Journal of Machine Learning Research 27 (2026) 1-43 Submitted 3/25; Revised 10/25; Published 5/26

A Single-Loop Stochastic Proximal Quasi-Newton Method for Large-Scale Nonsmooth Convex Optimization

Yongcun Song YONGCUN.SONG@NTU.EDU.SG
Division of Mathematical Sciences, School of Physical and Mathematical Sciences
Nanyang Technological University
21 Nanyang Link, 637371, Singapore.

Zimeng Wang ZIMENG.WANG@UMU.SE
Department of Mathematics and Mathematical Statistics
Umeå University
90187 Umeå, Sweden

Xiaoming Yuan XMYUAN@HKU.HK
Department of Mathematics
The University of Hong Kong
Hong Kong, China

Hangrui Yue YUEHANGRUI@GMAIL.COM
School of Mathematical Sciences
Nankai University
Tianjin, 300071, China

Editor: Silvia Villa

Abstract

We propose a new stochastic proximal quasi-Newton method for minimizing the sum of two convex functions in the particular context that one of the functions is the average of a large number of smooth functions and the other one is nonsmooth. The new method integrates a simple single-loop SVRG (L-SVRG) technique for sampling the gradient and a stochastic limited-memory BFGS (L-BFGS) scheme for approximating the Hessian of the smooth function components. The globally linear convergence rate of the new method is proved under mild assumptions. It is also shown that the new method covers a proximal variant of the L-SVRG as a special case, and it allows for various generalization through the integration with other variance reduction methods. For example, the L-SVRG can be replaced with the SAGA or SEGA in the proposed new method and thus other new stochastic proximal quasi-Newton methods with rigorously guaranteed convergence can be proposed accordingly. Moreover, we meticulously analyze the resulting nonsmooth subproblem at each iteration and leverage a compact representation of the L-BFGS matrix with the storage of some auxiliary matrices. As a result, we propose a very efficient and easily implementable semismooth Newton solver for solving the involved subproblems, whose arithmetic operations per iteration are merely order of $O(d)$, where d denotes the dimensionality of the problem. With this efficient inner solver, the new method performs well and its numerical efficiency is validated through extensive experiments on a regularized logistic regression problem.

Keywords: convex optimization, stochastic optimization, nonsmooth optimization, quasi-Newton, variance reduction, semismooth Newton, large-scale, globally linear convergence

MSC codes. 65K05, 90C25, 90C53, 90C15, 90C06

1. Introduction

We consider the following nonsmooth convex optimization problem:

$$\min_{x\in\mathbb{R}^d} F(x) := \underbrace{\frac{1}{n}\sum_{i=1}^{n} f_i(x)}_{f(x)} + h(x), \tag{1.1}$$

in which $f : \mathbb{R}^d \to \mathbb{R}$ is the average of n function components and each component $f_i : \mathbb{R}^d \to \mathbb{R}$ is convex and differentiable (hence f), and the function $h : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ is convex, proper, and lower semicontinuous, but possibly nonsmooth. Problem (1.1) with large n is prevalent within the machine learning community, known as regularized empirical risk minimization (see, e.g., (Hastie et al., 2009)), where each component f_i represents the loss associated with the i-th data sample, while h is a manually incorporated regularization function aimed at improving model stability or promoting sparsity in the model parameters. Typical machine learning models that fall into the frame of problem (1.1) include and are not limited to LASSO (Tibshirani, 1996), support vector machine (Cortes and Vapnik, 1995), and regularized logistic regression (Berkson, 1944).

1.1 Deterministic methods

For the generic f, it is common to solve problem (1.1) via considering its approximation scheme

$$x_{k+1} = \arg\min_{x\in\mathbb{R}^d} \left\{ f(x_k) + \nabla f(x_k)^\top (x - x_k) + \frac{1}{2\eta_k}\|x - x_k\|^2 + h(x) \right\}, \tag{1.2}$$

with $\eta_k > 0$. That is, the smooth function f is approximated by a simpler quadratic function iteratively. With the notation of proximal operator $\mathrm{prox}_{\eta,\, h}(\cdot)$ (see (2.2) for the definition), (1.2) can be equivalently written as

$$x_{k+1} = \mathrm{prox}_{\eta_k,\, h}(x_k - \eta_k \nabla f(x_k)). \tag{1.3}$$

The iterative scheme (1.3) is also called the proximal gradient (PG) method (Lions and Mercier, 1979; Passty, 1979). Depending on h, (1.3) could be easy enough to have a closed-form solution, and a set of classic algorithms can be rendered from (1.3) with different specifications of h. Examples include the projected gradient method (Levitin and Polyak, 1966) when h is the indicator function of a convex and closed set in $\mathbb{R}^d$, and the iterative shrinkage-thresholding algorithm (ISTA), as well as its faster version (FISTA) in (Beck and Teboulle, 2009) when $h = \lambda\|\cdot\|_1$ for some $\lambda > 0$. It is analyzed in, e.g., (Beck and Teboulle, 2009; Nesterov, 2013), that the PG methods exhibit sublinear and linear convergence rates when the smooth function f is convex and strongly convex, respectively. Nevertheless, the approximated model (1.2) only involves the gradient information of f, while no information

of the Hessian of f is considered. Hence, implementation of the resulting iterations is relatively easier, yet approximation to the original problem (1.1) is less accurate and thus the theoretical convergence rate of such an algorithm is of lower order.

To achieve higher-order convergence rates and further acceleration, it is necessary to incorporate the Hessian of f or its approximation into the approximation of the original problem (1.1). Some proximal Newton-type methods were thus proposed in the literature, to mention a few, see (Becker and Fadili, 2012; Becker et al., 2019; Byrd et al., 2016a,c; Lee et al., 2014; Mordukhovich and Sarabi, 2021; Mordukhovich et al., 2023; Stella et al., 2017). These proximal Newton-type methods have a common feature that the original problem (1.1) is approximated via the scheme

$$x_{k+1} = \underset{x\in\mathbb{R}^d}{\arg\min} \left\{ f(x_k) + \nabla f(x_k)^\top (x - x_k) + \frac{1}{2\eta_k}(x - x_k)^\top B_k (x - x_k) + h(x) \right\}, \quad (1.4)$$

where $\eta_k > 0$ can be regarded as the step size and $B_k \in \mathbb{R}^{d\times d}$ is the Hessian matrix $\nabla^2 f(x_k)$ or its approximation. In particular, if B_k is set as $\nabla^2 f(x_k)$ or constructed by a quasi-Newton strategy, we obtain the proximal Newton or proximal quasi-Newton method (Lee et al., 2014) accordingly. Note that the PG method (1.3) is recovered from (1.4) when B_k is the identity matrix. Of course, compared with (1.2), solving (1.4) is generally more difficult due to the presence of the general B_k. Hence, the computation per iteration of such a resulting algorithm is more time-consuming. Nevertheless, the curvature information of f can be well captured in (1.4) and the original problem (1.1) is approximated more accurately. Hence, the theoretical convergence rate of such an algorithm based on (1.4) is of higher order. Indeed, it has been shown in (Lee et al., 2014) that, under the same assumptions as those for the PG, the proximal Newton and the proximal quasi-Newton methods with unit length step sizes can achieve locally quadratic and superlinear convergence, respectively.

For proximal quasi-Newton methods, it is crucial to construct the Hessian approximation B_k efficiently. In this regard, the BFGS (Fletcher, 2000) and the limited-memory BFGS (L-BFGS) (Liu and Nocedal, 1989) are widely used in the literature. In the BFGS, the inverse Hessian approximation $H_k := B_k^{-1}$ is necessary to be stored to update H_{k+1}, which requires $O(d^2)$ memory. By contrast, the L-BFGS only needs to store m correction pairs

$$\{(s_i, y_i)\}_{i=k-m+1}^{k} \text{ with } s_i = x_{i+1} - x_i \text{ and } y_i = \nabla f(x_{i+1}) - \nabla f(x_i)$$

to construct H_{k+1} and hence reduces the storage costs to $O(md)$, where $m > 0$ is the memory size. Consequently, the L-BFGS is preferred for practical implementations, particularly when d is large.

These deterministic methods are efficient when the full gradient $\nabla f := \frac{1}{n}\sum_{i=1}^n \nabla f_i$ is accessible. They find extensive applications across various fields, including barrier representation of feasible set (Nesterov and Nemirovskii, 1994), LASSO (Tibshirani, 1996), and many others. However, in modern machine learning tasks, the number of components n involved in (1.1) is usually very large, which makes the full gradient ∇f very expensive to compute and thus deteriorates the numerical efficiency.

1.2 Stochastic methods

In the context of machine learning, a common idea for applying gradient-based methods on large-scale optimization problems is to replace a full gradient with a certain stochastic

approximation, such as a single or a small batch of gradient components. This results in the class of stochastic algorithms, which can be traced back to the seminal work (Robbins and Monro, 1951). The predominant methodology within this class advocates the stochastic gradient descent (SGD) method (Robbins and Monro, 1951) and its variants (Duchi et al., 2011; Kingma and Ba, 2014; Loizou and Richtárik, 2020; Qian, 1999; Zeiler, 2012), which have demonstrated significant success across a wide range of tasks in machine learning (Bottou and Bousquet, 2007; Bottou et al., 1991; LeCun et al., 1998). The per-iteration cost of SGD and its variants is relatively low but their worst-case convergence rates are only sublinear even for strongly convex functions, as the step sizes are necessary to vanish to ensure convergence. To address this issue, various approaches have been proposed in the literature, including variance reduction (VR) methods (Bottou et al., 2018; Defazio et al., 2014; Gower et al., 2020; Hanzely et al., 2018; Johnson and Zhang, 2013; Roux et al., 2012; Shalev-Shwartz and Zhang, 2013; Xiao and Zhang, 2014) and stochastic Newton-type methods (Byrd et al., 2016b; Guo et al., 2023; Kasai et al., 2018; Lucchi et al., 2015; Moritz et al., 2016; Schraudolph et al., 2007; Zhao et al., 2017).

Different from the SGD that uses one or more gradient components directly as the full gradient approximation, VR methods use them to update it, such that the variance of the gradient approximation vanishes as iterations progress, and hence often exhibit faster convergence both in theory and practice. Typical VR methods include SAG (Roux et al., 2012), SAGA (Defazio et al., 2014), SDCA (Shalev-Shwartz and Zhang, 2013), SVRG (Johnson and Zhang, 2013; Xiao and Zhang, 2014), SEGA (Hanzely et al., 2018) and so on. These VR methods admit constant step sizes and thus can achieve linear convergence rates for strongly convex objectives. Among these VR methods, the SVRG stands out as a popular choice due to its low storage cost and satisfactory performance in various tasks (Johnson and Zhang, 2013; Li et al., 2020). The SVRG method is mainly featured by a double-loop structure: in the outer loop, a full gradient is computed at a reference point, and this full gradient is then used in the inner loop to modify the gradient approximations.

Stochastic Newton-type methods incorporate Hessian information and hence often exhibit faster convergence than stochastic first-order methods. Representative stochastic Newton-type methods include the online BFGS (oBFGS), the online L-BFGS (oL-BFGS) (Schraudolph et al., 2007), the stochastic quasi-Newton (SQN) method (Byrd et al., 2016b) and the stochastic L-BFGS method (Moritz et al., 2016). The oBFGS and the oL-BFGS represent pioneering approaches that generalize the BFGS and the L-BFGS to stochastic settings. The SQN method leverages a modified L-BFGS scheme that applies a Hessian-vector product to update the Hessian approximation. While the worst-case convergence rate of SQN is sublinear, practical performance of SQN surpasses that of SGD significantly, indicating the substantial advantage offered by incorporating Hessian information. The stochastic L-BFGS (Moritz et al., 2016) builds upon the SQN and the SVRG, and is the first stochastic quasi-Newton algorithm achieving globally linear convergence for strongly convex and smooth objectives. Some methods that combine SQN with the SVRG can be referred to (Kasai et al., 2018; Lucchi et al., 2015; Zhao et al., 2017).

For solving (1.1), some stochastic PG methods have been studied in (Bertsekas, 2011; Duchi and Singer, 2009; Langford et al., 2009). Additionally, some methods combining the VR techniques and stochastic Newton-type methods have been proposed in the literature. For instance, a general framework designed in (Yang et al., 2021) performs an additional

stochastic proximal gradient step at each iteration after a Newton-type update and allows for integration with VR methods. The subsampled Newton method with cubic regularization proposed in (Zhang et al., 2022) assumes access to the true Hessian of each component function, and aggregates the stochastic gradient and Hessian approximation using a subset of components according to the SVRG scheme. An inexact subsampled proximal Newton method combined with the SVRG is developed in (Wang and Zhang, 2019), and its convergence results are established for a special case of (1.1), where each component f_i is a loss of a linear predictor. In (Luo et al., 2016), a stochastic proximal quasi-Newton method that combines the SQN method with the SVRG technique is introduced and analyzed. Specifically, at the k-th iteration of this method, the new iterate x_{k+1} is generated via (1.4) or its approximation, with $\nabla f(x_k)$ replaced by a stochastic gradient generated via the SVRG.

Note that the aforementioned stochastic methods that incorporate VR techniques simply adopt the SVRG scheme, see (Kasai et al., 2018; Lucchi et al., 2015; Luo et al., 2016; Moritz et al., 2016; Wang and Zhang, 2019; Zhang et al., 2022; Zhao et al., 2017). Despite achieving fast theoretical convergence, these methods are limited by the inherent drawbacks of the SVRG resulting from its double-loop structure. For instance, existing convergence results of the SVRG are established for the reference point, updating which can be computationally expensive since a full gradient computation is required. Additionally, as pointed out in (Kovalev et al., 2020), the practical performance of the SVRG is heavily influenced by the number of inner iterations, but there lacks theoretical guidance on selecting its value optimally. Moreover, empirical observations have indicated that a simplified implementation of the SVRG often exhibits superior performance in practice but without convergence guarantee, resulting in a discrepancy between theory and practice (see Section 3.1 for the details).

1.3 Our contributions

Existing stochastic quasi-Newton methods for solving the nonsmooth problem (1.1) suffer from either slow sublinear convergence or practical issues arising from the double-loop structure of the SVRG. To overcome these limitations, we combine a stochastic L-BFGS scheme (Byrd et al., 2016b) based on the Hessian-vector product with the recently introduced loopless SVRG (L-SVRG) (Kovalev et al., 2020) and hence propose a novel stochastic proximal quasi-Newton method for solving (1.1). Moreover, we provide rigorous analysis demonstrating its rapid globally linear convergence and present a highly efficient way for numerical implementation.

To adapt the L-BFGS for stochastic optimization, an intuitive way is to generate the correction vector y by differencing stochastic gradients based on small samples. Unfortunately, as pointed out in (Byrd et al., 2016b), this approach may lead to a biased estimate of the Hessian approximation. To address this issue, we follow (Byrd et al., 2016b) and decouple the calculations of the stochastic gradients and Hessian approximations using Hessian-vector products, which also provide the flexibility for periodic updates of the Hessian approximations. Moreover, to overcome the limitations brought by the double-loop structure of the SVRG, the L-SVRG eliminates the inner loop by incorporating a probability-based update of the reference point at each iteration. It was shown in (Kovalev et al., 2020) that, compared with the SVRG, the L-SVRG achieves a linear convergence rate for strongly convex

objectives without increasing storage cost while exhibiting a simplified structure. Furthermore, empirical results demonstrate that the L-SVRG outperforms the SVRG in practical applications.

Combining the ideas of the stochastic L-BFGS and the L-SVRG, our proposed method proceeds by performing the following iterative scheme

$$x_{k+1} = \arg\min_{x\in\mathbb{R}^d} \left\{ f(x_k) + v_k^\top (x - x_k) + \frac{1}{2\eta_k}(x - x_k)^\top B_k (x - x_k) + h(x) \right\}, \tag{1.5}$$

where $v_k \in \mathbb{R}^d$ denotes the stochastic gradient obtained by the L-SVRG, and $B_k \in \mathbb{R}^{d\times d}$ is constructed via m correction pairs generated by the approach outlined in (Byrd et al., 2016b). The proposed method surpasses SVRG-based proximal quasi-Newton methods by featuring a unique single-loop structure to update v_k, which makes it easier and cheaper to implement. In addition to the simple structure, our method demonstrates a rapid globally linear convergence rate under the same assumptions as those for the existing stochastic proximal quasi-Newton methods in the literature. To the best of our knowledge, our method seems to be the first stochastic proximal quasi-Newton method that incorporates a single-loop stochastic gradient updating scheme while preserving the desirable property of linear convergence. Moreover, as a special case of our method, we obtain a proximal extension of the L-SVRG for addressing (1.1). We also explore some generalizations of our method, where the stochastic gradients are generated by other VR methods like the SAGA and the SEGA. Notably, linear convergence results for these variants can be established by extending the mathematical paradigm used in analyzing the original method (see Section 4.3 for more details).

Like other proximal quasi-Newton methods, the practical efficiency of our method heavily relies on the rapid solution of the subproblem (1.5). If a first-order algorithm is employed for this purpose, one may see slow convergence and struggle in pursuing highly accurate solutions. To enhance the practical applicability of our method, we analyze the nonsmooth subproblem (1.5) meticulously and propose a new inner solver for this subproblem that can obtain highly accurate solutions very fast. Specifically, we first transform the nonsmooth subproblem into an equivalent smooth dual formulation. Then, we propose a Semismooth Newton (SSN) method (Qi and Sun, 1993) along with a line search scheme to efficiently solve the dual problem by carefully exploring its specific mathematical structure. To further reduce the computational cost, we develop an efficient numerical implementation for the proposed SSN method using a compact representation of the L-BFGS matrix and introducing auxiliary matrices that can be updated in a highly efficient manner. Generally, such an implementation requires only $O(\iota md)$ multiplications and $O(\iota m^2 d)$ additions, along with $O(\iota d)$ simple non-linear operations (e.g., projections). Here, $\iota > 0$ (typically a single-digit number) denotes the number of SSN iterations. It is noteworthy that our SSN method can effectively address the nonsmooth problem (1.5) with general v_k and positive definite matrix B_k, which is commonly incorporated as an inner subproblem in various proximal Newton-type methods, such as the proximal quasi-Newton method (Becker and Fadili, 2012; Becker et al., 2019; Byrd et al., 2016c; Lee et al., 2014), the stochastic proximal Newton-type method integrated with the SVRG (Luo et al., 2016; Wang and Zhang, 2019) and the proximal subsampled Newton method (Liu et al., 2017). Consequently, the proposed SSN

solver can be readily integrated into these proximal Newton-type methods as a subroutine, leading to immediate enhancements in their numerical performance.

