Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Unified Algorithm Framework for Nonconvex Stochastic Optimization in Deep Neural Networks

YINI ZHU[1], **HIDEAKI IIDUKA**[2]
[1]Computer Science Course, Graduate School of Science and Technology, Meiji University, Kanagawa 214-8571, Japan (e-mail: yini@cs.meiji.ac.jp)
[2]Department of Computer Science, Meiji University, Kanagawa 214-8571, Japan (e-mail: iiduka@cs.meiji.ac.jp)

Corresponding author: Yini Zhu (e-mail: yini@cs.meiji.ac.jp).

This work was supported by JSPS KAKENHI Grant Number 21K11773.

ABSTRACT This paper presents a unified algorithmic framework for nonconvex stochastic optimization, which is needed to train deep neural networks. The unified algorithm includes the existing adaptive-learning-rate optimization algorithms, such as Adaptive Moment Estimation (Adam), Adaptive Mean Square Gradient (AMSGrad), Adam with weighted gradient and dynamic bound of learning rate (GWDC), AMSGrad with weighted gradient and dynamic bound of learning rate (AMSGWDC), and Adapting stepsizes by the belief in observed gradients (AdaBelief). The paper also gives convergence analyses of the unified algorithm for constant and diminishing learning rates. When using a constant learning rate, the algorithm can approximate a stationary point of a nonconvex stochastic optimization problem. When using a diminishing rate, it converges to a stationary point of the problem. Hence, the analyses lead to the finding that the existing adaptive-learning-rate optimization algorithms can be applied to nonconvex stochastic optimization in deep neural networks in theory. Additionally, this paper provides numerical results showing that the unified algorithm can train deep neural networks in practice. Moreover, it provides numerical comparisons for unconstrained minimization using benchmark functions of the unified algorithm with certain heuristic intelligent optimization algorithms. The numerical comparisons show that a teaching-learning-based optimization algorithm and the unified algorithm perform well.

INDEX TERMS Adam, adaptive-learning-rate optimization algorithm, AMSGrad, AMSGWDC, deep neural network, GWDC, heuristic intelligent optimization methods, learning rate, nonconvex stochastic optimization, stationary point problem.

I. INTRODUCTION

A USEFUL way to train deep neural networks is to solve a nonconvex optimization problem in terms of deep neural networks [1], [2], [3] and find suitable parameters for them. Many algorithms have been presented to solve nonconvex optimization problems. The simplest algorithm for the problem is stochastic gradient descent (SGD) (see, e.g., [4], [5] for recent studies for SGD). Adaptive-learning-rate optimization algorithms (see Subsection 8.5 in [6]) are powerful methods that adapt the learning rates of the model parameters to solve the problem quickly. Examples include Adaptive Gradient (AdaGrad) [7], Root Mean Square Propagation (RMSProp) [6, Algorithm 8.5], Adaptive Moment Estimation (Adam) [8], Adaptive Mean Square Gradient (AMSGrad) [9], Adam with weighted gradient and dynamic bound of learning rate (GWDC) [2, Algorithm 2], AMSGrad with weighted gradient and dynamic bound of learning rate (AMSGWDC) [2, Algorithm 3], and Adapting stepsizes by the belief in observed gradients (AdaBelief) [10]. Note that the existing adaptive-learning-rate optimization algorithms use the inverses of certain positive-definite matrices at each iteration; in other words, they are framed depending on the definitions of such positive-definite matrices.

In this paper, we first show that the positive-definite matrices used in the existing adaptive-learning-rate optimization algorithms assume common conditions (Assumption III.1). Next, we present an algorithm [11] (Algorithm 1) for which convergence is guaranteed under Assumption III.1. This implies that the algorithm is a unification of the existing ones (see also Example III.1). For the convergence analyses of the

TABLE 1. Convergence rates for SGD and adaptive-learning-rate optimization algorithms (GWDC, AMSGWDC, AdaBelief, and its unified algorithm) on recent papers published in 2020 and 2021

Algorithm [Ref.] (Authors; year)	Nonconvex optimization		Convex optimization	
	Constant learning rate	Diminishing learning rate	Constant learning rate	Diminishing learning rate
SGD [4] (Scaman et al.; 2020)	$\mathcal{O}\left(\frac{1}{n}\right)+C$	$\mathcal{O}\left(\frac{1}{\sqrt{n}}\right)$	$\mathcal{O}\left(\frac{1}{T}\right)+C$	$\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$
SGD [5] (Loizou et al.; 2021)	——	$\mathcal{O}\left(\frac{1}{n}\right)+C$	——	$\mathcal{O}\left(\frac{1}{T}\right)+C$
GWDC [2] (Liang et al.; 2020)	——	——	——	$\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$
AMSGWDC [2] (Liang et al.; 2020)	——	——	——	$\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$
AdaBelief [10] (Zhuang et al.; 2020)	——	$\mathcal{O}\left(\frac{\log n}{\sqrt{n}}\right)$	——	$\mathcal{O}\left(\frac{\log T}{\sqrt{T}}\right)$
Algorithm 1 [11] (Iiduka; 2021)	$\mathcal{O}\left(\frac{1}{n}\right)+C_1\alpha+C_2\beta$	$\mathcal{O}\left(\frac{1}{\sqrt{n}}\right)$	$\mathcal{O}\left(\frac{1}{T}\right)+C_1\alpha+C_2\beta$	$\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$

The convergence rate of nonconvex (resp. convex) optimization is measured by $\min_{k=1,2,\ldots,n}\mathbb{E}[\|\nabla f(\boldsymbol{x}_k)\|^2]$ (resp. $R(T)/T$), where n denotes the number of iterations and T denotes the number of training samples (see Table 2 for the definitions of the notation). C and C_i $(i = 1,2)$ are constants that are independent of n, T, and the constant learning rates α and β. A diminishing learning rate is $\alpha_n = 1/\sqrt{n}$.

algorithm, we use constant and diminishing learning rates. We show that, for a constant learning rate, the algorithm can approximate a stationary point of the nonconvex stochastic optimization problem (Theorem III.1), while for a diminishing rate, it converges to a stationary point of the problem (Theorem III.2). Table 1 summarizes convergence rate results of SGD and adaptive-learning-rate optimization algorithms for nonconvex and convex optimization that were studied in 2020 and 2021.

Our first contribution is to provide convergence analyses of the existing adaptive-learning-rate optimization algorithms, i.e., analyses which could not be done using the methods in [2], [4], [5], [8], [9], [10]. The previously reported results (see, e.g., [2], [8], [9]) tried to minimize the regret (see Table 2 for the definition) for convex optimization. However, regret minimization does not always lead to solutions of optimization problems in deep learning (Subsection IV-E). In contrast to the previous results, this paper explicitly shows that the existing adaptive-learning-rate optimization algorithms can solve such problems (Subsections IV-A–IV-D). For nonconvex optimization, AdaBelief [10] has an $\mathcal{O}(\log n/\sqrt{n})$ convergence rate, where n denotes the number of iterations (see Table 1). Theorem III.2 ensures that using a diminishing learning rate $\alpha_n = 1/\sqrt{n}$ allows the proposed algorithm (Algorithm 1) including AdaBelief to have an $\mathcal{O}(1/\sqrt{n})$ convergence rate (see also [11]). While GWDC and AMSGWDC [2] with diminishing learning rates can only be applied to convex optimization (see Table 1), the proposed algorithm (Algorithm 1) can be applied to both convex and nonconvex optimization (see Table 1).

In particular, we would like to emphasize that the existing algorithms with constant learning rates can be applied to the problem (Subsection IV-E and Table 1). The results for a constant learning rate are significant from the viewpoints of both theory and practice, since algorithms with a constant learning rate work well whereas the algorithms with a diminishing rate converging to zero do not work. Moreover, we also would like to emphasize that not only Adam and AMSGrad but also GWDC and AMSGWDC can be applied to the problem. This is in contrast to [2], which presented a regret minimization only for GWDC and AMSGWDC with a diminishing learning rate, and [11], which presented convergence analyses only for Adam and AMSGrad for constant and diminishing learning rates. Using constant learning rates allows Algorithm 1 for nonconvex optimization to have approximately an $\mathcal{O}(1/n)$ convergence rate (see Table 1).

The second contribution of this paper is to show that the algorithm with a constant learning rate tends to be superior for training neural networks, while the one with a diminishing rate tends not to be good for training neural networks. Here, we focus on image classification using several neural networks and show the effectiveness of the algorithm with a constant learning rate (Subsections V-A–V-B). Moreover, we consider unconstrained optimization in [12] and provide numerical comparisons for Algorithm 1 with heuristic intelligent optimization algorithms, such as the genetic algorithm (GA), particle swarm optimization (PSO), biogeography-based optimization (BBO), an Atom search optimization (ASO), and a teaching-learning-based optimization (TLBO). The numerical comparisons show that, in particular, TLBO and Algorithm 1 perform well (Subsection V-C).

This paper is organized as follows. Section II states the main problem. Section III presents the proposed algorithm for solving the main problem and analyzes its convergence. Section IV compares the analyses in Section III with the ones in the previous reports. Section V numerically compares the behaviors of the proposed algorithm with those of the existing ones. Section VI concludes the paper with a brief summary.

II. OPTIMIZATION IN DEEP NEURAL NETWORKS

The notation used in this paper is summarized in Table 2.

In general, an optimization problem in a deep neural network can be expressed as the following nonconvex optimization problem:

Problem II.1 Assume that

(A1) $X \subset \mathbb{R}^d$ is a nonempty, closed convex set onto which

TABLE 2. Notation List

Notation	Description
$\mathbb{N}$	The set of all positive integers and zero
$\mathbb{R}^d$	A d-dimensional Euclidean space with inner product $\langle \cdot, \cdot \rangle$, which induces the norm $\|\cdot\|$
$\mathbb{R}^d_+$	$\mathbb{R}^d_+ := \{\boldsymbol{x} = (x_i) \in \mathbb{R}^d \colon x_i \geq 0 \ (i = 1, 2, \ldots, d)\}$
$\mathbb{R}^d_{++}$	$\mathbb{R}^d_{++} := \{\boldsymbol{x} = (x_i) \in \mathbb{R}^d \colon x_i > 0 \ (i = 1, 2, \ldots, d)\}$
$\mathbb{S}^d$	The set of $d \times d$ symmetric matrices, i.e., $\mathbb{S}^d = \{M \in \mathbb{R}^{d \times d} \colon M = M^\top\}$
$\mathbb{S}^d_{++}$	The set of $d \times d$ symmetric positive-definite matrices, i.e., $\mathbb{S}^d_{++} = \{M \in \mathbb{S}^d \colon M \succ O\}$
$\mathbb{D}^d$	The set of $d \times d$ diagonal matrices, i.e., $\mathbb{D}^d = \{M \in \mathbb{R}^{d \times d} \colon M = \mathsf{diag}(x_i), \ x_i \in \mathbb{R} \ (i = 1, 2, \ldots, d)\}$
$A \odot B$	The Hadamard product of matrices A and B ($\boldsymbol{x} \odot \boldsymbol{x} := (x_i^2) \in \mathbb{R}^d$ ($\boldsymbol{x} := (x_i) \in \mathbb{R}^d$))
$\langle \boldsymbol{x}, \boldsymbol{y} \rangle_H$	The H-inner product of $\mathbb{R}^d$, where $H \in \mathbb{S}^d_{++}$, i.e., $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_H := \langle \boldsymbol{x}, H\boldsymbol{y} \rangle$
$\|\boldsymbol{x}\|_H^2$	The H-norm, where $H \in \mathbb{S}^d_{++}$, i.e., $\|\boldsymbol{x}\|_H^2 := \langle \boldsymbol{x}, H\boldsymbol{x} \rangle$
P_X	The metric projection onto a nonempty, closed convex set X ($\subset \mathbb{R}^d$)
$P_{X,H}$	The metric projection onto X under the H-norm
$\mathbb{E}[Y]$	The expectation of a random variable Y
$\boldsymbol{\xi}$	A random vector whose probability distribution P is supported on a set $\Xi \subset \mathbb{R}^{d_1}$
$F(\cdot, \boldsymbol{\xi})$	A function from $\mathbb{R}^d$ to $\mathbb{R}$ that is continuously differentiable for all $\boldsymbol{\xi} \in \Xi$
f	The objective function defined by $f(\boldsymbol{x}) := \mathbb{E}[F(\boldsymbol{x}, \boldsymbol{\xi})]$ for all $\boldsymbol{x} \in \mathbb{R}^d$
∇f	The gradient of f
$\mathsf{G}(\boldsymbol{x}, \boldsymbol{\xi})$	The stochastic gradient for a given $(\boldsymbol{x}, \boldsymbol{\xi}) \in \mathbb{R}^d \times \Xi$ which satisfies $\mathbb{E}[\mathsf{G}(\boldsymbol{x}, \boldsymbol{\xi})] = \nabla f(\boldsymbol{x})$
$X^\star$	The set of stationary points of the problem of minimizing f over X
$f^\star$	The optimal objective function value for the problem of minimizing f over X
$R(T)$	The regret on a sequence $(\sum_{t=1}^T f_t(\boldsymbol{x}_t))$ defined by $R(T) := \sum_{t=1}^T f_t(\boldsymbol{x}_t) - f^\star$
$y_n = \mathcal{O}(x_n)$	There exist $c \in \mathbb{R}$ and $n_0 \in \mathbb{N}$ such that, for all $n \geq n_0$, $y_n \leq c x_n$, where $(x_n)_{n \in \mathbb{N}}, (y_n)_{n \in \mathbb{N}} \subset \mathbb{R}_+$

the projection can be easily computed;

(A2) $f \colon \mathbb{R}^d \to \mathbb{R}$, which is defined for all $\boldsymbol{x} \in \mathbb{R}^d$ by $f(\boldsymbol{x}) := \mathbb{E}[F(\boldsymbol{x}, \boldsymbol{\xi})]$, is well defined, where $F(\cdot, \boldsymbol{\xi})$ is continuously differentiable for almost every $\boldsymbol{\xi} \in \Xi$, where $\boldsymbol{\xi} \in \Xi$ is a random vector whose probability distribution P is supported on a set $\Xi \subset \mathbb{R}^{d_1}$.

