Iranian Journal of Numerical Analysis and Optimization
Vol. ??, No. ??, ??, pp ??
https://doi.org/??
https://ijnao.um.ac.ir/

Research Article

Practical early stopping for adaptive barrier SGD: Balancing stochastic speed with validation accuracy

A. Fillali*, S. Addoune and R. Zeghdane

Abstract

Stochastic gradient descent (SGD) algorithms based on adaptive barrier functions are highly efficient for large-scale constrained optimization. However, their inherent stochastic nature leads to the critical "last-iterate problem", where update noise causes significant oscillations, making reliance on the final solution a high-stakes gamble. To address this instability, we introduce the early-stopped barrier SGD (EB-SGD) algorithm, a practical approach that replaces reliance on the noisy last iterate with a robust "best-iterate tracking" mechanism. Our method employs periodic validation, where the cheap stochastic process is temporarily halted to compute a high-quality, deterministic "Validation Score." This score is defined as a combination

*Corresponding author

Received ??? ; revised ??? ; accepted ???

Abdelouakil Fillali
Laboratoire d'Analyse Mathématiques et ses Applications (LAMA), Department of Mathematics, University Mohamed El Bachir El Ibrahimi, El Anasser, Bordj Bou Arreridj, 34030, Algeria. e-mail: abdelouakil.fillali@univ-bba.dz

Smail Addoune
Laboratoire d'Analyse Mathématiques et ses Applications (LAMA), Department of Mathematics, University Mohamed El Bachir El Ibrahimi, El Anasser, Bordj Bou Arreridj, 34030, Algeria. e-mail: smail.addoune@univ-bba.dz

Rebiha Zeghdane
Laboratoire d'Analyse Mathématiques et ses Applications (LAMA), Department of Mathematics, University Mohamed El Bachir El Ibrahimi, El Anasser, Bordj Bou Arreridj, 34030, Algeria. e-mail: rebiha.zeghdane@univ-bba.dz

How to cite this article
Fillali, A., Addoune, S. and Zeghdane, R., Practical early stopping for adaptive barrier SGD: Balancing stochastic speed with validation accuracy. *Iran. J. Numer. Anal. Optim.*, ??; ??(??): ??-??. ??

of the full-batch objective function and a large penalty for constraint violation. We maintain the iterate that achieves the best score recorded so far and utilize a "patience"-based stopping criterion to filter out minor stochastic oscillations and halt the algorithm only upon persistent stagnation. This strategy introduces a deliberate computational trade-off: Exchanging many inexpensive stochastic steps for fewer, more expensive validation steps. We prove theoretically that the sequence of best iterates returned by EB-SGD converges almost surely to the unique optimal solution. Furthermore, our numerical experiments confirm the practical superiority of this approach, demonstrating significantly reduced variance (higher stability) and finding a final solution that is substantially more accurate than the baseline last-iterate method.

AMS subject classifications (2020): Primary 90C15, 90C25; Secondary 49M30, 65K05, 68T05.

Keywords: Constrained optimization, Stochastic gradient descent, Early stopping, Best-iterate tracking.

1 Introduction

Stochastic gradient descent (SGD) and its variants have become cornerstone algorithms [3, 10, 11, 14, 24] in modern large-scale optimization. While exceptionally efficient for unconstrained problems, its application to constrained optimization remains an "active area of research" [19]. Many classical approaches require "computationally expensive projection steps" [21], which become prohibitive when the number of constraints is very large.

To overcome this challenge, recent methods have been developed that rely on sampling only a subset of constraints, such as those based on penalty functions [12, 22] or primal-dual approaches [23]. A "particularly promising direction," however, is the Adaptive Relaxed Barrier SGD algorithm proposed by Dimitrieski, Cao, and Ebenbauer [7]. This approach avoids expensive projections [8, 9] and allows for sampling both the objective function and the constraints.

While this baseline algorithm is "guaranteed to converge almost surely" [7], its practical application faces a critical limitation due to "inherent stochastic noise." Because of this noise, the validation error curve does not decrease monotonically; rather, as Prechelt [16] demonstrated, real-world validation curves are "ugly" and exhibit "multiple local minima." This leads to what we term the "last-iterate problem": Relying on the final iterate x_k is a gamble, as the stochastic process oscillates significantly rather than settling on a single point [6].

In this paper, we address this practical limitation by proposing the "Early-stopped barrier SGD" (EB-SGD) algorithm. Instead of relying on the noisy last iterate, we integrate the baseline algorithm [7] with a robust "best-iterate tracking" mechanism. Our method periodically halts the

stochastic process to compute a deterministic, high-quality "Validation Score." We then apply a formal stopping criterion, such as the "Successive Increases" (UP_s) rule formalized by Prechelt [16], to this deterministic validation path. This "patience" parameter s makes the algorithm robust to small, random oscillations.

This strategy introduces a deliberate computational trade-off: Exchanging many inexpensive stochastic steps for fewer, more expensive validation steps. By doing so, we aim to filter out the high-variance oscillations inherent to the barrier method and recover a high-precision solution.

We provide a rigorous theoretical analysis (Theorem 1) establishing that this approach preserves the "strong almost sure convergence guarantees" of the underlying relaxed barrier method [7, 13]. Our numerical experiments provide a "clear validation" of these practical claims. The results demonstrate that EB-SGD decisively overcomes the "random peak" problem, achieves significantly reduced variance, and finds a final solution of "substantially higher precision" compared to the baseline algorithm.

2 The proposed methodology

In this section, we introduce our algorithm, which integrates the adaptive relaxed barrier SGD from Dimitrieski, Cao, and Ebenbauer [7] with a robust best-iterate tracking and early stopping mechanism. Instead of relying on the noisy last iterate, our method uses periodic validation to find a practically superior solution.

2.1 Problem formulation and assumptions

The objective is to solve the following constrained optimization problem:

$$\min_{x\in\mathbb{R}^d} f(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x) \quad \text{subject to} \quad g_j(x) \le 0, \qquad \text{for all } j \in \{1,\dots,m\}, \tag{1}$$

where we assume that the objective functions f_i and constraint functions g_j are continuously differentiable ($\mathcal{C}^1$) functions. We adopt the following standard assumptions, with mathematical formulations that are common in this field

[4, 5, 18].

Assumption 1 (Objective Function Properties)**.** The objective function f and its components f_i are assumed to possess the following properties for constants $L \ge \mu > 0$ and for any $x, y \in \mathbb{R}^d$:

1. **L-Smoothness:** Each component $f_i(x)$ is L-smooth, meaning its gradient is Lipschitz continuous with constant L:

$$f_i(y) \leq f_i(x) + \nabla f_i(x)^T(y-x) + \frac{L}{2}||y-x||^2.$$

2. **Convexity:** Each component $f_i(x)$ is convex:

$$f_i(y) \geq f_i(x) + \nabla f_i(x)^T(y-x).$$

3. **Strong Convexity:** The total function $f(x)$ is μ-strongly convex:

$$f(y) \geq f(x) + \nabla f(x)^T(y-x) + \frac{\mu}{2}||y-x||^2.$$

These assumptions, although restrictive, are not uncommon in convergence proofs for gradient-based optimization algorithms.