1.4 Organization

The rest of the paper is organized as follows. Section 2 provides some notations and preliminary results that are used throughout this paper. Then the proposed algorithm is introduced in Section 3, followed by the convergence analyses in Section 4. Section 5 presents an SSN approach for efficiently solving the subproblem (1.5). Section 6 is devoted to numerical experiments and Section 7 concludes the paper.

2. Preliminaries

In this section, we summarize some notations and preliminary results that are used throughout the paper. First, we denote by I_d the $d \times d$ identity matrix and omit the subscript d when it is clear from the context. For a vector $v \in \mathbb{R}^d$, we denote by $\|v\|$ its Euclidean norm and by $\|v\|_B := \sqrt{v^\top B v}$ the B-norm of v, with $B \in \mathbb{R}^{d\times d}$ being a positive definite matrix. For symmetric matrices $A, B \in \mathbb{R}^{d\times d}$, we write $A \preceq B$ if the matrix $B - A$ is positive semidefinite.

Let $(\Omega, \mathcal{F}, \mathrm{P})$ be the underlying probability space. We denote by $\{x_k\}_{k\geq 0} \subset \mathbb{R}^d$ the sequence of iterates generated by a stochastic algorithm, driven by a sequence of independent noise variables $\{\xi_k\}_{k\geq 1}$. Let $\{\mathcal{F}_k\}_{k\geq 0}$ be the natural filtration of this process, i.e., $\mathcal{F}_k = \sigma(x_0, \xi_1, \xi_2, \ldots, \xi_k)$ represents the σ-algebra generated by x_0 and $\{\xi_i\}_{i=1}^k$. We denote the conditional expectation given $\mathcal{F}_k$ by $\mathbb{E}_k[\cdot] := \mathbb{E}[\cdot \mid \mathcal{F}_k]$, which represents the expectation given all information available at step k.

For a set $\mathcal{S} \subset [n] := \{1, 2, \ldots, n\}$ with cardinality $|\mathcal{S}|$ and functions $\{f_i\}_{i=1}^n$, we define

$$f_{\mathcal{S}}(x) = \frac{1}{|\mathcal{S}|} \sum_{i\in\mathcal{S}} f_i(x), \ \forall x \in \mathbb{R}^d.$$

If each f_i is smooth enough, we define $\nabla f_{\mathcal{S}}(x)$ and $\nabla^2 f_{\mathcal{S}}(x)$ in the same way.

A function $f : \mathbb{R}^d \to \mathbb{R}$ is said to be L-smooth if it is continuously differentiable and its gradient ∇f is L-Lipschitz continuous. If f is L-smooth and convex, then we have

$$f(y) \geq f(x) + \nabla f(x)^\top (y - x) + \frac{1}{2L}\|\nabla f(y) - \nabla f(x)\|^2, \ \forall x, y \in \mathbb{R}^d. \tag{2.1}$$

One can refer to (Nesterov, 2003) for a proof of (2.1).

Using (2.1), we have the following result that is widely used in the convergence analysis of proximal-type algorithms.

Lemma 1 ((Xiao and Zhang, 2014, Lemma 1)) *Consider F as defined in* (1.1)*. Suppose for each $i \in [n]$, f_i is convex and L-smooth. If $x^* \in \arg\min_{x\in\mathbb{R}^d} F(x)$, then we have*

$$\frac{1}{n}\sum_{i=1}^n \|\nabla f_i(x) - \nabla f_i(x^*)\|^2 \leq 2L(F(x) - F(x^*)).$$

Let $F : \mathbb{R}^d \to \mathbb{R}$ be a convex function that attains a global minimizer, we denote $F^* := \min_{x\in\mathbb{R}^d} F(x)$. If F is moreover strongly convex, then it admits a unique minimizer $x^* := \arg\min_{x\in\mathbb{R}^d} F(x)$.

For a proper, lower semicontinuous, and convex function $h : \mathbb{R}^d \to \mathbb{R}\cup\{+\infty\}$, we denote by $\partial h(x)$ the set of all subgradients of h at $x \in \mathbb{R}^d$, i.e.,

$$\partial h(x) := \{\gamma \in \mathbb{R}^d | h(y) - h(x) - \langle \gamma, y - x\rangle \geq 0, \forall y \in \mathbb{R}^d\}.$$

Its proximal operator with parameter $\eta > 0$ is defined as

$$\mathrm{prox}_{\eta,\, h}(x) = \arg\min_{z\in\mathbb{R}^d} \left\{ h(z) + \frac{1}{2\eta}\|z - x\|^2 \right\},\ \forall x \in \mathbb{R}^d, \tag{2.2}$$

and the Moreau envelope of h with parameter $\eta > 0$ is given by

$$\mathcal{M}_{\eta,\, h}(x) = \min_{z\in\mathbb{R}^d}\ h(z) + \frac{1}{2\eta}\|z - x\|^2,\ \forall x \in \mathbb{R}^d.$$

It has been shown in (Rockafellar, 1997) that $\mathcal{M}_{\eta,\, h}$ is differentiable everywhere and its gradient is given by

$$\nabla\mathcal{M}_{\eta,\, h}(x) = \frac{1}{\eta}\left(x - \mathrm{prox}_{\eta,\, h}(x)\right),\ \forall x \in \mathbb{R}^d. \tag{2.3}$$

For simplicity, we denote $\mathrm{prox}_h(x) := \mathrm{prox}_{1,\, h}(x)$ and $\mathcal{M}_h(x) := \mathcal{M}_{1,\, h}(x)$.

Let $B \in \mathbb{R}^{d\times d}$ be a positive definite matrix, we define the scaled proximal operator of h with parameter $\eta > 0$ as

$$\mathrm{prox}^B_{\eta,\, h}(x) = \arg\min_{z\in\mathbb{R}^d} \left\{ h(z) + \frac{1}{2\eta}\|z - x\|_B^2 \right\},\ \forall x \in \mathbb{R}^d.$$

For any $\eta > 0$, the scaled proximal operator $\mathrm{prox}^B_{\eta,\, h}(\cdot)$ is nonexpansive in the B-norm, that is,

$$\left\|\mathrm{prox}^B_{\eta,\, h}(x) - \mathrm{prox}^B_{\eta,\, h}(y)\right\|_B \leq \|x - y\|_B,\ \forall x, y \in \mathbb{R}^d. \tag{2.4}$$

Next, we introduce an important lemma that represents an extension of (Xiao and Zhang, 2014, Lemma 3) by incorporating additional Hessian information. A similar result was mentioned in (Luo et al., 2016) but without proof. For completeness, we present a proof of the lemma.

Lemma 2 *Let $F(x)$ be defined in (1.1) and suppose that f is μ-strongly convex and L-smooth. For any $\eta > 0$, $x, v \in \mathbb{R}^d$ and positive definite matrix $B \in \mathbb{R}^{d\times d}$, let $x^+ := \mathrm{prox}^B_{\eta,\, h}(x - \eta B^{-1}v)$. Define $g := \frac{1}{\eta}(x - x^+)$ and $\Delta := v - \nabla f(x)$, then for any $y \in \mathbb{R}^d$, it holds that*

$$F(y) \geq F(x^+) + g^\top B(y - x) + \Delta^\top(x^+ - y) + \eta\|g\|_B^2 - \frac{L\eta^2}{2}\|g\|^2 + \frac{\mu}{2}\|y - x\|^2.$$

Proof Let $H = B^{-1}$. It follows from the definition of x^+ that

$$x^+ = \arg\min_{z\in\mathbb{R}^d} \left\{ h(z) + \frac{1}{2\eta} \|z - (x - \eta H v)\|_B^2 \right\}. \tag{2.5}$$

Then it is easy to show that the first-order optimality condition of (2.5) reads

$$\exists\, \xi \in \partial h(x^+), s.t. B(x^+ - x + \eta H v) + \eta\xi = 0,$$

which together with $g := \frac{1}{\eta}(x - x^+)$ implies that

$$\xi = Bg - v.$$

Since f is μ-strongly convex and h is convex, we have

$$F(y) = f(y) + h(y) \geq f(x) + \nabla f(x)^\top (y - x) + \frac{\mu}{2}\|y - x\|^2 + h(x^+) + \xi^\top (y - x^+). \tag{2.6}$$

Since f is L-smooth, we have that

$$f(x) \geq f(x^+) - \nabla f(x)^\top (x^+ - x) - \frac{L}{2}\|x^+ - x\|^2. \tag{2.7}$$

Applying (2.7) on (2.6) leads to

$$\begin{aligned}
F(y) \geq & f(x^+) - \nabla f(x)^\top (x^+ - x) - \frac{L}{2}\|x^+ - x\|^2 + \nabla f(x)^\top (y - x) + \frac{\mu}{2}\|y - x\|^2 \\
& + h(x^+) + \xi^\top (y - x^+) \\
= & F(x^+) + \nabla f(x)^\top (y - x^+) + (Bg - v)^\top (y - x^+) + \frac{\mu}{2}\|y - x\|^2 - \frac{L}{2}\|x^+ - x\|^2 \\
= & F(x^+) + (Bg)^\top (y - x + x - x^+) + \Delta^\top (x^+ - y) - \frac{L\eta^2}{2}\|g\|^2 + \frac{\mu}{2}\|y - x\|^2 \\
= & F(x^+) + g^\top B(y - x) + \Delta^\top (x^+ - y) + \eta\|g\|_B^2 - \frac{L\eta^2}{2}\|g\|^2 + \frac{\mu}{2}\|y - x\|^2,
\end{aligned}$$

which completes the proof. ■

3. Algorithm

In this section, we present the proposed single-loop stochastic proximal quasi-Newton method for solving problem (1.1). Typically, the iterative scheme of stochastic proximal Newton-type methods for solving (1.1) can be uniformly written as

$$\begin{aligned}
x_{k+1} &= \arg\min_{x\in\mathbb{R}^d} \left\{ v_k^\top (x - x_k) + \frac{1}{2\eta_k} \|x - x_k\|_{B_k}^2 + h(x) \right\} \\
&= \mathrm{prox}_{\eta_k,\, h}^{B_k}(x_k - \eta_k H_k v_k),
\end{aligned} \tag{3.1}$$

where $v_k \in \mathbb{R}^d$ represents a stochastic gradient, $B_k \in \mathbb{R}^{d\times d}$ serves as an approximation of the Hessian matrix $\nabla^2 f(x_k)$, and $H_k := B_k^{-1} \in \mathbb{R}^{d\times d}$. It is easy to see that the effectiveness

of (3.1) hinges on well-designed v_k and B_k (Byrd et al., 2016b; Yang et al., 2021). In the rest part of this section, we shall elaborate on the construction of v_k and B_k. In particular, we advocate combing the L-SVRG (Kovalev et al., 2020) with the stochastic L-BFGS based on the Hessian-vector product (Byrd et al., 2016b) to compute v_k and B_k, and thus propose an efficient and easily implementable stochastic proximal quasi-Newton method for solving (1.1).

3.1 Construction of v_k

In general, the stochastic gradient v_k serves as an unbiased estimator of $\nabla f(x_k)$ (see, e.g., (Defazio et al., 2014; Duchi et al., 2011; Johnson and Zhang, 2013; Kovalev et al., 2020; Nemirovski et al., 2009; Robbins and Monro, 1951)). In the design of SQN (Byrd et al., 2016b), v_k is set as $\nabla f_{\mathcal{S}_k}(x_k)$, where $\mathcal{S}_k \subset [n]$ is randomly selected. However, approximating $\nabla f(x_k)$ in this way necessitates the use of a sequence of decreasing step sizes to guarantee the convergence. This requirement leads to a sublinear convergence rate for SQN, even if $h(x) \equiv 0$ and f is strongly convex. To accelerate the convergence, a widely adopted strategy is to use VR techniques, with the SVRG method (Johnson and Zhang, 2013) being a popular and effective choice. The SVRG method uses reference points and has a double-loop structure. To be concrete, at the s-th outer loop, given a reference point $\tilde{w}^s$, we set the initial iterate x_1^s of the inner loop to be $\tilde{w}^s$. Then, at the k-th inner loop, the new iterate x_{k+1}^s is generated by $x_{k+1}^s = x_k^s - \eta v_k^s$, where $\eta > 0$ is a predefined step size and v_k^s is defined as

$$v_k^s = \nabla f_{i_k^s}(x_k^s) - \nabla f_{i_k^s}(\tilde{w}^s) + \nabla f(\tilde{w}^s), \tag{3.2}$$

where $i_k^s \in [n]$ is randomly selected at each iteration. Upon completion of the inner loop, the new reference point $\tilde{w}^{s+1}$ is updated using the sequence $\{x_k^s\}_{k=1}^{l_s+1}$, where $l_s > 0$ denotes the number of inner iterations for the s-th outer loop. There are two practical options for updating $\tilde{w}^{s+1}$. The first one is to simply set

$$\tilde{w}^{s+1} = x_{l_s+1}^s. \tag{3.3}$$

Alternatively, one can update $\tilde{w}^{s+1}$ as the average of the sequence $\{x_k^s\}_{k=1}^{l_s}$, i.e.,

$$\tilde{w}^{s+1} = \frac{1}{l_s}\sum_{i=1}^{l_s} x_i^s. \tag{3.4}$$

It has been shown in (Johnson and Zhang, 2013; Xiao and Zhang, 2014) that, with the update scheme (3.4), the SVRG and its proximal variant converge linearly. In practice, the update (3.3) is often preferred due to its superior empirical performance and ease of implementation. However, convergence guarantee for the SVRG with (3.3) is still lacking in the literature, which creates a discrepancy between mathematical theory and practice for the SVRG method. The double-loop structure of the SVRG method also introduces additional practical challenges. For instance, as mentioned in (Kovalev et al., 2020), the SVRG is sensitive to the number of inner iterations, for which rigorous theoretical guidance has not yet been established. Additionally, the SVRG requires the reference point $\tilde{w}^s$ to be updated at each outer loop. For a new reference point $\tilde{w}^s$, a full gradient $\nabla f(\tilde{w}^s)$ is required to update v_k in (3.2), which may be computationally expensive.

To overcome the aforementioned limitations of the SVRG, we advocate the L-SVRG scheme (Kovalev et al., 2020). Compared with the SVRG, the L-SVRG eliminates the inner loops, resulting in a more efficient and simplified framework for designing v_k. Precisely, the L-SVRG scheme computes v_k as follows:

$$v_k = \nabla f_{i_k}(x_k) - \nabla f_{i_k}(w_k) + \nabla f(w_k), \tag{3.5}$$

where $i_k \in [n]$ is a randomly selected index and $w_k \in \mathbb{R}^d$ is a reference point. Different from the SVRG, we update the new reference point w_{k+1} as x_k with a small probability $p \in (0, 1)$, while keeping it unchanged with a probability of $1 - p$, i.e.,

$$w_{k+1} = \begin{cases} x_k & \text{with probability } p, \\ w_k & \text{with probability } 1 - p. \end{cases} \tag{3.6}$$

In practice, it is very common to choose a relatively small value for p (e.g., $p = O(1/n)$) to ensure that w_k remains unchanged for most of the iterations, resulting in substantial savings in computational cost. Moreover, similar to the SVRG, the L-SVRG also achieves a linear convergence rate as demonstrated in (Kovalev et al., 2020). Therefore, the L-SVRG provides a more practical and efficient alternative to the SVRG, offering both simple implementation and desirable linear convergence.

Remark 3 *Notice that it is natural to extend the stochastic gradient v_k to the batch version, i,e., replace v_k defined in* (3.5) *with*

$$v_k = \nabla f_{\mathcal{B}_k}(x_k) - \nabla f_{\mathcal{B}_k}(w_k) + \nabla f(w_k), \tag{3.7}$$

where $\mathcal{B}_k \subset [n]$ is independently selected at each iteration.

3.2 Construction of B_k

We consider the stochastic L-BFGS scheme and use a set of correction pairs $\{(s_j, y_j)\}_{j=1}^m$ with $s_j^\top y_j > 0$ to construct B_k, where $m > 0$ is the memory size. The correction pairs are generated using the Hessian-vector product scheme described in (Byrd et al., 2016b). Specifically, new correction pairs are computed every r iterations, and at most a specified number (denoted by l) of the latest computed correction pairs are stored. We introduce a separate superscript t to denote the number of correction pairs computed so far, where t is the integer division of k by r. Denoting the new correction pair as (s^t, y^t), it is generated based on a collection of average iterates, i.e.,

$$s^t = \bar{x}^t - \bar{x}^{t-1} \text{ and } y^t = \nabla^2 f_{\mathcal{S}^t}\left(\bar{x}^t\right) s^t, \tag{3.8}$$

where $\bar{x}^t := \frac{1}{r}\sum_{j=k-r+1}^{k} x_j$ and $\mathcal{S}^t$ is randomly sampled from $[n]$. It is noteworthy that the matrix $\nabla^2 f_{\mathcal{S}^t}\left(\bar{x}^t\right)$ does not need to be constructed explicitly, one can directly compute the Hessian-vector product $\nabla^2 f_{\mathcal{S}^t}\left(\bar{x}^t\right) s^t$, which can save lots of computational and storage costs. Taking $\{(s_j, y_j)\}_{j=1}^m (m \le l)$ as the latest computed m correction pairs, i.e., $s_j := s^{t-m+j}$ and $y_j := y^{t-m+j}$ for $j \in [m]$, the compact representation of the L-BFGS matrix B^t (see (Nocedal and Wright, 1999)) reads as :

$$B^t := \sigma_0 I - \begin{bmatrix} \sigma_0 S & Y \end{bmatrix} \begin{bmatrix} \sigma_0 S^\top S & L \\ L^\top & -D \end{bmatrix}^{-1} \begin{bmatrix} \sigma_0 S^\top \\ Y^\top \end{bmatrix}, \ \sigma_0 := \frac{(y_m)^\top y_m}{(y_m)^\top s_m}, \tag{3.9}$$

where

$$S := [s_1, \ldots, s_m] \in \mathbb{R}^{d\times m},\ Y := [y_1, \ldots, y_m] \in \mathbb{R}^{d\times m},\ D := \mathrm{diag}[s_1^\top y_1 \ldots s_m^\top y_m] \in \mathbb{R}^{m\times m},$$

and $L \in \mathbb{R}^{m\times m}$ is defined as

$$(L)_{i,j} = \begin{cases} s_i^\top y_j & \text{if } i > j, \\ 0 & \text{otherwise.} \end{cases}$$

At the initial stage (i.e., $k < r$), we set $B_k = I$. For $k \geq r$, we set $B_k = B^t$ and update it every r iterations. In addition, the matrix B^t does not need to be explicitly computed, as we will discuss in detail in Section 5.

