seen by that client into attributes that fall into four groups: radio and load indicators (physical resource block utilization, channel quality, signal to noise ratio, active users, handovers); link reliability indicators (block error rates, retransmissions); performance indicators (uplink and downlink throughput, median and tail latency); and the slice traffic composition over the eMBB, URLLC, mMTC, XR, and best effort slices together with entropy and imbalance summaries. Each client is assigned one operating regime (light, medium, heavy, impaired), so the per client distributions are not identically distributed. The supervised target `sla_ok_next` is proactive: a window is labeled satisfied when the downlink throughput in the *next* window is at least 40 Mbps and its 95th percentile latency is at most 50 ms. After removing the client identifier, the target, and the three time index fields, and one hot encoding the regime, the model input has 26 features, including the current SLA flag. The positive class rate is 56.9%, and the next label equals the current label in 81.3% of all windows.

Secondary dataset: MEC latency regression. To assess robustness across task types we use a public task offloading dataset [43]. Each record describes a data transfer between a customer network and a service provider network: cell identifier, location area code (LAC), radio access type (RAT), uplink and downlink traffic volumes, duration, throughput, and latency. We predict $\log(1 + \text{latency})$ from the pre transfer attributes. Throughput is an outcome of the transfer and the cell identifier is a high cardinality index, so both are excluded. Traffic volumes and duration are log transformed and the RAT is one hot encoded, giving 5 input features. A fixed random subsample of 10,000 records is used in all experiments to keep the derivative free runs tractable.

5.2. Protocols

SLA task. We use a client grouped repeated hold out: in each of 20 repetitions, 30% of the clients (60 clients, about 9,000 windows) are held out for testing and the model is trained on the remaining 140 clients, so no client appears in both sets. This protocol is stricter than the temporal and stratified splits of earlier work on these data (Section 2) because it measures generalization to unseen sites. Features are standardized with training statistics. We report accuracy, F1, ROC-AUC, and the lift of accuracy over a persistence baseline that predicts the next flag to be equal to the current one. Because the absolute accuracy of every method moves with the persistence accuracy of the held out clients, all comparisons are paired by split. The MLP has $H = 24$ hidden units ($D = 673$ weights and biases) and weights bounded by $|w_j| \le 5$. To test whether the conclusions depend on the dimension of the search space we repeat the TLBO family runs with $H = 48$ hidden units ($D = 1345$). For comparison with Maitlo et al. [41] we also evaluate the temporal protocol of that study (Section 6.2).

MEC task. We use 30 random 70/30 splits, an MLP with $H = 10$ hidden units ($D = 71$), weights bounded by $|w_j| \le 2$, and a standardized target. We report R^2, RMSE, and MAE in log latency space.

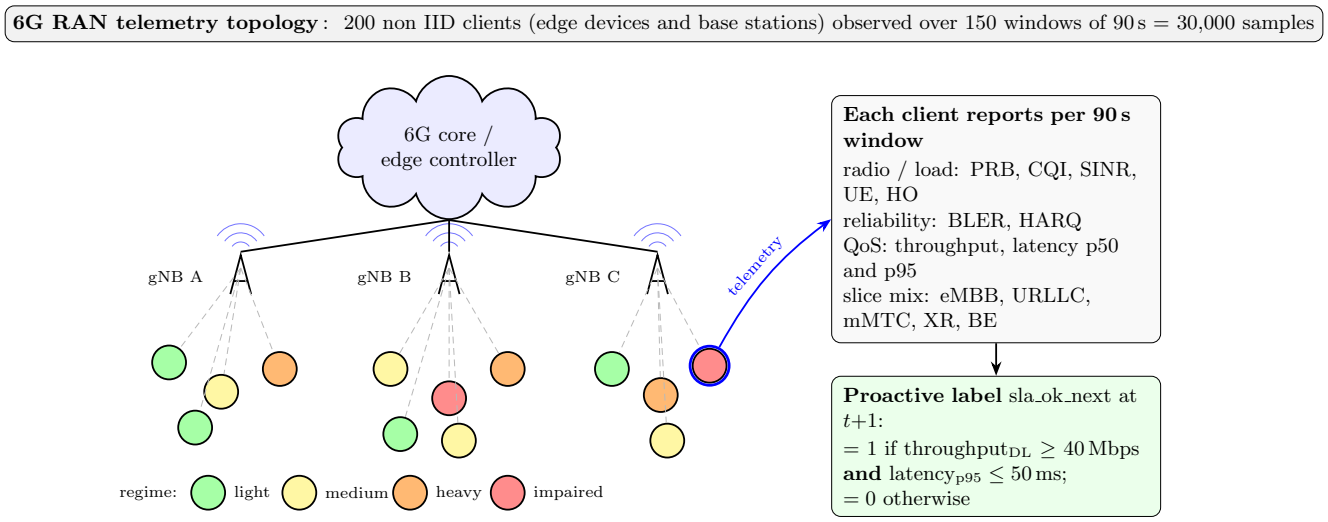


Figure 2. Structure of the 6G RAN telemetry data
Two hundred non identically distributed clients are observed over one hundred and fifty ninety second windows, and the proactive label is determined by the throughput and tail latency of the following window.

Reference models. To establish what accuracy is attainable on each task independently of the MLP trainer, we fit ℓ_2 regularized logistic regression (SLA) or ridge regression (MEC) and a histogram gradient boosting model [44] on the same splits, with library default settings. On the SLA task we also train Adam with early stopping (Adam-ES): 20% of the training clients are held out as a validation set, Adam is run for the compute matched number of steps on the remaining clients, and the weights with the lowest validation BCE are kept.

5.3. Evaluation budget and the cost of a gradient step

All optimizers receive the *same number of objective function evaluations* (FE), the budget $E_{\max}$, enforced by a counting wrapper that stops the optimizer at exactly the budget and returns the best point evaluated. One FE is one full batch forward pass of the MLP over the training set followed by the loss computation (or one call of a benchmark function). The budget is 6,400 FE for the SLA task, 3,600 FE for the MEC task, and 30,000 FE (= 1000D) for the benchmark functions, for which we also report the state after 6,000 FE (200D). For TLBO and PGTLBO, which use $2N_P$ FE per iteration, these budgets correspond to 80 iterations with $N_P = 40$ (SLA) and 60 iterations with $N_P = 30$ (MEC); GA, PSO, and HHO, which use about N_P FE per iteration, therefore run about twice as many iterations. Matching iterations instead of evaluations would give TLBO and PGTLBO about twice the evaluations of GA and PSO [16].

For gradient training (full batch Adam, learning rate 0.01) one *step* consists of one forward pass, one backward pass, and the parameter update. A forward pass is *not* computationally equivalent to a step: the backward pass of a dense network costs about two forward passes, so a step costs about three FE. On our hardware the measured wall clock ratio between one Adam step and one FE on the SLA training set was 2.7. We therefore report two gradient baselines: **Adam-C** (compute matched), which is granted $\lfloor E_{\max}/3 \rfloor$ steps, and **Adam-S** (step matched), which is granted $E_{\max}$ steps, that is, one step is charged as if it were one evaluation; Adam-S therefore uses about three times the computation of the metaheuristics and is reported only as an upper reference. Adam is not derivative free and has access to information that the metaheuristics do not have, so the comparison with Adam is a reference for what first order information buys on these problems rather than a like for like contest.

