RESEARCH-ARTICLE

Stochastic Variance-Reduced Iterative Hard Thresholding in Graph Sparsity Optimization

DEREK FOX, The University of North Carolina at Greensboro, Greensboro, NC, United States

SAMUEL HERNANDEZ, Texas A&M University, College Station, TX, United States

QIANQIAN TONG, The University of North Carolina at Greensboro, Greensboro, NC, United States

Published: 06 December 2024

CSAI 2024: 2024 8th International Conference on Computer Science and Artificial Intelligence (CSAI)
December 6 - 8, 2024
Beijing, China

https://doi.org/10.1145/3709026.3709040
ISBN: 9798400718182

Stochastic Variance-Reduced Iterative Hard Thresholding in Graph Sparsity Optimization

Derek Fox
UNCG
Greensboro, North Carolina, USA
defox@uncg.edu

Samuel Hernandez
Texas A&M University
College Station, Texas, USA
samuelhq11@tamu.edu

Qianqian Tong
UNCG
Greensboro, North Carolina, USA
q_tong@uncg.edu

Abstract

Stochastic optimization algorithms are widely used for large-scale data analysis due to their low per-iteration costs, but they often suffer from slow asymptotic convergence caused by inherent variance. Variance-reduced techniques have been therefore used to address this issue in structured sparse models utilizing sparsity-inducing norms or ℓ_0-norms. However, these techniques are not directly applicable to complex (non-convex) graph sparsity models, which are essential in applications like disease outbreak monitoring and social network analysis. In this paper, we introduce two stochastic variance-reduced gradient-based methods to solve graph sparsity optimization: GraphSVRG-IHT and GraphSCSG-IHT. We provide a general framework for theoretical analysis, demonstrating that our methods enjoy a linear convergence speed. Extensive experiments validate the efficiency and effectiveness of our proposed algorithms.

CCS Concepts

• **Computing methodologies** → **Machine learning algorithms**.

Keywords

graph sparsification, iterative hard thresholding, stochastic algorithms, variance reduction

ACM Reference Format:

Derek Fox, Samuel Hernandez, and Qianqian Tong. 2024. Stochastic Variance-Reduced Iterative Hard Thresholding in Graph Sparsity Optimization. In *2024 8th International Conference on Computer Science and Artificial Intelligence (CSAI) (CSAI 2024), December 06–08, 2024, Beijing, China.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3709026.3709040

1 Introduction

Graph structures enable the imposition of intricate sparsity constraints on the model, allowing them to better reflect relationships present in the data. For instance, graph-structured sparsity models are well-suited for predicting the spread of diseases or identifying social groups in networks. The search of connected subgraphs or clusters has a significant impact on identifying disease-related genes [1, 2, 22, 23]. Graph sparsification, which aims to reduce the complexity of large-scale graphs while preserving their essential structural properties, has garnered increasing attention as a crucial technique in modern data analysis and machine learning [6, 11–13, 30].

Graph sparsification can be formulated as the following optimization problem:

$$\min_{x\in\mathbb{R}^p} F(x), \quad F(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x), \tag{1}$$

which is known as the empirical risk minimization problem. Each $f_i(x)$ ($i \in [n]$) is convex and differentiable, and the graph structured sparsity is reflected by the constraint set $\mathbb{R}^p$ on x. The input vector x denotes the parameter of the model, and the output $f_i(x)$ is defined as the loss associated with sample i. By minimizing the loss $F(x)$, we guide the model towards the optimal solution. Typically, sparsity can be encoded by adding sparsity-inducing norms or penalties such as ℓ_0 norm, ℓ_1 norm and mixed norms [7, 8, 10, 16, 24–26, 28, 29]. These models often involve convex penalties and can be solved using convex optimization algorithms [3–5]. However, dealing with more complex sparse settings, such as graph-structured models, is more challenging.