It is worth mentioning an alternative stochastic L-BFGS scheme in (Berahas et al., 2016), which generates a correction pair (s_k, y_k) at every iteration. To be concrete, a small batch $S_k \subset [n]$ is generated at the k-th iteration such that $O_k := S_k \cap S_{k-1} \neq \emptyset$. Then the correction pair (s_k, y_k) is computed via

$$s_k = x_k - x_{k-1}, \quad y_k = \nabla f_{O_k}(x_k) - \nabla f_{O_k}(x_{k-1}).$$

Note that the construction of y_k in the above L-BFGS scheme does not require extra gradient computation, as O_k is a subset of both S_k and S_{k-1}. However, this approach may encounter practical challenges, such as the inability to ensure the independence of the samples $\{S_k\}$ or the insufficiency of the overlap set O_k to offer valuable Hessian information. In contrast, the stochastic L-BFGS Hessian-vector product scheme (3.8), as highlighted in (Byrd et al., 2016b), addresses this limitation and achieves a stable Hessian approximation by decoupling the computation of s and the construction of y. It also allows for the incorporation of new Hessian information periodically, and this flexibility enhances the performance and efficiency of the Hessian-vector product scheme (3.8).

Remark 4 *It is important to note that even the deterministic version of L-BFGS does not achieve locally superlinear convergence; generally, only linear convergence rate can be attained. This limitation can be attributed to the failure to meet the Dennis-Moré criterion (Dennis and Moré, 1974), which is both necessary and sufficient for the superlinear convergence of quasi-Newton methods (Nocedal and Wright, 1999). Therefore, in Section 4, we focus on studying the global convergence behavior of the proposed method.*

3.3 A single-loop stochastic proximal quasi-Newton method for (1.1)

Based on the discussions in Sections 3.1 and 3.2, we propose a single-loop stochastic proximal quasi-Newton method for solving (1.1), as summarized in Algorithm 1.

It is easy to see that if the Hessian approximation B_k is fixed as the identity matrix I_d, Algorithm 1 simplifies into a proximal variant of the L-SVRG, which can be considered as an extension of the L-SVRG for solving the nonsmooth problem (1.1). We present the proximal L-SVRG method in Algorithm 2. Additionally, note that Algorithm 1 can be generalized by generating the stochastic gradient v_k via other VR methods, such as the SAGA (Defazio et al., 2014) and the SEGA (Hanzely et al., 2018). As to be shown in Section 4.3, the resulting generalized algorithms exhibit linear convergence rates.

Algorithm 1 A single-loop stochastic proximal quasi-Newton method for (1.1)

Require: initial points $x_0 = w_0 \in \mathbb{R}^d$; step sizes $\{\eta_k\}_{k\geq 0}$; probability parameter $p \in (0,1)$; Hessian update frequency $r > 0$.

Set $t = 0, \bar{x}_0 = 0$.

for $k = 0, 1, 2, \cdots$ **do**

  if $\mathrm{mod}(k, r) = 0$ and $k \geq r$ **then**

    Set $t = t + 1$ and $\bar{x}^t = \frac{1}{r}\sum_{j=k-r+1}^{k} x_j$.

    Compute $s^t = \bar{x}^t - \bar{x}^{t-1}$.

    Sample $\mathcal{S}^t \subset [n]$ and compute $y^t = \nabla^2 f_{\mathcal{S}^t}(\bar{x}^t)s^t$.

    Update the correction pairs $\{(s_j, y_j)\}_{j=1}^{m}$ with (s^t, y^t).

  Sample an index i_k uniformly at random from $[n]$.

  Compute v_k according to (3.5) or (3.7).

  if $k < r$ **then**

    $x_{k+1} = \mathrm{prox}_{\eta_k,\, h}(x_k - \eta_k v_k)$.

  else

    Update the iterate x_{k+1} by solving (3.1) with $B_k := B^t$ as defined by (3.9).

  Update the reference point $w_{k+1} = \begin{cases} x_k & \text{with probability } p, \\ w_k & \text{with probability } 1-p. \end{cases}$

Like other proximal Newton-type methods, Algorithm 1 requires to solve the subproblem (3.1) to proceed, which generally lacks a closed-form solution. To tackle this challenge, we introduce a rapid inner solver in Section 5 that efficiently solves (3.1), thereby significantly reducing the overall computational cost. It's worth noting that this solver can be leveraged for any proximal Newton-type methods that require solving subproblems akin to (3.1), consequently improving their numerical efficiency as well.

Algorithm 2 Proximal Loopless SVRG for solving (1.1)

Require: initial points $x_0 = w_0 \in \mathbb{R}^d$; step sizes $\{\eta_k\}_{k\geq 0}$; probability parameter $p \in (0,1)$.

for $k = 0, 1, \ldots$ **do**

  Sample an index i_k uniformly at random from $[n]$.

  Compute v_k according to (3.5) or (3.7).

  Update $x_{k+1} = \mathrm{prox}_{\eta_k,\, h}(x_k - \eta_k v_k)$ and $w_{k+1} = \begin{cases} x_k & \text{with probability } p, \\ w_k & \text{with probability } 1-p. \end{cases}$

4. Convergence analysis

We present our convergence analyses in this section. Specifically, in Section 4.1 we focus on Algorithm 1 and demonstrate its globally linear convergence. The convergence of Algorithm 2 is established in Section 4.2. Finally, in Section 4.3 we generalize Algorithm 1 and provide corresponding convergence results. Note that while the results established in this section focus on the case (3.5), they can be readily extended to the batch version (3.7).

4.1 Convergence of Algorithm 1

In this subsection, we shall show that the iterative sequence generated by Algorithm 1 converges linearly towards the optimal solution in expectation. We start by making the following assumption, which is widely adopted in the related literature.

Assumption 5 *Each component f_i is twice continuously differentiable, and there exist constants $\mu, L > 0$ such that*

$$\mu I \preceq \nabla^2 f_{\mathcal{S}}(x) \preceq LI$$

for any $x \in \mathbb{R}^d$ and $\mathcal{S} \subset [n]$.

Note that Assumption 5 implies that each f_i and f are μ-strongly convex and L-smooth, and the objective F is μ-strongly convex, thereby ensuring the existence of a unique minimizer x^* of (1.1).

Under Assumption 5, it can be established that the eigenvalues of the Hessian approximations defined by (3.9) are uniformly bounded both from above and away from zero as summarized in the following lemma.

Lemma 6 ((Byrd et al., 2016b, Lemma 3.1)) *Let Assumption 5 hold, then there exist constants $0 < m_1 \leq m_2$ such that the Hessian approximations $\{B^t\}_{t\geq 1}$ defined by (3.9) satisfies*

$$m_1 I \preceq B^t \preceq m_2 I, \ \forall t \geq 1.$$

The upcoming lemma demonstrates that, in the L-SVRG scheme (3.5), if both $\{x_k\}$ and $\{w_k\}$ converge to x^*, then the variances of $\{v_k\}$ converges to zero. This result plays a crucial role in establishing linear convergence for Algorithm 1 with a constant step size.

Lemma 7 *Consider problem (1.1) and suppose that f_i is convex and L-smooth for each $i \in [n]$. Let v_k be defined as in (3.5), then for any $k \geq 0$, we have $\mathbb{E}_k[v_k] = \nabla f(x_k)$ and*

$$\mathbb{E}_k\left[\|v_k - \nabla f(x_k)\|^2\right] \leq 4L[F(x_k) - F^* + F(w_k) - F^*]. \tag{4.1}$$

Proof Firstly, it is easy to verify $\mathbb{E}_k[v_k] = \nabla f(x_k)$ by noting that $\mathbb{E}_k[\nabla f_{i_k}(x_k)] = \nabla f(x_k)$ and $\mathbb{E}_k[\nabla f_{i_k}(w_k)] = \nabla f(w_k)$. Moreover, from the fact that $\mathbb{E}_k\left[\|\zeta - \mathbb{E}_k[\zeta]\|^2\right] = \mathbb{E}_k\left[\|\zeta\|^2\right] - \|\mathbb{E}_k[\zeta]\|^2$ for any random vector ζ, we have

$$\begin{aligned}
\mathbb{E}_k\left[\|v_k - \nabla f\left(x_k\right)\|^2\right] &= \mathbb{E}_k\left[\|\nabla f_{i_k}\left(x_k\right) - \nabla f_{i_k}(w_k) + \nabla f(w_k) - \nabla f\left(x_k\right)\|^2\right] \\
&= \mathbb{E}_k\left[\|\nabla f_{i_k}\left(x_k\right) - \nabla f_{i_k}(w_k)\|^2\right] - \|\nabla f\left(x_k\right) - \nabla f(w_k)\|^2 \\
&\leq \mathbb{E}_k\left[\|\nabla f_{i_k}\left(x_k\right) - \nabla f_{i_k}\left(x^*\right) + \nabla f_{i_k}\left(x^*\right) - \nabla f_{i_k}(w_k)\|^2\right].
\end{aligned} \tag{4.2}$$

Apply Young's inequality on the RHS of (4.2), and then apply Lemma 1, we obtain

$$\begin{aligned}
&\mathbb{E}_k\left[\|v_k - \nabla f\left(x_k\right)\|^2\right] \\
\leq& 2\mathbb{E}_k\left[\|\nabla f_{i_k}\left(x_k\right) - \nabla f_{i_k}\left(x^*\right)\|^2\right] + 2\mathbb{E}_k\left[\|\nabla f_{i_k}(w_k) - \nabla f_{i_k}\left(x^*\right)\|^2\right] \\
=& \frac{2}{n}\sum_{i=1}^n \|\nabla f_i\left(x_k\right) - \nabla f_i\left(x^*\right)\|^2 + \frac{2}{n}\sum_{i=1}^n \|\nabla f_i(w_k) - \nabla f_i\left(x^*\right)\|^2 \\
\leq& 4L\left[F\left(x_k\right) - F^* + F(w_k) - F^*\right],
\end{aligned}$$

which completes the proof. ■

We are now ready to unveil the convergence behavior of the iterates $\{x_k\}$ produced by Algorithm 1 towards the optimal solution x^*. For this purpose, we introduce a Lyapunov sequence $\{\phi_k\}$ defined as

$$\phi_k := \|x_k - x^*\|_{B_k}^2 + A\eta_k^2 \left[F\left(w_k\right) - F^*\right] + 2\eta_k \left[F\left(x_k\right) - F^*\right], \forall k \geq 0, \tag{4.3}$$

where the constant $A > 0$ is introduced to balance the term $F(w_k) - F^*$ via the update rule of reference points $\{w_k\}$. In the following theorem, we show that, with an appropriate constant step size $\eta_k \equiv \eta > 0$ and Hessian update frequency $r > 0$, the sequence $\{\mathbb{E}[\phi_k]\}$ converges to 0 linearly, which implies that the distance between $\{x_k\}$ and x^* diminishes linearly to 0 in expectation.

Theorem 8 *Let Assumption 5 hold. Choose a positive constant A such that $A > \frac{8L\sqrt{m_2}}{pm_1\sqrt{m_1}}$, where $m_1, m_2 > 0$ are constants satisfying $m_1 I \preceq B^t \preceq m_2 I$ for any $t \geq 1$ as indicated by Lemma 6. Let $\{x_k\}$ be the iterates generated by Algorithm 1 with constant step size $\eta > 0$ and Hessian update frequency $r > 0$ satisfying*

$$\eta \leq \min\left\{\frac{m_1}{L}, \frac{2m_1 m_2\sqrt{m_1}}{8Lm_2\sqrt{m_2} + 2\mu m_1\sqrt{m_1} + Apm_1 m_2\sqrt{m_1}}\right\} \tag{4.4}$$

and

$$\left(\frac{m_2}{m_1}\right)^{1/r} \rho < 1,$$

where $\rho := \max\left\{1 - \frac{\eta\mu}{m_2}, \frac{8L\sqrt{m_2}}{Am_1\sqrt{m_1}} + 1 - p\right\}$. Then we have $\rho \in (0,1)$, and for any $k \geq r$, it holds that

$$\mathbb{E}\left[\phi_k\right] \leq \left(\frac{m_2}{m_1}\rho^r\right)^{\lfloor \frac{k}{r} \rfloor} \phi_0.$$

Proof Apply Lemma 2 with $x = x_k$, $v = v_k$, $B = B_k$, and $y = x^*$, then we have $x^+ = x_{k+1}$ and hence $g = g_k := \frac{1}{\eta}(x_k - x_{k+1})$ and $\Delta = \Delta_k := v_k - \nabla f(x_k)$, which gives

$$F^* \geq F(x_{k+1}) + g_k^\top B_k(x^* - x_k) + \Delta_k^\top(x_{k+1} - x^*) + \eta\|g_k\|_{B_k}^2 - \frac{L\eta^2}{2}\|g_k\|^2 + \frac{\mu}{2}\|x_k - x^*\|^2. \tag{4.5}$$

Using (4.5), we expand $\|x_{k+1} - x^*\|_{B_k}^2 = \|x_{k+1} - x_k + x_k - x^*\|_{B_k}^2$ as follows

$$\begin{aligned}
\|x_{k+1} - x^*\|_{B_k}^2 &= \|x_k - x^*\|_{B_k}^2 + 2\left(x_k - x^*\right)^\top B_k\left(x_{k+1} - x_k\right) + \|x_{k+1} - x_k\|_{B_k}^2 \\
&= \|x_k - x^*\|_{B_k}^2 - 2\eta g_k^\top B_k\left(x_k - x^*\right) + \eta^2\left\|g_k\right\|_{B_k}^2 \\
&\leq \|x_k - x^*\|_{B_k}^2 + 2\eta\left[F^* - F\left(x_{k+1}\right)\right] - 2\eta\Delta_k^\top\left(x_{k+1} - x^*\right) - \eta^2\|g_k\|_{B_k}^2 \\
&\quad + L\eta^3\|g_k\|^2 - \mu\eta\left\|x_k - x^*\right\|^2.
\end{aligned} \tag{4.6}$$

Since $m_1 I \preceq B_k \preceq m_2 I$ and $\eta \leq \frac{m_1}{L}$, we have from (4.6) that

$$\begin{aligned}\left\|x_{k+1}-x^*\right\|_{B_k}^2 \leq & \left\|x_k-x^*\right\|_{B_k}^2+2\eta\left[F^*-F\left(x_{k+1}\right)\right]-2\eta\Delta_k^\top\left(x_{k+1}-x^*\right) \\ & -\eta^2(m_1-L\eta)\|g_k\|^2-\frac{\mu\eta}{m_2}\left\|x_k-x^*\right\|_{B_k}^2 \\ \leq & \left(1-\frac{\mu\eta}{m_2}\right)\left\|x_k-x^*\right\|_{B_k}^2+2\eta\left[F^*-F\left(x_{k+1}\right)\right]-2\eta\Delta_k^\top\left(x_{k+1}-x^*\right).\end{aligned} \tag{4.7}$$

Next, we bound the last term $-2\eta\Delta_k^\top\left(x_{k+1}-x^*\right)$ on the RHS of (4.7). To this end, we define

$$\bar{x}_{k+1}=\operatorname{prox}_{\eta,\,h}^{B_k}\left(x_k-\eta H_k\nabla f\left(x_k\right)\right),$$

and split $-2\eta\Delta_k^\top\left(x_{k+1}-x^*\right)$ into two parts:

$$-2\eta\Delta_k^\top\left(x_{k+1}-x^*\right)=-2\eta\Delta_k^\top\left(x_{k+1}-\bar{x}_{k+1}\right)-2\eta\Delta_k^\top\left(\bar{x}_{k+1}-x^*\right). \tag{4.8}$$

Taking conditional expectation $\mathbb{E}_k[\cdot]$ on $\Delta_k^\top\left(\bar{x}_{k+1}-x^*\right)$ gives

$$\mathbb{E}_k\left[\Delta_k^\top\left(\bar{x}_{k+1}-x^*\right)\right]=\left(\mathbb{E}_k\left[\Delta_k\right]\right)^\top\mathbb{E}_k\left[\bar{x}_{k+1}-x^*\right]=0,$$

where the first equality holds due to the independence between i_k and $\mathcal{S}_t$ (Feller, 1971, Section V.2), and the second equality follows since $\mathbb{E}_k[\Delta_k]=0$ from Lemma 7. For the term $-\Delta_k^\top\left(x_{k+1}-\bar{x}_{k+1}\right)$ in (4.8), applying (2.4) and considering the fact that $\|H_k\|\leq\frac{1}{m_1}$, we bound it as follows

$$\begin{aligned}-\Delta_k^\top\left(x_{k+1}-\bar{x}_{k+1}\right) \leq & \left\|\Delta_k\right\|\cdot\left\|\operatorname{prox}_{\eta,\,h}^{B_k}\left(x_k-\eta H_k v_k\right)-\operatorname{prox}_{\eta,\,h}^{B_k}\left(x_k-\eta H_k\nabla f\left(x_k\right)\right)\right\| \\ \leq & \sqrt{\frac{m_2}{m_1}}\left\|\Delta_k\right\|\left\|\left(x_k-\eta H_k v_k\right)-\left(x_k-\eta H_k\nabla f\left(x_k\right)\right)\right\| \\ = & \sqrt{\frac{m_2}{m_1}}\eta\left\|\Delta_k\right\|\cdot\left\|H_k\Delta_k\right\| \\ \leq & \frac{\sqrt{m_2}}{m_1\sqrt{m_1}}\eta\left\|\Delta_k\right\|^2.\end{aligned}$$