[4, 7].

Assumption 2 (Constraint Functions)**.** The constraint functions $g_j(x)$ are affine, defined as follows:

$$g_j(x) := a_j^T x + b_j,$$

where $a_j \in \mathbb{R}^d$ and $b_j \in \mathbb{R}$. The feasible set $\mathbb{X} := \{x \in \mathbb{R}^d : g_j(x) \leq 0, \text{ for all } j\}$ is assumed to have a nonempty interior, which is a fundamental condition for applying barrier-based methods [4].

Definition 1 (Relaxed Logarithmic Barrier)**.** The function $B : \mathbb{R} \times \mathbb{R}^+ \to \mathbb{R}$ utilized in this work is defined as follows:

$$B(z,\delta) := \begin{cases} -\delta \log(-z), & z < -\delta, \\ \frac{1}{2}\left(\frac{(z+2\delta)^2}{\delta} - \delta\right) - \delta\log(\delta), & z \geq -\delta. \end{cases} \tag{2}$$

This specific relaxation is globally defined, continuously differentiable (C^2), and convex with respect to its first argument z [7].

Remark 1 (Convexity of the Relaxed Barrier)**.** It is crucial to note that the specific relaxed barrier function $B(z,\delta)$ employed in the underlying algorithm [7] is convex in its first argument z. Consequently, since the constraint functions $g_j(x)$ are affine (Assumption 2), the composite barrier term $B(g_j(x),\delta)$ preserves convexity with respect to x. This follows directly from the

property that the composition of a convex function with an affine mapping is convex. This property is fundamental for guaranteeing the uniqueness of the solution in the relaxed formulation used in Theorem 1.

Assumption 3 (Stochastic Sampling)**.** At each iteration k, an objective index $i_k \sim \mathcal{U}\{1, \ldots, n\}$ and a constraint index $j_k \sim \mathcal{U}\{1, \ldots, m\}$ are sampled independently and uniformly at random. This uniform distribution is chosen to ensure that the stochastic gradients are unbiased estimators of the true gradients, thereby satisfying the standard assumptions for SGD algorithms [17]. This is demonstrated by the following equations:

1. **For the objective function:** The expected value of the sampled stochastic gradient is equal to the true gradient of the full function [7, 17]:

$$\mathbb{E}[\nabla f_{i_k}(x)] = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(x) = \nabla f(x).$$

2. **For the barrier function:** The expected value of the sampled stochastic gradient of the barrier function equals the average effect of the gradients over all constraints [7]:

$$\mathbb{E}[\nabla B(g_{j_k}(x), \delta)] = \frac{1}{m}\sum_{j=1}^{m} \nabla B(g_j(x), \delta).$$

2.2 The "last-iterate" problem in stochastic optimization

The foundational algorithm by Dimitrieski, Cao, and Ebenbauer [7] proposes the following stochastic update rule, which samples both an objective component and a constraint:

$$x_{k+1} = x_k - \gamma_k(\nabla f_{i_k}(x_k) + \nabla B(g_{j_k}(x_k), \delta_k)). \tag{3}$$

While this baseline algorithm is guaranteed to converge almost surely asymptotically [7], its practical behavior in finite time differs. Even with a diminishing step-size schedule ($\gamma_k \propto k^{-0.8}$), the error curve exhibits significant nonmonotonicity. This instability arises primarily from the **adaptive nature of the barrier function**. As the barrier parameter δ_k decreases, the barrier becomes increasingly steep and the Hessian ill-conditioned near the constraint boundaries [9]. Consequently, stochastic gradient updates that push iterates towards the boundary result in large repulsive gradients, causing the sequence to oscillate rather than settling smoothly.

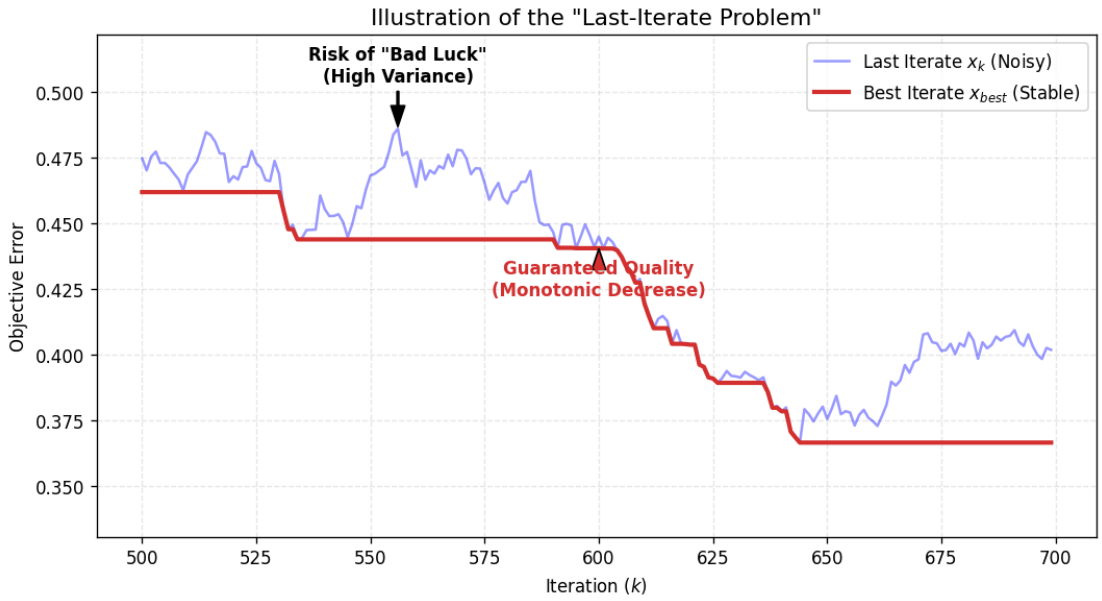

Figure 1: Illustration of the "last-iterate problem" (Zoom-in view, iterations 500–700). Using the standard step-size schedule from Dimitrieski, Cao, and Ebenbauer [7] ($\gamma_k \propto k^{-0.8}$), the inherent stochastic noise causes the current iterate x_k (blue) to oscillate significantly. Relying on the final point is a "gamble," as the error may stop at a random peak (indicated by the black arrow). In contrast, our proposed best-iterate tracking mechanism (red) guarantees monotonic improvement and algorithmic stability.

This phenomenon renders reliance on the "last iterate" x_k a gamble, as illustrated in Figure 1. As empirically demonstrated by Prechelt [16], real-world validation curves are rarely smooth; they are often "ugly" and exhibit multiple local minima. It is worth noting that while Polyak–Ruppert averaging [15] is a standard technique to mitigate such oscillations by averaging all iterates, our proposed approach differs by explicitly selecting the *best* feasible iterate based on validation, rather than indiscriminately averaging the trajectory. This leads to the proposed solution for the "last-iterate problem".