Thus, we would like to find a minimizer of f over X, i.e.,

$$\boldsymbol{x}^\star \in \underset{\boldsymbol{x} \in X}{\operatorname{argmin}} \, f(\boldsymbol{x}).$$

Even if Problem II.1 is deterministic, i.e., F does not depend on $\boldsymbol{\xi}$, the existing algorithms, such as the steepest descent method, Newton method, quasi-Newton methods, and conjugate gradient methods, can find a stationary point for the problem of minimizing f over X. From this fact, we will focus on the following stationary point problem [11] associated with Problem II.1:

Problem II.2 Under (A1) and (A2), we would like to find a stationary point $\boldsymbol{x}^\star$ of Problem II.1, i.e.,

$$\boldsymbol{x}^\star \in X^\star := \{\boldsymbol{x}^\star \in X \colon \langle \boldsymbol{x} - \boldsymbol{x}^\star, \nabla f(\boldsymbol{x}^\star) \rangle \geq 0 \ (\boldsymbol{x} \in X)\}.$$

The relationships between Problems II.1 and II.2 are as follows:

(F1) $\operatorname{argmin}_{\boldsymbol{x} \in X} f(\boldsymbol{x}) \subset X^\star$;

(F2) $\operatorname{argmin}_{\boldsymbol{x} \in X} f(\boldsymbol{x}) \supset X^\star$ when f is convex, i.e., $\operatorname{argmin}_{\boldsymbol{x} \in X} f(\boldsymbol{x}) = X^\star$.

We also have that $X^\star = \{\boldsymbol{x}^\star \in \mathbb{R}^d \colon \nabla f(\boldsymbol{x}^\star) = \boldsymbol{0}\}$ when $X = \mathbb{R}^d$.

We will consider Problem II.2 under the following conditions.

(C1) There is an independent and identically distributed sample $\boldsymbol{\xi}_0, \boldsymbol{\xi}_1, \ldots$ of realizations of the random vector $\boldsymbol{\xi}$;

(C2) There is an oracle which, for a given input point $(\boldsymbol{x}, \boldsymbol{\xi}) \in \mathbb{R}^d \times \Xi$, returns a stochastic gradient $\mathsf{G}(\boldsymbol{x}, \boldsymbol{\xi})$ such that $\mathbb{E}[\mathsf{G}(\boldsymbol{x}, \boldsymbol{\xi})] = \nabla f(\boldsymbol{x})$;

(C3) There exists a positive number M such that, for all $\boldsymbol{x} \in X$, $\mathbb{E}[\|\mathsf{G}(\boldsymbol{x}, \boldsymbol{\xi})\|^2] \leq M^2$.

III. ADAPTIVE-LEARNING-RATE OPTIMIZATION ALGORITHM

Algorithm 1 [11] is a unified algorithm for the existing adaptive-learning-rate optimization algorithms to solve Problem II.2 under (C1)–(C3).

Algorithm 1 Adaptive-learning-rate optimization algorithm for Problem II.2

```
Require: $(\alpha_n)_{n \in \mathbb{N}} \subset (0, 1)$, $(\beta_n)_{n \in \mathbb{N}} \subset [0, 1)$, $\gamma \in [0, 1)$
1: $n \leftarrow 0$, $\boldsymbol{x}_0, \boldsymbol{m}_{-1} \in \mathbb{R}^d$, $\mathsf{H}_0 \in \mathbb{S}^d_{++} \cap \mathbb{D}^d$
2: loop
3:     $\boldsymbol{m}_n := \beta_n \boldsymbol{m}_{n-1} + (1 - \beta_n) \mathsf{G}(\boldsymbol{x}_n, \boldsymbol{\xi}_n)$
4:     $\hat{\boldsymbol{m}}_n := \dfrac{\boldsymbol{m}_n}{1 - \gamma^{n+1}}$
5:     $\mathsf{H}_n \in \mathbb{S}^d_{++} \cap \mathbb{D}^d$
6:     Find $\mathbf{d}_n \in \mathbb{R}^d$ that solves $\mathsf{H}_n \mathbf{d} = -\hat{\boldsymbol{m}}_n$
7:     $\boldsymbol{x}_{n+1} := P_{X, \mathsf{H}_n}(\boldsymbol{x}_n + \alpha_n \mathbf{d}_n)$
8:     $n \leftarrow n + 1$
9: end loop
```

We need the following conditions to analyze Algorithm 1.

Assumption III.1 The sequence $(\mathsf{H}_n)_{n\in\mathbb{N}} \subset \mathbb{S}^d_{++} \cap \mathbb{D}^d$, denoted by $\mathsf{H}_n := \mathsf{diag}(h_{n,i})$, in Algorithm 1 satisfies the following conditions:

(A3) $h_{n+1,i} \geq h_{n,i}$ almost surely for all $n \in \mathbb{N}$ and all $i = 1, 2, \ldots, d$;

(A4) For all $i = 1, 2, \ldots, d$, a positive number B_i exists such that $\sup\{\mathbb{E}[h_{n,i}] \colon n \in \mathbb{N}\} \leq B_i$.

Moreover,

(A5) $D := \max_{i=1,2,\ldots,d} \sup\{(x_i - y_i)^2 \colon (x_i), (y_i) \in X\} < +\infty$.

Assumption (A5) holds under the boundedness condition of X, which is assumed in Theorem 4.1 in [8], [9], (A5) in [11]. We call Algorithm 1 an adaptive-learning-rate optimization algorithm (see Subsection 8.5 in [6]) since Algorithm 1 under (A3) and (A4) reduces to the previous algorithms that assume (A5).

Example III.1

(i) Adam [8]: Let us consider H_n and $\boldsymbol{v}_n$ $(n \in \mathbb{N})$, which are defined for all $n \in \mathbb{N}$ by

$$\begin{aligned} \boldsymbol{v}_n &:= \delta \boldsymbol{v}_{n-1} + (1-\delta)\mathsf{G}(\boldsymbol{x}_n, \boldsymbol{\xi}_n) \odot \mathsf{G}(\boldsymbol{x}_n, \boldsymbol{\xi}_n), \\ \bar{\boldsymbol{v}}_n &:= \frac{\boldsymbol{v}_n}{1-\delta^{n+1}}, \\ \hat{\boldsymbol{v}}_n &= (\hat{v}_{n,i}) := (\max\{\hat{v}_{n-1,i}, \bar{v}_{n,i}\}), \\ \mathsf{H}_n &:= \mathsf{diag}\left(\sqrt{\hat{v}_{n,i}}\right), \end{aligned} \tag{1}$$

where $\boldsymbol{v}_{-1} = \hat{\boldsymbol{v}}_{-1} = \mathbf{0} \in \mathbb{R}^d$ and $\delta \in [0,1)$. H_n and $\boldsymbol{v}_n$ defined by (1) satisfy (A3) and (A4) (see Section III in [11]). Hence, Algorithm 1 with (1) is based on the Adam algorithm.[1]

(ii) AMSGrad [9]: Let us consider H_n and $\boldsymbol{v}_n$ $(n \in \mathbb{N})$, which is defined for all $n \in \mathbb{N}$ by

$$\begin{aligned} \boldsymbol{v}_n &:= \delta \boldsymbol{v}_{n-1} + (1-\delta)\mathsf{G}(\boldsymbol{x}_n, \boldsymbol{\xi}_n) \odot \mathsf{G}(\boldsymbol{x}_n, \boldsymbol{\xi}_n), \\ \hat{\boldsymbol{v}}_n &= (\hat{v}_{n,i}) := (\max\{\hat{v}_{n-1,i}, v_{n,i}\}), \\ \mathsf{H}_n &:= \mathsf{diag}\left(\sqrt{\hat{v}_{n,i}}\right), \end{aligned} \tag{2}$$

where $\boldsymbol{v}_{-1} = \hat{\boldsymbol{v}}_{-1} = \mathbf{0} \in \mathbb{R}^d$ and $\delta \in [0,1)$. H_n and $\boldsymbol{v}_n$, which are defined by (2), satisfy (A3) and (A4) (see Section III in [11]). Algorithm 1 with (2) is the AMSGrad algorithm [9].

(iii) GWDC [2]: Suppose that $(l_n)_{n\in\mathbb{N}} \subset \mathbb{R}_{++}$ is monotone increasing and $(u_n)_{n\in\mathbb{N}} \subset \mathbb{R}_{++}$ is monotone decreasing with $l_n \leq u_n$ for all $n \in \mathbb{N}$. Let us consider H_n and $\boldsymbol{v}_n$ $(n \in \mathbb{N})$, which is defined for all $n \in \mathbb{N}$ by

$$\begin{aligned} \boldsymbol{v}_n &:= \delta \boldsymbol{v}_{n-1} + (1-\delta)\mathsf{G}(\boldsymbol{x}_n, \boldsymbol{\xi}_n) \odot \mathsf{G}(\boldsymbol{x}_n, \boldsymbol{\xi}_n), \\ \hat{\boldsymbol{v}}_n &= (\hat{v}_{n,i}) := \left(\mathrm{Clip}\left(\frac{1}{\sqrt{v_{n,i}}}, l_n, u_n\right)^{-1}\right), \\ \mathsf{H}_n &:= \mathsf{diag}\left(\sqrt{\hat{v}_{n,i}}\right), \end{aligned} \tag{3}$$

where $\boldsymbol{v}_{-1} = \mathbf{0} \in \mathbb{R}^d$, $\delta \in [0,1)$, and $\mathrm{Clip}(\cdot, l, u) \colon \mathbb{R} \to \mathbb{R}$ $(l, u \in \mathbb{R}$ with $l \leq u$ are given) is defined for all $x \in \mathbb{R}$ by

$$\mathrm{Clip}(x, l, u) := \begin{cases} l & \text{if } x < l, \\ x & \text{if } l \leq x \leq u, \\ u & \text{if } x > u. \end{cases}$$

Obviously, H_n and $\boldsymbol{v}_n$, which is defined by (3), satisfy (A3) (see also (13) in [2]). Moreover, we have that, for all $n \in \mathbb{N}$, $l_0 \leq l_n \leq \mathrm{Clip}(1/\sqrt{v_{n,i}}, l_n, u_n) \leq u_n \leq u_0$, which implies that (A4) holds. Algorithm 1 with (3) is the GWDC algorithm (see Algorithm 2 in [2]).

(iv) AMSGWDC [2]: Suppose that $(l_n)_{n\in\mathbb{N}}$ and $(u_n)_{n\in\mathbb{N}}$ satisfy the same conditions as in (iii). Let us consider H_n and $\boldsymbol{v}_n$ $(n \in \mathbb{N})$, which are defined for all $n \in \mathbb{N}$ by

$$\begin{aligned} \boldsymbol{v}_n &:= \delta \boldsymbol{v}_{n-1} + (1-\delta)\mathsf{G}(\boldsymbol{x}_n, \boldsymbol{\xi}_n) \odot \mathsf{G}(\boldsymbol{x}_n, \boldsymbol{\xi}_n), \\ \hat{\boldsymbol{v}}_n &= (\hat{v}_{n,i}) := (\max\{\hat{v}_{n-1,i}, v_{n,i}\}), \\ \tilde{\boldsymbol{v}}_n &= (\tilde{v}_{n,i}) := \left(\mathrm{Clip}\left(\frac{1}{\sqrt{\hat{v}_{n,i}}}, l_n, u_n\right)^{-1}\right), \\ \mathsf{H}_n &:= \mathsf{diag}\left(\sqrt{\tilde{v}_{n,i}}\right), \end{aligned} \tag{4}$$

where $\boldsymbol{v}_{-1} = \hat{\boldsymbol{v}}_{-1} = \mathbf{0} \in \mathbb{R}^d$ and $\delta \in [0,1)$. From Example III.1(ii) and (iii), H_n and $\boldsymbol{v}_n$, which are defined by (4), satisfy (A3) and (A4). Algorithm 1 with (4) is the AMSGWDC algorithm (see Algorithm 3 in [2]).

A. CONVERGENCE ANALYSIS OF ALGORITHM 1 WITH A CONSTANT LEARNING RATE

The following is the convergence analysis of Algorithm 1 with a constant learning rate. The proof of Theorem III.1 is given in the proof of Theorem 1 in [11].

Theorem III.1 Suppose that (A1)–(A5) and (C1)–(C3) hold and $(\boldsymbol{x}_n)_{n\in\mathbb{N}}$ is the sequence generated by Algorithm 1 with $\alpha_n := \alpha$ and $\beta_n := \beta$ $(n \in \mathbb{N})$. Then, for all $\boldsymbol{x} \in X$,

$$\begin{aligned} &\limsup_{n\to+\infty} \mathbb{E}\left[\langle \boldsymbol{x} - \boldsymbol{x}_n, \nabla f(\boldsymbol{x}_n)\rangle\right] \geq -\frac{\tilde{B}^2 \tilde{M}^2}{2\tilde{b}\tilde{\gamma}^2}\alpha - \frac{\tilde{M}\sqrt{Dd}}{\tilde{b}\tilde{\gamma}}\beta, \\ &\max_{k=1,2,\ldots,n} \mathbb{E}\left[\langle \boldsymbol{x} - \boldsymbol{x}_n, \nabla f(\boldsymbol{x}_n)\rangle\right] \geq -\mathcal{O}\left(\frac{1}{n}\right) - C_1\alpha - C_2\beta, \end{aligned}$$

where $\tilde{\gamma} := 1 - \gamma$, $\tilde{b} := 1 - \beta$, $\tilde{M}^2 := \max\{\|\boldsymbol{m}_{-1}\|^2, M^2\}$, D is defined as in (A5), $\tilde{B} := \sup\{\max_{i=1,2,\ldots,d} h_{n,i}^{-1/2} \colon n \in \mathbb{N}\} < +\infty$, and C_i $(i = 1, 2)$ are positive constants that are independent of n, α, and β.

The following proposition (see also [11, Proposition 1]) enables us to compare Algorithm 1 with Adam [8], AMSGrad [9], GWDC [2], and AMSGWDC [2].