Therefore, taking conditional expectation $\mathbb{E}_k[\cdot]$ on both sides of (4.8) and applying Lemma 7, we have

$$\begin{aligned}-2\eta\mathbb{E}_k[\Delta_k^\top\left(x_{k+1}-x^*\right)] \leq & \frac{2\sqrt{m_2}}{m_1\sqrt{m_1}}\eta^2\mathbb{E}_k[\|\Delta_k\|^2] \\ \leq & \frac{8L\sqrt{m_2}}{m_1\sqrt{m_1}}\eta^2\left[F\left(x_k\right)-F^*+F\left(w_k\right)-F^*\right].\end{aligned} \tag{4.9}$$

Then, taking conditional expectation $\mathbb{E}_k[\cdot]$ on both sides of (4.7) and it follows from (4.9) that

$$\begin{aligned}\mathbb{E}_k\left[\left\|x_{k+1}-x^*\right\|_{B_k}^2\right] \leq & \left(1-\frac{\mu\eta}{m_2}\right)\left\|x_k-x^*\right\|_{B_k}^2-2\eta\mathbb{E}_k\left[F\left(x_{k+1}\right)-F^*\right] \\ & +\frac{8L\sqrt{m_2}}{m_1\sqrt{m_1}}\eta^2\left[F\left(x_k\right)-F^*+F\left(w_k\right)-F^*\right].\end{aligned} \tag{4.10}$$

On the other hand, from the update rule (3.6), we have

$$\mathbb{E}_k\left[F\left(w_{k+1}\right)-F^*\right]=(1-p)\left[F\left(w_k\right)-F^*\right]+p\left[F\left(x_k\right)-F^*\right]. \tag{4.11}$$

Adding $A\eta^2\mathbb{E}_k\left[F\left(w_{k+1}\right)-F^*\right]$ on both sides of (4.10), then it follows from (4.11) that

$$\begin{aligned}
&\mathbb{E}_k\left[\left\|x_{k+1}-x^*\right\|_{B_k}^2+A\eta^2\left(F\left(w_{k+1}\right)-F^*\right)\right]\\
\leq&\left(1-\frac{\mu\eta}{m_2}\right)\left\|x_k-x^*\right\|_{B_k}^2-2\eta\mathbb{E}_k\left[F\left(x_{k+1}\right)-F^*\right]+\left(\frac{8L\sqrt{m_2}}{m_1\sqrt{m_1}}+Ap\right)\eta^2\left[F\left(x_k\right)-F^*\right]\\
&+\left(\frac{8L\sqrt{m_2}}{Am_1\sqrt{m_1}}+1-p\right)A\eta^2\left[F\left(w_k\right)-F^*\right].
\end{aligned} \tag{4.12}$$

Note that $\eta>0$ guarantees $1-\frac{\mu\eta}{m_2}<1$, and the condition $A>\frac{8L\sqrt{m_2}}{pm_1\sqrt{m_1}}$ ensures that $0<\frac{8L\sqrt{m_2}}{Am_1\sqrt{m_1}}+1-p<1$. Therefore, we have $\rho\in(0,1)$. With the definition of ρ, we obtain from (4.12) that

$$\begin{aligned}
&\mathbb{E}_k\left[\left\|x_{k+1}-x^*\right\|_{B_k}^2+A\eta^2\left(F\left(w_{k+1}\right)-F^*\right)+2\eta\left(F\left(x_{k+1}\right)-F^*\right)\right]\\
\leq&\rho\left[\left\|x_k-x^*\right\|_{B_k}^2+A\eta^2\left(F\left(w_k\right)-F^*\right)+\frac{\eta^2}{\rho}\left(\frac{8L\sqrt{m_2}}{m_1\sqrt{m_1}}+Ap\right)\left(F\left(x_k\right)-F^*\right)\right].
\end{aligned} \tag{4.13}$$

Since

$$\eta\leq\frac{2m_1m_2\sqrt{m_1}}{8Lm_2\sqrt{m_2}+2\mu m_1\sqrt{m_1}+Apm_1m_2\sqrt{m_1}},$$

we have that

$$\eta\leq\frac{2-\frac{2\eta\mu}{m_2}}{\frac{8L\sqrt{m_2}}{m_1\sqrt{m_1}}+Ap}. \tag{4.14}$$

Then, it follows from $\rho\geq1-\frac{\mu\eta}{m_2}$ and (4.14) that

$$\frac{\eta^2}{\rho}\left(\frac{8L\sqrt{m_2}}{m_1\sqrt{m_1}}+Ap\right)\leq2\eta. \tag{4.15}$$

Hence, combining (4.13) with (4.15) we have

$$\begin{aligned}
&\mathbb{E}_k\left[\left\|x_{k+1}-x^*\right\|_{B_k}^2+A\eta^2\left(F\left(w_{k+1}\right)-F^*\right)+2\eta\left(F\left(x_{k+1}\right)-F^*\right)\right]\\
\leq&\rho\left[\left\|x_k-x^*\right\|_{B_k}^2+A\eta^2\left(F\left(w_k\right)-F^*\right)+2\eta\left(F\left(x_k\right)-F^*\right)\right].
\end{aligned} \tag{4.16}$$

For any $k\geq r$, from the condition $m_1I\preceq B_k\preceq m_2I$ we have

$$\left\|x_{k+1}-x^*\right\|_{B_k}^2\geq m_1\left\|x_{k+1}-x^*\right\|^2\geq\frac{m_1}{m_2}\left\|x_{k+1}-x^*\right\|_{B_{k+1}}^2. \tag{4.17}$$

Also notice that within any consecutive r iterations, H_k only changes once. Therefore, it follows from (4.16) and (4.17) that

$$\mathbb{E}\left[\phi_k\right]\leq\left(\frac{m_2}{m_1}\rho^r\right)\mathbb{E}\left[\phi_{k-r}\right]\leq\left(\frac{m_2}{m_1}\rho^r\right)^{\lfloor\frac{k}{r}\rfloor}\mathbb{E}\left[\phi_{k-r\lfloor\frac{k}{r}\rfloor}\right], \tag{4.18}$$

where ϕ_k is the Lyapunov function defined by (4.3). On the other hand, since $B_k = I$ for all $k < r$, we have

$$\phi_k = \|x_k - x^*\|^2 + A\eta^2 \left[F\left(w_k\right) - F^*\right] + 2\eta \left[F\left(x_k\right) - F^*\right], ; \forall k < r,$$

which together with (4.16) gives

$$\mathbb{E}\left[\phi_k\right] \leq \rho \cdot \mathbb{E}\left[\phi_{k-1}\right] \leq \rho^k \phi_0 \leq \phi_0, \ \forall k < r. \tag{4.19}$$

Note that $k - r\lfloor \frac{k}{r} \rfloor < r$. Then combining (4.18) with (4.19) leads to

$$\mathbb{E}\left[\phi_k\right] \leq \left(\frac{m_2}{m_1}\rho^r\right)^{\lfloor \frac{k}{r} \rfloor} \phi_0, \tag{4.20}$$

which completes the proof. ■

Theorem 8 shows that the the sequence of Lyapunov functions $\{\phi_k\}$ converges to 0 linearly in expectation since $\left(m_2/m_1\right)^{1/r} \rho < 1$. As a direct consequence, we have that $\{\|x_k - x^*\|_{B_k}\}$ converges to 0 linearly, which together with $m_1 I \preceq B_k$ implies that $\{x_k\}$ converges to x^* linearly. Moreover, the two sequences of the objective values $\{F(x_k)\}$ and $\{F(w_k)\}$ also converge to F^* at linear rates.

4.2 Convergence of Algorithm 2

In this subsection, we establish a linear convergence result and complexity analysis for Algorithm 2. To this end, we first note that Algorithm 2 can be regarded as a special case of Algorithm 1 wherein the matrix B_k remains fixed as the identity matrix. Therefore, by setting $m_1 = m_2 = 1$ in Theorem 8, we can derive the following convergence result for Algorithm 2.

Theorem 9 *Let Assumption 5 hold. Let $\{x_k\}$ be the iterates generated by Algorithm 2 with a constant step size $\eta > 0$ satisfying*

$$0 < \eta \leq \min\left\{\frac{1}{L}, \frac{1}{9L + \mu}\right\}, \tag{4.21}$$

we then have

$$\mathbb{E}\left[\phi_k\right] \leq \max\left\{1 - \eta\mu, 1 - \frac{p}{5}\right\}^k \phi_0, \tag{4.22}$$

where ϕ_k is the Lyapunov function defined in (4.3) *with $A = \frac{10L}{p}$ and $B_k = I$, i.e.,*

$$\phi_k := \|x_k - x^*\|^2 + \frac{10L}{p}\eta^2 \left[F\left(w_k\right) - F^*\right] + 2\eta \left[F\left(x_k\right) - F^*\right].$$

Proof Setting $m_1 = m_2 = 1$ in Theorem 8, we have that the step size requirement (4.4) reduces to

$$0 < \eta \leq \min\left\{\frac{1}{L}, \frac{2}{8L + 2\mu + Ap}\right\} \text{ and } \rho = \max\left\{1 - \eta\mu, \frac{8L}{A} + 1 - p\right\},$$

which, together with $A = \frac{10L}{p}$, gives (4.21) and ensures that $\rho \in (0,1)$, so that the condition $\left(\frac{m_2}{m_1}\right)^{1/r} \rho < 1$ is satisfied. Then, substituting $B_k = I$ and $A = \frac{10L}{p}$ in (4.16), and noting that $r = 1$ in Algorithm 2, we obtain

$$\mathbb{E}\left[\phi_k\right] \leq \rho^k \phi_0 = \max\left\{1 - \eta\mu, 1 - \frac{p}{5}\right\}^k \phi_0,$$

which completes the proof. ■

Theorem 9 implies the following iteration complexity result for Algorithm 2.

Corollary 10 *Consider the same setting as that in Theorem 9. Suppose we choose $\eta = \frac{1}{10L}$. Then for any $\epsilon > 0$, we have $\mathbb{E}\left[\phi_k\right] \leq \epsilon\phi_0$ if*

$$k \geq 10\left(\frac{1}{p} + \frac{L}{\mu}\right)\log\left(\frac{1}{\epsilon}\right). \tag{4.23}$$

Furthermore, if we select $p \in \left[\min\left\{\frac{c}{n}, \frac{c\mu}{L}\right\}, \max\left\{\frac{c}{n}, \frac{c\mu}{L}\right\}\right]$ for any constant $c > 0$, then the expected total number of stochastic gradient oracles is

$$O\left(\left(n + \frac{L}{\mu}\right)\log\frac{1}{\epsilon}\right). \tag{4.24}$$

Proof To start off, note that $\eta = \frac{1}{10L}$ satisfies the step size requirement (4.21) since $\mu \leq L$, hence plugging $\eta = \frac{1}{10L}$ into (4.22) gives

$$\mathbb{E}\left[\phi_k\right] \leq \rho^k \phi_0, \tag{4.25}$$

with $\rho = \max\left\{1 - \frac{\mu}{10L}, 1 - \frac{p}{5}\right\}$. From (4.25) and the inequality $\log(\beta) \leq \beta - 1$ for any $\beta > 0$, we know that $\mathbb{E}\left[\phi_k\right] \leq \epsilon\phi_0$ as long as

$$k \geq \frac{1}{1-\rho}\log\left(\frac{1}{\epsilon}\right). \tag{4.26}$$

In addition, notice that

$$10\left(\frac{L}{\mu} + \frac{1}{p}\right) > \max\left\{\frac{10L}{\mu}, \frac{5}{p}\right\} = \frac{1}{1-\rho}.$$

Therefore, choosing $k = 10\left(\frac{1}{p} + \frac{L}{\mu}\right)\log\left(\frac{1}{\epsilon}\right)$ satisfies (4.26), which proves (4.23).

To show (4.24), note that at each iteration, Algorithm 2 calls $O(1 + pn)$ stochastic gradient in expectation. Combining it with the iteration complexity (4.23) leads to the expected total number of stochastic gradient oracles

$$O\left(\left(\frac{1}{p} + n + \frac{L}{\mu} + \frac{Lpn}{\mu}\right)\log\frac{1}{\epsilon}\right).$$

Therefore, given any $c > 0$, any choice of $p \in [\min\{c/n, c\mu/L\}, \max\{c/n, c\mu/L\}]$ leads to the total complexity (4.24), which completes the proof. ■

Remark 11 *The complexity result* $O\left(\left(n+\frac{L}{\mu}\right)\log\frac{1}{\epsilon}\right)$ *of Algorithm 2 in terms of stochastic gradient oracles aligns with the one of the L-SVRG presented in (Kovalev et al., 2020). Consequently, we generalize the L-SVRG to a proximal variant and establish its linear convergence by leveraging the techniques developed in the proof of Theorem 8.*

4.3 Generalization of Algorithm 1

As mentioned in Section 3.3, it is possible to generalize Algorithm 1 by defining the stochastic gradient v_k using VR methods different from the L-SVRG. In this subsection, we establish linear convergence results for these generalized versions. To this end, recall that in the proof of Theorem 8, we resort to two properties of v_k defined by the L-SVRG scheme (3.5), namely, (4.1) and (4.11). Note that these two inequalities are special cases of the following general form:

$$\begin{cases}\mathbb{E}_k\left[\left\|v_k-\nabla f\left(x_k\right)\right\|^2\right]\leq a\left(F\left(x_k\right)-F^*\right)+b\xi_k,\\ \mathbb{E}_k\left[\xi_{k+1}\right]\leq(1-\gamma)\xi_k+c\left(F\left(x_k\right)-F^*\right),\end{cases}\tag{4.27}$$

where $a,b,c\geq 0$ and $\gamma\in(0,1]$ are constants unrelated to the index k, and $\{\xi_k\}_{k\geq 0}$ is a sequence of random variables. Indeed, it is easy to verify that v_k defined by the L-SVRG (3.5) satisfies (4.27) with $a=b=4L$, $c=\gamma=p$, and $\xi_k=F(w_k)-F^*$.

In the remaining part of this subsection, rather than specifying an explicit expression on v_k, we make the following assumption concerning its general properties.

Assumption 12 *For any* $k\geq 0$*, the stochastic gradient* v_k *is an unbiased estimator of* $\nabla f(x_k)$*, i.e.,* $\mathbb{E}_k[v_k]=\nabla f(x_k)$*. Furthermore, there exist constants* $a,b,c\geq 0$ *and* $\gamma\in(0,1]$*, and a sequence of random variables* $\{\xi_k\}_{k\geq 0}$ *such that* (4.27) *is satisfied.*

It can be demonstrated that, in addition to the L-SVRG (Kovalev et al., 2020), the stochastic gradient v_k generated by other VR methods like the SAGA (Defazio et al., 2014) and the SEGA (Hanzely et al., 2018) also satisfies Assumption 12. For instance, in the SAGA, each component f_i is assigned with a reference point w_k^i. At the k-th iteration, a random index i_k is sampled from $[n]$, and the update $w_{k+1}^{i_k}=x_k$ is performed while other reference points remain unchanged. Then the stochastic gradient v_k is computed as

$$v_k=\nabla f_{i_k}(x_k)-\nabla f_{i_k}(w_k^{i_k})+\frac{1}{n}\sum_{j=1}^n\nabla f_j(w_k^j).\tag{4.28}$$

Next, we show that v_k defined by (4.28) satisfies Assumption 12, in which case $\mathbb{E}_k[\cdot]$ denotes the expectation conditioned on x_k and $\{w_k^i\}_{i=1}^n$. Similar results can be established for some other VR methods.