5.4. Optimizers and benchmark functions

All population based methods use $N_P = 40$ on the SLA task and $N_P = 30$ on the MEC task and the benchmark suite. GA uses tournament selection (size 3), blend crossover (rate 0.9), Gaussian mutation (rate 0.1, standard deviation 3% of the search range), and two elites; PSO uses inertia 0.7, acceleration coefficients 1.5, and velocity clamping at 20% of the range; HHO uses the standard schedule [3], with the escaping energy decaying with the fraction of the FE budget used. Runs of different methods with the same index share the random seed, so TLBO, PGTLBO, and TLBO-RD start from the same initial population. The benchmark suite contains ten functions in $D = 30$: six shifted and rotated (SR) functions (Sphere, Rosenbrock, Rastrigin, Ackley,

Griewank, Zakharov), in which a random shift removes the origin bias and a random orthogonal rotation removes separability, and four deceptive or strongly multimodal functions (Schwefel 2.26, Michalewicz, Styblinski–Tang, Levy). Each method is run 30 times per function.

5.5. Statistical analysis

Comparisons are paired by split, repeat, or run. We use the two sided Wilcoxon signed rank test with Holm's correction for the family of comparisons of PGTLBO against the other methods [45–47], report win/tie/loss counts, the Vargha–Delaney A_{12} effect size [48], and bootstrap 95% confidence intervals of mean paired differences. Across the benchmark functions we use the Friedman test on average ranks. We regard PGTLBO as better than TLBO only when the Holm adjusted p value is below 0.05.

6. Results and Discussion

This section first reports the primary 6G SLA task, the attainable accuracy on these data, and the comparison with published results. It then reports the MEC regression task and the benchmark functions, analyzes what the pseudo gradient contributes at the level of individual teacher steps, and closes with a discussion of the findings and their limitations.

6.1. Primary task: 6G SLA prediction

Table 2 reports the SLA prediction results over the 20 client held out splits; the mean persistence accuracy of the held out clients is 0.811. Figure 3(a) shows the per split lift over persistence, which removes the split to split variation of the absolute accuracy, and Figure 3(b) the optimization histories on a common axis of evaluations.

PGTLBO versus TLBO. PGTLBO and TLBO reach mean accuracies of 0.846 and 0.844. The mean paired difference is 0.0016 (95% bootstrap CI [−0.0002, 0.0034]), PGTLBO wins on 13/0/7 splits (W/T/L), and the unadjusted Wilcoxon test gives $p = 0.113$ (Holm adjusted $p = 0.338$); the corresponding differences in ROC-AUC (0.0010, $p = 0.294$) and in final training loss (−0.0032, W/T/L 15/0/5, $p = 0.143$) are also not significant. The random direction control TLBO-RD is likewise indistinguishable from PGTLBO in accuracy (unadjusted $p = 0.313$; Holm adjusted $p = 0.395$ in Table 2), although PGTLBO reaches a lower training loss than TLBO-RD on 15/0/5 splits ($p = 0.048$, unadjusted). On this task, therefore, the pseudo gradient does not produce a statistically significant change in test performance relative to TLBO, and we do not claim one.

Comparison with the other trainers. All three TLBO variants clearly outperform GA, PSO, and HHO, which win on none of the 20 splits against PGTLBO (Holm adjusted $p < 0.001$). GA and PSO end close to the persistence baseline and HHO falls well below it; their training losses remain far above those of the teaching methods (Figure 3(b)). This ordering holds although GA, PSO, and HHO receive the same number of evaluations as TLBO and therefore about twice as many iterations. Gradient training reaches the lowest training loss by a wide margin, yet its test accuracy is lower than that of PGTLBO (Adam-C: 0.841, $p = 0.014$; Adam-S: 0.839). Table 3 shows why: the test accuracy of Adam-C is highest after the smallest budget (0.849 at 800 FE equivalents) and declines as training continues (0.841 at 6,400), so the unregularized network overfits the 140 training clients, whereas the metaheuristics stop at a higher training loss. The advantage of the teaching methods over Adam at a fixed budget is therefore an implicit regularization effect, not evidence that they optimize better. With a client grouped validation split for early stopping (Adam-ES), Adam reaches 0.849 ± 0.023 after a median of 155 steps, which is higher than PGTLBO on most splits (W/T/L from the perspective of PGTLBO 1/0/19, $p < 0.001$) although Adam-ES is trained on only 80% of the training clients.

Larger search space. With $H = 48$ hidden units ($D = 1345$) the picture is unchanged (Table 3, lower block): PGTLBO, TLBO, and TLBO-RD all reach a mean accuracy of 0.845 ($p = 0.490$ for PGTLBO versus TLBO), and their final training losses differ by less than one percent (W/T/L 13/0/7, $p = 0.076$). Doubling the dimension therefore provides no evidence that the effect of the pseudo gradient grows with the size of the network.

6.2. Attainable accuracy and comparison with published results

The reference models in Table 2 calibrate what is attainable on the client held out protocol. A logistic regression reaches 0.853, the highest mean accuracy in Table 2, and is more accurate than PGTLBO on all 20 splits; gradient boosting reaches 0.847, tied with PGTLBO. Together with the early stopped Adam result above, this indicates that the test accuracy on these data is limited mainly by the information in the features and by the distribution shift between clients, not by the optimizer: once a trainer reaches a training BCE of about 0.33, further reductions do not translate into better test accuracy.

For a direct comparison with Maitlo et al. [41] we also evaluated the temporal protocol of that study (first 80% of windows for training, last 20% for testing; rows of the last training window are removed because

Table 2. 6G SLA prediction over 20 client held out splits (mean ± std; $H = 24$, $D = 673$, 6,400 FE per run) Lift is accuracy minus persistence accuracy on the same split. Train loss is the final training BCE. p is the Holm adjusted two sided Wilcoxon signed rank p value of the accuracy comparison with PGTLBO; W/T/L counts the splits on which PGTLBO wins, ties, and loses. Time: median wall clock seconds of a single threaded run without diagnostics on one split.

Method	Accuracy	F1	ROC-AUC	Lift over persist.	Train loss	Time (s)	p (acc.)	W/T/L
PGTLBO	0.846 ± 0.022	0.859	0.927	+0.035 ± 0.006	0.324	15.6		
TLBO	0.844 ± 0.021	0.858	0.926	+0.034 ± 0.005	0.327	15.9	0.338	13/0/7
TLBO-RD	0.845 ± 0.021	0.858	0.926	+0.034 ± 0.005	0.328	15.6	0.395	10/0/10
Adam-C†	0.841 ± 0.022	0.855	0.924	+0.030 ± 0.005	0.287	14.1	0.014	14/0/6
Adam-S†	0.839 ± 0.023	0.853	0.922	+0.028 ± 0.004	0.284	40.6	<0.001	17/0/3
Adam-ES†	0.849 ± 0.023	0.863	0.930	+0.039 ± 0.004	–	–	–	–
GA	0.814 ± 0.027	0.829	0.898	+0.003 ± 0.012	0.509	15.6	<0.001	20/0/0
PSO	0.809 ± 0.025	0.825	0.886	−0.002 ± 0.015	1.362	15.7	<0.001	20/0/0
HHO	0.765 ± 0.035	0.791	0.842	−0.046 ± 0.033	0.477	15.4	<0.001	20/0/0
Logistic regr.‡	0.853 ± 0.021	0.865	0.933	+0.042 ± 0.004	–	–	<0.001	0/0/20
Hist. grad. boosting‡	0.847 ± 0.023	0.861	0.929	+0.037 ± 0.005	–	–	0.395	10/0/10

† Gradient based (not derivative free). Adam-C: 2,133 steps (3 FE per step); Adam-S: 6,400 steps (about three times the computation of the other trainers). Adam-ES: Adam-C budget with early stopping on a validation set of 20% of the training clients; it is compared with PGTLBO in the text.
‡ Reference models that are not MLPs, fitted on the same splits; no FE budget applies.