In stochastic optimization, iterative hard thresholding (IHT) methods include gradient descent IHT (GD-IHT) [16], stochastic gradient descent IHT (SGD-IHT) [21], hybrid stochastic gradient IHT (HSG-IHT) [31], and variance-reduced methods such as stochastic variance reduced gradient IHT (SVRG-IHT) [19], stochastically controlled stochastic gradient IHT (SCSG-IHT) [20]. These methods update the parameter iterate x via gradient descent or its variants, and then apply a hard thresholding (HT) operator to enforce sparsity of x, preserving the top s elements in x while setting other elements to zero. In the context of graph-structured sparse optimization, the SGD based IHT method, named GraphSto-IHT, achieves HT through the use of Head and Tail Projections, first described by [30]. Head and Tail Projections map arbitrary vectors from the data onto the graph while simultaneously enforcing model sparsity [12–14]. Specifically, Head Projection identifies and preserves the largest entries in x, while Tail Projection identifies the smallest entries and sets them to zero. By ignoring the small magnitude entries in the vector, these projections help prevent overfitting and ensure sparse solutions. Meanwhile, stochastic sampling of the data is used to speed up gradient calculations. A single data point or small batch of the data is selected and a gradient is calculated only with respect to that batch. This greatly decreases the computational costs associated with the gradient calculations. However, if the selected batch does not represent the whole dataset, the gradient may not accurately point towards a local minimum of the function, introducing variance into the gradient descent process.

CSAI 2024, Beijing, China

ACM ISBN 979-8-4007-1818-2/24/12
https://doi.org/10.1145/3709026.3709040

To reduce the randomness inherent in SGD, variance reduction techniques may be used such as SVRG [17], SAGA [9], or SCSG [18]. During each iteration, the history of the stochastic process is considered to regulate the newly calculated gradient and minimize large changes in direction. This improvement on SGD is our main interest; both proposed algorithms utilize this technique to leverage fast gradient calculations while still enjoying quick convergence. In this paper we leverage the recent success of stochastic variance-reduced algorithms for non-convex problems and propose a series of efficient stochastic optimization algorithms for graph-structured sparsity constraint problems. Specifically, we introduce two new stochastic variance-reduced gradient based methods, GraphSVRG-IHT and GraphSCSG-IHT, designed to solve graph sparsity optimization problems. By incorporating stochastic variance-reduced techniques and graph approximated projections (head and tail), our algorithms are specifically tailored for non-convex graph-structured sparsity constraints, leading to faster convergence and improved performance. We provide a comprehensive theoretical framework and conduct extensive experiments to validate the efficiency and effectiveness of our proposed methods. **Our main contributions** are summarized as follows.

- This work is the first to explore the application of stochastic variance-reduced methods to graph-structured sparsity problems. By using batch gradients to approximate computationally expensive full gradients, we enhance the efficiency of variance reduction. We further propose GraphSCSG-IHT by parameterizing the variance reduction technique for greater control. It employs two different batch sizes for gradient calculations and randomly selects the update rate of the current position from a geometric distribution.
- We theoretically prove GraphSCSG-IHT enjoys linear convergence rate with a constant learning rate. This analysis provides a robust theoretical framework for analyzing graph sparsification optimization.
- We conduct extensive experiments to validate our proposed algorithms. In addition to simulation tests, we also evaluate our methods on a real-world breast cancer dataset. Our experiments empirically demonstrate the efficiency and effectiveness of our methods compared to GraphSto-IHT and other deterministic methods.

2 Preliminaries

Notations. We use lowercase letters, e.g. x, to denote a vector and use $\|\cdot\|$ to denote the l_2-norm of a vector. The operator E represents taking expectation over all random variables, $[n]$ denotes the integer set $\{1, ..., n\}$. The notation $supp(x)$ means the support of x or the index set of non-zero elements in x.

Definition 1. *(Subspace model) [14] Given the space $\mathbb{R}^p$, a subspace model $\mathcal{M}$ is defined as a family of linear subspaces of $\mathbb{R}^p$:*

$$\mathcal{M} = \{S_1, S_2, \ldots, S_k, \ldots\}$$

where each S_k is a subspace of $\mathbb{R}^p$. The set of corresponding vectors in these subspaces is denoted as

$$\mathcal{M}(\mathcal{M}) = \{x : x \in V \text{ for some } V \in \mathcal{M}\}.$$

Definition 2. *(Weighted graph model) [13] Given an underlying graph $G = (V, E)$ defined on the coefficients of the unknown vector x, where $V = [p]$ and $E \subseteq V \times V$, then the weighted graph model (G, s, g, C)- WGM can be defined as the following set of supports*

$$\mathcal{M} = \{S : |S| \leq s, \exists F \subseteq V \text{ with } V_F = S, \gamma(F) = g, \text{ and } w(F) \leq C\},$$

where C is the budget on weight of edges w, g is the number of connected components of F, and s is the sparsity.