Proposition 13 *Consider problem* (1.1)*. Suppose each component* f_i *is convex and* L*-smooth, and* $x^*\in\arg\min_{x\in\mathbb{R}^d}F(x)$*. Then, the stochastic gradient* v_k *defined by* (4.28) *satisfies Assumption 12 with* $a=4L, b=2, \gamma=\frac{1}{n}, c=\frac{2L}{n}$ *and*

$$\xi_k=\frac{1}{n}\sum_{i=1}^n\left\|\nabla f_i\left(w_k^i\right)-\nabla f_i\left(x^*\right)\right\|^2.$$

Proof To start off, it is straightforward to verify $\mathbb{E}_k[v_k] = \nabla f(x_k)$. Now, we estimate the variance of the stochastic gradient v_k defined by (4.28). First, it follows from (4.28) that

$$\begin{aligned}\mathbb{E}_k\left[\left\|v_k - \nabla f\left(x_k\right)\right\|^2\right] =& \mathbb{E}_k\left[\left\|\nabla f_{i_k}\left(x_k\right) - \nabla f_{i_k}\left(w_k^{i_k}\right) + \frac{1}{n}\sum_{j=1}^n \nabla f_j\left(w_k^j\right) - \nabla f\left(x_k\right)\right\|^2\right] \\ =& \mathbb{E}_k\left[\left\|\nabla f_{i_k}\left(x_k\right) - \nabla f_{i_k}\left(w_k^{i_k}\right) - \mathbb{E}_k\left[\nabla f_{i_k}\left(x_k\right) - \nabla f_{i_k}\left(w_k^{i_k}\right)\right]\right\|^2\right] \\ \leq& \mathbb{E}_k\left[\left\|\nabla f_{i_k}\left(x_k\right) - \nabla f_{i_k}\left(x^*\right) + \nabla f_{i_k}\left(x^*\right) - \nabla f_{i_k}\left(w_k^{i_k}\right)\right\|^2\right].\end{aligned} \tag{4.29}$$

Apply Young's inequality on the RHS of (4.29) and then apply Lemma 1, we have

$$\begin{aligned}\mathbb{E}_k\left[\left\|v_k - \nabla f\left(x_k\right)\right\|^2\right] \leq& 2\mathbb{E}_k\left[\left\|\nabla f_{i_k}\left(x_k\right) - \nabla f_{i_k}\left(x^*\right)\right\|^2\right] + 2\mathbb{E}_k\left[\left\|\nabla f_{i_k}\left(w_k^{i_k}\right) - \nabla f_{i_k}\left(x^*\right)\right\|^2\right] \\ \leq& 4L\left(F\left(x_k\right) - F^*\right) + \frac{2}{n}\sum_{i=1}^n \left\|\nabla f_i\left(w_k^i\right) - \nabla f_i\left(x^*\right)\right\|^2 \\ =& 4L\left(F\left(x_k\right) - F^*\right) + 2\xi_k,\end{aligned}$$

Next, we estimate $\mathbb{E}_k[\xi_{k+1}]$ by Lemma 1, which gives

$$\begin{aligned}\mathbb{E}_k\left[\xi_{k+1}\right] &= \frac{1}{n}\sum_{i=1}^n \mathbb{E}_k\left[\left\|\nabla f_i\left(w_{k+1}^i\right) - \nabla f_i\left(x^*\right)\right\|^2\right] \\ &= \frac{1}{n}\sum_{i=1}^n \left(\frac{n-1}{n}\left\|\nabla f_i\left(w_k^i\right) - \nabla f_i\left(x^*\right)\right\|^2 + \frac{1}{n}\left\|\nabla f_i\left(x_k\right) - \nabla f_i\left(x^*\right)\right\|^2\right) \\ &\leq \left(1 - \frac{1}{n}\right)\xi_k + \frac{2L}{n}\left(F\left(x_k\right) - F^*\right).\end{aligned}$$

The proof is now complete. ■

In the subsequent theorem, we extend Theorem 8 and demonstrate that a linear convergence rate can be achieved by generalizations of Algorithm 1, wherein the stochastic gradient v_k satisfies Assumption 12.

Theorem 14 *Let Assumption 5 hold and consider implementing a variant of Algorithm 1, wherein the stochastic gradients $\{v_k\}$ are defined such that Assumption 12 is satisfied. Suppose, we choose constant $A > 0$ and step size $\eta > 0$ such that*

$$\rho := \max\left\{1 - \frac{\eta\mu}{m_2}, \frac{2b\sqrt{m_2}}{Am_1\sqrt{m_1}} + 1 - \gamma\right\} \in (0,1), \ \left(\frac{2a\sqrt{m_2}}{m_1\sqrt{m_1}} + Ac\right)\eta \leq 2\rho,$$

where m_1 and m_2 are positive constants such that $m_1 I \preceq B_k \preceq m_2 I$ for any $k \geq 0$. Besides, the Hessian update frequency $r > 0$ is chosen to satisfy $\left(\frac{m_2}{m_1}\right)^{1/r}\rho < 1$. Then for any $k \geq r$, we have

$$\mathbb{E}\left[V_k\right] \leq \left(\frac{m_2}{m_1}\rho^r\right)^{\lfloor \frac{k}{r} \rfloor} V_0,$$

where V_k is a Lyapunov function defined by

$$V_k = \|x_k - x^*\|_{B_k}^2 + A\eta^2\xi_k + 2\eta\left[F\left(x_k\right) - F^*\right].$$

Proof Imitating the proof of Theorem 8 from (4.5) to (4.9) and applying the first inequality in (4.27), we have

$$\begin{aligned}\mathbb{E}_k\left[\|x_{k+1} - x^*\|_{B_k}^2\right] \leq & \left(1 - \frac{\eta\mu}{m_2}\right)\|x_k - x^*\|_{B_k}^2 - 2\eta\mathbb{E}_k\left[F\left(x_{k+1}\right) - F^*\right] \\ & + \frac{2\sqrt{m_2}}{m_1\sqrt{m_1}}\eta^2\mathbb{E}_k\left[\|v_k - \nabla f(x_k)\|^2\right] \\ \leq & \left(1 - \frac{\eta\mu}{m_2}\right)\|x_k - x^*\|_{B_k}^2 - 2\eta\mathbb{E}_k\left[F\left(x_{k+1}\right) - F^*\right] \\ & + \frac{2a\sqrt{m_2}}{m_1\sqrt{m_1}}\eta^2\left(F\left(x_k\right) - F^*\right) + \frac{2b\sqrt{m_2}}{m_1\sqrt{m_1}}\eta^2\xi_k.\end{aligned} \tag{4.30}$$

Adding $A\eta^2\mathbb{E}_k[\xi_{k+1}]$ on both sides of (4.30) and applying the second inequality in (4.27) leads to

$$\begin{aligned}&\mathbb{E}_k\left[\|x_{k+1} - x^*\|_{B_k}^2 + A\eta^2\xi_{k+1} + 2\eta\left[F\left(x_{k+1}\right) - F^*\right]\right] \\ \leq & \left(1 - \frac{\eta\mu}{m_2}\right)\|x_k - x^*\|_{B_k}^2 + \left(\frac{2b\sqrt{m_2}}{Am_1\sqrt{m_1}} + 1 - \gamma\right)A\eta^2\xi_k + \left(\frac{2a\sqrt{m_2}}{m_1\sqrt{m_1}} + Ac\right)\eta^2\left[F\left(x_k\right) - F^*\right].\end{aligned}$$

Since $\rho = \max\left\{1 - \frac{\eta\mu}{m_2}, \frac{2b\sqrt{m_2}}{Am_1\sqrt{m_1}} + 1 - \gamma\right\} \in (0,1)$, we have

$$\begin{aligned}&\mathbb{E}_k\left[\|x_{k+1} - x^*\|_{B_k}^2 + A\eta^2\xi_{k+1} + 2\eta\left[F\left(x_{k+1}\right) - F^*\right]\right] \\ \leq & \rho\left[\|x_k - x^*\|_{B_k}^2 + A\eta^2\xi_k + \frac{\eta^2}{\rho}\left(\frac{2a\sqrt{m_2}}{m_1\sqrt{m_1}} + Ac\right)\left(F\left(x_k\right) - F^*\right)\right].\end{aligned}$$

Since $\left(\frac{2a\sqrt{m_2}}{m_1\sqrt{m_1}} + Ac\right)\eta \leq 2\rho$, it holds that $\frac{\eta^2}{\rho}\left(\frac{2a\sqrt{m_2}}{m_1\sqrt{m_1}} + Ac\right) \leq 2\eta$. Therefore, we have

$$\begin{aligned}&\mathbb{E}_k\left[\|x_{k+1} - x^*\|_{B_k}^2 + A\eta^2\xi_{k+1} + 2\eta\left(F\left(x_{k+1}\right) - F^*\right)\right] \\ \leq & \rho\left[\|x_k - x^*\|_{B_k}^2 + A\eta^2\xi_k + 2\eta\left(F\left(x_k\right) - F^*\right)\right].\end{aligned}$$

The remaining of the proof resembles the proof of Theorem 8 starting from (4.17). ■

Remark 15 *Similar to Theorem 9, we can establish convergence results for the proximal extensions of various VR methods satisfying Assumption 12 by setting $m_1 = m_2 = 1$ in Theorem 14.*

5. An SSN method for (3.1) and its numerical implementation

In this section, we provide a fast SSN method for efficiently solving the subproblem (3.1). The subproblem (3.1) is equivalent to the following minimization problem:

$$\min_{x\in\mathbb{R}^d} \quad v_k^\top (x - x_k) + \frac{1}{2\eta_k} (x - x_k)^\top B_k(x - x_k) + h(x). \tag{5.1}$$

To simplify the formulation of (5.1), we introduce

$$g := \eta_k v_k - B_k x_k, \quad \theta(\cdot) := \eta_k h(\cdot).$$

Then, we omit the subscript k in B_k and reformulate (5.1) as

$$\min_{x\in\mathbb{R}^d} \quad g^\top x + \frac{1}{2} x^\top B x + \theta(x). \tag{5.2}$$

Let $\{(s_i, y_i)\}_{i=1}^m$ be the correction pairs that generate B, where m is the current memory size. For simplicity and with a slight abuse of notation, we denote by B_i the i-th matrix in the generation of B throughout this section. That is, given B_{i-1}, B_i is updated via

$$B_i = B_{i-1} - \frac{s_i y_i^\top}{y_i^\top s_i} + \frac{y_i y_i^\top}{y_i^\top s_i},$$

and we have $B = B_m$.

In Section 5.1, we present our SSN method for solving (5.2), and we provide an efficient numerical implementation for the proposed SSN method in Section 5.2.

5.1 An SSN method for (5.2)

It is known that SSN methods are typically used in conjunction with a suitable step size search strategy (Ali et al., 2017; Nakayama et al., 2024). However, the nonsmoothness of the objective function in (5.2) prevents the direct application of commonly used line search conditions. To address this issue, we first transform problem (5.2) into an unconstrained smooth problem. For this purpose, we first rewrite (5.2) as

$$\begin{aligned} \min_{x,z\in\mathbb{R}^d} \quad & g^\top x + \frac{1}{2} x^\top (B - \alpha I) x + \frac{\alpha}{2} z^\top z + \theta(z), \\ \text{s.t.} \quad & x = z, \end{aligned} \tag{5.3}$$

where $\alpha > 0$. Note that if the matrix $B_\alpha := B - \alpha I$ is positive definite, then the objective function of (5.3) is convex. The proposition below offers guidance for selecting α to ensure the positive definiteness of B_α.

Proposition 16 *For any α satisfying $0 \le \alpha < \bar{\alpha} := \frac{1}{\frac{1}{\sigma_0} + \sum_{i=1}^m \frac{s_i^\top s_i}{y_i^\top s_i}}$, where σ_0 is defined in (3.9), the matrix $B_\alpha = B - \alpha I$ is positive definite.*

Proof Recall $y_i^\top s_i > 0$ for all $i \in [m]$. From (Nocedal and Wright, 1999), the inverse L-BFGS matrix $H_i = B_i^{-1}$ satisfies for any $i \in [m]$,

$$H_i = \Big(I - \frac{s_i y_i^\top}{y_i^\top s_i}\Big) H_{i-1} \Big(I - \frac{y_i s_i^\top}{y_i^\top s_i}\Big) + \frac{s_i s_i^\top}{y_i^\top s_i}.$$

Note that the largest eigenvalue of H_i denoted by $\sigma_{\max}(H_i)$ satisfies

$$\begin{aligned}
\sigma_{\max}(H_i) &= \max_{\|x\|=1} x^\top H_i x \\
&\le \max_{\|x\|=1} x^\top \Big(I - \frac{s_i y_i^\top}{y_i^\top s_i}\Big) H_{i-1} \Big(I - \frac{y_i s_i^\top}{y_i^\top s_i}\Big) x + \max_{\|x\|=1} x^\top \frac{s_i s_i^\top}{y_i^\top s_i} x \\
&= \sigma_{\max}\Big(\Big(I - \frac{s_i y_i^\top}{y_i^\top s_i}\Big) H_{i-1} \Big(I - \frac{y_i s_i^\top}{y_i^\top s_i}\Big)\Big) + \frac{s_i^\top s_i}{y_i^\top s_i}.
\end{aligned} \tag{5.4}$$

On the other hand, note that the eigenvalues of matrix $\frac{y_i s_i^\top}{y_i^\top s_i}$ correspond to 0 with multiplicity $d-1$ and 1 with multiplicity 1, hence there exist orthogonal matrices $U_i, V_i \in \mathbb{R}^{d\times d}$ and diagonal matrix $\Sigma = \mathrm{diag}(0, 1, \ldots, 1) \in \mathbb{R}^{d\times d}$ such that

$$I - \frac{y_i s_i^\top}{y_i^\top s_i} = U_i \Sigma V_i.$$

Thus it holds that

$$\begin{aligned}
\sigma_{\max}\Big(\Big(I - \frac{s_i y_i^\top}{y_i^\top s_i}\Big) H_{i-1} \Big(I - \frac{y_i s_i^\top}{y_i^\top s_i}\Big)\Big) &= \sigma_{\max}(V_i^\top \Sigma U_i^\top H_{i-1} U_i \Sigma V_i) = \sigma_{\max}(\Sigma U_i^\top H_{i-1} U_i \Sigma) \\
&\le \sigma_{\max}(U_i^\top H_{i-1} U_i) = \sigma_{\max}(H_{i-1}),
\end{aligned} \tag{5.5}$$

where the inequality comes from the interlacing property for eigenvalues of Hermitian matrices (see (Golub and Van Loan, 2013)) by noting that the nonzero $(d-1)\times(d-1)$ principle submatrix of $\Sigma U_i^\top H_{i-1} U_i \Sigma$ is also a principle submatrix of $U_i^\top H_{i-1} U_i$. Then from (5.4) and (5.5) we have

$$\sigma_{\max}(H_i) \le \sigma_{\max}(H_{i-1}) + \frac{s_i^\top s_i}{y_i^\top s_i}. \tag{5.6}$$

Summing over (5.6) for $i = 1, \ldots, m$ gives

$$\sigma_{\max}(H_m) \le \sigma_{\max}(H_0) + \sum_{i=1}^m \frac{s_i^\top s_i}{y_i^\top s_i} = \frac{1}{\sigma_0} + \sum_{i=1}^m \frac{s_i^\top s_i}{y_i^\top s_i}.$$

Denote by $\sigma_{\min}(B_m)$ the smallest eigenvalue of B_m, then we have

$$\sigma_{\min}(B_m) = \frac{1}{\sigma_{\max}(H_m)} \ge \frac{1}{\frac{1}{\sigma_0} + \sum_{i=1}^m \frac{s_i^\top s_i}{y_i^\top s_i}} := \bar{\alpha},$$

which together with $B = B_m$ implies that $\sigma_{\min}(B_\alpha) > 0$ for all $\alpha \in [0, \bar{\alpha})$, hence completes the proof. ■

To proceed, let's consider the dual problem of (5.3), which reads as

$$\max_{\lambda\in\mathbb{R}^d} J^*(\lambda), \tag{5.7}$$

where $J^*(\lambda) = \min_{x,z\in\mathbb{R}^d} L(x,z;\lambda)$ with $L(x,z;\lambda) = \frac{1}{2}x^\top B_\alpha x + g^\top x + \frac{\alpha}{2}z^\top z + \theta(z) - \lambda^\top(x - z)$. Since problem (5.3) is convex with $0 \le \alpha < \bar{\alpha}$, the strong duality holds (Boyd and Vandenberghe, 2004). Thus, the solutions of (5.3) can be obtained by solving its dual problem (5.7). For convenience, we define $\Lambda(\lambda) := -J^*(\lambda)$ and consider the following equivalent formulation of (5.7):

$$\min_{\lambda\in\mathbb{R}^d} \Lambda(\lambda). \tag{5.8}$$

Remark 17 *We transform the subproblem* (5.2) *into the dual reformulation* (5.8)*, which can be conducted for general positive definite matrix B. It is worth noting that an alternative reformulation of* (5.2) *is provided by the proximal calculus framework in Becker et al. (2019). This reformulation is applicable when the matrix B admits a decomposition $B = D - UU^\top$, where $D \in \mathbb{R}^{d\times d}$ is a diagonal matrix and $U \in \mathbb{R}^{d\times m}$. In this case, the solution of* (5.2)*, expressed as the scaled proximal operator $\mathrm{prox}_\theta^B(u)$ with $u := -B^{-1}g$, satisfies*

$$\mathrm{prox}_\theta^B(u) = \mathrm{prox}_\theta^D\left(u + D^{-1}U\alpha^\star\right),$$

where $\alpha^\star$ is the unique root of $\mathcal{L}(\alpha) := U^\top\left(u - \mathrm{prox}_\theta^D\left(u + D^{-1}U\alpha\right)\right) + \alpha$. Consequently, the computational challenge shifts to solving the nonlinear, nonsmooth equation $\mathcal{L}(\alpha) = 0$, which typically requires an iterative solver such as an SSN method. However, for L-BFGS matrices with a general memory size $m > 1$, obtaining the required decomposition $B = D - UU^\top$ is not straightforward. As seen from the compact representation (3.9)*, this decomposition necessitates computing the square root of the inverse of a matrix J, which is one of the key computational bottlenecks that our method avoids (see Section 5.2).*

Next, we show that the objective function Λ in (5.8) is differentiable if $0 < \alpha < \bar{\alpha}$.

Proposition 18 *If $0 < \alpha < \bar{\alpha}$, then the objective function Λ in* (5.8) *is differentiable and its gradient is given by*

$$\nabla\Lambda(\lambda) = B_\alpha^{-1}(\lambda - g) - \mathrm{prox}_{\frac{1}{\alpha},\,\theta}\Big(-\frac{\lambda}{\alpha}\Big),\ \forall\lambda\in\mathbb{R}^d.$$

Proof To simplify the notation, we introduce two auxiliary functions $\Psi(\cdot)$ and $\Theta(\cdot)$ defined by

$$\Psi(\lambda) := -\min_{x\in\mathbb{R}^d}\ \frac{1}{2}x^\top B_\alpha x + g^\top x - \lambda^\top x = \frac{1}{2}(\lambda - g)^\top B_\alpha^{-1}(\lambda - g),$$
$$\Theta(\lambda) := -\min_{z\in\mathbb{R}^d}\ \frac{\alpha}{2}\|z\|^2 + \theta(z) + \lambda^\top z.$$

Then, it is easy to see that

$$\Lambda(\lambda) = \Psi(\lambda) + \Theta(\lambda).$$

Therefore, it suffices to prove that both Ψ and Θ are differentiable.