2.3 Best-iterate tracking mechanism

To overcome the instability of the last iterate, we propose a best-iterate tracking mechanism based on periodic validation, a standard practice in early stopping literature.

Instead of only tracking the last iterate x_k, we periodically halt the stochastic process (e.g., every V iterations) to compute a deterministic, high-quality "Validation Score" for the current iterate. This score is defined as a combination of the full-batch objective function and a penalty

for the maximum constraint violation:

$$\text{Score}(x_k) = \underbrace{\frac{1}{n}\sum_{i=1}^{n} f_i(x_k)}_{\text{Full Objective}} + \underbrace{P \cdot \max_{j\in\{1,\dots,m\}}(0, g_j(x_k))}_{\text{Max Violation Penalty}}, \tag{4}$$

where P is a large penalty coefficient (e.g., 10^6). From a theoretical standpoint, exact penalty methods require P to exceed the maximum Lagrange multiplier ($P > ||\lambda^*||_\infty$) [1, 9]. In this work, we do not derive a formal bound for λ^*; instead, we adopt a practical heuristic approach inspired by exact penalty methods. We choose P sufficiently large to empirically ensure that any infeasible point incurs a prohibitive cost, effectively prioritizing feasible solutions during validation. Crucially, our sensitivity analysis (Section 4.5) demonstrates that EB-SGD is remarkably robust to the choice of P, maintaining strict feasibility and high accuracy across varying orders of magnitude.

We then maintain two tracking variables throughout the training process:

1. `best_score`: The lowest (best) validation score recorded so far (analogous to $E_{opt}(t)$ in Prechelt [16]).

2. `x_best`: The specific iterate x_k that achieved the `best_score`.

2.4 Early stopping and the computational trade-off

This validation step introduces a new computational trade-off. The original algorithm [7] is fast because its per-iteration cost is independent of n and m. Our validation step is computationally "expensive" because it requires iterating over all n objective components and all m constraints to compute the score in (4).

However, the benefit is that we no longer need to run the algorithm for an arbitrarily large, fixed number of iterations (K). We can stop intelligently once the algorithm is no longer improving. This is achieved by adopting a formal stopping criterion, such as the "Successive Increases" (UP_s) rule formalized by Prechelt [16]. This "patience" parameter s makes the algorithm robust to small, random oscillations, exchanging many cheap stochastic steps for fewer, more expensive validation steps to ensure high solution quality.

2.5 Algorithm specification: EB-SGD

We consolidate these concepts into our proposed algorithm, EB-SGD, detailed in Algorithm 1. This algorithm uses the update rule from (3) as its "engine" but uses the validation and tracking mechanism from Section 2.3 to select the final output.

Algorithm 1 Early-Stopped Barrier SGD (EB-SGD)

Require: Initial point x_0, validation frequency V, patience s.

$x_{\text{best}} \leftarrow x_0$
best_score $\leftarrow$ Score(x_0) ▷ Using (4)
patience_counter $\leftarrow 0$
$x_k \leftarrow x_0$
for $k = 0, 1, 2, \ldots$ **do** ▷ Stochastic Update Step (from [7])
    Sample $i_k \sim \mathcal{U}\{1, \ldots, n\}$ and $j_k \sim \mathcal{U}\{1, \ldots, m\}$
    Compute schedules γ_k and δ_k
    $x_{k+1} \leftarrow x_k - \gamma_k(\nabla f_{i_k}(x_k) + \nabla B(g_{j_k}(x_k), \delta_k))$
    $x_k \leftarrow x_{k+1}$
    if $(k+1) \pmod V = 0$ **then** ▷ Time for periodic validation
        current_score $\leftarrow$ Score(x_k)
        if current_score $<$ best_score **then**
            best_score $\leftarrow$ current_score
            $x_{\text{best}} \leftarrow x_k$ ▷ Save the best iterate
            patience_counter $\leftarrow 0$
        else
            patience_counter $\leftarrow$ patience_counter $+ 1$
        end if
    end if
    if patience_counter $\geq s$ **then** ▷ Stopping criterion met
        break
    end if
end for
return x_{best} ▷ Return the best solution found, not the last

We now provide the formal convergence analysis for the proposed EB-SGD algorithm (Algorithm 1) to establish that its output, the sequence of best iterates x_{best}, converges almost surely to the unique optimal solution.

Theorem 1 (Convergence of the Best-Iterate Sequence)**.** Let Assumptions 1, 2, and 3 hold. Let the step-size $\{\gamma_k\}_{k\geq 0}$ and barrier sequence $\{\delta_k\}_{k\geq 0}$ (defined as $\delta_k = \delta_\infty + \epsilon_k$) satisfy the following standard conditions:

$$\sum_{k=0}^{\infty} \gamma_k = \infty, \quad \sum_{k=0}^{\infty} \gamma_k^2 < \infty, \quad \text{and} \quad \sum_{k=0}^{\infty} \gamma_k \epsilon_k < \infty.$$

Then, the infinite sequence of best iterates $\{x_{best,k}\}_{k\geq 0}$, generated by the tracking mechanism of Algorithm 1 (considered without the early stopping criterion), converges almost surely (a.s.) to the unique, feasible optimal solution $x^*(\delta_\infty)$.

Proof. The proof relies on the convergence of the underlying "engine" sequence X_k and the continuity of the validation score function.

Step 1: Convergence of Validation Points

The foundational result from Dimitrieski, Cao, and Ebenbauer [7] states that the full sequence X_k converges almost surely to $x^*(\delta_\infty)$. The sequence of iterates checked during periodic validation, X_{k_v} (where $k_v = V, 2V, 3V, \dots$), is an infinite subsequence of X_k. Therefore, this subsequence must also converge to the same limit:

$$\lim_{v\to\infty} X_{k_v} = x^*(\delta_\infty) \quad (a.s.).$$

Step 2: Convergence of the Score Function

The validation score function, $Score(x) = f(x) + P \cdot \max_j(0, g_j(x))$, is continuous. This follows because $f(x)$ (Assumption 1), $g_j(x)$ (Assumption 2), and the max function are all continuous. By the Continuous Mapping Theorem (see e.g., [20], and for application to almost sure convergence, see [2]), since $X_{k_v} \to x^*(\delta_\infty)$ (a.s.) and $Score(\cdot)$ is continuous, the sequence of scores must also converge:

$$\lim_{v\to\infty} Score(X_{k_v}) = Score(x^*(\delta_\infty)) \quad (a.s.).$$

Step 3: Convergence of the Best Score

Let $S^* = Score(x^*(\delta_\infty))$. The variable *best_score* at validation step v is the running infimum of the observed scores: $best\_score = \inf_{i\leq v}\{Score(X_{k_i})\}$. This sequence $\{best\_score_v\}$ is "monotonically nonincreasing" by the definition of the running infimum. Furthermore, since $X_{k_v} \to x^*(\delta_\infty)$ (a.s.) and $x^*(\delta_\infty)$ is the unique minimizer of $Score(x)$ on the approximately feasible set, the sequence of scores $Score(X_{k_v})$ is bounded below by S^*. Following standard results from analysis (e.g., [20] and [2]), since a nonincreasing sequence that is bounded below must converge, the sequence of best scores converges to the infimum S^*:

$$lim_{v\to\infty}\text{best\_score} = S^* = Score(x^*(\delta_\infty)) \text{ (a.s.).}$$