Definition 3. *(Projection Operator) [14] We define a projection operator onto $\mathcal{M}(\mathbb{M})$, i.e, $P(\cdot, \mathcal{M}(\mathbb{M})) : \mathbb{R}^p \to \mathbb{R}^p$ defined as*

$$P(x, \mathcal{M}(\mathbb{M})) = \arg \min_{y \in \mathcal{M}(\mathcal{M})} \|x - y\|^2.$$

With this projection, we introduce Head Projection to retain large magnitudes and Tail Projection to zero out small magnitudes.

Assumption 1. *(Head Projection)[14] Let M and M_H be the predefined subspace models. Given any vector x, there exists a (c_H, M, M_H)-Head-Projection which is to find a subspace $H \in M_H$ such that*

$$\|P(x, H)\|_2 \geq c_H \cdot \max_{S \in M} \|P(x, S)\|_2, \tag{1}$$

where $0 < c_H \leq 1$. We denote $P(x, H)$ as $P(x, M, M_H)$.

Assumption 2. *(Tail Projection)[14] Let M and M_T be the predefined subspace models. Given any vector x, there exists a (c_T, M, M_T)-Tail-Projection which is to find a subspace $T \in M_T$ such that*

$$\|P(x, T) - x\|_2 \leq c_T \cdot \min_{S \in M} \|x - P(x, S)\|_2, \tag{2}$$

where $c_T \geq 1$. We denote $P(x, T)$ as $P(x, M, M_T)$.

We introduce the widely used RSC/RSS definition, along with an additional assumption, to characterize the properties of the objective function.

Definition 4. *($(\alpha, \beta, M(M))$-RSC/RSS Properties)[14] We say a differentiable function $f(\cdot)$ satisfies the $(\alpha, \beta, M(M))$-Restricted Strong Convexity (RSC)/Smoothness (RSS) property if there exist positive constants α and β such that*

$$\frac{\alpha}{2}\|x - y\|_2^2 \leq B_f(x, y) \leq \frac{\beta}{2}\|x - y\|_2^2, \tag{3}$$

for all $x, y \in M(M)$, where $B_f(x, y)$ is the Bregman divergence of f, i.e.,

$$B_f(x, y) = f(x) - f(y) - \langle \nabla f(y), x - y \rangle.$$

α and β are the strong convexity parameter and strong smoothness parameter, respectively.

Assumption 3. *[14] Given the objective function $F(x)$ in (1), we assume that $F(x)$ satisfies α-RSC in subspace model $\mathcal{M}(\mathcal{M} \oplus \mathcal{M}_H \oplus \mathcal{M}_T)$. Each function $f_i(x)$ satisfies β-RSS in $\mathcal{M}(\mathcal{M} \oplus \mathcal{M}_H \oplus \mathcal{M}_T)$, where $\oplus$ of two models $\mathcal{M}_1$ and $\mathcal{M}_2$ is defined as $\mathcal{M}_1 \oplus \mathcal{M}_2 := \{S_1 \cup S_2 : S_1 \in \mathcal{M}_1, S_2 \in \mathcal{M}_2\}$.*

3 Methods

In this section, we introduce two proposed algorithms: GraphSVRG-IHT and GraphSCSG-IHT. Both algorithms employ the variance-reduced techniques derived from SVRG [17] and SCSG [18] respectively, while also utilizing the graph projection operators found in GraphSto-IHT[30]. This results in methods that are applicable to graph-structured sparsity problems and effectively reduce the stochastic variance. Therefore, our algorithms converge faster and more accurately than their predecessors.

3.1 GraphSVRG-IHT

Our proposed GraphSVRG-IHT (Algorithm 1) utilizes variance reduction by periodically computing the full gradient, significantly reducing the inherent variance in stochastic gradient methods. By incorporating graph projection operators, our GraphSVRG-IHT adapts to non-convex graph sparsity constraints, enhancing its applicability and efficiency. The key steps are outlined below:

(1) Calculate the full gradient, $\tilde{v}^j$, with the position at the start of each outer loop, $\tilde{x}^j$ (Line 4).
(2) In the inner loop, compute two gradients from a single sampled data point: one at the copied position x_k^j and the other at $\tilde{x}^j$. Then calculate the stochastic variance reduced gradient, v_k^j (Line 7-8).
(3) Pass v_k^j through the Head Projection operator (Line 9); and use the resulting gradient to update the next iterate x_k^j, through the Tail Projection operator (Line 10).
(4) After a fixed number ($\mathcal{K}$) of inner loop iterations, update the outer loop position $\tilde{x}^j$ and re-calculate the new full gradient.