To this end, first note that Ψ is differentiable with its gradient given by

$$\nabla\Psi(\lambda) = B_\alpha^{-1}(\lambda - g),\ \forall \lambda \in \mathbb{R}^d. \tag{5.9}$$

To show that Θ is also differentiable, notice that

$$\Theta(\lambda) = -\min_{z\in\mathbb{R}^d}\left\{\theta(z) + \frac{\alpha}{2}\|z + \frac{\lambda}{\alpha}\|^2 - \frac{1}{2\alpha}\|\lambda\|^2\right\} = -\mathcal{M}_{\frac{1}{\alpha},\,\theta}\Big(-\frac{\lambda}{\alpha}\Big) + \frac{1}{2\alpha}\|\lambda\|^2.$$

Hence, from (2.3) we know that Θ is differentiable, and

$$\nabla\Theta(\lambda) = \frac{1}{\alpha}\nabla\mathcal{M}_{\frac{1}{\alpha},\,\theta}\Big(-\frac{\lambda}{\alpha}\Big) + \frac{\lambda}{\alpha} = -\operatorname{prox}_{\frac{1}{\alpha},\,\theta}\Big(-\frac{\lambda}{\alpha}\Big). \tag{5.10}$$

Consequently, Λ is differentiable, and combining (5.9) with (5.10) gives

$$\nabla\Lambda(\lambda) = B_\alpha^{-1}(\lambda - g) - \operatorname{prox}_{\frac{1}{\alpha},\,\theta}\Big(-\frac{\lambda}{\alpha}\Big),$$

which completes the proof. ■

As a consequence of Proposition 18, if we let $\mathcal{D}_\theta(\cdot)$ be the generalized Jacobian (Clarke, 1990) of $\operatorname{prox}_\theta(\cdot)$, then the matrix

$$B_\alpha^{-1} + \frac{1}{\alpha}\mathcal{D}_{\frac{\theta}{\alpha}}\Big(-\frac{\lambda}{\alpha}\Big)$$

is the generalized Jacobian of $\nabla\Lambda$ at λ. Now applying an SSN method to (5.8) readily yields the following iterative scheme

$$\lambda_{j+1} = \lambda_j + \rho_j d_j,$$

where $\rho_j > 0$ is a step size and $d_j := -(B_\alpha^{-1} + \mathcal{D}_j)^{-1}\nabla\Lambda(\lambda_j)$ with $\mathcal{D}_j := \frac{1}{\alpha}\mathcal{D}_{\frac{\theta}{\alpha}}\big(-\frac{\lambda_j}{\alpha}\big)$.

To determine the step size ρ_j, one can apply a backtracking scheme until the Wolfe condition (Wolfe, 1969) is met. Alternatively, we propose an exact line search by solving

$$\rho_j = \arg\min_{\rho\geq 0} R_j(\rho) := \Lambda(\lambda_j + \rho d_j).$$

Since R_j is convex and differentiable, various methods can be applied to minimize it. In particular, we advocate applying an SSN method due to its fast convergence. For this purpose, a straightforward calculation gives

$$R_j'(\rho) = d_j^\top\nabla\Lambda(\lambda_j + \rho d_j) = d_j^\top B_\alpha^{-1}(\lambda_j - g) + \rho d_j^\top B_\alpha^{-1}d_j - d_j^\top\operatorname{prox}_{\frac{1}{\alpha},\,\theta}\Big(-\frac{\lambda_j + \rho d_j}{\alpha}\Big),$$

which implies that

$$d_j^\top B_\alpha^{-1}d_j + \frac{1}{\alpha}d_j^\top\mathcal{D}_{\frac{\theta}{\alpha}}\Big(-\frac{\lambda_j + \rho d_j}{\alpha}\Big)d_j$$

is an element of the generalized Jacobian of $R_j'(\rho)$. Therefore, an SSN method for minimizing $R_j(\cdot)$ proceeds by updating

$$\rho_{i+1,j} = \rho_{i,j} - \frac{d_j^\top B_\alpha^{-1}(\lambda_j - g) + \rho_{i,j}d_j^\top B_\alpha^{-1}d_j - d_j^\top\operatorname{prox}_{\frac{1}{\alpha},\,\theta}\big(-\frac{\lambda_j + \rho_{i,j}d_j}{\alpha}\big)}{d_j^\top B_\alpha^{-1}d_j + \frac{1}{\alpha}d_j^\top\mathcal{D}_{\frac{\theta}{\alpha}}\big(-\frac{\lambda_j + \rho_{i,j}d_j}{\alpha}\big)d_j}. \tag{5.11}$$

Note that $\rho_{0,j} = 1$ is a natural choice of the initial guess, and we set ρ_j as the output of (5.11).

With the above discussions, we propose an SSN method for solving problem (5.8) and list it in Algorithm 3.

Algorithm 3 An SSN method for solving problem (5.8).

Require: initial point $\lambda_0 \in \mathbb{R}^d$.
for $j = 0, 1, \ldots$ **do**
Compute $x_j = B_\alpha^{-1}(\lambda_j - g)$ and $z_j = \mathrm{prox}_{\frac{1}{\alpha},\theta}\big(-\frac{\lambda_j}{\alpha}\big)$
Set $\mathcal{D}_j = \frac{1}{\alpha}\mathcal{D}_{\frac{\theta}{\alpha}}\big(-\frac{\lambda_j}{\alpha}\big)$ and $\nabla\Lambda(\lambda_j) = x_j - z_j$
Compute $d_j = -(B_\alpha^{-1} + \mathcal{D}_j)^{-1}\nabla\Lambda(\lambda_j)$
Update $\lambda_{j+1} = \lambda_j + \rho_j d_j$ with the step size ρ_j obtained by (5.11).

Remark 19 *From the equivalence between problems* (5.2) *and* (5.3) *and the fact that strong duality holds between* (5.3) *and its dual* (5.8)*, we have that* $x^* = B_\alpha^{-1}(\lambda^* - g)$ *is a global minimizer of* (5.2) *if* λ^* *is a global minimizer of* (5.8)*. In other words, Algorithm 3 is an SSN method from a dual perspective to solve* (5.2)*.*

5.2 A fast numerical implementation of Algorithm 3

In this subsection, we explore an efficient numerical implementation of Algorithm 3. Assuming that θ has a simple structure with an efficiently computable proximal operator, the computational cost of Algorithm 3 primarily involves computing x_j and d_j, as well as the computations involved in (5.11). As a result, the key challenge in implementing Algorithm 3 is: *given a vector* $z \in \mathbb{R}^d$*, how to efficiently compute the matrix-vector products* $B_\alpha^{-1}z$ *and* $(B_\alpha^{-1} + \mathcal{D}_j)^{-1}z$*.*

Recall that at most $l \ll d$ correction pairs $(s_j, y_j), j = 1, \ldots, m \le l$, are stored. From (3.9), we can represent B_α as a scaled difference between a diagonal matrix and a matrix with rank $2m \le 2l \ll d$, i.e.,

$$B_\alpha = (\sigma_0 - \alpha)(I - UJ^{-1}U^\top), \tag{5.12}$$

where

$$U := \begin{bmatrix}\sigma_0 S & Y\end{bmatrix} \in \mathbb{R}^{d\times 2m} \text{ and } J := (\sigma_0 - \alpha)\begin{bmatrix}\sigma_0 S^\top S & L\\ L^\top & -D\end{bmatrix} \in \mathbb{R}^{2m\times 2m}.$$

Leveraging this specific structure, we can convert the complex d-dimensional linear system in Algorithm 3 into an easily solvable $2m$-dimensional linear system.

For simplicity, we assume that $\theta = \|\cdot\|_1$ or $\theta = I_C$, with I_C being the indicator function of a convex closed set $C \subset \mathbb{R}^d$. In both cases, the generalized Jacobian $\mathcal{D}_\theta(\cdot)$ of $\mathrm{prox}_\theta(\cdot)$ (see (Clarke, 1990)) can be represented as a diagonal matrix with entries being either 0 or 1. This distinct structure of $\mathcal{D}_\theta(\cdot)$ implies that many of the multiplication operations employed in computing $(B_\alpha^{-1} + \mathcal{D}_j)^{-1}z$ are the same as those used in computing $B_\alpha^{-1}z$. Then as shown below, we can reduce the redundant multiplication operations by computing and storing

some auxiliary matrices. This approach significantly reduces the computational load of the entire algorithm. We summarize these conditions in Assumption 20. It is worth mentioning that the techniques developed subsequently can be easily generalized to cases where $\mathcal{D}_\theta$ and the basic matrix B_0 are general diagonal matrices.

Assumption 20 *For any $u \in \mathbb{R}^d$, the generalized Jacobian $\mathcal{D}_\theta(u)$ is a diagonal matrix with diagonal entries being either 0 or 1.*

5.2.1 Computation of auxiliary matrices

In addition to the matrices $S = [s_1, \ldots, s_m]$ and $Y = [y_1, \ldots, y_m]$, we introduce some supplementary auxiliary matrices to track all multiplicative operations required for computing $S^\top Y$, $S^\top S$, and $Y^\top Y$. To simplify our notation, for any vectors $u, v \in \mathbb{R}^m$ we define the operation:

$$u \otimes v := uv^\top = \begin{bmatrix} u_1v_1 & u_1v_2 & \ldots & u_1v_m \\ u_2v_1 & u_2v_2 & \ldots & u_2v_m \\ \vdots & \vdots & \ddots & \vdots \\ u_mv_1 & u_mv_2 & \ldots & u_mv_m \end{bmatrix} \in \mathbb{R}^{m\times m}.$$

For $i \in [d]$, let $\bar{s}_i, \bar{y}_i \in \mathbb{R}^m$ represent the transpose of the i-th row of S and Y, respectively, such that:

$$S = \begin{bmatrix} \bar{s}_1^\top \\ \bar{s}_2^\top \\ \vdots \\ \bar{s}_d^\top \end{bmatrix} \in \mathbb{R}^{d\times m} \text{ and } Y = \begin{bmatrix} \bar{y}_1^\top \\ \bar{y}_2^\top \\ \vdots \\ \bar{y}_d^\top \end{bmatrix} \in \mathbb{R}^{d\times m}.$$

We compute and store the following auxiliary matrices:

$$\bar{s}_i \otimes \bar{y}_i,\ \bar{s}_i \otimes \bar{s}_i,\ \text{and } \bar{y}_i \otimes \bar{y}_i \in \mathbb{R}^{m\times m},\ \text{for } i = 1, 2, \ldots, d.$$

After computing a new correction pair (s^t, y^t), these auxiliary matrices do not need to be completely recomputed. In fact, for $i \in [d]$, let $s_{j,i}, s_i^t \in \mathbb{R}$ denote the i-th entry of s_j and s^t, respectively ($y_{j,i}, y_i^t \in \mathbb{R}$ are defined similarly). Then the updates only include computing the entries:

$$\begin{cases} s_{j,i}\ y_i^t \\ s_i^t\ y_{j,i} \\ s_{j,i}\ s_i^t \\ y_{j,i}\ y_i^t \end{cases} \text{ for } j \in \begin{cases} [m] \text{ if } t \le l \\ \{2, \ldots, m\} \text{ otherwise} \end{cases} \text{ and } \begin{cases} s_i^t\ y_i^t, \\ s_i^t\ s_i^t, \\ y_i^t\ y_i^t. \end{cases} \tag{5.13}$$

When $t \le l$, we update S and Y by adding new columns $s_{m+1} = s^t$ and $y_{m+1} = y^t$ respectively, and update $\bar{s}_i \otimes \bar{y}_i,\ \bar{s}_i \otimes \bar{s}_i,$ and $\bar{y}_i \otimes \bar{y}_i$ by appending a new column and row with entries calculated in (5.13) on the right and bottom correspondingly. When $t > l$, the matrices S, Y and the auxiliary matrices $\bar{s}_i \otimes \bar{y}_i,\ \bar{s}_i \otimes \bar{s}_i,$ and $\bar{y}_i \otimes \bar{y}_i$ are updated by deleting and appending certain rows and columns with s^t, y^t, and those computed in (5.13). In total, computing and storing the auxiliary matrices $\bar{s}_i \otimes \bar{y}_i,\ \bar{s}_i \otimes \bar{s}_i,$ and $\bar{y}_i \otimes \bar{y}_i$ for $i \in [d]$ totally require approximately $4md$ multiplication operations and $2m^2d$ units of storage.

5.2.2 Computation of $B_\alpha^{-1}z$

Recall (5.12), we have from Woodbury's formula that

$$B_\alpha^{-1}z = \frac{1}{\sigma_0-\alpha}[z - U(U^\top U - J)^{-1}U^\top z].$$

Therefore, $B_\alpha^{-1}z$ can be easily computed once we obtain the explicit expression of $(U^\top U - J)^{-1}$. First, it holds that

$$J = (\sigma_0-\alpha)\begin{bmatrix}\sigma_0 S^\top S & L\\ L^\top & -D\end{bmatrix} \in \mathbb{R}^{2m\times 2m} \text{ and } U^\top U = \begin{bmatrix}\sigma_0^2 S^\top S & \sigma_0 S^\top Y\\ \sigma_0 Y^\top S & Y^\top Y\end{bmatrix} \in \mathbb{R}^{2m\times 2m}.$$

Additionally, note that

$$S^\top Y = \sum_{i=1}^d \bar{s}_i \otimes \bar{y}_i,\ S^\top S = \sum_{i=1}^d \bar{s}_i \otimes \bar{s}_i,\ Y^\top Y = \sum_{i=1}^d \bar{y}_i \otimes \bar{y}_i,$$

which only involves assembling the auxiliary matrices $\bar{s}_i\otimes\bar{y}_i$, $\bar{s}_i\otimes\bar{s}_i$, and $\bar{y}_i\otimes\bar{y}_i$, $i=1,\dots,d$. Therefore, the matrices $S^\top Y$, $S^\top S$, and $Y^\top Y$ can be very efficiently computed since they do not require multiplications but only approximately $3m^2d$ additions in total. Moreover, recall that L is the lower triangular part of $S^\top Y$ and D is the diagonal matrix of $S^\top Y$, hence obtaining the expressions of L and D is straightforward. Therefore, the explicit expression of $U^\top U - J$ can be obtained correspondingly, and it costs $O(m^3)$ operations to compute its inverse, which is negligible since $m \ll d$. Then the remaining computations for $B_\alpha^{-1}z$ mainly consist of matrix-vector products, which cost about $4md$ multiplications and additions.

5.2.3 Computation of $(B_\alpha^{-1}+\mathcal{D}_j)^{-1}z$

From the compact formulation (3.9), we have from Woodbury's formula that

$$\begin{aligned}(B_\alpha^{-1}+\mathcal{D}_j)^{-1}z =& \left(\frac{1}{\sigma_0-\alpha}I + \mathcal{D}_j - \frac{1}{\sigma_0-\alpha}U(U^\top U - J)^{-1}U^\top\right)^{-1} z\\ =& C_{\alpha,j}z - C_{\alpha,j}U\left[U^\top C_{\alpha,j}U - (\sigma_0-\alpha)(U^\top U - J)\right]^{-1}U^\top C_{\alpha,j}z,\end{aligned} \tag{5.14}$$

where we define $C_{\alpha,j} = \left(\frac{1}{\sigma_0-\alpha}I + \mathcal{D}_j\right)^{-1} \in \mathbb{R}^{d\times d}$, with $\mathcal{D}_j := \frac{1}{\alpha}\mathcal{D}_{\frac{\theta}{\alpha}}\big(-\frac{\lambda_j}{\alpha}\big)$. Recall that from Assumption 20 we have $\mathcal{D}_j = \frac{1}{\alpha}\mathrm{diag}(a_1^j, a_2^j, \dots, a_d^j)$, where $a_i^j \in \{0,1\}$ for any $i \in [d]$. Therefore, the top-right block of $U^\top C_{\alpha,j}U$ reads as

$$\sigma_0 S^\top C_{\alpha,j}Y = \sum_{i=1}^d \frac{\sigma_0}{1/(\sigma_0-\alpha) + a_i^j/\alpha}\bar{s}_i\otimes\bar{y}_i. \tag{5.15}$$

Furthermore, denoting $I_j = \{1\le i\le d | a_i^j = 1\}$, then it follows from (5.15) that

$$\sigma_0 S^\top C_{\alpha,j}Y = \alpha(\sigma_0-\alpha)\sum_{i\in I_j}\bar{s}_i\otimes\bar{y}_i + \sigma_0(\sigma_0-\alpha)\left(S^\top Y - \sum_{i\in I_j}\bar{s}_i\otimes\bar{y}_i\right).$$

Similar computation can be established for other blocks in $U^\top C_{\alpha,j} U$, which implies that the explicit expression of $U^\top C_{\alpha,j} U$ can be computed efficiently via assembling the auxiliary matrices, which only require approximately $3m^2|I_j|$ additions. Given $J, U^\top U$ and $U^\top C_{\alpha,j} U$, it costs $O(m^3)$ operations to compute $\left[U^\top C_{\alpha,j} U - (\sigma_0 - \alpha)(U^\top U - J)\right]^{-1}$. Since $C_{\alpha,k}$ is diagonal and $m \ll d$, we obtain that the remaining computation of $(B_\alpha^{-1} + \mathcal{D}_j)^{-1} z$ by (5.14) requires about $4md$ multiplications and additions.

5.3 Computational complexity analysis

Based on the implementation described above, we can significantly reduce the computational complexity of Algorithm 3. Overall, the computational cost of our implementation includes:

- updating supplementary auxiliary matrices $\bar{s}_i \otimes \bar{y}_i,\ \bar{s}_i \otimes \bar{s}_i,$ and $\bar{y}_i \otimes \bar{y}_i,$ for $i \in [d]$;
- computing the gradient $\nabla \Lambda(\lambda_j)$;
- computing the direction $d_j = -(B_\alpha^{-1} + \mathcal{D}_j)^{-1} \nabla \Lambda(\lambda_j)$;
- performing step size searches with (5.11).

Assuming an implementation of Algorithm 3 involves ι SSN iterations (typically a single-digit number), we now estimate the computational cost of the aforementioned tasks.

Firstly, updating the auxiliary matrices incurs relatively minor computational cost. These auxiliary matrices are updated only after receiving a new correction pair, occurring every r iterations, with each update costing approximately $4md$ multiplications. Thus, on average, the computational cost for updating these auxiliary matrices in each implementation of Algorithm 3 is $4md/r$ multiplications.