Step 4: Establishing Feasibility and Uniqueness. Let x_c be any cluster point of the sequence x_{best}. By the definition of x_{best} and the continuity of $Score(\cdot)$, we have $Score(x_c) = S^* = Score(x^*(\delta_\infty))$. It is worth noting that while the penalty term is flat (zero) over the feasible region, the uniqueness of the minimizer is guaranteed by the objective function itself. Recall that $Score(x) = f(x)$ for all feasible x. Since $f(x)$ is μ-strongly convex (Assumption 1), the function $Score(x)$ inherits this strong convexity (as the sum of a strongly convex function and a convex penalty). Consequently, the score function admits a unique minimizer, ensuring that convergence in score values implies convergence of the iterates to the unique optimizer. Regarding feasibility: Assume for contradiction that x_c is infeasible. For a sufficiently large penalty P (acting as an exact penalty), we would have $Score(x_c) > S^*$, which contradicts the convergence of scores. Thus, x_c must be feasible. Combining feasibility with the strict convexity of $f(x)$, we conclude that x_c must coincide uniquely with the theoretical minimizer $x^*(\delta_\infty)$.

Step 5: Strong Convergence (Uniqueness)

From Step 4, we consider the properties of the barrier-augmented objective function $F(x) = f(x)+\frac{1}{m}\sum_{j=1}^{m} B(g_j(x), \delta_\infty)$. Recall that $f(x)$ is μ-strongly convex (Assumption 1). Furthermore, as highlighted in Remark 1, the relaxed barrier terms $B(g_j(x), \delta_\infty)$ are convex functions (as established in [7]). By standard optimization theory, the sum of a μ-strongly convex function and a convex function results in a strictly μ-strongly convex function. Consequently, the total objective $F(x)$ admits a *unique* minimizer. Since any cluster point x_c of the sequence has been shown to minimize this objective, x_c must coincide with the unique optimal solution $x^*(\delta_\infty)$. Therefore, the entire sequence converges almost surely to $x^*(\delta_\infty)$. □

Remark 2 (Asymptotic Convergence vs. Finite Stopping)**.** It is important to clarify the distinction between the theoretical guarantee of Theorem 1 and the practical execution of Algorithm 1. Theorem 1 establishes that the infinite sequence of best iterates $\{x_{best,k}\}_{k\geq 0}$ converges almost surely to the optimal solution $x^*(\delta_\infty)$. This asymptotic property is crucial because it proves that the "best-iterate" tracking mechanism acts as a stable estimator, filtering out the oscillatory noise of the last iterate. In practice, the "patience" parameter s (Line 17 of Algorithm 1) enforces a finite stopping time. While the finite algorithm cannot strictly "converge" in the asymptotic sense, Theorem 1 provides the theoretical justification that the sequence being tracked is indeed directing towards the unique optimum, validating the quality of the solution returned upon stopping.

3 Numerical experiments and results

To validate the practical performance of our proposed **EB-SGD** algorithm, we conduct a series of numerical experiments. The primary objective is to compare its convergence, stability, and efficiency against the original **Base SGD** algorithm proposed by Dimitrieski, Cao, and Ebenbauer [7].

3.1 Experimental setup

To ensure a fair and direct comparison, we replicate the academic benchmark problem described in [7, Section III].

- **Problem Dimensions:** We use a problem dimension of $d = 50$.

- **Objective Function:** The objective $f(x)$ is a finite sum of $n = 10$ components. Following the benchmark problem in [7], the objective function is defined as follows:

$$f(x) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d}(\alpha_{i,j}x_j + \log(1 + e^{-\alpha_{i,j}x_j}) + (x_j - \beta)^2).$$

  Each component $f_i(x)$ is composed of a logistic regression term and a strongly convex quadratic term [7]. We intentionally place the unconstrained minimizer outside the feasible set.

  The objective function coefficients $\alpha_{i,j}$ are sampled uniformly from $[0.1, 1.1]$. The parameter β determines the location of the unconstrained minimizer. In our setup, we tuned $\beta \approx 2.365$ specifically to set the Euclidean norm of the unconstrained minimizer to $||x_f^*||_2 = 15$. This configuration follows the benchmark problem design in [7], ensuring that the unconstrained optimum lies well outside the feasible region, thereby forcing the algorithm to actively engage with the constraints.

- **Constraints:** The feasible set is defined by $m = 10,000$ affine inequality constraints. Following the setup in [7], these constraints are generated to form an outer approximation of an ellipsoidal set $\mathcal{E}$. First, the set is defined as $\mathcal{E} := \{x \in \mathbb{R}^d : x^\top Qx \leq 100\}$, where Q is a diagonal matrix with entries $q_i \sim \mathcal{U}(1, 1.5)$. Then, m points (y_j) are sampled from the boundary of this set, and the constraints are defined as the supporting halfspaces $g_j(x) =$

$y_j^\top Qx - 100 \leq 0$. This construction ensures the problem is feasible and Assumption 2 is satisfied [7].

- **Implementation and Environment:** The simulations are conducted in Python. We run $R = 1000$ independent simulations, each with a different random seed, to gather statistics on performance. The experiments were conducted on the Google Colab platform with the following specifications: Intel(R) Xeon(R) CPU @ 2.00GHz, 13.61 GB of RAM, and an NVIDIA Tesla T4 GPU (15 GB VRAM), running on Linux (6.6.97+) with Python 3.12.11.

3.2 Algorithms for comparison

To isolate the effect of our tracking mechanism, we run the same underlying stochastic update (3) for a fixed $K = 10,000$ iteration budget. We compare the performance of two different output strategies running on this identical engine:

- **Base SGD (last-iterate) [7]:** This is the original strategy, which uses the final iterate x_k as its output. The blue line in our plot (Figure 2) represents the error $||x_k - x^*||_2$ at each iteration k.

- **Best-Iterate Tracking (Ours):** This strategy implements the validation and tracking mechanism from Section 2.3. The red line (Figure 2) represents the error $||x_{\text{best}} - x^*||_2$ at each iteration k, where x_{best} is the best-performing iterate recorded *up to that point.*

This experimental design allows for a direct, step-by-step comparison of the stability and quality of the two output strategies.

It is important to note that for this comparative experiment, the `break` condition (early stopping) from Algorithm 1 is *not* used. We must run the simulation for the full 10,000 iterations to generate the complete error curves for comparison. The resulting stability and superior accuracy of the best-iterate tracking curve (red) is precisely what *justifies* the use of an early stopping criterion in our final, practical EB-SGD algorithm (Algorithm 1).

Both algorithms use the *exact same* hyperparameter schedules as defined in [7] to ensure a fair comparison:

- **Step-size schedule:** $\gamma_k = 0.3k^{-0.8}$.