Algorithm 1 GraphSVRG-IHT

1: **Input**: η, f_i, $\mathbb{M}$, $\mathbb{M}_{\mathcal{H}}$, $\mathbb{M}_{\mathcal{T}}$, $\mathcal{J}$, $\mathcal{K}$
2: **Initialize**: $\tilde{x}^1$ such that $supp(\tilde{x}^1) \in \mathbb{M}$
3: **for** $j = 1$ to $\mathcal{J}$ **do**
4: $\quad \tilde{v}^j = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\tilde{x}^j)$
5: $\quad x_0^j = \tilde{x}^j$
6: $\quad$ **for** $k = 1$ to $\mathcal{K}$ **do**
7: $\quad\quad$ Randomly pick $i_k \in [n]$
8: $\quad\quad v_k^j = \nabla f_{i_k}(x_{k-1}^j) - \nabla f_{i_k}(\tilde{x}^j) + \tilde{v}^j$
9: $\quad\quad \tau_k = P(v_k^j, \mathbb{M} \oplus \mathbb{M}_{\mathcal{T}}, \mathbb{M}_{\mathcal{H}})$
10: $\quad\quad x_k^j = P(x_{k-1}^j - \eta\tau_k, \mathbb{M}, \mathbb{M}_{\mathcal{T}})$
11: $\quad$ **end for**
12: $\quad \tilde{x}^{j+1} = x_{\mathcal{K}}^j$
13: **end for**

Our proposed GraphSVRG-IHT algorithm differs from GraphSto-IHT in that GraphSVRG-IHT uses nested loops, allowing it to account for the history of the stochastic process, whereas GraphSto-IHT only considers a stochastic gradient. Both methods implement Head and Tail Projections for hard thresholding, making them applicable to graph-structured sparsity optimization problems. The inclusion of stochastic variance in GraphSVRG-IHT makes the theoretical analysis more complex and challenging.

3.2 GraphSCSG-IHT

To better understand the calculation of variance-reduced gradients and stochastically control the outer batch size, we propose the GraphSCSG-IHT (Algorithm 2). While similar to Algorithm 1 in its use of variance-reduced gradients, GraphSCSG-IHT has the following key characteristics:

(1) In the outer loop, the gradient is calculated using a batch of data of size B, whereas Algorithm 1 calculates a full gradient at this step (Line 4-5).
(2) In the inner loop, when calculating the stochastic variance reduced gradient, a mini-batch is used instead of a single data point (Line 10-11).
(3) The number of inner loops, $\mathcal{K}^j$, is not fixed. Instead, $\mathcal{K}^j$ is chosen from a geometric distribution (Line 7) or can be set as $\frac{B}{b}$ (Line 8).

Algorithm 2 GraphSCSG-IHT

1: **Input**: η, f_i, $\mathbb{M}$, $\mathbb{M}_{\mathcal{H}}$, $\mathbb{M}_{\mathcal{T}}$, B, b, $\mathcal{J}$
2: **Initialize**: $\tilde{x}^1$ such that $supp(\tilde{x}^1) \in \mathbb{M}$
3: **for** $j = 1$ to $\mathcal{J}$ **do**
4: $\quad$ Uniformly sample a batch $I^j \subset [n]$, s.t. $|I^j| = B$
5: $\quad \tilde{\mu}^j = \nabla f_{I^j}(\tilde{x}^j)$
6: $\quad x_0^j = \tilde{x}^j$
7: $\quad$ (Option I) Generate $\mathcal{K}^j \sim Geom\left(\frac{B}{B+b}\right)$
8: $\quad$ (Option II) $\mathcal{K}^j = \frac{B}{b}$
9: $\quad$ **for** $k = 1$ to $\mathcal{K}^j$ **do**
10: $\quad\quad$ Uniformly sample a mini-batch $I_k^j \subset [n]$, s.t. $|I_k^j| = b$
11: $\quad\quad \mu_k^j = \nabla f_{I_k^j}(x_{k-1}^j) - \nabla f_{I_k^j}(\tilde{x}^j) + \tilde{\mu}^j$
12: $\quad\quad \tau_k = P(\mu_k^j, \mathbb{M} \oplus \mathbb{M}_{\mathcal{T}}, \mathbb{M}_{\mathcal{H}})$
13: $\quad\quad x_k^j = P(x_{k-1}^j - \eta\tau_k, \mathbb{M}, \mathbb{M}_{\mathcal{T}})$
14: $\quad$ **end for**
15: $\quad \tilde{x}^{j+1} = x_{\mathcal{K}^j}^j$
16: **end for**