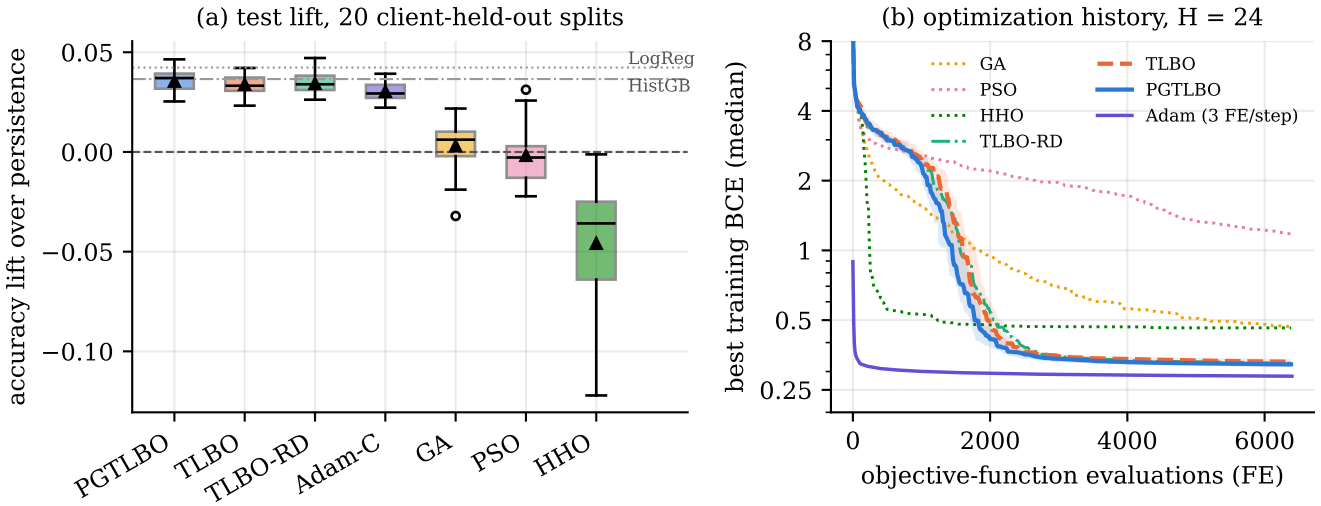


Figure 3. 6G SLA task
(a) Per split accuracy lift over persistence for the MLP trainers (boxes; triangles are means); the horizontal lines mark the mean lift of the logistic regression and gradient boosting references. (b) Median best training loss against the number of objective function evaluations (shaded: interquartile range for PGTLBO and TLBO); Adam is plotted at three evaluations per step.

their next window label is computed from the first test window). Table 4 shows that, under this protocol, PGTLBO (0.849), Adam-C (0.849), and the published centralized MLP with two hidden layers of 512 and 256 units (0.851) are within 0.002 of one another, with logistic regression again the most accurate model (0.855). The persistence baseline is 0.822. PGTLBO and TLBO do not differ in accuracy ($p = 0.193$, W/T/L 6/0/4 over 10 seeds), but PGTLBO reaches a lower training loss in 9/0/1 seeds ($p = 0.004$).

6.3. Secondary task: MEC latency regression

Table 5 and Figure 4 report the latency regression over 30 repeats. Among the derivative free trainers, GA, PGTLBO, PSO, and TLBO are statistically indistinguishable (Holm adjusted $p \geq 0.199$), while HHO is

Table 3. Anytime behavior on the SLA task (20 splits): median best training BCE and mean test accuracy after a given number of evaluations (FE; for Adam-C, FE = 3× steps). W/T/L and p (unadjusted Wilcoxon) compare the best training loss reached by PGTLBO and TLBO at each checkpoint.

		Median training BCE				PGTLBO vs TLBO		Test accuracy		
H	FE	PGTLBO	TLBO	TLBO-RD	Adam-C	W/T/L	p	PGTLBO	TLBO	Adam-C
24	800	2.685	2.762	2.706	0.303	14/0/6	0.261	0.807	0.808	0.849
	1,600	0.714	0.961	0.936	0.296	14/0/6	0.090	0.780	0.787	0.848
	3,200	0.338	0.346	0.344	0.291	11/0/9	0.330	0.839	0.834	0.844
	6,400	0.322	0.331	0.326	0.286	15/0/5	0.143	0.846	0.844	0.841
48	800	3.037	3.012	3.019	0.299	9/0/11	0.622	0.829	0.826	0.848
	1,600	2.006	1.908	2.104	0.286	7/0/13	0.154	0.795	0.789	0.843
	3,200	0.351	0.349	0.358	0.270	8/0/12	0.114	0.833	0.836	0.834
	6,400	0.323	0.326	0.328	0.261	13/0/7	0.076	0.845	0.845	0.829

Table 4. SLA prediction under the temporal protocol of Maitlo et al. [41] (mean ± std over 10 seeds for the MLP trainers; deterministic models fitted once).

Method	Accuracy	F1	ROC-AUC
Persistence	0.822	–	–
PGTLBO	0.849 ± 0.003	0.867	0.934
TLBO	0.847 ± 0.002	0.866	0.932
TLBO-RD	0.847 ± 0.002	0.866	0.931
Adam-C†	0.849 ± 0.002	0.867	0.932
Logistic regr.‡	0.855	0.872	0.938
Hist. grad. boosting‡	0.852	0.870	0.936
MLP 512–256*	0.851 ± 0.002*	–	–

∗Centralized MLP (512 and 256 hidden units, Adam) as reported by Maitlo et al. [41]: mean and 95% confidence half width over five seeds; their split does not remove the boundary window.

much worse. PGTLBO and TLBO differ by 0.0065 in R^2 (95% CI [−0.0022, 0.0150], W/T/L 19/0/11, unadjusted $p = 0.152$); PGTLBO again reaches a lower training loss more often than not (W/T/L 20/0/10, $p = 0.140$). In contrast to the SLA task, the MLP is not limited by the data here: Adam reaches $R^2 = 0.958$ with the compute matched budget, and gradient boosting reaches 0.975, far above every metaheuristic. This shows that for smooth, low dimensional regression problems derivative free training of the MLP is not competitive.