Proposition III.1 Suppose that (A1)–(A5) and (C1)–(C3) hold, $F(\cdot, \boldsymbol{\xi})$ is convex for almost every $\boldsymbol{\xi} \in \Xi$, and $(\boldsymbol{x}_n)_{n\in\mathbb{N}}$

[1] Adam uses $\mathsf{H}_n = \mathsf{diag}(\bar{v}_{n,i}^{1/2})$. We use $\hat{\boldsymbol{v}}_n = (\hat{v}_{n,i}) := (\max\{\hat{v}_{n-1,i}, \bar{v}_{n,i}\})$ in (1) so as to satisfy (A3). The modification of H_n defined by $\mathsf{diag}(\hat{v}_{n,i}^{1/2} + \epsilon)$ guarantees that $h_{n,i} \neq 0$, where $\epsilon > 0$ [8].

is the sequence generated by Algorithm 1 with $\alpha_n := \alpha$ and $\beta_n := \beta$ $(n \in \mathbb{N})$. Then,

$$\liminf_{n \to +\infty} \mathbb{E}\left[f(\boldsymbol{x}_n) - f^\star\right] \leq \frac{\tilde{B}^2 \tilde{M}^2}{2\tilde{b}\tilde{\gamma}^2}\alpha + \frac{\tilde{M}\sqrt{Dd}}{\tilde{b}\tilde{\gamma}}\beta,$$

where $f^\star$ denotes the optimal value of Problem II.1 (see (F2)), and $\tilde{\gamma}$, $\tilde{b}$, $\tilde{M}$, D, and $\tilde{B}$ are defined as in Theorem III.1. Additionally, for the regret on a sequence of $F(\cdot, t) := f_t(\cdot)$ $(t = 1, 2, \ldots, T)$, Algorithm 1 satisfies

$$\frac{R(T)}{T} \leq \mathcal{O}\left(\frac{1}{T}\right) + C_1 \alpha + C_2 \beta.$$

B. CONVERGENCE ANALYSIS OF ALGORITHM 1 WITH A DIMINISHING LEARNING RATE

The following is a convergence analysis of Algorithm 1 with a diminishing sub-learning rate. The proof of Theorem III.2 is given in the proof of Theorem 2 in [11].

Theorem III.2 Suppose that (A1)–(A5) and (C1)–(C3) hold and $(\boldsymbol{x}_n)_{n\in\mathbb{N}}$ is the sequence generated by Algorithm 1 with α_n and β_n $(n \in \mathbb{N})$ satisfying $\sum_{n=0}^{+\infty} \alpha_n = +\infty$, $\sum_{n=0}^{+\infty} \alpha_n^2 < +\infty$, and $\sum_{n=0}^{+\infty} \alpha_n \beta_n < +\infty$. Then, for all $\boldsymbol{x} \in X$,

$$\limsup_{n \to +\infty} \mathbb{E}\left[\langle \boldsymbol{x} - \boldsymbol{x}_n, \nabla f(\boldsymbol{x}_n)\rangle\right] \geq 0.$$

Moreover, if $\alpha_n := 1/n^\eta$ $(\eta \in [1/2, 1))$ and if $\beta_n := \lambda^n$ $(\lambda \in (0,1))$, then Algorithm 1 achieves the following convergence rate:

$$\max_{k=1,2,\ldots,n} \mathbb{E}\left[\langle \boldsymbol{x} - \boldsymbol{x}_k, \nabla f(\boldsymbol{x}_k)\rangle\right] \geq -\mathcal{O}\left(\frac{1}{n^{1-\eta}}\right).$$

The following proposition (see also [11, Proposition 2]) enables us to compare Algorithm 1 with Adam [8], AMSGrad [9], GWDC [2], and AMSGWDC [2].

Proposition III.2 Suppose that (A1)–(A5) and (C1)–(C3) hold, $F(\cdot, \boldsymbol{\xi})$ is convex for almost every $\boldsymbol{\xi} \in \Xi$, and $(\boldsymbol{x}_n)_{n\in\mathbb{N}}$ is the sequence generated by Algorithm 1 with $\alpha_n := 1/n^\eta$ and $\beta_n := \lambda^n$ $(n \in \mathbb{N})$, where $\eta \in [1/2, 1]$ and $\lambda \in (0,1)$. Then, under $\eta \in (1/2, 1]$,

$$\liminf_{n \to +\infty} \mathbb{E}\left[f(\boldsymbol{x}_n) - f^\star\right] = 0,$$

where $f^\star$ denotes the optimal value of the problem of minimizing f over X. Moreover, under $\eta \in [1/2, 1)$, any accumulation point of $(\tilde{\boldsymbol{x}}_n)_{n\in\mathbb{N}}$ defined by $\tilde{\boldsymbol{x}}_n := (1/n)\sum_{k=1}^n \boldsymbol{x}_k$ almost surely belongs to $X^\star = \operatorname{argmin}_{\boldsymbol{x}\in X} f(\boldsymbol{x})$, and Algorithm 1 achieves the following convergence rate:

$$\mathbb{E}\left[f(\tilde{\boldsymbol{x}}_n) - f^\star\right] = \mathcal{O}\left(\frac{1}{n^{1-\eta}}\right).$$

Additionally, for the regret on a sequence of $F(\cdot, t) := f_t(\cdot)$ $(t = 1, 2, \ldots, T)$, Algorithm 1 satisfies

$$\frac{R(T)}{T} \leq \mathcal{O}\left(\frac{1}{T^{1-\eta}}\right).$$

IV. COMPARISON OF OUR CONVERGENCE ANALYSES WITH CONVERGENCE ANALYSES ON EXISTING ALGORITHMS

To compare the previously reported results in [2], [8], [9] with the results in Section III, we will consider Problem II.1 when $f_t(\cdot) := F(\cdot, t)$ $(t = 1, 2, \ldots, T)$ is convex (see, e.g., [2, p.110932]), i.e.,

$$\text{Find } \boldsymbol{x}^\star \in \operatorname*{argmin}_{\boldsymbol{x}\in X} \underbrace{\sum_{t=1}^T f_t(\boldsymbol{x})}_{Tf(\boldsymbol{x})}. \tag{5}$$

See [11, Table II] and Table 1 for comparisons of SGD [4], AMSGrad [13], AdaBelief [10], and Algorithm 1 for nonconvex optimization. Since f is convex, (F2) implies that $X^\star = \operatorname{argmin}_{\boldsymbol{x}\in X} \sum_{t=1}^T f_t(\boldsymbol{x})$. The performance measure used for the previously reported results in [2], [8], [9] is the regret $R(T)$.

A. COMPARISON OF ADAM WITH ALGORITHM 1 IN THE CASE OF EXAMPLE III.1(I)

Theorem 4.1 in [8] indicates that Adam with $\alpha_n = \mathcal{O}(1/\sqrt{n})$ and $\beta_n = \lambda^n$ $(\lambda \in (0,1))$ ensures that there exists a positive real number D such that

$$\frac{R(T)}{T} = \frac{1}{T}\sum_{t=1}^T f_t(\boldsymbol{x}_t) - \frac{f^\star}{T} \leq \frac{D}{\sqrt{T}}. \tag{6}$$

Unfortunately, Theorem 1 in [9] shows that a counter-example to Theorem 4.1 in [8] exists. Meanwhile, Algorithm 1 with (1) resembles the Adam algorithm (see Footnote 1 for details). Proposition III.2 indicates that Algorithm 1 with (1), $\eta = 1/2$, and $\beta_n = \lambda^n$ satisfies

$$\mathbb{E}\left[f\left(\frac{1}{T}\sum_{t=1}^T \boldsymbol{x}_t\right) - f^\star\right] = \mathcal{O}\left(\frac{1}{\sqrt{T}}\right),$$
$$\frac{R(T)}{T} \leq \mathcal{O}\left(\frac{1}{\sqrt{T}}\right).$$

This implies that Algorithm 1 based on Adam achieves an $\mathcal{O}(1/\sqrt{T})$ convergence rate, which could not be done using the analysis in [8].

B. COMPARISON OF AMSGRAD WITH ALGORITHM 1 IN THE CASE OF EXAMPLE III.1(II)

Theorem 4 in [9] indicates that AMSGrad with $\alpha_n = \mathcal{O}(1/\sqrt{n})$ and $\beta_n = \lambda^n$ $(\lambda \in (0,1))$ ensures that there exists a positive real number $\hat{D}$ such that

$$\frac{R(T)}{T} = \frac{1}{T}\sum_{t=1}^T f_t(\boldsymbol{x}_t) - \frac{f^\star}{T} \leq \hat{D}\sqrt{\frac{1 + \ln T}{T}}. \tag{7}$$

Example III.1(ii) shows that Algorithm 1 with (2) coincides with AMSGrad. Proposition III.2 indicates that Algorithm 1 with (2), $\eta = 1/2$, and $\beta_n = \lambda^n$ (i.e., AMSGrad) satisfies

$$\mathbb{E}\left[f\left(\frac{1}{T}\sum_{t=1}^{T} \boldsymbol{x}_t\right) - f^\star \right] = \mathcal{O}\left(\frac{1}{\sqrt{T}}\right),$$
$$\frac{R(T)}{T} \leq \mathcal{O}\left(\frac{1}{\sqrt{T}}\right).$$

This implies that AMSGrad can achieve an $\mathcal{O}(1/\sqrt{T})$ convergence rate, which is better than $\mathcal{O}(\sqrt{(1+\ln T)/T})$ (see (7)).

C. COMPARISON OF GWDC WITH ALGORITHM 1 IN THE CASE OF EXAMPLE III.1(III)

Inequality (15) in [2] indicates that GWDC with $\alpha_n = \mathcal{O}(1/\sqrt{n})$ and $\beta_n = \hat{\beta} e^{-\tilde{\beta} n}$ ($\hat{\beta} \in \{0.9, 0.99\}, \tilde{\beta} \in (0,1)$) ensures that there exists a positive real number $\tilde{D}$ such that

$$\frac{R(T)}{T} = \frac{1}{T}\sum_{t=1}^{T} f_t(\boldsymbol{x}_t) - \frac{f^\star}{T} \leq \frac{\tilde{D}}{\sqrt{T}}. \tag{8}$$

Example III.1(iii) shows that Algorithm 1 with (3) coincides with GWDC. Proposition III.2 indicates that Algorithm 1 with (3), $\eta = 1/2$, and $\beta_n = \hat{\beta} e^{-\tilde{\beta} n}$ (i.e., GWDC) satisfies

$$\mathbb{E}\left[f\left(\frac{1}{T}\sum_{t=1}^{T} \boldsymbol{x}_t\right) - f^\star \right] = \mathcal{O}\left(\frac{1}{\sqrt{T}}\right),$$
$$\frac{R(T)}{T} \leq \mathcal{O}\left(\frac{1}{\sqrt{T}}\right).$$

D. COMPARISON OF AMSGWDC WITH ALGORITHM 1 IN THE CASE OF EXAMPLE III.1(IV)

Appendix B in [2] indicates that AMSGWDC with $\alpha_n = \mathcal{O}(1/\sqrt{n})$ and $\beta_n = \hat{\beta} e^{-\tilde{\beta} n}$ ($\hat{\beta} \in \{0.9, 0.99\}, \tilde{\beta} \in (0,1)$) ensures that there exists a positive real number $\bar{D}$ such that

$$\frac{R(T)}{T} = \frac{1}{T}\sum_{t=1}^{T} f_t(\boldsymbol{x}_t) - \frac{f^\star}{T} \leq \frac{\bar{D}}{\sqrt{T}}. \tag{9}$$

Example III.1(iv) shows that Algorithm 1 with (4) coincides with AMSGWDC. Proposition III.2 indicates that Algorithm 1 with (4), $\eta = 1/2$, and $\beta_n = \hat{\beta} e^{-\tilde{\beta} n}$ (i.e., AMSGWDC) satisfies

$$\mathbb{E}\left[f\left(\frac{1}{T}\sum_{t=1}^{T} \boldsymbol{x}_t\right) - f^\star \right] = \mathcal{O}\left(\frac{1}{\sqrt{T}}\right),$$
$$\frac{R(T)}{T} \leq \mathcal{O}\left(\frac{1}{\sqrt{T}}\right).$$

E. DISCUSSION

First of all, we would like to emphasize that regret minimization does not always lead to solutions of problem (5) (see also [14]). This is because, even if $(\boldsymbol{x}_t)_{t=1}^T$ is satisfied, for a sufficiently large number T,

$$\frac{R(T)}{T} = \frac{1}{T}\sum_{t=1}^{T} f_t(\boldsymbol{x}_t) - \frac{f^\star}{T} \approx 0,$$

we do not have that

$$f(\boldsymbol{x}_T) - \frac{f^\star}{T} = \frac{1}{T}\sum_{t=1}^{T} f_t(\boldsymbol{x}_T) - \frac{f^\star}{T} \approx 0.$$

Accordingly, from only (6), (7), (8), and (9), we cannot evaluate whether or not the output $\boldsymbol{x}_T$ generated by the existing adaptive-learning-rate optimization algorithms approximates the solution of problem (5). Meanwhile, Proposition III.2 and Subsections IV-A, IV-B, IV-C, and IV-D lead to the finding that Algorithm 1, including Adam, AMSGrad, GWDC, and AMSGWDC, ensures that, under $\eta \in (1/2, 1]$,

$$\liminf_{n \to +\infty} \mathbb{E}\left[\sum_{t=1}^{T} f_t(\boldsymbol{x}_n) - f^\star \right] = 0 \tag{10}$$

and that, under $\eta \in [1/2, 1)$,

$$\mathbb{E}\left[\sum_{t=1}^{T} f_t\left(\frac{1}{T}\sum_{t=1}^{T} \boldsymbol{x}_t\right) - f^\star \right] = \mathcal{O}\left(\frac{1}{\sqrt{T}}\right). \tag{11}$$

The results in (10) and (11) imply that Algorithm 1 can solve problem (5), in contrast to (6), (7), (8), and (9) that are results for regret minimization.