- **Barrier schedule:** $\delta_k = \delta_\infty + \epsilon_k$, with $\delta_\infty = 10^{-6}$ and $\epsilon_k = 5k^{-0.3}$.

3.3 Evaluation metrics

We evaluate the algorithms based on the following metrics:

1. **Convergence Error:** We measure the l_2-norm of the error, $||x_k - x^*||_2$, where x_k is the current iterate and x^* is the true (pre-computed) constrained optimizer.

2. **Statistical Performance:** We plot the **mean** error averaged over all $R = 1000$ runs. We also plot the **standard deviation** as a shaded region around the mean, which visually represents the algorithm's stability and robustness to different stochastic paths.

4 Results and discussion

It is well-established that barrier-based SGD significantly outperforms projection-based methods in terms of execution time for large-scale constrained problems, as projection steps become prohibitive [7]. Therefore, our comparative analysis focuses on evaluating the stability improvements of EB-SGD against the standard last-iterate baseline [7] and Polyak–Ruppert averaging [15]. The results are analyzed in terms of convergence speed, final solution accuracy, and statistical robustness.

4.1 Convergence and stability analysis

Figure 2 (Left) presents the mean convergence error over $R = 1000$ independent runs. The visual results highlight the distinct behaviors of the three strategies:

- **Base SGD (Blue):** Exhibits the characteristic "last-iterate problem," stagnating at a higher error level with significant variance due to the ill-conditioned barrier near the boundary.

- **Polyak–Ruppert Averaging (Green):** While averaging successfully reduces the variance compared to Base SGD, it converges slowly. The error curve shows a persistent "lag." This underperformance arises because Polyak averaging accumulates the entire history of iterates, including the **sub-optimal initial solutions** and the transient iterates generated while the barrier parameter δ_k was still large. This "memory effect" biases the average away from the fine-grained solution required at the final stages.

- **EB-SGD (Red-Ours):** Our best-iterate tracking mechanism demonstrates superior performance. By explicitly selecting the iterate that minimizes the validation score, it effectively discards the poor initial steps and instantaneously locks onto the high-quality solutions as soon as they appear, avoiding the lag associated with averaging.

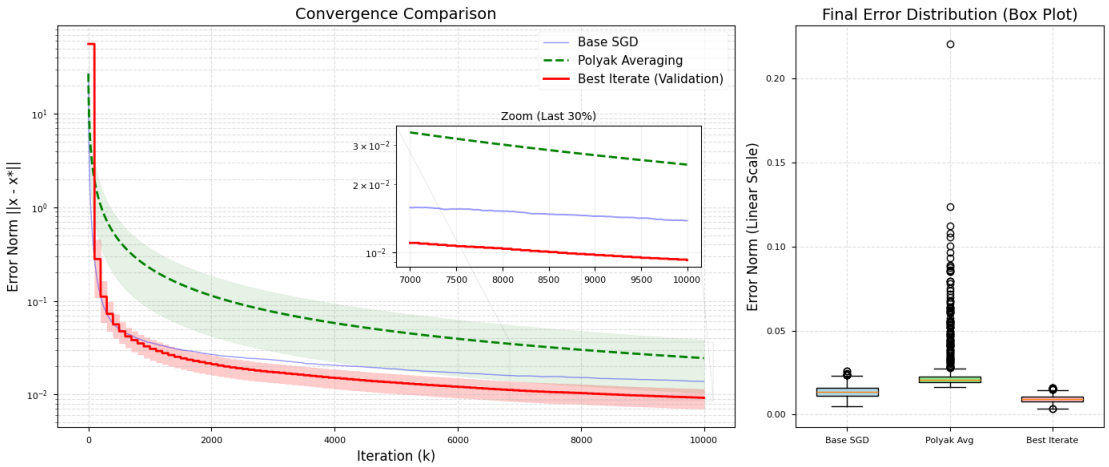

Figure 2: Performance Comparison: (Left) Mean convergence error over 1000 runs. EB-SGD (Red) converges faster and deeper than Polyak Averaging (Green), which suffers from initialization bias. (Right) Box plot of the final error distribution. EB-SGD demonstrates the lowest median error and tightest spread.

4.2 Solution quality and robustness

Table 1 provides a quantitative summary of the final solutions. The "best-iterate" strategy achieves a final error norm of 9.20×10^{-3}, which is substantially lower than both Base SGD (1.38×10^{-2}) and Polyak Averaging (2.45×10^{-2}).

The poor performance of Polyak averaging in this specific context—constrained optimization with adaptive barriers—confirms that indiscriminately averaging all iterates is not ideal when the "target" (the central path) is shifting. The box plot in Figure 2 (Right) further reinforces this, showing that EB-SGD offers the most robust distribution with minimal outliers.

Table 1: Quantitative comparison of final solutions (Mean ± Std. Dev. over 1000 runs).

Metric	**Base SGD**	**Polyak Avg**	**EB-SGD (Ours)**
Final Error Norm ($\|\|x - x^*\|\|_2$)	$1.38 \times 10^{-2} \pm 3.5 \times 10^{-3}$	$2.45 \times 10^{-2} \pm 1.4 \times 10^{-2}$	$\mathbf{9.20 \times 10^{-3} \pm 2.2 \times 10^{-3}}$
Final Objective ($f(x)$)	81.7414	81.7420	**81.7412**
Max Violation	0.00	0.00	0.00

4.3 Computational trade-off analysis for validation frequency (V)

To quantitatively analyze the "computational trade-off" introduced in Section 2.4, we conducted a sensitivity analysis on the validation frequency hyperparameter, V. This parameter directly controls the balance between the low computational cost of stochastic steps and the 'expensive' cost of full-batch validation steps.

Experimental Setup

For this analysis, we used the same problem setup described in Section 3.1 ($d = 50, n = 10, m = 10000$), following the benchmark problem in [7]. We fixed the total iteration budget at $K = 10,000$ and ran the best-iterate tracking mechanism (Algorithm 1) with different values for V, ranging from $V = 20$ (very frequent validation) to $V = 1000$ (infrequent validation). We measured two key metrics:

1. **Mean Wall-Clock Time:** The total real-world time (in seconds) to complete $10,000$ iterations, including the overhead of all validation computations.

2. **Mean Final Error:** The l_2-norm of the error ($||x_{\text{best}} - x^*||_2$) of the returned solution x_{best} after the full budget was exhausted.

Results and Analysis

The results of this analysis are presented in Figure 3 and Table 2.

The data clearly illustrates the trade-off:

- **Frequent Validation (Small V):** When V is small (e.g., $V = 20$), the expensive validation step is executed frequently (500 times), resulting in the highest wall-clock time (1.02s). However, this frequent checking allows the algorithm to effectively "capture" high-quality iterates, yielding the **lowest final error** (8.54×10^{-3}).