6.4. Benchmark functions

Table 6 reports the mean best fitness after 30,000 FE together with paired comparisons of PGTLBO against TLBO and against the random direction control, and Figure 5 shows the median convergence histories. Table A.1 in the Appendix gives the same comparison after 6,000 FE ($200D$), a budget typical of short runs.

No method dominates the suite, and the Friedman test over the six methods does not reject equal performance at 30,000 FE ($\chi^2_F = 7.83$, five degrees of freedom, $p = 0.166$). HHO is strongest on the deceptive Schwefel 2.26, Styblinski–Tang, and Levy functions, GA on SR Rastrigin, SR Ackley, and Michalewicz, PSO on SR Zakharov, TLBO-RD on SR Sphere and SR Rosenbrock, and PGTLBO on SR Griewank. Against its parent algorithm, PGTLBO is significantly better than TLBO on four functions (SR Sphere, SR Ackley, SR Griewank, SR Zakharov) and significantly worse on none, and it obtains the best average rank (2.50, followed by TLBO-RD at 2.80, GA at 3.30, and TLBO at 4.10). After 6,000 FE the picture is similar for PGTLBO against TLBO (significantly better on SR Sphere, SR Rosenbrock, SR Ackley, SR Griewank; worse on none), whereas GA has the best average rank at that budget (2.20, versus 3.60 for PGTLBO).

The random direction control qualifies this result. TLBO-RD is itself significantly better than TLBO on four functions at both budgets (SR Sphere, SR Rosenbrock, SR Ackley, SR Griewank; Holm adjusted Wilcoxon tests), three of which are among the four on which PGTLBO improves on TLBO, and PGTLBO is better than TLBO-RD on only two functions (SR Rastrigin, SR Zakharov) and worse on two (SR Sphere, SR Ackley) at 30,000 FE, and better on one (SR Griewank) and worse on none at 6,000 FE. A large part of the improvement of PGTLBO over TLBO on the shifted and rotated functions is therefore produced by adding a population scaled perturbation to the teacher step, irrespective of its direction. The direction alone is not sufficient either: the PG-only ablation, which keeps $\mathbf{d}$ but drops the teacher term, is significantly worse than PGTLBO on six functions and better on none, so the pseudo gradient is useful as a complement to the teacher update and not as a replacement for it.

6.5. What does the pseudo gradient contribute?

The comparisons of final results above are coarse, because the pseudo gradient changes only one of the two TLBO phases and because the outcome of a run aggregates thousands of stochastic steps. We therefore

Table 5. MEC latency regression in log latency space over 30 repeats (mean ± std; $D = 71$, 3,600 FE per run) Train loss is the final standardized training MSE. p: Holm adjusted Wilcoxon signed rank test of R^2 against PGTLBO; W/T/L from the perspective of PGTLBO. Time: median wall clock seconds of a single threaded run on one split; differences of a few tenths of a second are within measurement noise at this problem size.

Method	R^2	RMSE	MAE	Train loss	Time (s)	p (R^2)	W/T/L
PGTLBO	0.878 ± 0.019	0.564	0.411	0.120	0.74		
TLBO	0.872 ± 0.015	0.580	0.427	0.127	0.90	0.456	19/0/11
TLBO-RD	0.872 ± 0.019	0.581	0.429	0.128	0.94	0.598	17/0/13
GA	0.888 ± 0.015	0.542	0.390	0.111	0.94	0.199	11/0/19
PSO	0.877 ± 0.023	0.567	0.413	0.121	0.87	0.935	15/0/15
HHO	0.607 ± 0.126	1.006	0.811	0.386	0.91	<0.001	30/0/0
Adam-C†	0.958 ± 0.014	0.330	0.183	0.042	0.95	<0.001	0/0/30
Adam-S†	0.974 ± 0.003	0.262	0.126	0.026	2.73	<0.001	0/0/30
Ridge regr.‡	0.850 ± 0.006	0.628	0.488	–	–	<0.001	28/0/2
Hist. grad. boosting‡	0.975 ± 0.002	0.257	0.105	–	–	<0.001	0/0/30

† Gradient based. Adam-C: 1,200 steps; Adam-S: 3,600 steps. ‡ Reference models, no FE budget.

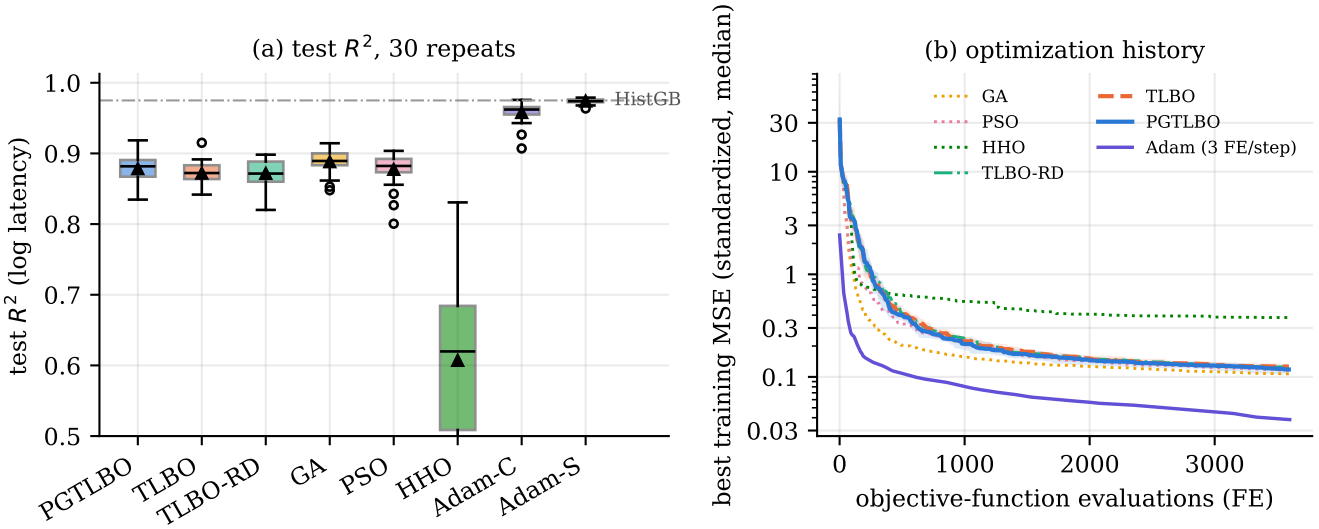


Figure 4. MEC latency regression
(a) Test R^2 over 30 repeats; the dash dotted line marks gradient boosting. (b) Median best training MSE against evaluations; Adam is plotted at three evaluations per step.

looked at the teacher step itself through the off budget counterfactual described in Section 4.3: in every teacher update of PGTLBO, the candidates with and without **d** and with a random direction of the same length were built from the same state and the same random numbers. Table 7 and Figure 6 report how often each candidate improves the learner it was generated for, split into the first, middle, and last third of the iterations.