Moreover, we would like to emphasize that Algorithm 1 with a constant learning rate can approximate the solution of problem (5) in the sense of the result in Proposition III.1, i.e.,

$$\liminf_{n \to +\infty} \mathbb{E}\left[\sum_{t=1}^{T} f_t(\boldsymbol{x}_n) - f^\star \right] \leq T\left(\frac{\tilde{B}^2 \tilde{M}^2}{2\tilde{b}\tilde{\gamma}^2}\alpha + \frac{\tilde{M}\sqrt{Dd}}{\tilde{b}\tilde{\gamma}}\beta \right).$$

The above result indicates that using small constant learning rates would be a good way to solve problem (5). Proposition III.2 shows that any accumulation point of $\tilde{\boldsymbol{x}}_n := (1/n)\sum_{k=1}^{n} \boldsymbol{x}_k$ belongs to $\mathrm{argmin}_{\boldsymbol{x} \in X} \sum_{t=1}^{T} f_t(\boldsymbol{x})$. Accordingly, using a diminishing learning rate is a more robust way to solve problem (5) than using a constant learning rate in theory, as shown in [11], [14], [15]. However, Algorithm 1 with a diminishing learning rate might not work for a rather large number N of iterations, since step 7 in Algorithm 1 with $\alpha_N \approx 0$ satisfies

$$\boldsymbol{x}_{N+1} = P_{X, \mathsf{H}_N}(\boldsymbol{x}_N + \alpha_N \mathbf{d}_N) \approx P_{X, \mathsf{H}_N}(\boldsymbol{x}_N) = \boldsymbol{x}_N.$$

This implies that using a diminishing learning rate would not be good in practice. The next section shows that using a constant learning rate tends to be superior to using a diminishing one. This tendency was also observed in [11], [14], [15].

V. NUMERICAL EXPERIMENTS

We examined the behavior of Algorithm 1 with different learning rates. The adaptive-learning-rate optimization algorithms with $\delta = 0.999$ [8], [9] used in the experiments were as follows, where the initial points initialized automatically by PyTorch were used and l_n and u_n were based on [16, Section 4].

Algorithm 1 with constant learning rates:

- ADAM-C1: Algorithm 1 with (1), $\gamma = 0.9$, $\alpha_n = 10^{-3}$, and $\beta_n = 0.9$
- ADAM-C2: Algorithm 1 with (1), $\gamma = 0.9$, $\alpha_n = 10^{-3}$, and $\beta_n = 10^{-3}$
- ADAM-C3: Algorithm 1 with (1), $\gamma = 0.9$, $\alpha_n = 10^{-2}$, and $\beta_n = 10^{-2}$
- AMSG-C1: Algorithm 1 with (2), $\gamma = 0$, $\alpha_n = 10^{-3}$, and $\beta_n = 0.9$
- AMSG-C2: Algorithm 1 with (2), $\gamma = 0$, $\alpha_n = 10^{-3}$, and $\beta_n = 10^{-3}$
- AMSG-C3: Algorithm 1 with (2), $\gamma = 0$, $\alpha_n = 10^{-2}$, and $\beta_n = 10^{-2}$
- GWDC-C1: Algorithm 1 with (3), $\gamma = 0.9$, $l_n := 0.1 - 0.1/((1-\delta)n+1)$, $u_n := 0.1 + 0.1/(1-\delta)n$, $\alpha_n = 10^{-3}$, and $\beta_n = 0.9$
- GWDC-C2: Algorithm 1 with (3), $\gamma = 0.9$, $l_n := 0.1 - 0.1/((1-\delta)n+1)$, $u_n := 0.1 + 0.1/(1-\delta)n$, $\alpha_n = 10^{-3}$, and $\beta_n = 10^{-3}$
- GWDC-C3: Algorithm 1 with (3), $\gamma = 0.9$, $l_n := 0.1 - 0.1/((1-\delta)n+1)$, $u_n := 0.1 + 0.1/(1-\delta)n$, $\alpha_n = 10^{-2}$, and $\beta_n = 10^{-2}$
- AMSGWDC-C1: Algorithm 1 with (4), $\gamma = 0$, $l_n := 0.1 - 0.1/((1-\delta)n+1)$, $u_n := 0.1 + 0.1/(1-\delta)n$, $\alpha_n = 10^{-3}$, and $\beta_n = 0.9$
- AMSGWDC-C2: Algorithm 1 with (4), $\gamma = 0$, $l_n := 0.1 - 0.1/((1-\delta)n+1)$, $u_n := 0.1 + 0.1/(1-\delta)n$, $\alpha_n = 10^{-3}$, and $\beta_n = 10^{-3}$
- AMSGWDC-C3: Algorithm 1 with (4), $\gamma = 0$, $l_n := 0.1 - 0.1/((1-\delta)n+1)$, $u_n := 0.1 + 0.1/(1-\delta)n$, $\alpha_n = 10^{-2}$, and $\beta_n = 10^{-2}$

Algorithm 1 with a diminishing learning rate:

- ADAM-D1 [8]: Algorithm 1 with (1), $\gamma = 0.9$, $\alpha_n = 1/\sqrt{n}$, and $\beta_n = 1/2^n$
- ADAM-D2: Algorithm 1 with (1), $\gamma = 0.9$, $\alpha_n = 1/n^{3/4}$, and $\beta_n = 1/2^n$
- ADAM-D3: Algorithm 1 with (1), $\gamma = 0.9$, $\alpha_n = 1/n$, and $\beta_n = 1/2^n$
- AMSG-D1 [9]: Algorithm 1 with (2), $\gamma = 0$, $\alpha_n = 1/\sqrt{n}$, and $\beta_n = 1/2^n$
- AMSG-D2: Algorithm 1 with (2), $\gamma = 0$, $\alpha_n = 1/n^{3/4}$, and $\beta_n = 1/2^n$
- AMSG-D3: Algorithm 1 with (2), $\gamma = 0$, $\alpha_n = 1/n$, and $\beta_n = 1/2^n$
- GWDC-D1 [2]: Algorithm 1 with (3), $\gamma = 0.9$, $l_n := 0.1 - 0.1/((1-\delta)n+1)$, $u_n := 0.1 + 0.1/(1-\delta)n$, $\alpha_n = 1/\sqrt{n}$, and $\beta_n = 0.9e^{-n/10^4}$
- GWDC-D2: Algorithm 1 with (3), $\gamma = 0.9$, $l_n := 0.1 - 0.1/((1-\delta)n+1)$, $u_n := 0.1 + 0.1/(1-\delta)n$, $\alpha_n = 1/n^{3/4}$, and $\beta_n = 1/2^n$
- GWDC-D3: Algorithm 1 with (3), $\gamma = 0.9$, $l_n := 0.1 - 0.1/((1-\delta)n+1)$, $u_n := 0.1+0.1/(1-\delta)n$, $\alpha_n = 1/n$, and $\beta_n = 1/2^n$
- AMSGWDC-D1 [2]: Algorithm 1 with (4), $\gamma = 0$, $l_n := 0.1 - 0.1/((1-\delta)n+1)$, $u_n := 0.1 + 0.1/(1-\delta)n$, $\alpha_n = 1/\sqrt{n}$, and $\beta_n = 0.9e^{-n/10^4}$
- AMSGWDC-D2: Algorithm 1 with (4), $\gamma = 0$, $l_n := 0.1 - 0.1/((1-\delta)n+1)$, $u_n := 0.1 + 0.1/(1-\delta)n$, $\alpha_n = 1/n^{3/4}$, and $\beta_n = 1/2^n$
- AMSGWDC-D3: Algorithm 1 with (4), $\gamma = 0$, $l_n := 0.1 - 0.1/((1-\delta)n+1)$, $u_n := 0.1 + 0.1/(1-\delta)n$, $\alpha_n = 1/n$, and $\beta_n = 1/2^n$

ADAM-C1, AMSG-C1, GWDC-C1, and AMSGWDC-C1 are based on the numerical results sections in [2], [8], and [9]. We implemented ADAM-Ci, AMSG-Ci, GWDC-Ci, and AMSGWDC-Ci ($i = 2, 3$) so that we could compare ADAM-C1, AMSG-C1, GWDC-C1, and AMSGWDC-C1 with the proposed algorithms with small constant learning rates, which are based on Theorem III.1 and Proposition III.1.

ADAM-D1, AMSG-D1, GWDC-D1, and AMSGWDC-D1 are based on the previous results in [2], [8], and [9] (see also (6), (7), (8), and (9)). To check how the algorithms work with different learning rates, we implemented ADAM-Di, AMSG-Di, GWDC-Di, and AMSGWDC-Di ($i = 2, 3$).

The experiments used a fast scalar computation server at Meiji University. The environment has two Intel(R) Xeon(R) Gold 6148 (2.4 GHz, 20 cores) CPUs, an NVIDIA Tesla V100 (16 GB, 900 Gbps) GPU and a Red Hat Enterprise Linux 7.6 operating system. The experimental code was written in Python 3.8.2, and we used the NumPy 1.17.3 package and PyTorch 1.3.0 package.

A. FEEDFORWARD NEURAL NETWORK MODELS

First, we experimented with two feedforward neural network models, a single-layer perceptron and a double-layer perceptron, having different hidden layers for image classification. We used the MNIST dataset, which is a multi-class handwritten digits dataset (0–9), collected by the National Institute of Standards and Technology (NIST). The training data were gathered from Census Bureau employees and high-school students. The training data contains 60,000 grayscale images (28×28), and the test data contains 10,000 grayscale images (28×28).

1) Single-layer perceptron

Figures 1 and 2 indicate that Algorithm 1 with a constant learning rate tended to perform better than Algorithm 1 with a diminishing one in terms of training and test accuracy score. Moreover, Figure 2 shows that ADAM-C1 and ADAM-C2 minimized the training loss function faster than the other algorithms. Meanwhile, Figure 4 shows that GWDC-C1, AMSGWDC-C1, and AMSGWDC-C2 minimized the test loss function faster than the other algorithms. Figures 4 and 8 also show that the algorithm with a constant learning rate outperformed the one with a diminishing rate.

2) Double-layer perceptron

Figures 9,11 and Figures13,15 indicate that Algorithm 1 with a constant learning rate tended to perform better than Algorithm 1 with a diminishing one in terms of training and test accuracy score. Figure 10 shows that AMSG-C1 and AMSG-C2 minimized the training loss function faster than the other

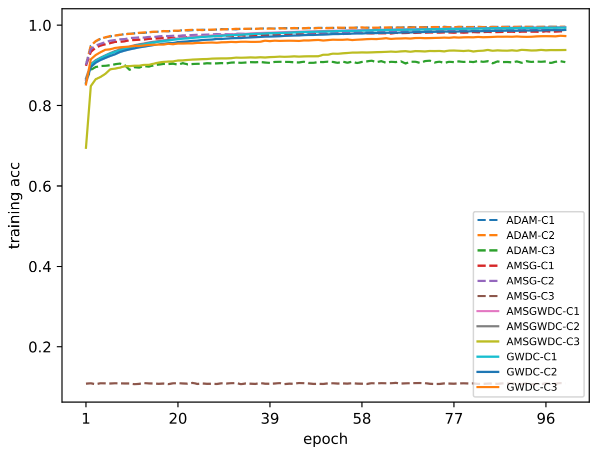


FIGURE 1. Training classification accuracy score for Algorithm 1 with a constant learning rate versus the number of epochs on the MNIST dataset using a single-layer perceptron

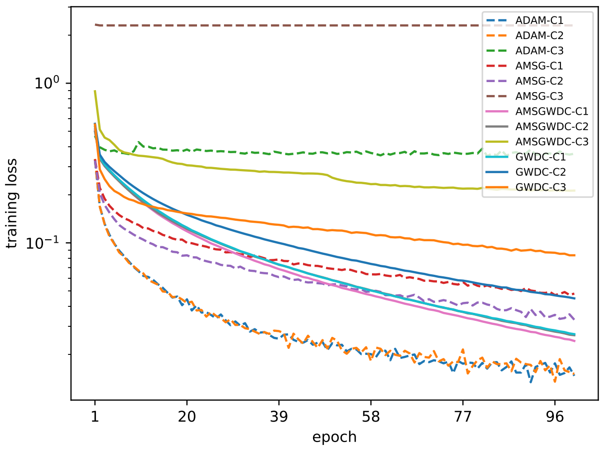


FIGURE 2. Training loss function value for Algorithm 1 with a constant learning rate versus the number of epochs on the MNIST dataset using a single-layer perceptron

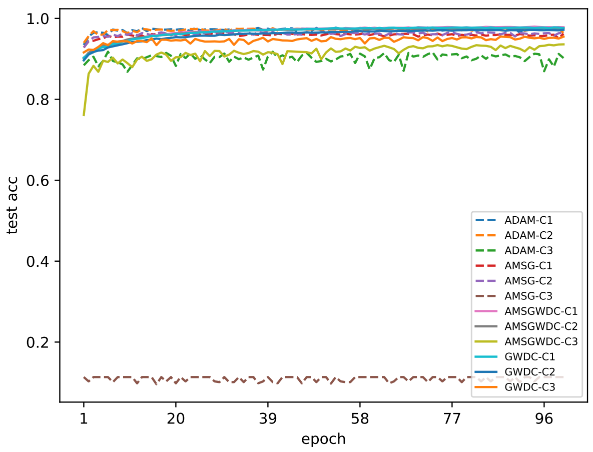


FIGURE 3. Test classification accuracy score for Algorithm 1 with a constant learning rate versus the number of epochs on the MNIST dataset using a single-layer perceptron

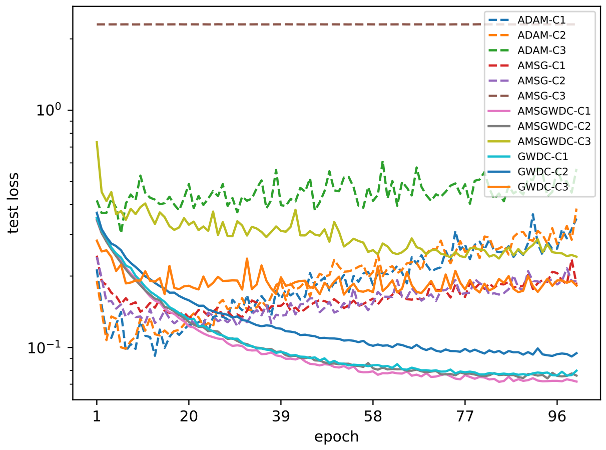


FIGURE 4. Test loss function value for Algorithm 1 with a constant learning rate versus the number of epochs on the MNIST dataset using a single-layer perceptron

algorithms. On the other hand, Figure 12 shows that GWDC-C1, GWDC-C2, and AMSGWDC-C2 performed well. We can see that GWDC and AMGWDC with constant learning rates were superior at training the neural network. Figures 12 and 16 also show that the algorithm with a constant learning rate outperformed the one with a diminishing rate.