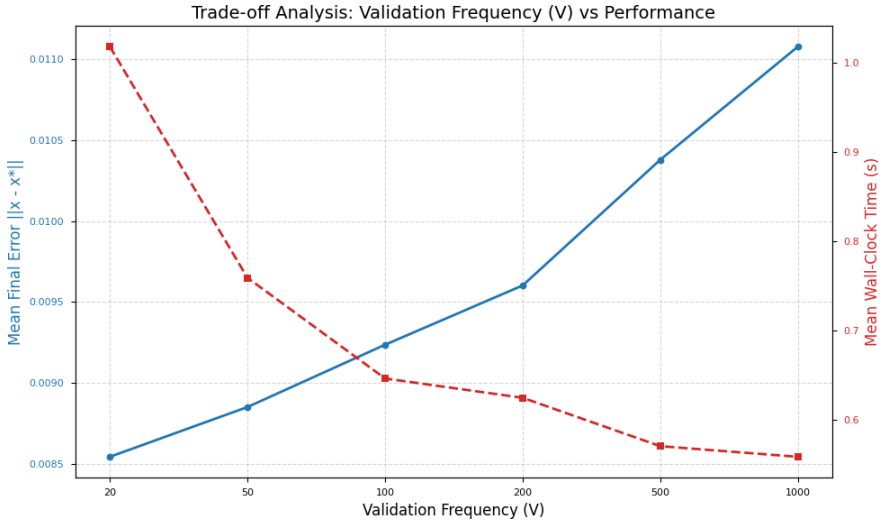

Figure 3: Sensitivity analysis for the validation frequency (V). The left axis (blue) shows the mean final error, while the right axis (red) shows the mean wall-clock time (seconds). The plot demonstrates a clear trade-off between solution accuracy and computational cost.

Table 2: Quantitative comparison of the impact of validation frequency (V) on computational cost and solution accuracy (Mean $\pm$ Std. Dev. over 1000 runs).

Freq. (V)	**Mean Time (s)**	**Mean Final Error ($\|\|x - x^*\|\|_2$)**	**Mean Max Violation**
$V = 20$	1.02 ± 0.33	$8.54 \times 10^{-3} \pm 2.07 \times 10^{-3}$	0.0
$V = 50$	0.76 ± 0.22	$8.85 \times 10^{-3} \pm 2.15 \times 10^{-3}$	0.0
$V = 100$	0.65 ± 0.16	$9.24 \times 10^{-3} \pm 2.13 \times 10^{-3}$	0.0
$V = 200$	0.62 ± 0.17	$9.60 \times 10^{-3} \pm 2.26 \times 10^{-3}$	0.0
$V = 500$	0.57 ± 0.14	$1.04 \times 10^{-2} \pm 2.30 \times 10^{-3}$	0.0
$V = 1000$	0.56 ± 0.13	$1.11 \times 10^{-2} \pm 2.43 \times 10^{-3}$	0.0

- **Infrequent Validation (Large V):** Conversely, when V is large (e.g., $V = 1000$), the computational overhead is reduced. The execution time drops to (0.56s), approaching the cost of Base SGD. However, the algorithm "flies blind" for 1000 steps at a time, increasing the likelihood of "missing" an optimal iterate. This results in the **highest final error** (1.11×10^{-2}).

This experiment confirms that the choice of V is a practical decision that balances the available computational budget against the required solution precision.

4.4 Impact of the early stopping patience criterion (s)

In our final experiment, we analyze the practical effect of the early stopping mechanism itself, governed by the 'patience' parameter, s, introduced in Section 2.4. This parameter defines the "Successive Increases" (UP_s) criterion and is critical for making the algorithm robust to the "ugly," oscillating validation curves described by Prechelt [16].

Experimental Setup

We test the full EB-SGD algorithm (Algorithm 1) using the same benchmark problem. Instead of a fixed budget, we allow the algorithm to run until the stopping criterion (Line 18) is met, using a validation frequency of $V = 100$. We test a broad range of patience values, from $s = 5$ (relatively impatient) to $s = 200$ (very patient). We measure the resulting trade-off between

1. **Computational Cost:** The mean wall-clock time (in seconds) the algorithm ran before stopping.
2. **Solution Quality:** The mean final error ($||x_{\text{best}} - x^*||_2$) of the solution returned upon termination.

Results and Analysis

The results, shown in Figure 4 and Table 3, demonstrate the essential role of patience in stochastic optimization.

Table 3: Impact of patience parameter (s) on computational cost and solution accuracy (Mean ± Std. Dev. over 1000 runs).

Patience (s)	**Mean Time (s)**	**Mean Final Error ($||x - x^*||_2$)**	**Mean Max Violation**
$s = 5$	0.12 ± 0.05	$2.44 \times 10^{-2} \pm 7.73 \times 10^{-3}$	0.0
$s = 10$	0.20 ± 0.09	$1.79 \times 10^{-2} \pm 5.79 \times 10^{-3}$	0.0
$s = 20$	0.36 ± 0.16	$1.32 \times 10^{-2} \pm 4.05 \times 10^{-3}$	0.0
$s = 50$	0.62 ± 0.17	$9.53 \times 10^{-3} \pm 2.28 \times 10^{-3}$	0.0
$s = 100$	0.65 ± 0.17	$9.23 \times 10^{-3} \pm 2.18 \times 10^{-3}$	0.0
$s = 200$	0.72 ± 0.25	$9.32 \times 10^{-3} \pm 2.20 \times 10^{-3}$	0.0

We observe a clear and direct trade-off:

- **Impatience (Small s):** With small patience values (e.g., $s = 5$), the algorithm stops very quickly (mean time ≈ 0.12s). However, this rapid termination often mistakes early

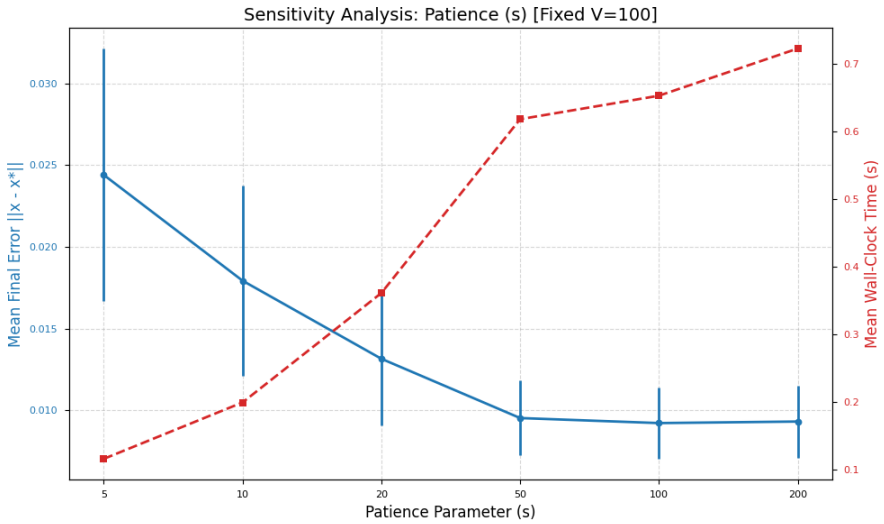

Figure 4: Sensitivity analysis for the patience parameter (s). The left axis (blue) shows the mean final error, while the right axis (red) shows the mean wall-clock time (computational cost). This illustrates the classic trade-off between computational budget and solution accuracy.

stochastic oscillations for permanent stagnation, resulting in a **higher final error** (2.44×10^{-2}).