Secondly, recall that the gradient $\nabla \Lambda(\lambda_j) = x_j - z_j$. As previously mentioned, computing $x_j = B_\alpha^{-1}(\lambda_j - g)$ requires roughly $3m^2 d$ additions to calculate $U^\top U - J$, $O(m^3)$ operations to compute $(U^\top U - J)^{-1}$, and $4md$ multiplications and additions for matrix-vector multiplication. Since we assume $m \leq l \ll d$, the computational cost of $O(m^3)$ operations compared to $3m^2 d$ additions can be neglected. Furthermore, the matrix $(U^\top U - J)^{-1}$ is only recomputed during correction pair updates and the matrix-vector multiplication is executed per iteration of SSN. Therefore, on average, computing $x_j = B_\alpha^{-1}(\lambda_j - g)$ for each implementation of Algorithm 3 cost approximately $4\iota md$ multiplications and $4\iota md + 3m^2 d/r$ additions. On the other hand, computing z_j involves ιd non-linear operations in each implementation of Algorithm 3, with the specific complexity depending on the form of the function θ. Under Assumption 20, these operations are easy and cheap to implement. That is to say, on average, the computational complexity for computing the gradient $\nabla \Lambda(\lambda_j)$ in each implementation of Algorithm 3 includes approximately $4\iota md$ multiplications and $4\iota md + 3m^2 d/r$ additions, along with ιd simple non-linear operations.

Thirdly, the computation of the direction d_j is inherently the most costly. It involves forming the diagonal matrix D_j and solving the linear system with the coefficient matrix $B_\alpha^{-1} + \mathcal{D}_j$. The estimation of its computational cost is similar to that of z_j and x_j, with the difference being that $U^\top C_{\alpha,j} U$ needs to be reassembled at each SSN iteration. Then, it is easy to show that computing d_j requires approximately $4\iota md$ multiplications and approximately $4\iota md + 3\iota m^2 d$ additions, along with ιd simple non-linear operations.

	Update auxiliary matrices	Compute $\nabla\Lambda(\lambda_j)$	Compute d_j	Total
Multiplications	$4md/r$	$4\iota md$	$4\iota md$	$O(\iota md)$
Additions	-	$4\iota md + 3m^2d/r$	$4\iota md + 3\iota m^2 d$	$O(\iota m^2 d)$
Non-linear Opts	-	ιd	ιd	$O(\iota d)$

Table 1: Summary of computational complexity of Algorithm 3.

Finally, we need to perform step size searches with (5.11). Due to the potentially varying number of step size searches in each SSN iteration, it's difficult to quantify the computational complexity. However, we note that it is relatively low. In fact, the terms $d_j^\top B_\alpha^{-1}(\lambda_j - g) = d_j^\top x_j$ and $d_j^\top B_\alpha^{-1} d_j$ in (5.11) can be computed once in each SSN iteration and then reused during the step size searches. Therefore, most of the computational cost of the step size search comes from executing the proximal operator and computing its generalized Jacobian, which consist of $O(\iota d)$ simple non-linear operations, which incurs small computational overhead under Assumption 20.

In conclusion, each implementation of Algorithm 3 requires $O(\iota md)$ multiplications and $O(\iota m^2 d)$ additions, along with $O(\iota d)$ simple non-linear operations. We summarize our discussions on computational complexity in Table 1.

6. Experiments

In this section, we present a comprehensive set of experiments to validate the performance, accuracy, and advantages offered by Algorithm 1 and Algorithm 3. Following the experimental settings of some stochastic Newton-type methods (e.g., (Byrd et al., 2016b)), we focus on the logistic regression for binary classification, incorporating the elastic net regularizer (Zou and Hastie, 2005), which reads

$$\min_{x\in\mathbb{R}^d} F(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x) + h(x), \tag{6.1}$$

where, for all $x \in \mathbb{R}^d$,

$$\begin{aligned} f_i(x) &:= f\left(x; a_i, b_i\right) = b_i \log\left(c\left(x; a_i\right)\right) + \left(1 - b_i\right)\log\left(1 - c\left(x; a_i\right)\right), \\ h(x) &:= \frac{\mu}{2}\|x\|^2 + \lambda\|x\|_1. \end{aligned} \tag{6.2}$$

In (6.2), $(a_i, b_i) \in \mathbb{R}^d \times \{0, 1\}$ represents the i-th sample, where $a_i \in \mathbb{R}^d$ corresponds to the feature vector and $b_i \in \{0, 1\}$ represents the label. The function $c(\cdot;\cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is defined as:

$$c\left(x; a_i\right) = \frac{1}{1 + \exp\left(-a_i^\top x\right)},\ \forall x \in \mathbb{R}^d.$$

It is worth noting that the square term $\frac{\mu}{2}\|\cdot\|^2$ in the function h is smooth and can be combined with f_i, which leads to the following equivalent formulation for all $x \in \mathbb{R}^d$:

$$\begin{aligned} f_i(x) &:= f\left(x; a_i, b_i\right) = b_i \log\left(c\left(x; a_i\right)\right) + \left(1 - b_i\right)\log\left(1 - c\left(x; a_i\right)\right) + \frac{\mu}{2}\|x\|^2, \\ h(x) &:= \lambda\|x\|_1. \end{aligned} \tag{6.3}$$

Algorithm	Algorithm 1	SPQN-SVRG	SPQN	P-LSVRG
Step size choice	constant	constant	$\eta_k = O(\frac{1}{k})$	constant

Table 2: Summary of algorithms and their step size choices.

Dataset	n	d	*Sparsity*
*Synthetic*1	10^4	5000	dense
*Synthetic*2	10^4	10^6	sparsity=0.1%
*Synthetic*3	10^4	10^6	sparsity=1%
*rcv*1	23149	47236	sparse
*a*9*a*	32561	123	sparse
*w*8*a*	49749	300	sparse
mushrooms	8124	112	sparse

Table 3: Summary of datasets.

An important advantage of the formulation (6.3) is that each f_i becomes μ-strongly convex and $(\|a_i\|^2 + \mu)$-smooth.

In our experimental evaluation, we compare the performance of Algorithm 1 with other three algorithms for solving the regularized logistic regression problem (6.1), namely the stochastic proximal quasi-Newton method combined with the SVRG (SPQN-SVRG) (e.g., (Luo et al., 2016)), the stochastic proximal quasi-Newton (SPQN) built upon (Byrd et al., 2016b), and Algorithm 2 (P-LSVRG). For the step size selection, we follow (Byrd et al., 2016b) to gradually decrease the step sizes of SPQN. For the other three algorithms, we use a constant step size, chosen for each algorithm from the set $\{2^{-20}, \ldots, 2^{-1}, 1\}$ that gives the best overall performance. We summarize the test algorithms and their step size choices in Table 2.

In all the experiments, we set the regularization parameter $\mu = \lambda = 10^{-3}$. The stochastic gradient of each algorithm is computed using a small batch of samples. In addition, since the subproblem (3.1) has no closed-form solution, we solve it iteratively by the ISTA (Daubechies et al., 2004), the FISTA (Beck and Teboulle, 2009), or Algorithm 3.

Our experiments for solving (6.1) consist of the following three scenarios:

- Comparison among Algorithm 1, SPQN-SVRG, SPQN, and P-LSVRG, as well as inner solvers Algorithm 3, FISTA, and ISTA on synthetic datasets with various dimensions and levels of sparsity.
- Comparison between Algorithm 1 and SPQN-SVRG with different parameter settings on real datasets.
- Comparison among testing errors and prediction accuracy of Algorithm 1, SPQN, and P-LSVRG on real datasets.

To perform the experiments, we use both synthetic and real datasets given below (see also Table 3).

- Three synthetic datasets consisting of 10,000 training samples, where the features are generated from a standard multivariate Gaussian distribution. These datasets include

a dense one ($d = 5000$), and two sparse ones ($d = 10^6$) with sparsity levels of 0.1% and 1% respectively.

- Four sparse real datasets including the *rcv*1 dataset (Lewis et al., 2004), as well as three datasets sourced from the LIBSVM library (Chang, 2008): *a*9*a*, *w*8*a*, and *mushrooms*.

In all the figures, we define the *training error* as $F(x) - F^*$, where $F(x)$ is determined based on the data points from the training set, and F^* is obtained by running FISTA for a large number of iterations. On the other hand, the *testing error* is defined as $F(x)$, excluding the regularization term, and using the data points from the testing set. Our codes are available at `https://github.com/wzm0213/single-loop-stochastic-proximal-L-BFGS-code.git`, written in Python with *PyTorch*, and all the numerical experiments were conducted on a Windows 11 system, using a laptop equipped with an Intel(R) Core(TM) i5-12500H CPU operating at 2.50 GHz and 16 GB of memory.

6.1 Experiments on synthetic datasets

In this section, we perform experiments on the comparison among Algorithm 1, SPQN-SVRG, SPQN, and P-LSVRG, as well as the inner solvers Algorithm 3, FISTA, and ISTA on the synthetic datasets presented in Table 3.

6.1.1 Performance of all four algorithms

We train the logistic regression model (6.1) using the four algorithms listed in Table 2 with the three synthetic datasets presented in Table 3. The subproblems involved in Algorithm 1, SPQN-SVRG and SPQN are solved by FISTA. Each subproblem takes the form of (5.2) and its first-order optimality condition suggests that its solution is a root of

$$\mathcal{E}(x) := x - \mathrm{prox}_{\theta}\left(x - Bx - g\right).$$

We terminate the inner solver and return $\tilde{x}$ whenever $\mathcal{E}(\tilde{x}) < 10^{-8}$. For all the three datasets, the batch size for computing the stochastic gradients is chosen to be $b = 128$, while the sample size $b_H := |\mathcal{S}^t|$ for implementing the Hessian-vector product (3.8) is fixed as 600 for each algorithm. We also fix the Hessian update frequency $r = 10$ for updating the correction pairs, and the maximum memory size $l = 10$ for Algorithm 1, SPQN-SVRG, and SPQN for a fair comparison. Moreover, the probability parameter in Algorithm 1 is fixed as $p = \frac{b}{n}$, and the number of inner iterations in SPQN-SVRG is chosen as $l_s = \frac{1}{p} = \frac{n}{b}$. The initial points are set as $0.01 \cdot \mathbf{1}$ for all the algorithms, with $\mathbf{1} \in \mathbb{R}^d$ being the all-ones vector, while the step sizes are carefully tuned for each algorithm on each dataset.

Figure 1 displays the training errors on the three synthetic datasets. We can observe that Algorithm 1 and SPQN-SVRG significantly outperform SPQN and P-LSVRG across all three datasets, indicating that integrating VR techniques and Hessian information can expedite algorithmic convergence.

6.1.2 Comparison among inner solvers

In this subsection, we compare the efficiency of Algorithm 3 with ISTA and FISTA. For this purpose, we apply Algorithm 1 with the above-mentioned three inner solvers to solve

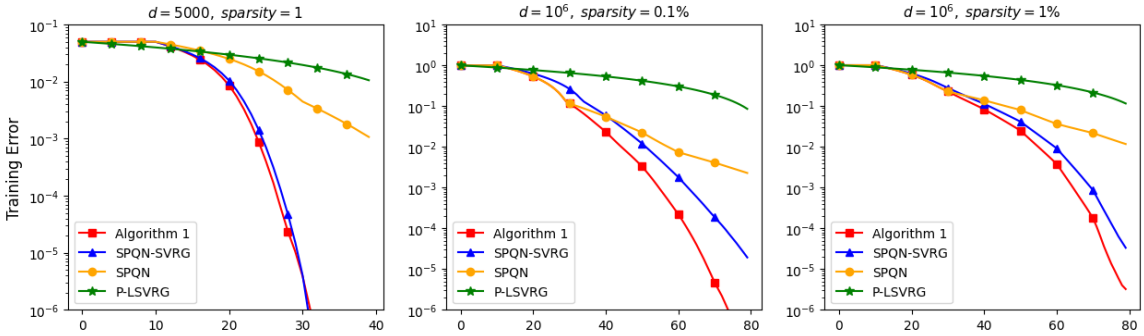


Figure 1: Comparisons for training logistic regression model with three synthetic datasets.

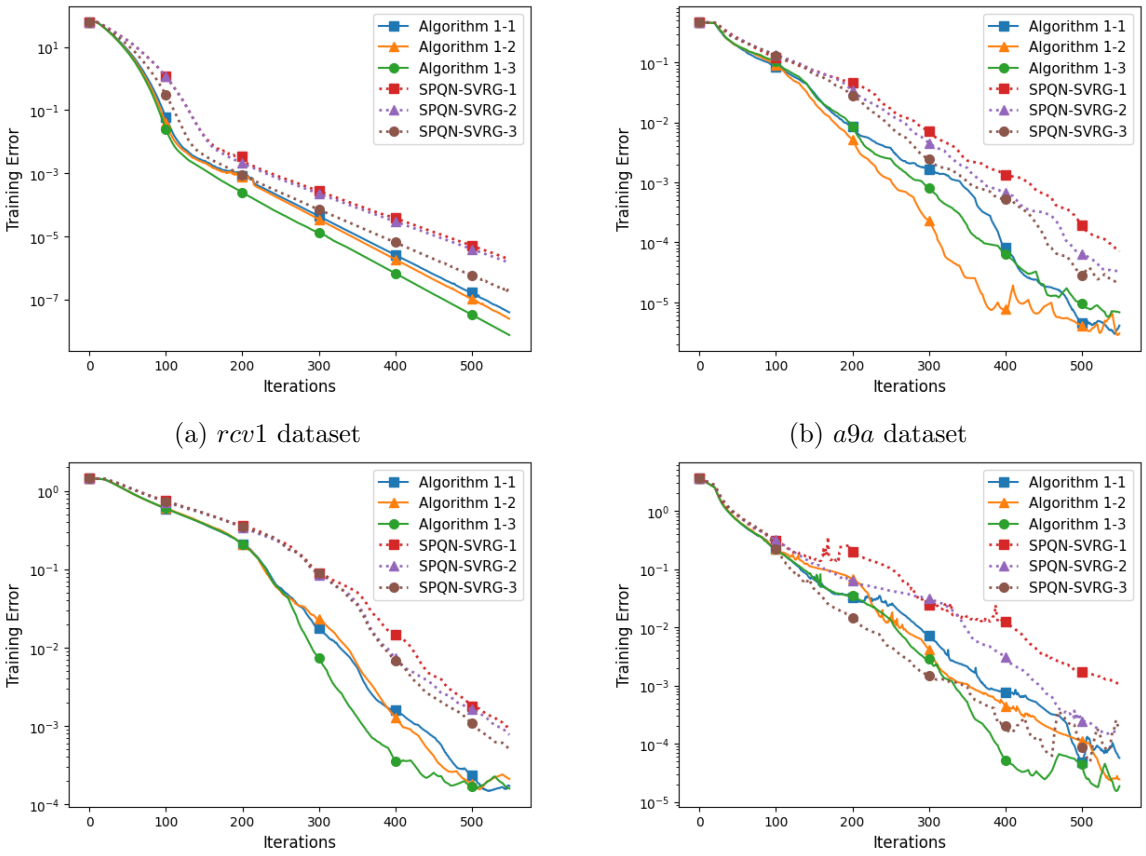
Dataset	Ave Time (s)	Ave Iter	Max Iter
Algorithm 3			
*Synthetic*1	0.039	7.61	19
*Synthetic*2	0.125	8.26	23
*Synthetic*3	0.122	8.07	23
FISTA			
*Synthetic*1	0.315	113.46	304
*Synthetic*2	1.877	132.51	381
*Synthetic*3	1.901	137.82	452
ISTA			
*Synthetic*1	0.481	187.95	491
*Synthetic*2	2.753	201.68	606
*Synthetic*3	2.686	197.67	622

Table 4: Comparison among three inner solvers when implementing Algorithm 1 on three synthetic datasets. Ave Time: average time for solving the subproblems; Ave Iter: average number of inner iterations; Max Iter: maximum number of inner iterations.

(6.1) on the three synthetic datasets. We terminate Algorithm 1 when the relative error $[F(x) - F^*]/F^* < 10^{-6}$. The stopping criterion for inner iterations is set as $\mathcal{E}(x) < 10^{-8}$, the initial point is set as $0.01 \cdot \mathbf{1}$ for all inner solvers, while the other settings of Algorithm 1 are kept identical for all the cases to ensure fair comparisons. The performance of the three inner solvers is presented in Table 4, and we can conclude that Algorithm 3 demonstrates superior performance compared to FISTA and ISTA across various problem sizes, as evidenced by significantly better results in all aspects. Consequently, Algorithm 3 has the potential to be employed as a subroutine in diverse proximal Newton-type methods, enabling them to handle high-dimensional problems.

6.2 Comparison between Algorithm 1 and SPQN-SVRG

The L-SVRG (Kovalev et al., 2020) presents experimental evidence supporting the claim that L-SVRG exhibits faster convergence compared to the SVRG across various tasks. In our study, we extend this comparison by evaluating the performance of Algorithm 1 in



(a) *rcv*1 dataset

(b) *a*9*a* dataset

(c) *w*8*a* dataset

(d) *mushrooms* dataset

Figure 2: Comparison between Algorithm 1 and SPQN-SVRG over four real datasets using different choices of parameters p and $l_s = 1/p$, where the indexes **-1, -2, -3** correspond to the cases where $p = \frac{b}{n}$, $\frac{1}{2}(\frac{b}{n} + \frac{\mu b}{L})$, and $\frac{\mu b}{L}$ respectively.

comparison to SPQN-SVRG. To this end, we use four real datasets: *rcv*1, *a*9*a*, *w*8*a*, and *mushrooms*, see Table 3.