B. CONVOLUTIONAL NEURAL NETWORK MODELS

Next, we experimented with a dense convolutional network (DenseNet) and residual network (ResNet); both are relatively deep models based on convolution neural networks (CNN) for image classification. We used the CIFAR-10 dataset, which is a benchmark for image classification. The dataset is a collection of color image data collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. It includes ten categories of images, each of which has 6,000 color images (32×32). There are 50,000 training images and 10,000 test images.

1) DenseNet

Figures 17,19 and Figures 21,23 indicate that Algorithm 1 with a constant learning rate outperformed Algorithm 1 with a diminishing one in terms of the training and test accuracy score. In particular, Algorithm 1 with a constant learning rate had high accuracies. Figures 18 and 22 show that AMSG-C1 and AMSG-D2 minimized the training loss function faster than the other algorithms, and Figures 20 and 24 show that all algorithms except for GWDC-C2, ADAM-D1, and AMSG-D1 minimized the test loss function.

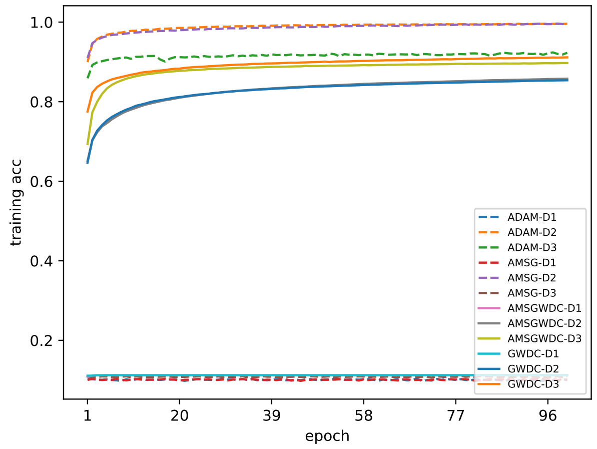


FIGURE 5. Training classification accuracy score for Algorithm 1 with a diminishing learning rate versus the number of epochs on the MNIST dataset using a single-layer perceptron

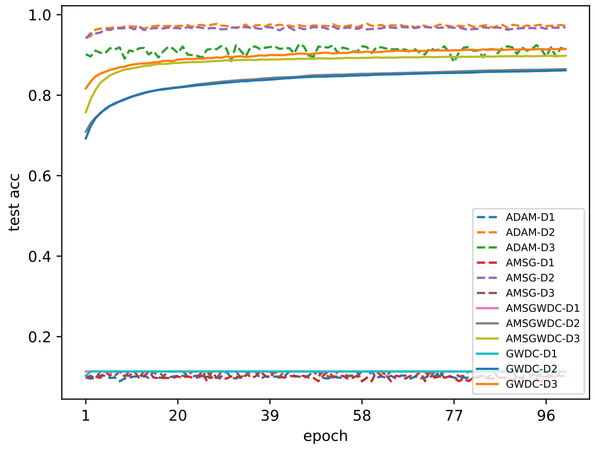


FIGURE 7. Test classification accuracy score for Algorithm 1 with a diminishing learning rate versus the number of epochs on the MNIST dataset using a single-layer perceptron

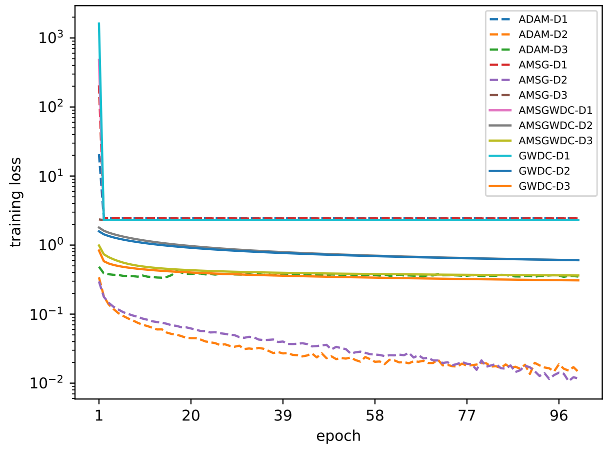


FIGURE 6. Training loss function value for Algorithm 1 with a diminishing learning rate versus the number of epochs on the MNIST dataset using a single-layer perceptron

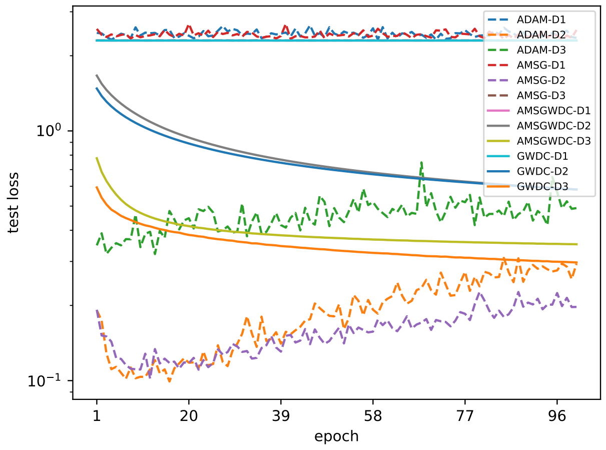


FIGURE 8. Test loss function value for Algorithm 1 with a diminishing learning rate versus the number of epochs on the MNIST dataset using a single-layer perceptron

2) ResNet

Figures 25,27 and Figures 29,31 indicate that Algorithm 1 with a constant learning rate outperformed Algorithm 1 with a diminishing one in terms of the training and test accuracy score. In particular, ADAM-D1 and AMSG-D1 did not work and had low accuracies. Figure 28 shows that AMSG-C2, AMSG-C3, and GWDC-C1 minimized the training loss function, while Figure 30 shows that ADAM-D2, ADAM-D3, AMSG-D2, and AMSG-D3 minimized the test loss function faster than other algorithms, such as GWDC-Di and AMSGWDC-Di ($i = 1, 2, 3$). Meanwhile, Figure 32 shows that all algorithms except for ADAM-D1 and AMSG-D1 minimized the test loss function, and Figure 28 and Figure 32 show that the algorithm with a constant learning rate performed better than the one with a diminishing rate.

C. UNCONSTRAINED OPTIMIZATION USING BENCHMARK FUNCTIONS

We performed optimizations for the seven unimodal benchmark functions in Table 3 [12, Table 2] and compared Algorithm 1 with heuristic intelligent optimization methods[2], as follows:

- GA[3]: Genetic algorithm, where the population size was 100, the probability of performing crossover was 0.95, the probability of mutation was 0.025, and the maximum iteration number was 500.
- PSO [17]: Particle swarm optimization algorithm, where the population size was 100, the acceleration constant

[2]https://github.com/thieu1995/mealpy

[3]https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_quick_guide.htm

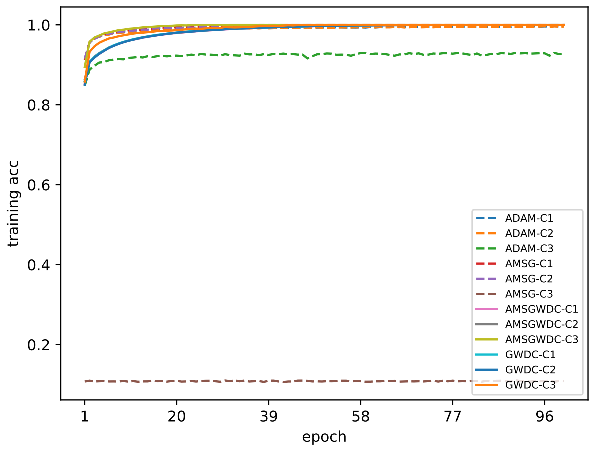


FIGURE 9. Training classification accuracy score for Algorithm 1 with a constant learning rate versus the number of epochs on the MNIST dataset using a double-layer perceptron

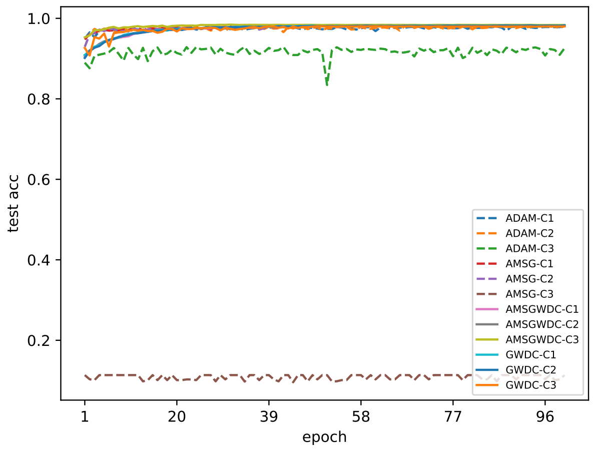


FIGURE 11. Test classification accuracy score for Algorithm 1 with a constant learning rate versus the number of epochs on the MNIST dataset using a double-layer perceptron

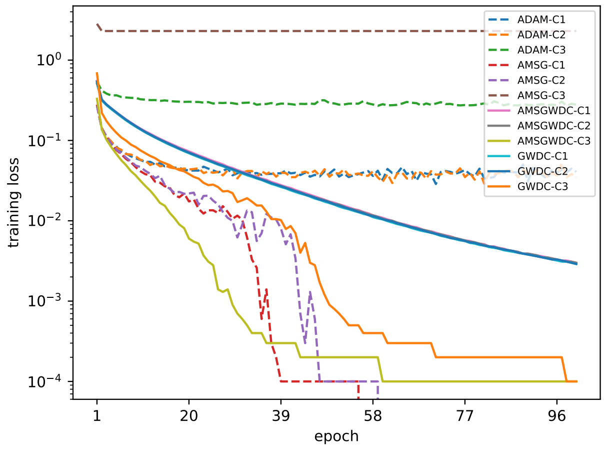


FIGURE 10. Training loss function value for Algorithm 1 with a constant learning rate versus the number of epochs on the MNIST dataset using a double-layer perceptron

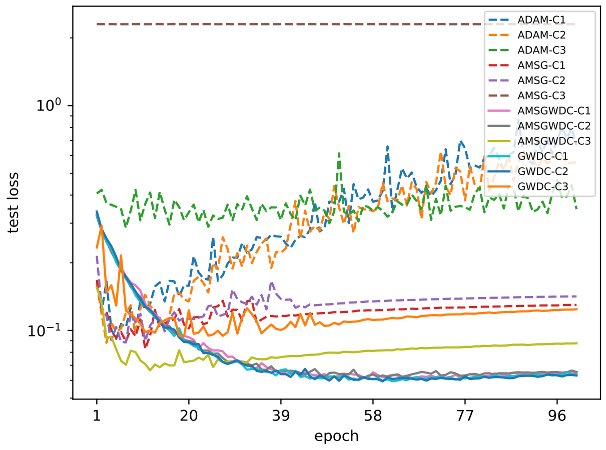


FIGURE 12. Test loss function value for Algorithm 1 with a constant learning rate versus the number of epochs on the MNIST dataset using a double-layer perceptron

was 1.2, and the maximum iteration number was 500.

- BBO [18]: Biogeography-based optimization algorithm, where the population size was 100, the mutation probability was 0.01, the number of elites was 2, and the maximum iteration number was 500.
- ASO [19]: Atom search optimization algorithm, where the population size was 100, the depth weight was 50, the multiplier weight was 0.2, and the maximum iteration number was 500.
- TLBO [20]: Teaching-learning-based optimization algorithm, where the population size was 100 and the maximum iteration number was 500.

The stopping condition for all the algorithms was $n = 500$.

Figures 33~39 plots the values of the Sphere, Schwefel2.22, Schwefel1.2, Schwefel2.21, Resonbrock, Step, and Quartic benchmark functions for Algorithm 1 using constant learning rates with heuristic intelligent optimization methods (GA, PSO, BBO, ASO, and TLBO) versus elapsed time. These figures show that all the algorithms minimized the seven benchmark functions. For example, TLBO, AMSGWDC-C3, and GWDC-C3 performed well for Sphere and Schwefel1.2, while TLBO converged to a solution faster than Algorithm 1 for Schwefel2.22. In contrast, in the case of the Resonbrock benchmark function, TLBO still had not converged to the solution when the stopping condition was reached, but AMSGWDC-C3 and GWDC-C3 had converged to a solution.

Figures 40~46 plots the values of Sphere, Schwefel2.22, Schwefel1.2, Schwefel2.21, Resonbrock, Step, and Quartic benchmark functions for Algorithm 1 using diminishing

TABLE 3. Unimodal benchmark functions [12, Table 2]

Name	Benchmark Function	Dimension	Range	$f_{\min}$
Sphere	$f_1(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$	50	$x_i \in [-100, 100]$	0
Schwefel2.22	$f_2(\boldsymbol{x}) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	50	$x_i \in [-10, 10]$	0
Schwefel1.2	$f_3(\boldsymbol{x}) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	50	$x_i \in [-100, 100]$	0
Schwefel2.21	$f_4(\boldsymbol{x}) = \max_{1 \le i \le D} \{\lvert x_i \rvert\}$	50	$x_i \in [-100, 100]$	0
Resonbrock	$f_5(\boldsymbol{x}) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	50	$x_i \in [-30, 30]$	0
Step	$f_6(\boldsymbol{x}) = \sum_{i=1}^{n} (\lvert x_i + 0.5 \rvert)^2$	50	$x_i \in [-100, 100]$	0
Quartic	$f_7(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^4 + \mathrm{random}(0, 1)$	50	$x_i \in [-1.28, 1.28]$	0

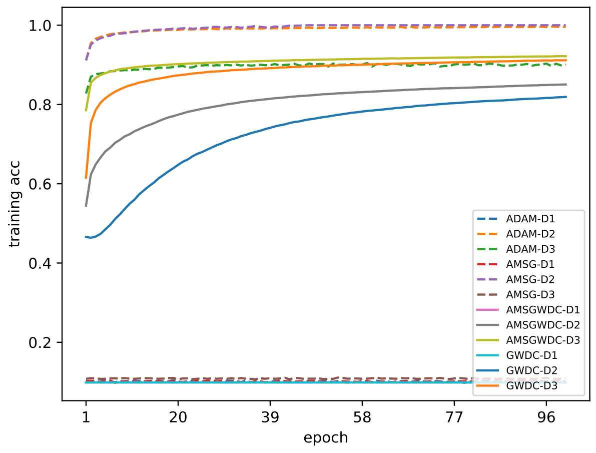


FIGURE 13. Training classification accuracy score for Algorithm 1 with a diminishing learning rate versus the number of epochs on the MNIST dataset using a double-layer perceptron

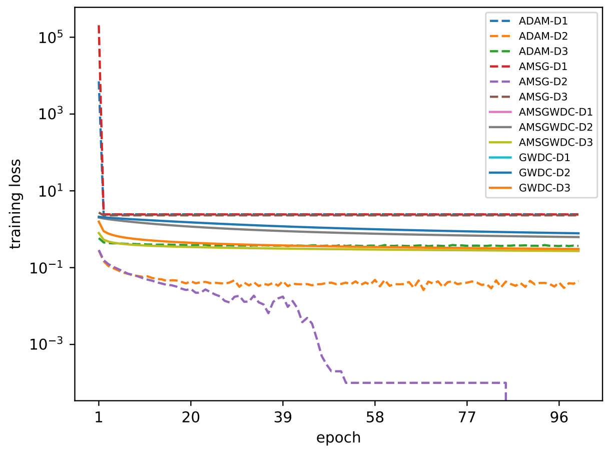


FIGURE 14. Training loss function value for Algorithm 1 with a diminishing learning rate versus the number of epochs on the MNIST dataset using a double-layer perceptron

learning rates with GA, PSO, BBO, ASO, and TLBO versus elapsed time. These figures also show that all the algorithms minimized the seven benchmark functions. For example, for Schwefel2.22, TLBO converged to the solution faster than Algorithm 1. For Step, Algorithm 1 performed better than TLBO.