Three findings emerge. First, the *direction* of **d** carries information: in every task and at every stage, the PGTLBO candidate improves the learner significantly more often than the random direction candidate of equal length (for example 0.301 versus 0.218 on the SLA task and 0.144 versus 0.100 on the MEC task; both $p < 0.001$), and it is the better of the two candidates in 62.6% and 59.6% of all teacher updates, respectively. The pseudo gradient is also positively aligned with the true descent direction $-\nabla f$ at the population mean: the mean cosine similarity is 0.025 and 0.123 on the SLA and MEC tasks and between 0.005 and 0.310 on the benchmark functions, against approximately zero for random directions (0.001, 0.006). Second, adding **d** to the TLBO step is *not* uniformly beneficial. On the benchmark suite the PGTLBO candidate improves the learner more often than the plain TLBO candidate (0.234 versus 0.223, $p < 0.001$), but on the learning tasks it does so less often in the early iterations

Table 6. Mean best fitness on the benchmark suite after 30,000 FE, 30 paired runs, $D = 30$ (lower is better; best mean in bold). The first six functions are shifted and rotated (SR). W/T/L counts the runs on which PGTLBO is better, tied, and worse than the named method; p is the two sided Wilcoxon signed rank p value, Holm adjusted over the ten functions. Average ranks are computed from the unrounded means.

							PGTLBO vs TLBO		PGTLBO vs TLBO-RD	
Function	GA	PSO	HHO	TLBO	TLBO-RD	PGTLBO	W/T/L	p	W/T/L	p
SR Sphere	3.37e0	1.20e2	1.65e1	2.04e-1	**7.25e-7**	1.16e-4	29/0/1	<0.001	4/0/26	<0.001
SR Rosenbrock	3.13e2	8.72e4	3.73e3	3.39e2	**1.40e2**	2.61e2	22/0/8	0.298	10/0/20	0.249
SR Rastrigin	**7.94e1**	8.69e1	1.12e2	1.07e2	1.34e2	9.91e1	15/0/15	1.000	21/0/9	0.021
SR Ackley	**1.53e0**	3.72e0	8.35e0	7.02e0	4.44e0	5.66e0	23/0/7	0.023	8/0/22	0.020
SR Griewank	1.03e0	9.21e-1	3.48e0	1.37e-1	1.48e-2	**1.38e-2**	30/0/0	<0.001	15/0/15	1.000
SR Zakharov	7.06e-1	**2.05e-1**	3.71e1	2.57e0	2.16e0	9.50e-1	23/0/7	<0.001	25/0/5	<0.001
Schwefel 2.26	5.73e3	5.25e3	**2.72e2**	5.10e3	4.88e3	4.74e3	15/0/15	1.000	16/0/14	1.000
Michalewicz	**-2.53e1**	-2.47e1	-1.28e1	-2.45e1	-2.45e1	-2.48e1	15/0/15	1.000	16/0/14	1.000
Styblinski-Tang	-1.01e3	-9.94e2	**-1.17e3**	-1.03e3	-1.04e3	-1.03e3	19/0/11	1.000	12/0/18	0.149
Levy	4.75e0	6.89e0	**1.94e-4**	6.68e-1	5.88e-1	6.42e-1	17/0/13	1.000	11/0/19	1.000
Average rank (6 methods)	3.30	4.10	4.20	4.10	2.80	2.50				

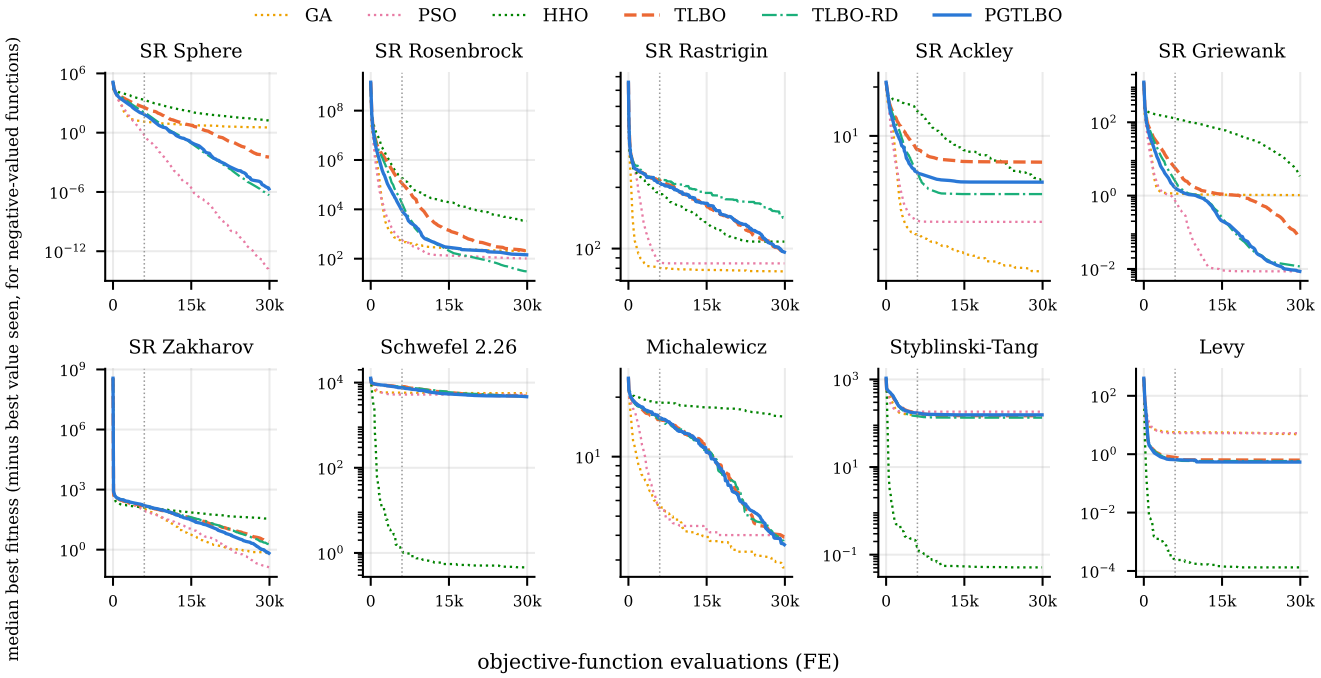


Figure 5. Median best fitness against evaluations on the benchmark suite (30 runs) The dotted vertical line marks 6,000 FE ($200D$). For Michalewicz and Styblinski–Tang, whose optima are negative, the best value observed by any method is subtracted before the logarithm is taken.

(SLA: 0.298 versus 0.320, $p < 0.001$; MEC: 0.144 versus 0.182 over all iterations), when the population is still spread over the box constraints and the added step overshoots, and more often only late in the search for the larger network (SLA, $H = 48$: 0.423 versus 0.388, $p = 0.002$). The cosine similarities also show that **d** is only weakly aligned with the local gradient in the high dimensional weight spaces, which is consistent with its interpretation as a population scale secant direction rather than a local gradient estimate. Third, these per step effects net out to a small but consistent advantage in optimization: across the 80 paired runs of the learning tasks, PGTLBO ends with a lower training loss than TLBO in 57 runs (SLA $H = 24$: 15/20; SLA $H = 48$: 13/20; temporal protocol: 9/10; MEC: 20/30; sign test $p < 0.001$, or $p < 0.001$ with 44 of 60 when the $H = 48$ runs, which share their splits with the $H = 24$ runs, are excluded), with a mean relative reduction of 2.2% (median 1.5%). It also ends lower than TLBO-RD in 57 of 80 runs ($p < 0.001$).