We implement Algorithm 1 and SPQN-SVRG with different parameters choices for p and l_s. Recall that in this context, p represents the probability of updating the reference point in Algorithm 1 (see (3.6)), while l_s denotes the number of inner iterations in SPQN-SVRG. For fair comparisons, we use Algorithm 3 as the inner solver for both algorithms across all datasets, given its high efficiency demonstrated in Section 6.1.2. Additionally, we maintain consistent batch sizes of $b = 128$ and $b_H = 600$ across all cases. As it is expensive to estimate the bounds m_1 and m_2 for the Hessian approximations B_k as in Lemma 6, we follow Corollary 10 to select three distinct values of p: $\frac{b}{n}$, $\frac{\mu b}{L}$, and $\frac{1}{2}(\frac{b}{n} + \frac{\mu b}{L})$. We set $l_s = \frac{1}{p}$ to ensure that the expected number of iterations for updating the reference point remain the same for both algorithms. Furthermore, all the other parameters are the same as those specified in Section 6.1.1 for both Algorithm 1 and SPQN-SVRG.

Figure 2 reports the resulting training errors on the four real datasets. The figures effectively demonstrate that the training errors obtained by Algorithm 1 exhibit faster decreases compared to those obtained by SPQN-SVRG across various parameters choices, and therefore highlight the superior performance of Algorithm 1 for these tasks.

6.3 Testing error and prediction accuracy

In our previous experiments, we observed a rapid decrease in the training errors when using Algorithm 1. Moreover, it consistently outperformed SPQN-SVRG across various parameters choices. Nevertheless, it is essential to evaluate the testing error of the algorithm to assess its performance in avoiding overfitting. For this purpose, we use two real datasets $a9a$ and $w8a$, which have been split into the training set and testing set beforehand. For each dataset, we train the logistic regression model (1.1) using three different algorithms: Algorithm 1, SPQN, and P-LSVRG. The purpose of comparing these three methods is to highlight the advantages gained through the integration of Hessian information and VR techniques in the design of stochastic algorithms.

In the experiments, the parameters μ, λ, b, and b_H are kept consistent with the values specified in Section 6.1.1. The subproblems involved in Algorithm 1 and SPQN are solved by Algorithm 3. We record the iterates $\{x_k\}$ along the training process to calculate the corresponding testing errors on the testing dataset for each algorithm.

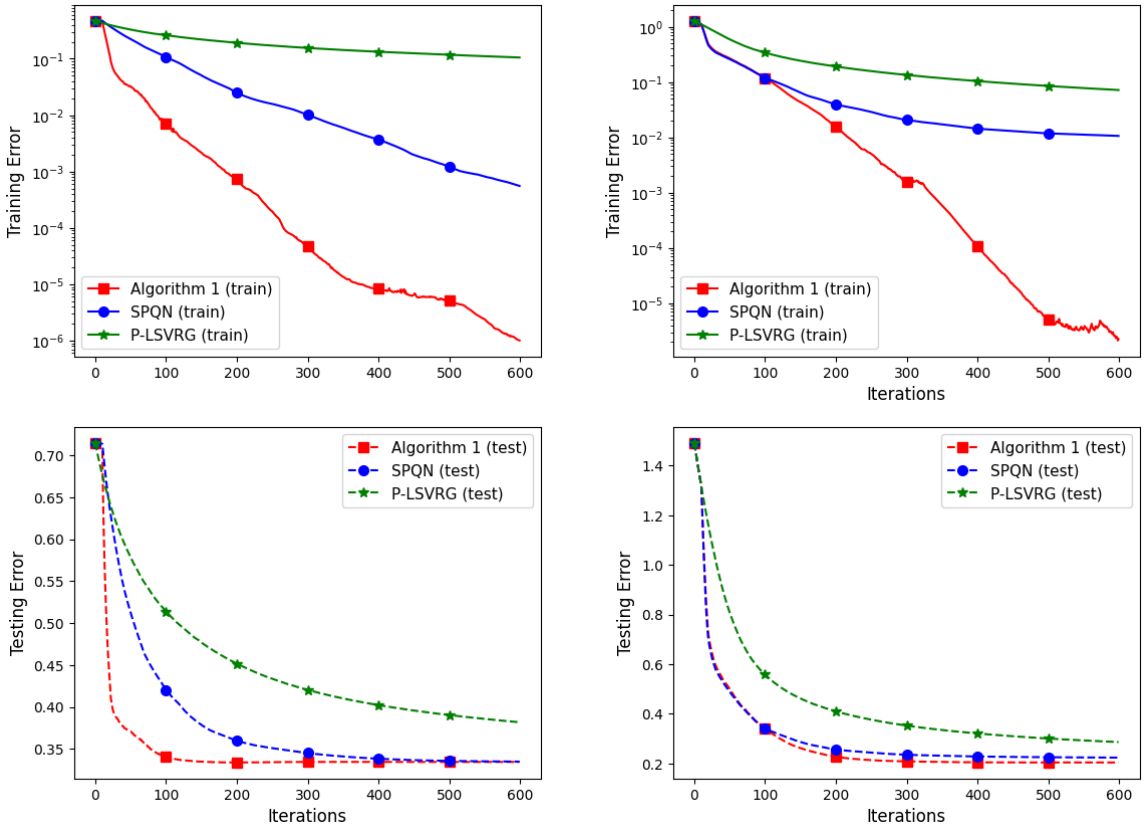
Figure 3 displays the training and testing errors for each algorithm. Based on the figures, we observe that Algorithm 1 consistently outperforms the other two algorithms in terms of both training and testing errors. This observation supports our expectation that the integration of Hessian information and VR techniques in Algorithm 1 leads to significant improvements in training efficiency while effectively avoiding overfitting.

We also present the prediction accuracy in Figure 4 to provide additional insights into the performance of these algorithms. From Figure 4, it can be observed that while both Algorithm 1 and SPQN achieve high prediction accuracy within a small number of iterations, Algorithm 1 demonstrates relatively better performance in terms of prediction accuracy compared to SPQN. In contrast, the prediction accuracy of the P-LSVRG increases relatively slowly. This observation further supports the superior performance of Algorithm 1 in terms of efficient convergence and achieving high prediction accuracy on the logistic binary classification task.

7. Conclusions

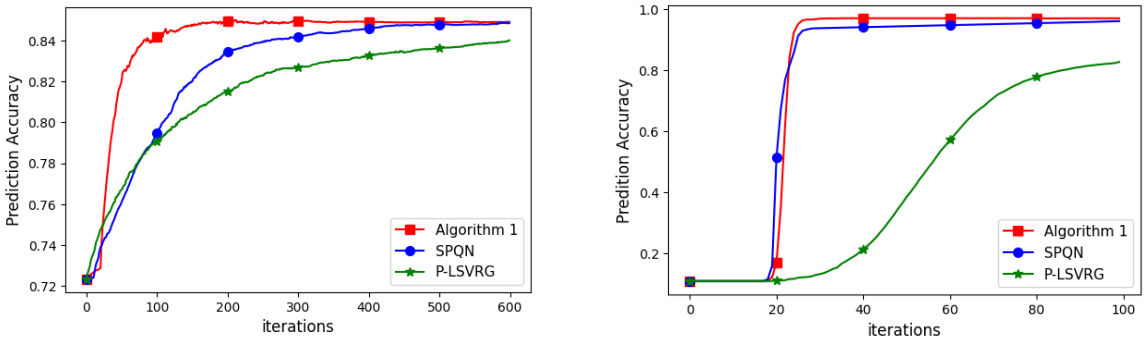
In this paper, we present a novel stochastic proximal quasi-Newton method for efficiently solving a class of convex optimization problems commonly encountered in machine learning, which involve structured objective functions consist of both smooth and nonsmooth components. The proposed method offers several advantages, which we summarize as follows:

- Structurally, our method combines the loopless SVRG technique with a stochastic L-BFGS scheme. By incorporating Hessian information and leveraging stochastic gradients with reduced variances, our method produces high-quality search directions in large-scale scenarios. Furthermore, compared to the commonly used SVRG scheme,



(a) $a9a$ dataset (b) $w8a$ dataset

Figure 3: Training and testing error for the logistic regression model on $a9a$ and $w8a$ datasets trained with three algorithms: Algorithm 1, SPQN, and P-LSVRG.



(a) $a9a$ dataset (b) $w8a$ dataset

Figure 4: Prediction accuracy for the logistic regression model on $a9a$ and $w8a$ datasets trained with three algorithms: Algorithm 1, SPQN, and P-LSVRG.

the generation of stochastic gradients via the loopless SVRG only exhibits a single loop, simplifying the implementation of our method.

- Theoretically, we establish a worst-case globally linear convergence rate for the proposed method under mild assumptions. We also discuss and demonstrate linear convergence for special cases and generalizations of our method, which allows for incorporating other variance reduction techniques.

- Numerically, we propose a fast Semismooth Newton (SSN) method along with a line search scheme to efficiently solve the subproblems. By leveraging a compact representation of the L-BFGS matrix and storing a few auxiliary matrices, we significantly reduce the computational burdens for implementing our method.

Both synthetic and real datasets are tested on a regularized logistic regression problem to validate the computational efficiency of our method along with the SSN solver. The results suggest that the proposed method outperforms several state-of-the-art algorithms, and the SSN-based numerical implementation significantly reduces the overall computational cost.

Acknowledgments

The authors are grateful to two anonymous referees for their very valuable comments which have helped improve the paper substantially. The work of Y. Song was supported by a Start-Up Grant from Nanyang Technological University. The work of X. Yuan was supported by the Hong Kong Research Grants Council under the GRF project 17309824. The work of H. Yue was supported by the National Natural Science Foundation of China (No. 12301399).

References

Alnur Ali, Eric Wong, and J Zico Kolter. A semismooth newton method for fast, generic convex programming. In *International Conference on Machine Learning*, pages 70–79. PMLR, 2017.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

Stephen Becker and Jalal Fadili. A quasi-Newton proximal splitting method. *Advances in Neural Information Processing Systems*, 25, 2012.

Stephen Becker, Jalal Fadili, and Peter Ochs. On quasi-Newton forward-backward splitting: Proximal calculus and convergence. *SIAM Journal on Optimization*, 29(4):2445–2481, 2019.

Albert S Berahas, Jorge Nocedal, and Martin Takác. A multi-batch L-BFGS method for machine learning. *Advances in Neural Information Processing Systems*, 29, 2016.

Joseph Berkson. Application of the logistic function to bio-assay. *Journal of the American Statistical Association*, 39(227):357–365, 1944.

Dimitri P Bertsekas. Incremental proximal methods for large scale convex optimization. *Mathematical Programming*, 129(2):163–195, 2011.

Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. *Advances in Neural Information Processing Systems*, 20, 2007.

Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

Léon Bottou et al. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, 91(8):12, 1991.

Stephen P Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Richard H Byrd, Gillian M Chin, Jorge Nocedal, and Figen Oztoprak. A family of second-order methods for convex l1 -regularized optimization. *Mathematical Programming*, 159: 435–467, 2016a.

Richard H Byrd, Samantha L Hansen, Jorge Nocedal, and Yoram Singer. A stochastic quasi-Newton method for large-scale optimization. *SIAM Journal on Optimization*, 26 (2):1008–1031, 2016b.

Richard H Byrd, Jorge Nocedal, and Figen Oztoprak. An inexact successive quadratic approximation method for l-1 regularized optimization. *Mathematical Programming*, 157 (2):375–396, 2016c.

Chih-Chung Chang. Libsvm data: Classification, regression, and multi-label. *http://www. csie. ntu. edu. tw/˜ cjlin/libsvmtools/datasets/*, 2008.

Frank H Clarke. *Optimization and Nonsmooth Analysis*. SIAM, 1990.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20: 273–297, 1995.

Ingrid Daubechies, Michel Defrise, and Christine De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 57(11):1413–1457, 2004.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in Neural Information Processing Systems*, 27, 2014.

John E Dennis and Jorge J Moré. A characterization of superlinear convergence and its application to quasi-Newton methods. *Mathematics of Computation*, 28(126):549–560, 1974.

John Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10:2899–2934, 2009.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(7), 2011.

William Feller. *An Introduction to Probability Theory and Its Applications, Volume 2*, volume 2. John Wiley & Sons, 1971.

Roger Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, 2000.

G-H Golub and C-F Van Loan. *Matrix Computations*. JHU Press, 2013.

Robert M Gower, Mark Schmidt, Francis Bach, and Peter Richtárik. Variance-reduced methods for machine learning. *Proceedings of the IEEE*, 108(11):1968–1983, 2020.

Tian-De Guo, Yan Liu, and Cong-Ying Han. An overview of stochastic quasi-Newton methods for large-scale machine learning. *Journal of the Operations Research Society of China*, 11(2):245–275, 2023.

Filip Hanzely, Konstantin Mishchenko, and Peter Richtárik. SEGA: Variance reduction via gradient sketching. *Advances in Neural Information Processing Systems*, 31, 2018.

Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, volume 2. Springer, 2009.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in Neural Information Processing Systems*, 26, 2013.

Hiroyuki Kasai, Hiroyuki Sato, and Bamdev Mishra. Riemannian stochastic quasi-Newton algorithm with variance reduction and its convergence analysis. In *International Conference on Artificial Intelligence and Statistics*, pages 269–278. PMLR, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Dmitry Kovalev, Samuel Horváth, and Peter Richtárik. Don't jump through hoops and remove those loops: SVRG and Katyusha are better without the outer loop. In *Algorithmic Learning Theory*, pages 451–467. PMLR, 2020.

John Langford, Lihong Li, and Tong Zhang. Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, 10(3), 2009.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Jason D Lee, Yuekai Sun, and Michael A Saunders. Proximal Newton-type methods for minimizing composite functions. *SIAM Journal on Optimization*, 24(3):1420–1443, 2014.

Evgeny S Levitin and Boris T Polyak. Constrained minimization methods. *USSR Computational Mathematics and Mathematical Physics*, 6(5):1–50, 1966.

David D Lewis, Yiming Yang, Tony Russell-Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5(Apr): 361–397, 2004.

Huan Li, Cong Fang, and Zhouchen Lin. Accelerated first-order optimization algorithms for machine learning. *Proceedings of the IEEE*, 108(11):2067–2082, 2020.

Pierre-Louis Lions and Bertrand Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 16(6):964–979, 1979.

Dong C Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, 1989.

Xuanqing Liu, Cho-Jui Hsieh, Jason D Lee, and Yuekai Sun. An inexact subsampled proximal Newton-type method for large-scale machine learning. *arXiv preprint arXiv:1708.08552*, 2017.

Nicolas Loizou and Peter Richtárik. Momentum and stochastic momentum for stochastic gradient, Newton, proximal point and subspace descent methods. *Computational Optimization and Applications*, 77(3):653–710, 2020.

Aurelien Lucchi, Brian McWilliams, and Thomas Hofmann. A variance reduced stochastic newton method. *arXiv preprint arXiv:1503.08316*, 2015.

Luo Luo, Zihao Chen, Zhihua Zhang, and Wu-Jun Li. A proximal stochastic quasi-Newton algorithm. *arXiv preprint arXiv:1602.00223*, 2016.

Boris S Mordukhovich and M Ebrahim Sarabi. Generalized Newton algorithms for tilt-stable minimizers in nonsmooth optimization. *SIAM Journal on Optimization*, 31(2): 1184–1214, 2021.

Boris S Mordukhovich, Xiaoming Yuan, Shangzhi Zeng, and Jin Zhang. A globally convergent proximal Newton-type method in nonsmooth convex optimization. *Mathematical Programming*, 198(1):899–936, 2023.

Philipp Moritz, Robert Nishihara, and Michael Jordan. A linearly-convergent stochastic L-BFGS algorithm. In *Artificial Intelligence and Statistics*, pages 249–258. PMLR, 2016.

Shummin Nakayama, Yasushi Narushima, and Hiroshi Yabe. Inexact proximal dc newton-type method for nonconvex composite functions. *Computational Optimization and Applications*, 87(2):611–640, 2024.

Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

Yu Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.

Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*, volume 87. Springer Science & Business Media, 2003.

Yurii Nesterov and Arkadii Nemirovskii. *Interior-point Polynomial Algorithms in Convex Programming*. SIAM, 1994.

Jorge Nocedal and Stephen J Wright. *Numerical Optimization*. Springer, 1999.

Gregory B Passty. Ergodic convergence to a zero of the sum of monotone operators in Hilbert space. *Journal of Mathematical Analysis and Applications*, 72(2):383–390, 1979.

Liqun Qi and Jie Sun. A nonsmooth version of Newton's method. *Mathematical Programming*, 58(1-3):353–367, 1993.

Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, 1999.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.

R Tyrrell Rockafellar. *Convex Analysis*, volume 11. Princeton University Press, 1997.

Nicolas Roux, Mark Schmidt, and Francis Bach. A stochastic gradient method with an exponential convergence _rate for finite training sets. *Advances in Neural Information Processing Systems*, 25, 2012.

Nicol N Schraudolph, Jin Yu, and Simon Günter. A stochastic quasi-Newton method for online convex optimization. In *Artificial Intelligence and Statistics*, pages 436–443. PMLR, 2007.

Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(1), 2013.

Lorenzo Stella, Andreas Themelis, and Panagiotis Patrinos. Forward–backward quasi-Newton methods for nonsmooth optimization problems. *Computational Optimization and Applications*, 67(3):443–487, 2017.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288, 1996.

Jialei Wang and Tong Zhang. Utilizing second order information in minibatch stochastic variance reduced proximal iterations. *Journal of Machine Learning Research*, 20(42): 1–56, 2019.

Philip Wolfe. Convergence conditions for ascent methods. *SIAM Review*, 11(2):226–235, 1969.

Lin Xiao and Tong Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

Minghan Yang, Andre Milzarek, Zaiwen Wen, and Tong Zhang. A stochastic extra-step quasi-Newton method for nonsmooth nonconvex optimization. *Mathematical Programming*, pages 1–47, 2021.

Matthew D Zeiler. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Junyu Zhang, Lin Xiao, and Shuzhong Zhang. Adaptive stochastic variance reduction for subsampled Newton method with cubic regularization. *INFORMS Journal on Optimization*, 4(1):45–64, 2022.

Renbo Zhao, William Benjamin Haskell, and Vincent YF Tan. Stochastic L-BFGS: improved convergence rates and practical acceleration strategies. *IEEE Transactions on Signal Processing*, 66(5):1155–1169, 2017.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 67(2):301–320, 2005.