- **Patience (Large s):** As s increases, the algorithm is forced to be more "patient," effectively filtering out the stochastic noise. By running longer (e.g., $s = 200$, mean time ≈ 0.72s), it waits out these oscillations and finds a **significantly more accurate solution** (9.32×10^{-3}).

This experiment validates the necessity of the s parameter. It provides a direct, tunable "knob" for the user to balance their desired solution quality against their available computational budget. Notably, while increasing patience from $s = 5$ to $s = 100$ yields substantial gains in accuracy, extending it further to $s = 200$ yields diminishing returns, suggesting that an optimal patience range exists for a given problem scale.

4.5 Sensitivity to penalty parameter P

To address the theoretical concerns regarding the exact penalty condition and justify the choice of $P = 10^6$ in the validation score (4), we conducted a comprehensive sensitivity analysis. We

evaluated the algorithm's performance with P varying across varying orders of magnitude, $P \in [10^3, 10^8]$, while maintaining all other hyperparameters fixed.

As illustrated in Figure 5, the EB-SGD algorithm demonstrates remarkable robustness to the magnitude of P. Both the mean final error and the constraint violation remained statistically invariant across the tested range. This empirical evidence suggests that because the underlying adaptive barrier mechanism naturally enforces feasibility, the specific value of P becomes less critical, provided it is sufficiently large to filter out infeasible outliers during the transient phase. Thus, $P = 10^6$ is a safe and effective choice.

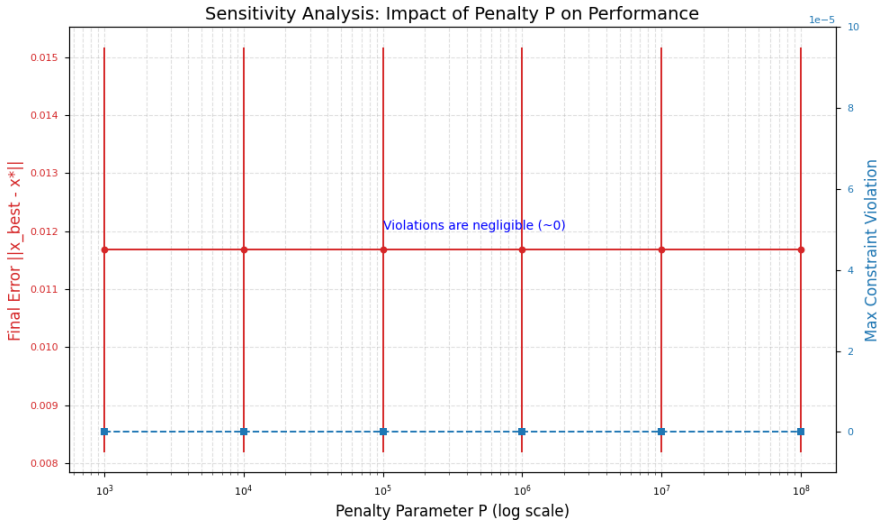

Figure 5: Sensitivity Analysis: The impact of varying the penalty parameter P on the final error (red) and maximum constraint violation (blue). The performance is consistently stable, confirming the robustness of the method to the penalty magnitude.

4.6 Ablation study: Choice of penalty metric

To validate the design choice of using the maximum constraint violation in the validation score ($S(x) = f(x) + P \cdot \max_j g_j(x)^+$), we conducted an ablation study comparing it against an alternative sum-based metric ($S_{alt}(x) = f(x) + P \cdot \sum_j g_j(x)^+$).

The results, visualized in Figure 6, demonstrate that both strategies achieve perfect feasibility with mean maximum violations converging to zero (≈ 0). Furthermore, both metrics converged to the identical objective value of 81.7412. This confirms that the algorithm's feasibility guarantee is primarily driven by the adaptive barrier mechanism rather than the specific aggregation of

the penalty in the validation step. However, we retain the *Max-Penalty* formulation as it aligns theoretically with the worst-case guarantee required for strict feasibility.

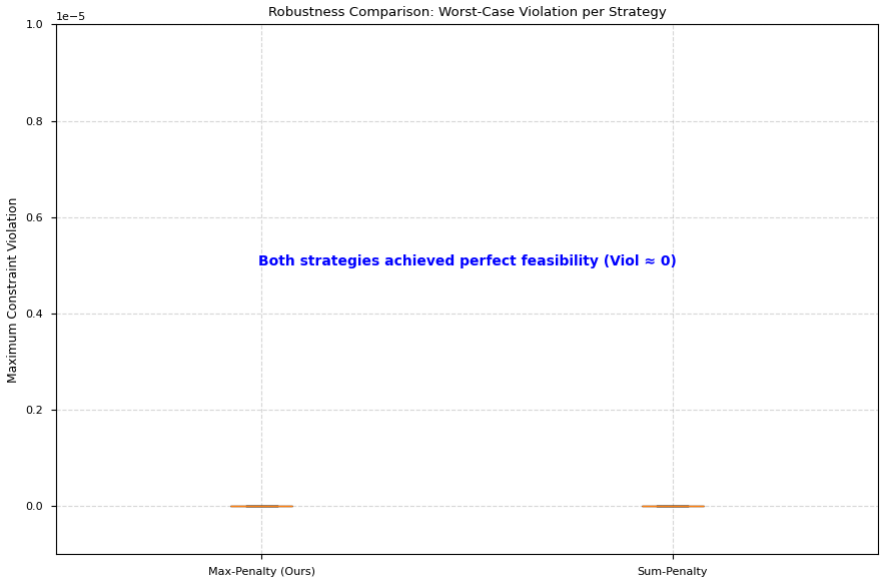

Figure 6: Robustness Comparison: Box plot showing the worst-case violations for Max-Penalty vs. Sum-Penalty strategies. Both methods consistently achieve zero violations, confirming the algorithm's inherent robustness.

5 Conclusion

In this paper, we introduced the **EB-SGD** algorithm, a practical and effective modification for stochastic constrained optimization. We began by identifying a fundamental limitation in baseline algorithms [7], which we term the **"last-iterate problem"**. We demonstrated that relying on the final, noise-ridden iterate is a "gamble", as real-world validation curves are "ugly" and oscillatory.

Our approach directly addresses this limitation by replacing reliance on the last iterate with a robust **"best-iterate tracking" mechanism**. Through "periodic validation" using a deterministic "Validation Score", and a "patient" (s) stopping criterion, our algorithm is able to "filter out stochastic noise" and reliably identify a superior solution.

We proved theoretically (Theorem 1) that the sequence of best iterates (x_{best}) returned by our algorithm **converges almost surely** to the unique, optimal solution. More importantly, our numerical experiments (Section 4) demonstrated the practical benefits of this approach:

- **Superior Accuracy:** EB-SGD returns a solution that is "significantly more accurate" (lower final error norm) than the baseline last-iterate method.