Table 7. Counterfactual analysis of the teacher step
For every teacher update of PGTLBO the plain TLBO candidate, the PGTLBO candidate (TLBO step $+\mathbf{r}_2 \odot \mathbf{d}$), and the random direction candidate (TLBO step $+\mathbf{r}_2 \odot \mathbf{u}$, $\|\mathbf{u}\| = \|\mathbf{d}\|$) are generated from the same state with the same random numbers. Columns 3 to 5: probability that the candidate improves the learner (mean over runs). p: paired Wilcoxon test over runs (unadjusted). Last column: probability that the PGTLBO candidate is better than the random direction candidate.

Task	Stage	P(candidate improves)			p, PGTLBO cand. vs		P(PGTLBO cand.
		TLBO	PGTLBO	Random dir.	TLBO	random	< random cand.)
SLA, H=24	all	0.306	0.301	0.218	0.648	<0.001	0.626
	early	0.320	0.298	0.262	<0.001	<0.001	0.559
	middle	0.267	0.264	0.187	0.575	<0.001	0.651
	late	0.333	0.342	0.205	0.093	<0.001	0.671
SLA, H=48	all	0.325	0.322	0.249	0.263	<0.001	0.611
	early	0.304	0.289	0.241	0.065	<0.001	0.575
	middle	0.285	0.257	0.211	0.881	0.010	0.573
	late	0.388	0.423	0.295	0.002	<0.001	0.690
MEC	all	0.182	0.144	0.100	<0.001	<0.001	0.596
	early	0.224	0.190	0.175	<0.001	<0.001	0.530
	middle	0.142	0.094	0.059	<0.001	0.002	0.622
	late	0.180	0.149	0.067	0.001	<0.001	0.636
Benchmarks (10 fn.)	all	0.223	0.234	0.179	<0.001	<0.001	0.538
	early	0.247	0.253	0.204	<0.001	<0.001	0.603
	middle	0.245	0.261	0.197	<0.001	<0.001	0.547
	late	0.176	0.187	0.136	<0.001	<0.001	0.464

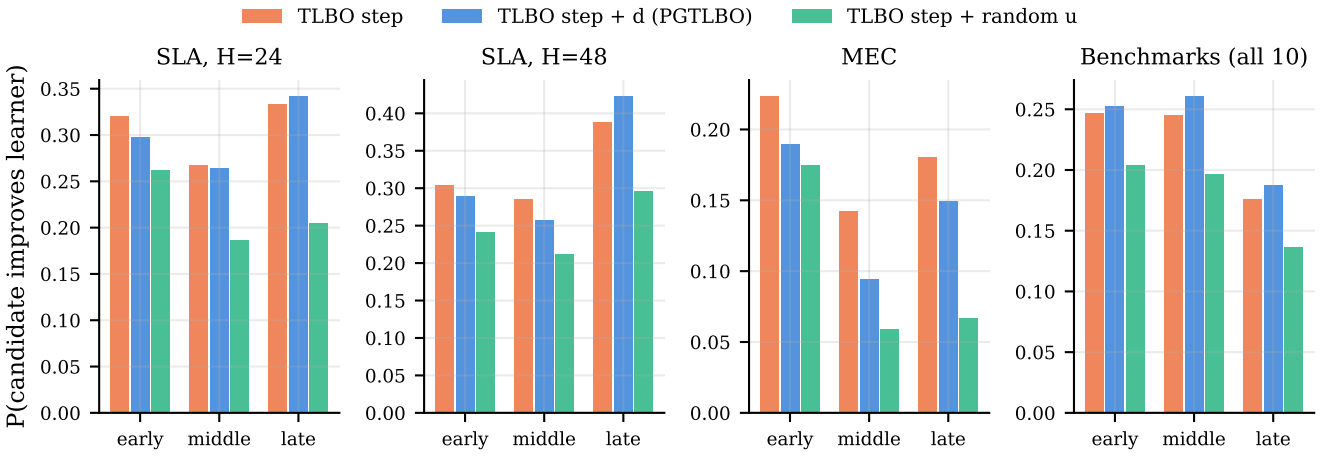


Figure 6. Probability that a teacher phase candidate improves its learner, for the plain TLBO step, the PGTLBO step, and a step with a random direction of the same length as $\mathbf{d}$, all generated from the same states with the same random numbers (off budget counterfactual; means over 20, 20, 30, and 300 runs)

6.6. Discussion and threats to validity

Table 8 collects the answers to the questions posed in Section 1 together with the evidence for each.

The evidence supports a deliberately narrow message. The ranked population does carry directional information that standard TLBO does not use, and a parameter free estimate of it measurably improves the teacher step relative to a random perturbation of equal length and lowers the training loss reached by TLBO within a fixed budget. On the shifted and rotated benchmark functions PGTLBO improves significantly on TLBO, although a random perturbation of the same length achieves a comparable improvement, so that the benefit there is mainly a matter of step length. On the two learning tasks the optimization gain is about two percent of the training loss and does not translate into a statistically significant improvement of test accuracy, ROC-AUC, or R^2 over TLBO. We therefore claim parity of PGTLBO with TLBO on the learning tasks, a significant advantage of both over PSO, GA,

Table 8. Summary of findings
Each question is answered from the indicated table or section

Question	Evidence	Answer
Does the direction of **d** carry information?	PGTLBO candidate beats the equal length random direction candidate in 62.6% (SLA) and 59.6% (MEC) of teacher updates; more often improving than random at every stage of every task, $p \leq 0.010$ (Table 7)	Yes
Does it improve optimization of the training loss?	Lower final training loss than TLBO in 57 of 80 paired runs, mean relative reduction 2.2%, sign test $p < 0.001$ (Section 6.5)	Yes, modestly
Does it improve test performance over TLBO?	SLA accuracy 0.846 versus 0.844 (Holm $p = 0.338$); MEC R^2 0.878 versus 0.872 (Holm p = 0.456) (Tables 2 and 5)	No significant difference
Does it help on benchmark functions?	Better than TLBO on 4 of 10 functions and worse on none, but TLBORD is also better than TLBO on 4 (Table 6)	Partly; mostly a step length effect
How do TLBO and PGTLBO compare with GA, PSO, and HHO on the 6G task?	PGTLBO wins on all 20 client held out splits against each (Holm $p <$ 0.001) (Table 2)	Clearly better
Is the MLP trainer the limiting factor?	Logistic regression 0.853 and early stopped Adam 0.849 on SLA; Adam 0.958 and gradient boosting 0.975 on MEC (Tables 2 and 5)	No; data (SLA) or first order information (MEC)

and HHO on the 6G task, and a small optimization advantage of PGTLBO over TLBO, and we do not claim that PGTLBO improves the predictive performance of the trained models.