VI. CONCLUSION

This paper presented a unification of the existing adaptive-learning-rate optimization algorithms for nonconvex stochastic optimization in deep neural networks. It also presented two convergence analyses of the algorithm. The first analysis showed that the algorithm approximates a stationary point of the problem when it uses a constant learning rate. The second analysis showed that the algorithm converges to a stationary point of the problem when it uses a diminishing learning rate. The advantage of the proposed convergence analyses over the existing ones is that they show that the existing adaptive-learning-rate optimization algorithms can be applied to stochastic optimization in deep neural networks. The experiments provided support for the convergence analyses. In particular, the numerical results for training neural networks showed that the algorithm with a constant learning rate performed better than the one with a diminishing rate, as promised by the convergence analyses. Moreover, the numerical results for minimizing benchmark functions showed that the proposed algorithm and TLBO performed well.

VII. ACKNOWLEDGMENT

We are sincerely grateful to the Associate Editor, Nishant Unnikrishnan, and the two anonymous reviewers for helping us improve the original manuscript.

REFERENCES

[1] M. A. Hanif, A. Manglik, and M. Shafique, "Resistive crossbar-aware neural network design and optimization," IEEE Access, vol. 8, pp. 229066–229085, 2020.

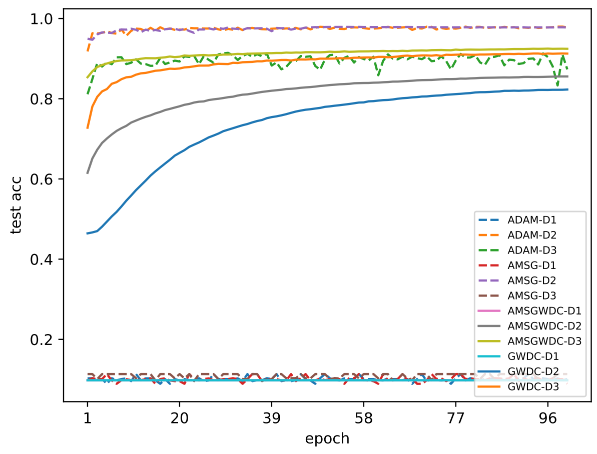


FIGURE 15. Test classification accuracy score for Algorithm 1 with a diminishing learning rate versus the number of epochs on the MNIST dataset using a double-layer perceptron

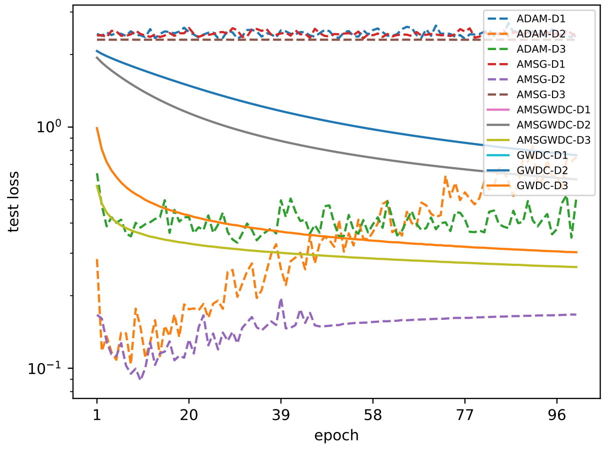


FIGURE 16. Test loss function value for Algorithm 1 with a diminishing learning rate versus the number of epochs on the MNIST dataset using a double-layer perceptron

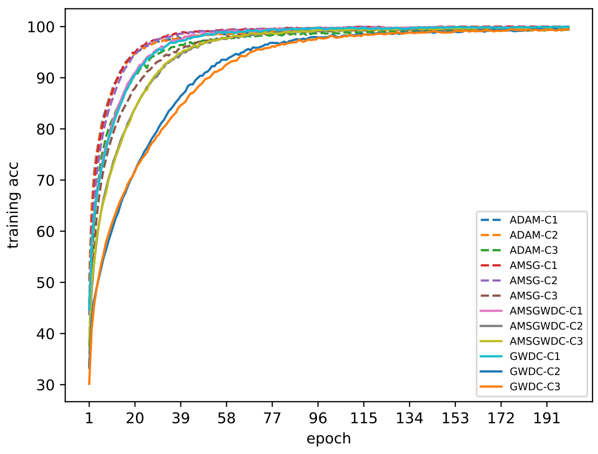


FIGURE 17. Training classification accuracy score for Algorithm 1 with a constant learning rate versus the number of epochs on the CIFAR-10 dataset using DenseNet

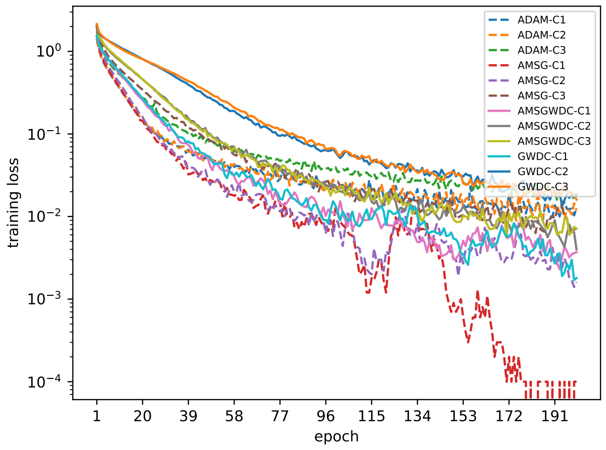


FIGURE 18. Training loss function value for Algorithm 1 with a constant learning rate versus the number of epochs on the CIFAR-10 dataset using DenseNet

[2] D. Liang, F. Ma, and W. Li, "New gradient-weighted adaptive gradient methods with dynamic constraints," IEEE Access, vol. 8, pp. 110929–110942, 2020.

[3] Y. Yu and F. Liu, "Effective neural network training with a new weighting mechanism-based optimization algorithm," IEEE Access, vol. 7, pp. 72403–72410, 2019.

[4] K. Scaman and C. Malherbe, "Robustness analysis of non-convex stochastic gradient descent using biased expectations," Advances in Neural Information Processing Systems, vol. 33, pp. 1–11, 2020.

[5] N. Loizou, S. Vaswani, I. Laradji, and S. Lacoste-Julien, "Stochastic polyak step-size for SGD: An adaptive learning rate for fast convergence," Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS), vol. 130, pp. 1–11, 2021.

[6] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, Cambridge, 2016.

[7] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," Journal of Machine Learning Research, vol. 12, pp. 2121–2159, 2011.

[8] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," Proceedings of The International Conference on Learning Representations, pp. 1–15, 2015.

[9] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of Adam and beyond," Proceedings of The International Conference on Learning Representations, pp. 1–23, 2018.

[10] J. Zhuang, T. Tang, Y. Ding, S. Tatikonda, N. Dvornek, X. Papademetris, and J. S. Duncan, "AdaBelief optimizer: Adapting stepsizes by the belief in observed gradients," Advances in Neural Information Processing Systems, vol. 33, pp. 1–29, 2020.

[11] H. Iiduka, "Appropriate learning rates of adaptive learning rate optimization algorithms for training deep neural networks," IEEE Transactions on Cybernetics, 2021.

[12] Y. Ling, Y. Zhou, and Q. Luo, "Lévy flight trajectory-based whale optimization algorithm for global optimization," IEEE Access, vol. 5, pp. 6168–6186, 2017.

[13] X. Chen, S. Liu, R. Sun, and M. Hong, "On the convergence of a class of Adam-type algorithms for non-convex optimization," Proceedings of The International Conference on Learning Representations, pp. 1–30, 2019.

[14] H. Sakai and H. Iiduka, "Riemannian adaptive optimization algorithm and its application to natural language processing," IEEE Transactions on Cybernetics, 2021.

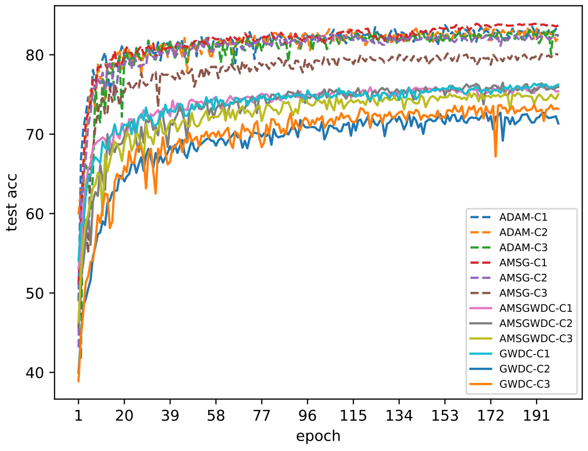


FIGURE 19. Test classification accuracy score for Algorithm 1 with a constant learning rate versus the number of epochs on the CIFAR-10 dataset using DenseNet

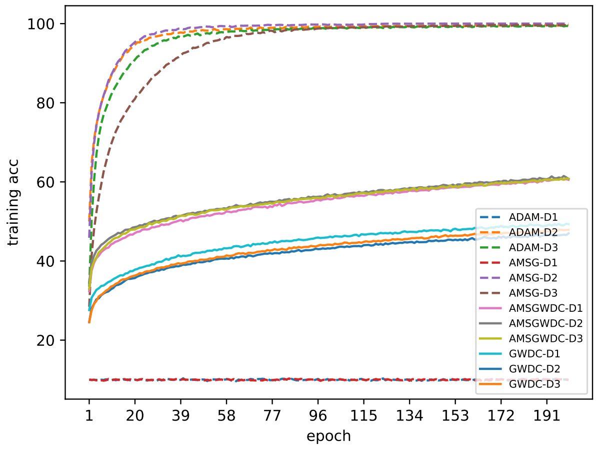


FIGURE 21. Training classification accuracy score for Algorithm 1 with a diminishing learning rate versus the number of epochs on the CIFAR-10 dataset using DenseNet

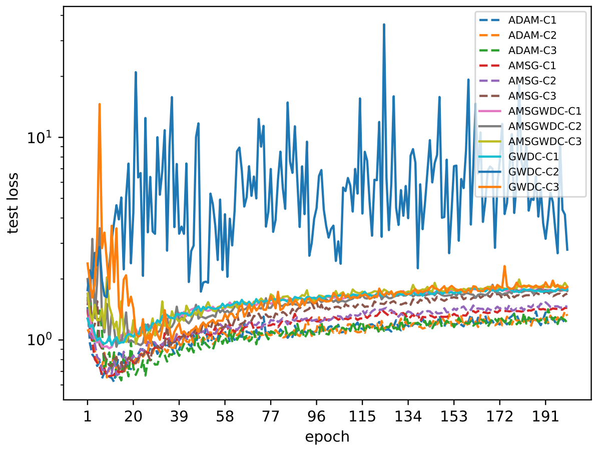


FIGURE 20. Test loss function value for Algorithm 1 with a constant learning rate versus the number of epochs on the CIFAR-10 dataset using DenseNet

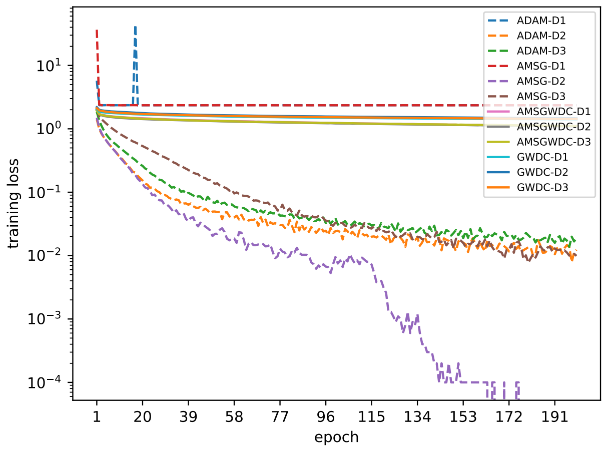
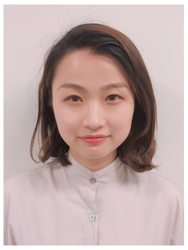


FIGURE 22. Training loss function value for Algorithm 1 with a diminishing learning rate versus the number of epochs on the CIFAR-10 dataset using DenseNet

[15] H. Iiduka, "Stochastic fixed point optimization algorithm for classifier ensemble," IEEE Transactions on Cybernetics, vol. 50, no. 10, pp. 4370–4380, 2020.
[16] L. Luo, Y. Xiong, Y. Liu, and X. Sun, "Adaptive gradient methods with dynamic bound of learning rate," Proceedings of The International Conference on Learning Representations, pp. 1–21, 2019.
[17] F. van den Bergh, An Analysis of Particle Swarm Optimizers. PhD thesis, The Faculty of Natural and Agricultural Science, University of Pretoria, 2001.
[18] D. Simon, "Biogeography-based optimization," IEEE Transactions on Evolutionary Computation, vol. 12, no. 6, pp. 702–713, 2008.
[19] W. Zhao, L. Wang, and Z. Zhang, "Atom search optimization and its application to solve a hydrogeologic parameter estimation problem," Knowledge-Based Systems, vol. 163, pp. 283–304, 2019.
[20] R. Venkata Rao and V. Patel, "An elitist teaching-learning-based optimization algorithm for solving complex constrained optimization problems," International Journal of Industrial Engineering Computations, vol. 3, no. 4, pp. 535–560, 2012.