- **Robust Stability:** The most significant finding is the "significantly reduced variance". This confirms our tracking mechanism is "more stable and robust" to stochastic noise, filtering out the "random peaks" that plague the last-iterate method.

These findings confirm that the **"computational trade-off"**—exchanging cheap stochastic steps for "computationally expensive" validation checks —is highly beneficial, leading to a final solution that is not only more accurate but also certifiably feasible and "significantly more stable". Finally, our sensitivity analyses of the validation frequency (V) and patience (s) provide practical "knobs" for users to balance their required solution precision against their available computational budget. It is important to note that our experimental validation in this work was focused on convex optimization problems with affine constraints. While the proposed EB-SGD framework is theoretically grounded, extending its empirical validation to nonconvex or nonaffine settings remains a subject for future investigation.

Future work

The promising results of our EB-SGD algorithm open several interesting avenues for future research. Based on our findings, we suggest the following directions:

- **Adaptive and Autonomous Hyperparameter Tuning:** Developing a self-tuning mechanism for the validation frequency V and the patience parameter s to automatically detect the optimal balance between computational budget and solution quality, thereby reducing user dependence.

- **Variance-Based Validation Frequency:** Designing a heuristic to dynamically adjust the validation frequency V based on the observed variance in the deterministic Validation Score, ensuring frequent checks only when stochastic noise is high.

- **Trend Analysis for Patience (s):** Employing advanced statistical monitoring or time-series analysis techniques to robustly differentiate between minor stochastic oscillation and persistent performance stagnation, allowing for an automatic adjustment of s.

- **Scalable Stochastic Validation Schemes:** Investigating the replacement of the expensive full-batch validation step with a low-cost, statistically sound mini-batch validation

scheme, leveraging concentration inequalities to guarantee high-confidence early stopping decisions.

- **Extension to Nonaffine Constraints:** Investigating the performance, stability, and theoretical guarantees of the EB-SGD approach when applied to problems with general nonlinear (nonaffine) convex constraints, moving beyond the current focus on affine constraints.

- **Generalization to Nonconvex Optimization:** Exploring the empirical performance and potential convergence guarantees of the EB-SGD algorithm within the domain of nonconvex optimization, specifically for large-scale training of deep neural networks.

- **Tail-Averaging Variants:** Since our experiments demonstrated that standard Polyak–Ruppert averaging suffers from "memory lag" due to the shifting central path, future work should investigate "Tail-Averaging" (or windowed averaging) schemes. These methods average only the most recent iterates (e.g., the last suffix of the trajectory), potentially combining the variance-reduction benefits of averaging with the dynamic tracking capability required for adaptive barrier methods.

- **High-Dimensional Scalability:** While this study established the stability and robustness of EB-SGD on the standard benchmark ($d = 50$), investigating the algorithm's scalability to significantly higher-dimensional problems (e.g., $d \geq 100$) remains a critical direction. Future work will focus on analyzing the convergence rates and computational trade-offs in such high-dimensional regimes under varying constraint geometries.

Funding

This research received no external funding.

Conflicts of Interest

The authors declare no conflicts of interest. article.

References

[1] Bertsekas, D.P. *Nonlinear programming*, 2nd ed., Athena Scientific, 1999.

[2] Billingsley, P. *Convergence of probability measures*, John Wiley & Sons, 1968.

[3] Bottou, L., Curtis, F.E. and Nocedal, J. *Optimization methods for large-scale machine learning,* SIAM Rev. 60(2) (2018), 223–311.

[4] Boyd, S. and Vandenberghe, L. *Convex optimization*, Cambridge University Press, 2004.

[5] Bubeck, S. *Convex optimization: Algorithms and complexity,* Found. Trends Mach. Learn. 8(3–4) (2015), 231–358.

[6] Dieuleveut, A., Durmus, A. and Bach, F. *Bridging the gap between constant step size stochastic gradient descent and Markov chains,* arXiv preprint arXiv:1707.06386 (2018).

[7] Dimitrieski, N., Cao, J. and Ebenbauer, C. *Stochastic gradient descent for constrained optimization based on adaptive relaxed barrier functions,* Int. J. Robust. Nonlinear Control. (2025).

[8] Feller, C. and Ebenbauer, C. *A stabilizing iteration scheme for model predictive control based on relaxed barrier functions,* Automatica 80 (2017), 328–339.

[9] Fiacco, A.V. and McCormick, G.P. *Nonlinear programming: Sequential unconstrained minimization techniques*, John Wiley & Sons, 1968.

[10] Garrigos, G. and Gower, R.M. *Handbook of convergence theorems for (stochastic) gradient methods,* arXiv preprint arXiv:2301.11235 (2023).

[11] Gower, R.M., Loizou, N., Qian, X., Sailanbayev, A., Shulgin, E. and Richtárik, P. *SGD: General analysis and improved rates,* arXiv preprint arXiv:1901.09401 (2019).

[12] Li, M., Grigas, P. and Atamtürk, A. *New penalized stochastic gradient methods for linearly constrained strongly convex optimization,* SIAM J. Optim. 32(2) (2022), 859–887.

[13] Liu, J. and Yuan, Y. *On almost sure convergence rates of stochastic gradient methods*, In Proceedings of the 35th Annual Conference on Learning Theory, PMLR, 178 (2022), 1–21.

[14] Nocedal, J. and Wright, S.J. *Numerical optimization*, 2nd ed., Springer, 2006.

[15] Polyak, B.T. and Juditsky, A.B. *Acceleration of stochastic approximation by averaging,* SIAM J. Control Optim. 30(4) (1992), 838–855.

[16] Prechelt, L. Early stopping but when?, In: Orr, G.B., Müller, K.R. (eds.) *Neural networks: Tricks of the trade*, Lecture Notes in Computer Science, vol 1524, Springer, Berlin, Heidelberg, 1998, 55–69.

[17] Robbins, H. and Monro, S. *A stochastic approximation method,* Ann. Math. Stat. 22(3) (1951), 400–407.

[18] Rockafellar, R.T. *Convex analysis*, Princeton University Press, 1970.

[19] Roy, S.K. and Harandi, M. *Constrained stochastic gradient descent: The good practice*, In Int. Conf. Digit. Image Comput. Tech. Appl. (DICTA), IEEE, 2017, 1–8.

[20] Rudin, W. *Principles of mathematical analysis*, 3rd ed., McGraw-Hill, 1976.

[21] Wang, M. and Bertsekas, D.P. *Incremental constraint projection methods for variational inequalities,* Math. Program. 150 (2015), 321–363.

[22] Wang, X., Ma, S. and Yuan, Y. *Penalty methods with stochastic approximation for stochastic nonlinear programming,* arXiv preprint arXiv:1605.05609 (2016).

[23] Yan, Y. and Xu, Y. *Adaptive primal-dual stochastic gradient method for expectation-constrained convex stochastic programs,* Math. Program. Comput. 14(2) (2022), 319–363.

[24] Zhang, T. *Solving large scale linear prediction problems using stochastic gradient descent algorithms,* In Proc. 21st Int. Conf. Mach. Learn. (ICML), 2004, 116.