Two observations explain why better optimization does not help the test metrics here. On the 6G task the attainable accuracy is limited by the data: logistic regression is the most accurate model, unregularized Adam reaches a much lower training loss but generalizes worse unless it is stopped early, and all trainers that reach a training BCE of about 0.33 are within a few thousandths of one another on held out clients. On the MEC task the opposite holds: the network can fit the data much better, but gradient training does so far more efficiently than any derivative free method, so that the relevant gap is between first order and zeroth order training rather than between TLBO variants.

These results have three practical implications for 6G controllers. First, when an SLA predictor must be trained without gradients, for example with a non differentiable objective or on hardware without automatic differentiation, a teaching based trainer such as TLBO or PGTLBO is preferable to GA, PSO, or HHO, which stayed at or below the persistence baseline on unseen clients. Second, PGTLBO can replace TLBO at no extra cost and with no new coefficient, and it reaches a slightly lower training loss, but this should not be expected to change the accuracy of the deployed model. Third, because the next SLA state equals the current one in most windows, a predictor should be judged by its lift over persistence on held out clients rather than by raw accuracy on windows of clients seen in training.

Several limitations remain. The networks are small (at most 1,345 weights), and the conclusions about step length and direction may change for larger models, for which derivative free training is in any case expensive. The pseudo gradient enters with a unit random scale; the counterfactual analysis suggests that a smaller scale in the early iterations would avoid the overshooting observed on the learning tasks, but tuning it would reintroduce a coefficient. The benchmark results depend on the operator settings of GA, PSO, and HHO, which we fixed to common defaults without tuning. Finally, both learning tasks use public data sets in which the SLA label is derived from thresholds on throughput and latency of the next window, and results on operational network data may differ.

7. Conclusion

We studied PGTLBO, a TLBO variant that adds to the teacher update the direction from the centroid of the worse half to the centroid of the better half of the ranked population, a parameter free, rank based search direction related to evolution strategy gradient estimates. Under an identical evaluation budget, with gradient steps charged for their backward pass and with a random direction control of equal step length, the pseudo gradient direction proved better than random at every stage of the search, and PGTLBO reached a lower training loss than TLBO in 57 of 80 paired runs on the learning tasks and significantly better results than TLBO on four of ten benchmark functions, without being significantly worse on any. These gains did not translate into a statistically significant improvement of test performance over TLBO on proactive 6G SLA prediction or on MEC latency regression. On the 6G task PGTLBO and TLBO were tied and both clearly outperformed PSO, GA, and HHO, but test accuracy was limited by the data, with logistic regression and early stopped gradient training at least as accurate as any metaheuristic trainer; on the MEC task gradient

training and gradient boosting were far superior to all derivative free trainers.

Future work will examine a schedule for the scale of the pseudo gradient that avoids the early overshooting identified by the counterfactual analysis, evaluate the method on larger networks and non differentiable training objectives, where a derivative free trainer is better motivated, and study federated settings and fairness across heterogeneous clients.

Code and data availability

The 6G RAN telemetry [41] and the task offloading data [43] are publicly available from their sources. The code used in this study is publicly available at https://github.com/hilalnuha/pgtlbo/tree/main. The revised experiments (evaluation counted optimizers, controls, diagnostics, statistical analysis, and the scripts that generate every table and figure from the raw results) are included in the repository.

Appendix A. Benchmark results after 6,000 evaluations

Table A.1 repeats the benchmark comparison of Table 6 at the smaller budget of 6,000 FE ($200D$); the results are discussed in Section 6.4.

Acknowledgements

This research is funded by Telkom University PPM Grant No. 6414/LIT06/PPM-LIT/2026.

Author contributions

H.H. Nuha: conceptualization, methodology, software, formal analysis, and writing of the original draft. P.A. Atmajaya: validation, investigation, and writing, review, and editing. G.M. Firdaus: software, investigation (simulations), and writing, review, and editing (manuscript revision). All authors have read and approved the final manuscript.

Declaration on the use of generative AI

During the preparation of this work the authors used Claude (Anthropic), a large language model, to help write and debug the Python experiment scripts and the code that draws the figures, to draft and revise passages of the manuscript, and to proofread the text and check that the numbers in the text agree with the tables. The authors reviewed, edited, and verified all code, results, and text, and take full responsibility for the content of this article. No AI generated images are included; all figures are plots or diagrams produced from code.

References

[1] Goldberg DE. Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, MA: Addison Wesley; 1989.

[2] Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of the International Conference on Neural Networks; 1995. p. 1942-8.

[3] Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafarja M, Chen H. Harris hawks optimization: algorithm and applications. Future Generation Computer Systems. 2019;97:849-72.

[4] Ojha VK, Abraham A, Snášel V. Metaheuristic design of feedforward neural networks: a review of two decades of research. Engineering Applications of Artificial Intelligence. 2017;60:97-116.

[5] Nuha HH, Ariyanto E, Nurma Yulita II, Sailellah HRP. Clustered packet optimization for human powered energy generation prediction. Discover Internet of Things. 2026;6(1):111.

[6] Fauzan MN, Munadi R, Sumaryo S, Nuha HH. Enhanced wireless sensor network lifetime using EGWO optimized neural network approach. Network. 2026;6(1):5.

[7] Al-Shaikhi A, Nuha H, Mohandes M, Rehman S, Adrian M. Vertical wind speed extrapolation model using long short term memory and particle swarm optimization. Energy Science & Engineering. 2022;10(12):4580-94.

[8] Fauzan MN, Munadi R, Sumaryo S, Nuha HH. Enhanced grey wolf optimization for efficient transmission power optimization in wireless sensor network. Applied System Innovation. 2025;8(2):36.

[9] Suyanto S, Ariyanto AA, Ariyanto AF. Komodo mlipir algorithm. Applied Soft Computing. 2022;114:108043.

[10] Setiawan D, Suyanto S, Erfianto B, Gozali AA. Battlefield optimization algorithm. Expert Systems with Applications. 2025;266:126097.

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

[12] Hansen N, Ostermeier A. Completely derandomized self adaptation in evolution strategies. Evolutionary Computation. 2001;9(2):159-95.

[13] Wierstra D, Schaul T, Glasmachers T, Sun Y, Peters J, Schmidhuber J. Natural evolution strategies. Journal of Machine Learning Research. 2014;15:949-80. Available from: https://jmlr.org/papers/v15/wierstra14a.html.

[14] Gong W, Cai Z. Differential evolution with ranking based mutation operators. IEEE Transactions on Cybernetics. 2013;43(6):2066-81.

[15] Zhang Y, Jin Z. Group teaching optimization algorithm: a novel metaheuristic method for solving global optimization problems. Expert Systems with Applications. 2020;148:113246.

[16] Črepinšek M, Liu SH, Mernik L. A note on teaching learning based optimization algorithm. Information Sciences. 2012;212:79-93.

[17] Salimans T, Ho J, Chen X, Sidor S, Sutskever I. Evolution strategies as a scalable alternative to reinforcement learning; 2017. arXiv preprint arXiv:1703.03864. Available from: https://arxiv.org/abs/1703.03864.

[18] Such FP, Madhavan V, Conti E, Lehman J, Stanley KO, Clune J. Deep neuroevolution: genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning; 2017. arXiv preprint arXiv:1712.06567. Available from: https://arxiv.org/abs/1712.06567.