YINI ZHU has been a masters student in Computer Science Course, Graduate School of Science and Technology, Meiji University, Kanagawa, Japan, since April 2020. Her research field is optimization theory and its applications.

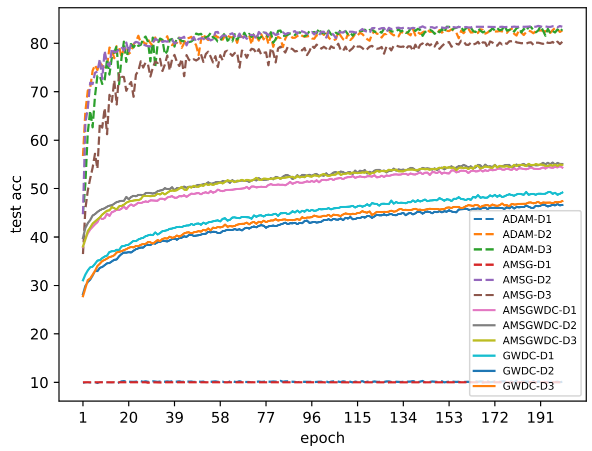


FIGURE 23. Test classification accuracy score for Algorithm 1 with a diminishing learning rate versus the number of epochs on the CIFAR-10 dataset using DenseNet

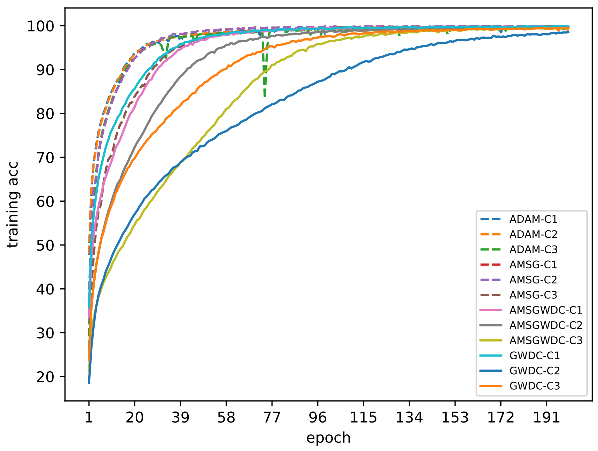


FIGURE 25. Training classification accuracy score for Algorithm 1 with a constant learning rate versus the number of epochs on the CIFAR-10 dataset using ResNet

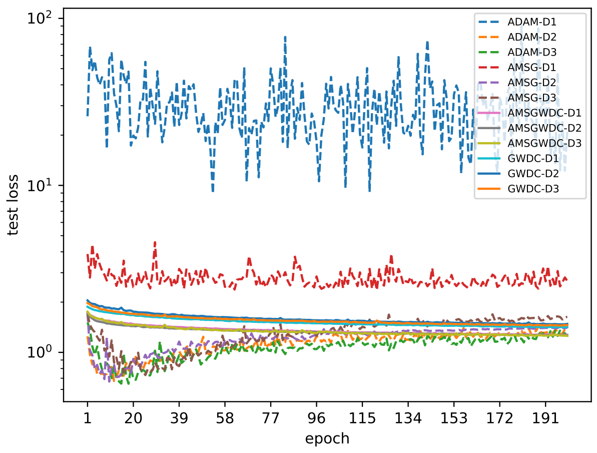


FIGURE 24. Test loss function value for Algorithm 1 with a diminishing learning rate versus the number of epochs on the CIFAR-10 dataset using DenseNet

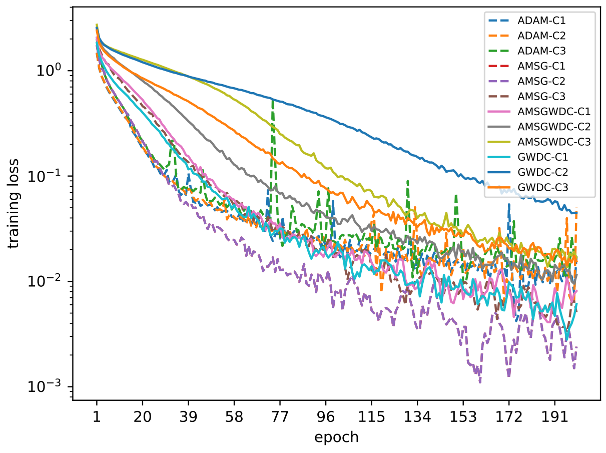


FIGURE 26. Training loss function value for Algorithm 1 with a constant learning rate versus the number of epochs on the CIFAR-10 dataset using ResNet

HIDEAKI IIDUKA received the Ph.D. degree in mathematical and computing science from Tokyo Institute of Technology, Tokyo, Japan, in 2005. From 2005 to 2007, he was a Research Assistant in the Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, Tokyo, Japan. From 2007 to 2008, he was a Research Fellow (PD) of the Japan Society for the Promotion of Science. From October 2008 to March 2013, he was an Associate Professor in the Network Design Research Center, Kyushu Institute of Technology, Tokyo, Japan. From April 2013 to March 2019, he was an Associate Professor in the Department of Computer Science, School of Science and Technology, Meiji University, Kanagawa, Japan. Since April 2019, he has been a Professor in the same department. He was awarded the 4th Research Encourage Award of ORSJ in August 2014 and the 9th Research Award of ORSJ in September 2019. His research field is optimization theory and its applications to mathematical information science. He is a Fellow of ORSJ and a member of MOS and SIAM.

• • •

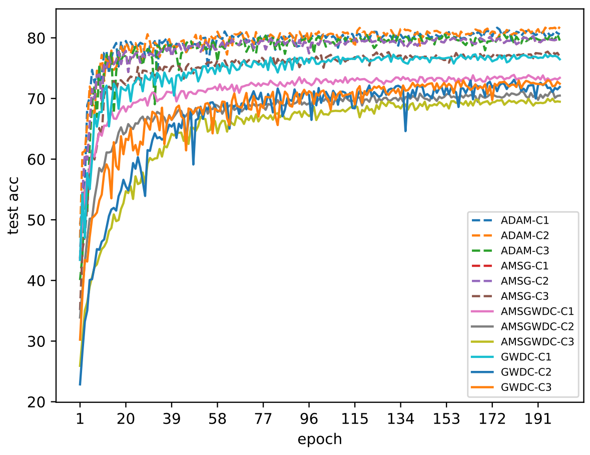


FIGURE 27. Test classification accuracy score for Algorithm 1 with a constant learning rate versus the number of epochs on the CIFAR-10 dataset using ResNet

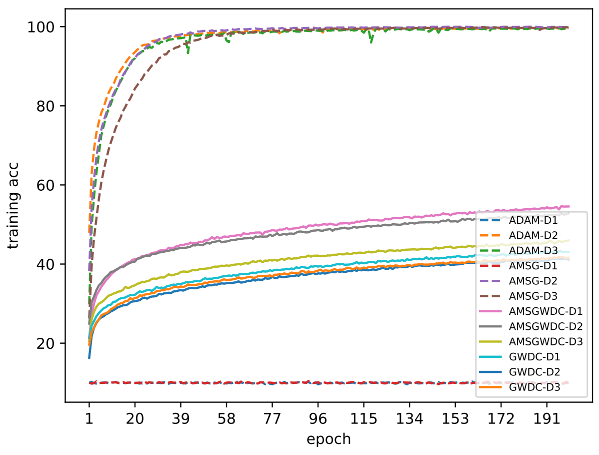


FIGURE 29. Training classification accuracy score for Algorithm 1 with a diminishing learning rate versus the number of epochs on the CIFAR-10 dataset using ResNet

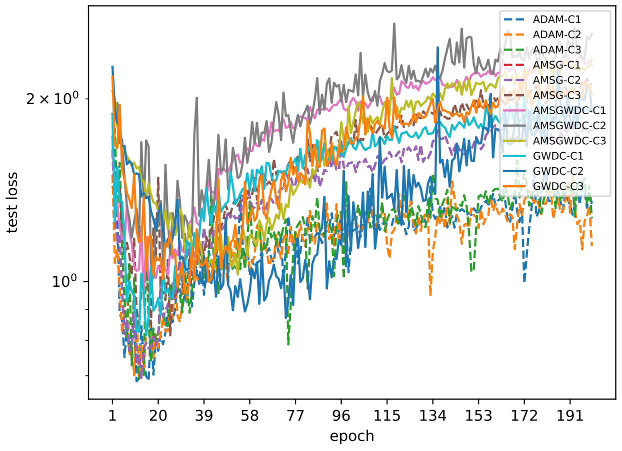


FIGURE 28. Test loss function value for Algorithm 1 with a constant learning rate versus the number of epochs on the CIFAR-10 dataset using ResNet

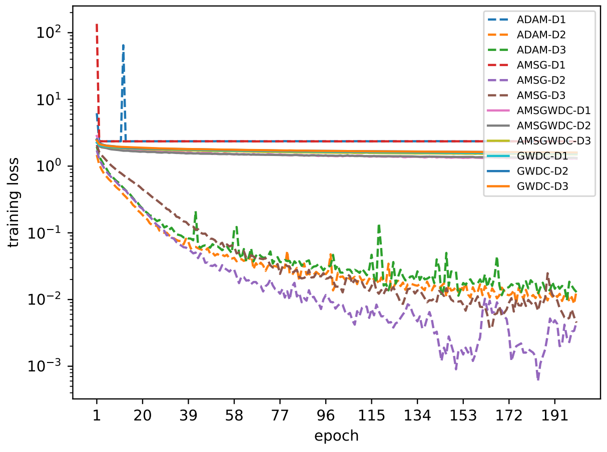


FIGURE 30. Training loss function value for Algorithm 1 with a diminishing learning rate versus the number of epochs on the CIFAR-10 dataset using ResNet

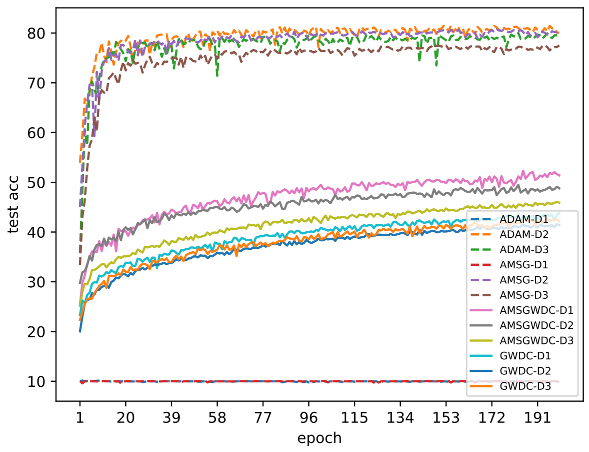


FIGURE 31. Test classification accuracy score for Algorithm 1 with a diminishing learning rate versus the number of epochs on the CIFAR-10 dataset using ResNet

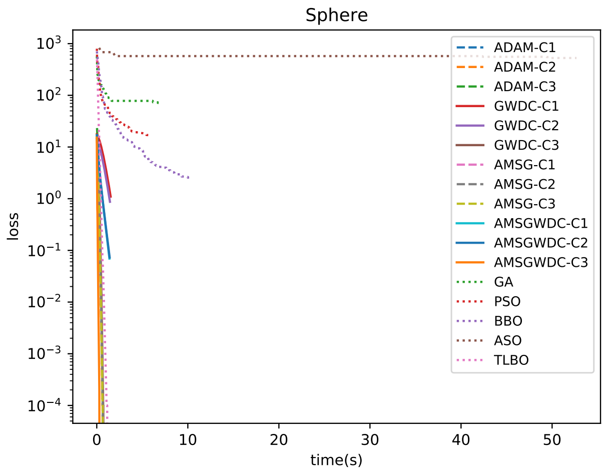


FIGURE 33. Sphere values for Algorithm 1 using constant learning rates with heuristic intelligent optimization methods (GA, PSO, BBO, ASO, and TLBO) versus elapsed time

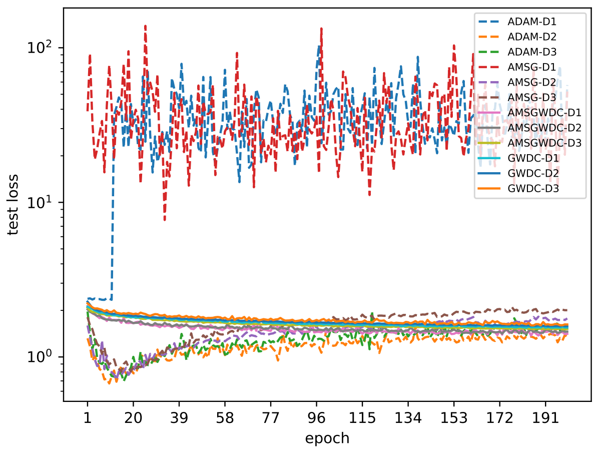


FIGURE 32. Test loss function value for Algorithm 1 with a diminishing learning rate versus the number of epochs on the CIFAR-10 dataset using ResNet

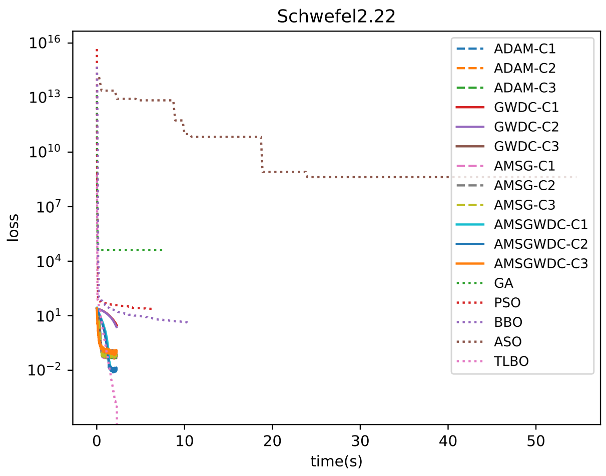


FIGURE 34. Schwefel2.22 values for Algorithm 1 using constant learning rates with heuristic intelligent optimization methods (GA, PSO, BBO, ASO, and TLBO) versus elapsed time