[19] Kingma DP, Ba J. Adam: a method for stochastic optimization. In: International Conference on Learning Representations (ICLR); 2015. Available from: https:

Table A.1. Mean best fitness on the benchmark suite after 6,000 FE (200D, a small budget), 30 paired runs, $D = 30$; layout as in Table 6, with average ranks computed from the unrounded means.

Function	GA	PSO	HHO	TLBO	TLBO-RD	PGTLBO	PGTLBO vs TLBO		PGTLBO vs TLBO-RD	
							W/T/L	p	W/T/L	p
SR Sphere	**1.33e1**	1.22e2	2.07e3	3.75e2	1.29e2	9.65e1	28/0/2	<0.001	21/0/9	0.311
SR Rosenbrock	**6.09e2**	9.67e4	2.30e5	1.20e5	1.92e4	1.41e4	29/0/1	<0.001	18/0/12	0.629
SR Rastrigin	**8.39e1**	9.10e1	1.90e2	2.13e2	2.18e2	2.11e2	16/0/14	1.000	18/0/12	0.629
SR Ackley	**2.45e0**	3.77e0	1.40e1	8.11e0	5.64e0	6.26e0	23/0/7	0.001	15/0/15	0.947
SR Griewank	**1.13e0**	1.63e0	1.26e2	6.16e0	2.63e0	2.14e0	29/0/1	<0.001	25/0/5	0.007
SR Zakharov	1.14e2	**9.85e1**	1.27e2	1.53e2	1.58e2	1.59e2	14/0/16	1.000	14/0/16	1.000
Schwefel 2.26	5.88e3	5.26e3	**3.97e2**	7.80e3	7.79e3	7.31e3	17/0/13	0.657	20/0/10	0.341
Michalewicz	-2.27e1	**-2.28e1**	-9.87e0	-1.34e1	-1.34e1	-1.30e1	15/0/15	1.000	13/0/17	0.947
Styblinski-Tang	-1.01e3	-9.94e2	**-1.17e3**	-1.01e3	-1.03e3	-1.01e3	16/0/14	1.000	11/0/19	0.384
Levy	5.52e0	6.92e0	**4.78e-4**	7.46e-1	6.32e-1	6.82e-1	16/0/14	1.000	17/0/13	1.000
Average rank (6 methods)	2.20	2.90	3.90	4.70	3.70	3.60				

//arxiv.org/abs/1412.6980.

[20] Hartono R, Mustika IW, Sulistyo S. Metaheuristic approaches for energy optimization in wireless sensor networks: a systematic review of trends, challenges, and future directions. EAI Endorsed Transactions on Internet of Things. 2026;11.

[21] Sabat NR, Sahoo RR, Acharya B, Kumar R. Hybrid genetic algorithm and water wave optimization approach for QoS aware multi objective task scheduling and load balancing in cloud environments. EAI Endorsed Transactions on Internet of Things. 2026;11.

[22] Yang Q, Chen B, Huang Y, Hu X. Optimal scheduling of renewable power distribution systems combining deep learning and particle swarm optimization. EAI Endorsed Transactions on Energy Web. 2026;13.

[23] Tran DT, Ha DB, Ansere JA. Overview of quantum machine learning for 6G. EAI Endorsed Transactions on Tourism, Technology and Intelligence. 2025;2(3).

[24] Rao RV. Teaching Learning Based Optimization Algorithm and Its Engineering Applications. Cham: Springer; 2016.

[25] Zou F, Chen D, Xu Q. A survey of teaching learning based optimization. Neurocomputing. 2019;335:366-83.

[26] Rao RV, Patel V. An elitist teaching learning based optimization algorithm for solving complex constrained optimization problems. International Journal of Industrial Engineering Computations. 2012;3(4):535-60.

[27] Satapathy SC, Naik A. Modified teaching learning based optimization algorithm for global numerical optimization: a comparative study. Swarm and Evolutionary Computation. 2014;16:28-37.

[28] Zou F, Wang L, Hei X, Chen D, Yang D. Teaching learning based optimization with dynamic group strategy for global optimization. Information Sciences. 2014;273:112-31.

[29] Rao RV. 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.

[30] Jastrebski GA, Arnold DV. Improving evolution strategies through active covariance matrix adaptation. In: Proceedings of the IEEE Congress on Evolutionary Computation; 2006. p. 2814-21.

[31] Rubinstein R. The cross entropy method for combinatorial and continuous optimization. Methodology and Computing in Applied Probability. 1999;1(2):127-90.

[32] Larrañaga P, Lozano JA, editors. Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Boston, MA: Kluwer Academic Publishers; 2002.

[33] Zhang J, Sanderson AC. JADE: adaptive differential evolution with optional external archive. IEEE Transactions on Evolutionary Computation. 2009;13(5):945-58.

[34] Tanabe R, Fukunaga A. Success history based parameter adaptation for differential evolution. In: Proceedings of the IEEE Congress on Evolutionary Computation; 2013. p. 71-8.

[35] Storn R, Price K. Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization. 1997;11(4):341-59.

[36] Cheng R, Jin Y. A competitive swarm optimizer for large scale optimization. IEEE Transactions on Cybernetics. 2015;45(2):191-204.

[37] Cheng R, Jin Y. A social learning particle swarm optimization algorithm for scalable optimization. Information Sciences. 2015;291:43-60.

[38] Mirjalili S, Mirjalili SM, Lewis A. Grey wolf optimizer. Advances in Engineering Software. 2014;69:46-61.

[39] Ahmadianfar I, Bozorg-Haddad O, Chu X. Gradient based optimizer: a new metaheuristic optimization algorithm. Information Sciences. 2020;540:131-59.

[40] Mao Y, You C, Zhang J, Huang K, Letaief KB. A survey on mobile edge computing: the communication perspective. IEEE Communications Surveys and Tutorials. 2017;19(4):2322-58.

[41] Maitlo N, Shah MH, Maitlo A, Mustafa G, Arshid K, Noonari N. Communication fairness trade offs in federated learning for 6G resource allocation: a 200 client study. Inventions. 2026;11(2):31.

[42] Maitlo N, Rajper S, Shaikh M, Kareem H. Robust privacy preserving federated learning for 6G network resource allocation. ILMA Journal of Technology & Software Management. 2026;7(1):1-8. Available from: https://ijtsm.ilmauniversity.edu.pk/arc/Vol7/i1P1.

[43] UCI Machine Learning. Task Offloading Dataset; 2023. Kaggle. Available from: https://www.

kaggle.com/datasets/ucimachinelearning/task-offloading-dataset.

[44] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: machine learning in Python. Journal of Machine Learning Research. 2011;12:2825-30. Available from: https://jmlr.org/papers/v12/pedregosa11a.html.

[45] Demšar J. Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research. 2006;7:1-30. Available from: https://jmlr.org/papers/v7/demsar06a.html.

[46] Holm S. A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics. 1979;6(2):65-70. Available from: https://www.jstor.org/stable/4615733.

[47] Derrac J, García S, Molina D, Herrera F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation. 2011;1(1):3-18.

[48] Vargha A, Delaney HD. A critique and improvement of the CL common language effect size statistics of McGraw and Wong. Journal of Educational and Behavioral Statistics. 2000;25(2):101-32.