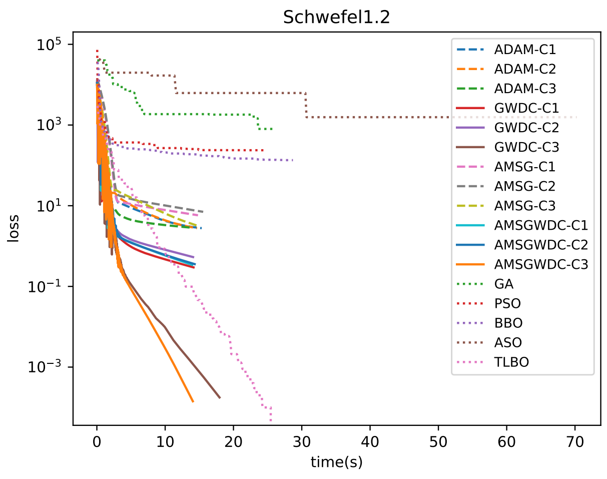


FIGURE 35. Schwefel1.2 values for Algorithm 1 using constant learning rates with heuristic intelligent optimization methods (GA, PSO, BBO, ASO, and TLBO) versus elapsed time

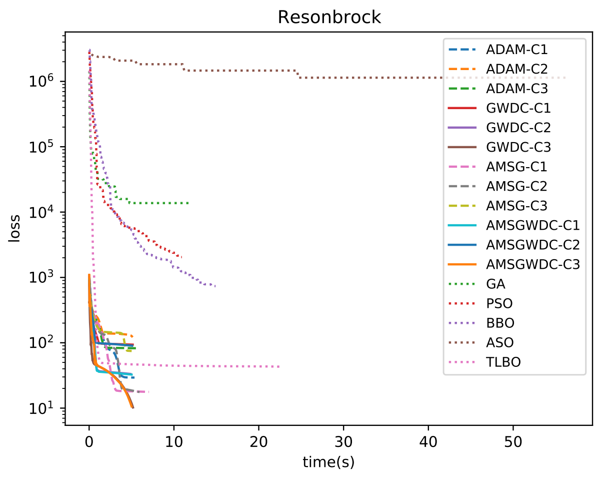


FIGURE 37. Resonbrock values for Algorithm 1 using constant learning rates with heuristic intelligent optimization methods (GA, PSO, BBO, ASO, and TLBO) versus elapsed time

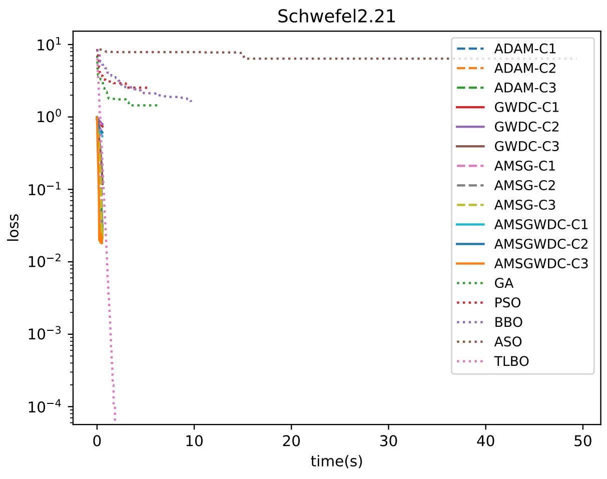


FIGURE 36. Schwefel2.21 values for Algorithm 1 using constant learning rates with heuristic intelligent optimization methods (GA, PSO, BBO, ASO, and TLBO) versus elapsed time

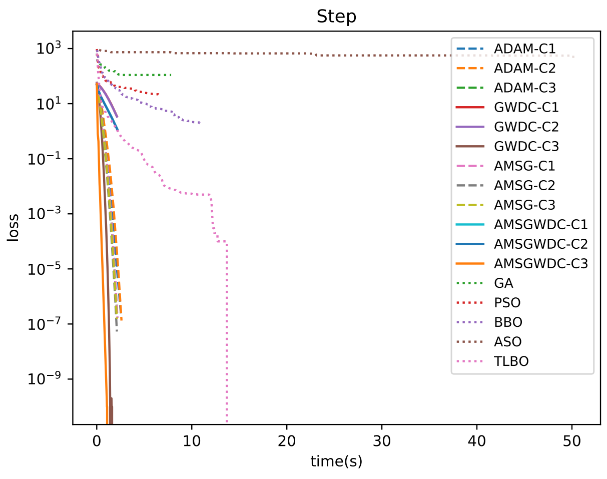


FIGURE 38. Step Function values for Algorithm 1 using constant learning rates with heuristic intelligent optimization methods (GA, PSO, BBO, ASO, and TLBO) versus elapsed time

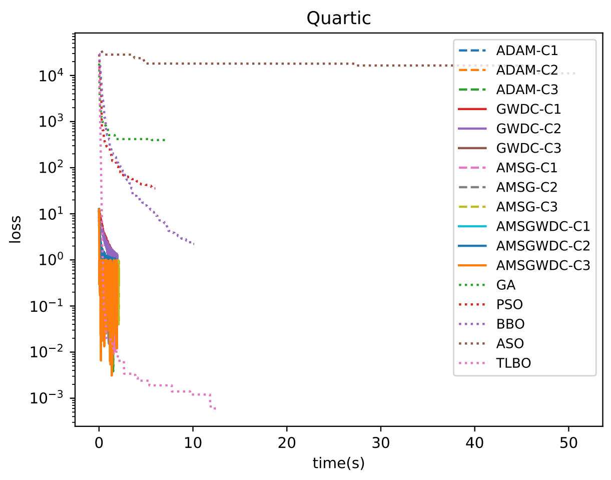


FIGURE 39. Quartic function values for Algorithm 1 using constant learning rates with heuristic intelligent optimization methods (GA, PSO, BBO, ASO, and TLBO) versus elapsed time

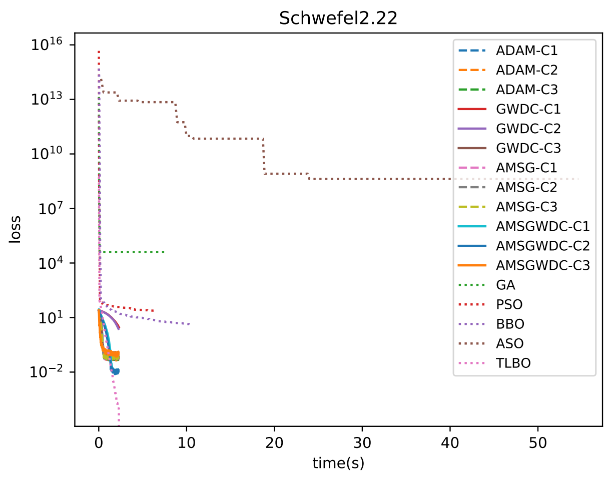


FIGURE 41. Schwefel2.22 values for Algorithm 1 using diminishing learning rates with heuristic intelligent optimization methods (GA, PSO, BBO, ASO, and TLBO) versus elapsed time

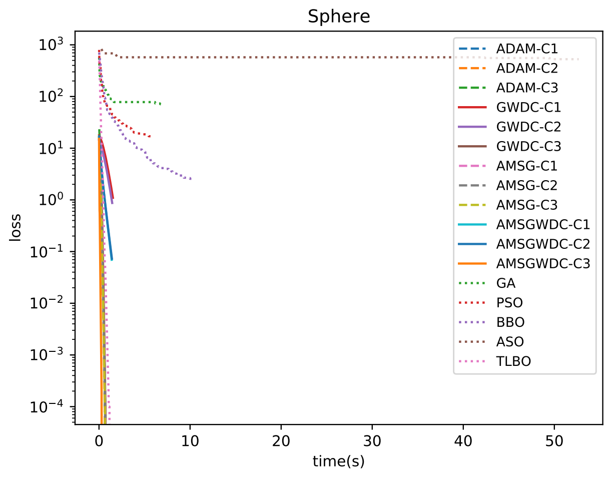


FIGURE 40. Sphere values for Algorithm 1 using diminishing learning rates with heuristic intelligent optimization methods (GA, PSO, BBO, ASO, and TLBO) versus elapsed time

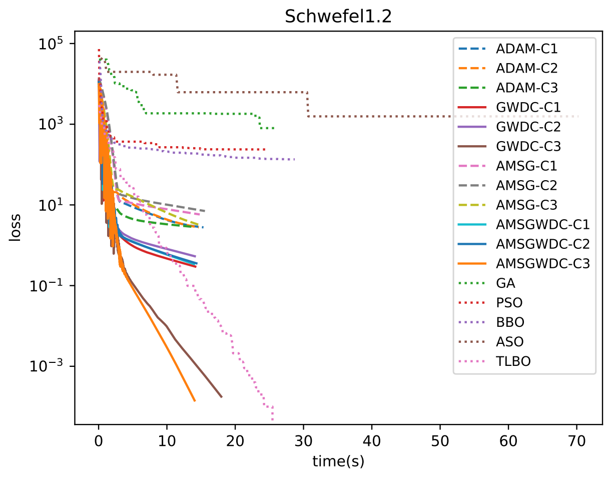


FIGURE 42. Schwefel1.2 values for Algorithm 1 using diminishing learning rates with heuristic intelligent optimization methods (GA, PSO, BBO, ASO, and TLBO) versus elapsed time

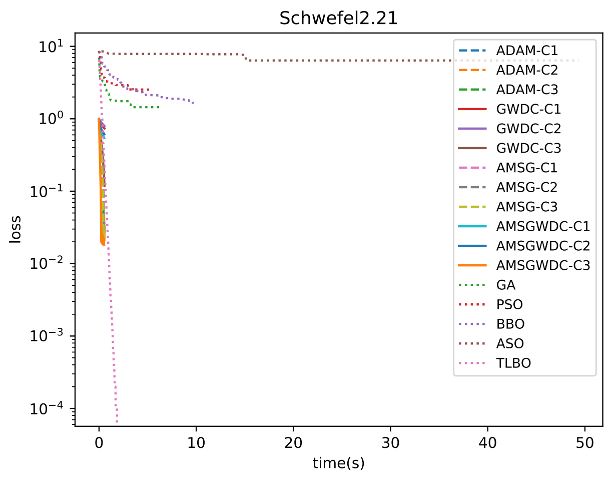


FIGURE 43. Schwefel2.21 values for Algorithm 1 using diminishing learning rates with heuristic intelligent optimization methods (GA, PSO, BBO, ASO, and TLBO) versus elapsed time

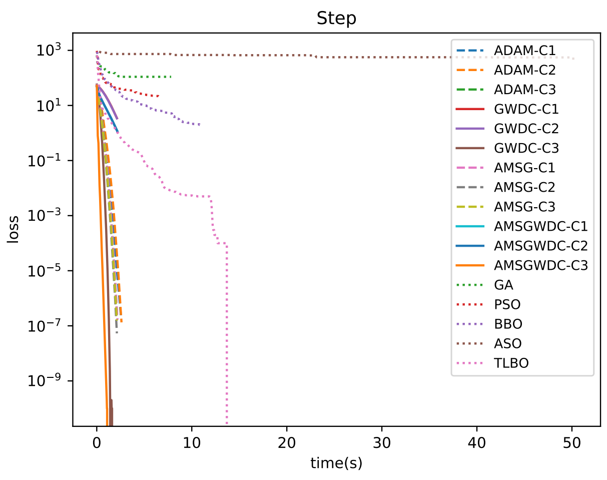


FIGURE 45. Step Function values for Algorithm 1 using diminishing learning rates with heuristic intelligent optimization methods (GA, PSO, BBO, ASO, and TLBO) versus elapsed time

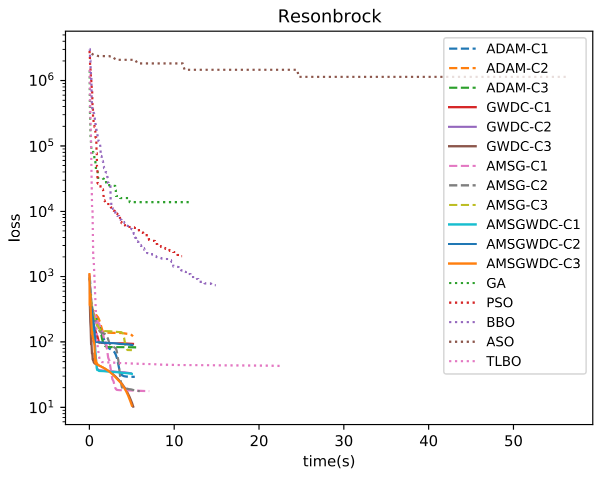


FIGURE 44. Resonbrock values for Algorithm 1 using diminishing learning rates with heuristic intelligent optimization methods (GA, PSO, BBO, ASO, and TLBO) versus elapsed time

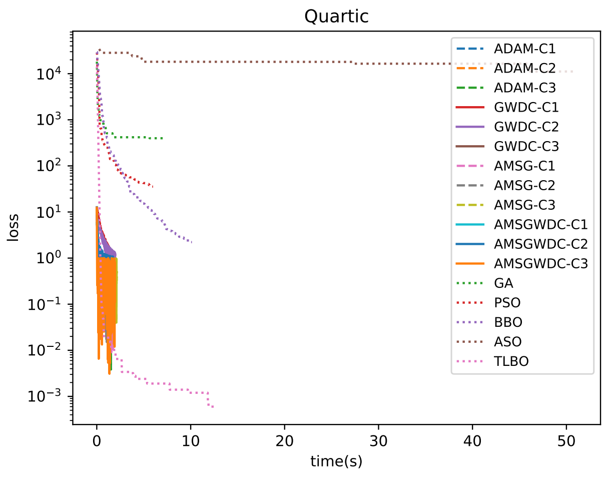


FIGURE 46. Quartic function values for Algorithm 1 using diminishing learning rates with heuristic intelligent optimization methods (GA, PSO, BBO, ASO, and TLBO) versus elapsed time