In contrast to GraphSto-IHT, the inner loop of GraphSCSG-IHT computes the gradient over a random set of functions rather than a randomly selected function. GraphSCSG-IHT also employs two loops with different batch sizes for gradient calculation, making it more flexible than GraphSto-IHTand GraphSVRG-IHT. This flexibility allows GraphSCSG-IHT to serve as a general framework for graph constrained optimization.

In summary, compared with traditional IHT and StoIHT methods, graph-structured hard thresholding steps mainly differ in their use of Head and Tail Projections. Both proposed new algorithms are much more complex than GraphSto-IHT, introducing distinct variance reduction techniques while maintaining the same sparsification enforcement. It is also important to note that GraphSVRG-IHT is a specific scenario of GraphSCSG-IHT, hence, we provide theoretical analysis of GraphSCSG-IHT, which can be easily extended to GraphSVRG-IHT case.

4 Theoretical Analysis

In this section, we present our main theoretical results characterizing the estimation error of parameters x. The proof provides a general framework based on the gradients from GraphSCSG-IHT. We demonstrate the convergence of our algorithms by bounding the final error using the ℓ^2 norm of the initial and the optimal distance. Additionally, we consider the history of the stochastic process up to iteration $j * \mathcal{K}$ with the notation $I_{\mathcal{K}}^j$. Due to the limited space, all of our formal proofs are provided in the Supplementary Material.

Lemma 1. *[30] If each $f_{\xi_t}(\cdot)$ and $F(x)$ satisfy Assumption 3, and given head projection model $(c_H, M \oplus M_T, M_H)$ and tail projection model (c_T, M, M_T), then we have the following inequality*

$$\mathbb{E}_{\xi_t}\|(x^t - x^*)_H\| \leq \sqrt{1-\alpha_0}\mathbb{E}_{\xi_t}\|x^t - x^*\| + \sigma_1,$$

where $\sigma_1 = \left(\frac{\beta_0}{\alpha_0} + \sqrt{\frac{\alpha_0\beta_0}{1-\alpha_0}}\right)\mathbb{E}_{\xi_t}\|\nabla_I f_{\xi_t}(x^*)\|$, $H = supp(P(\nabla f_{\xi_t}(x^t), M \oplus M_T, M_H))$, $\alpha_0 = c_H\alpha\tau - \sqrt{\alpha\beta\tau^2 - 2\alpha\tau + 1}$, $\beta_0 = (1+c_H)\tau$, $I = \arg\max_{S\in M\oplus M_T\oplus M_H}\mathbb{E}_{\xi_t}\|\nabla_S f_{\xi_t}(x^*)\|$, *and* $\tau \in (0, 2/\beta)$.

With the prepared lemmas and appropriate assumptions, we can now present our main result.

Theorem 2. *Assume that Definition 4 and Assumption 3 hold. Under the same setting of Lemma 1, let* x_0 *be the start point. If we choose a constant learning rate* η *within* $\eta \in \left(\frac{2\alpha - \sqrt{4\alpha^2 - 3.75\alpha\beta}}{2\alpha\beta}, \frac{2\alpha + \sqrt{4\alpha^2 - 3.75\alpha\beta}}{2\alpha\beta}\right)$ *then the solution* $\tilde{x}^{j+1}$ *of GraphSCSG-IHT satisfies*

$$\mathbb{E}_{I_K^j}\|\tilde{x}^{j+1} - x^*\| \le \left[\left(\frac{\delta}{1-\lambda}\right)^S + \lambda^j\left(\frac{1-\lambda}{1-\lambda-\delta}\right)\right]\|x^0 - x^*\| + \frac{\gamma}{1-\lambda-\delta}\mathbb{E}_{I_K^j}\|\nabla_I f_{I^j}(x^*)\|,$$

where $\delta = (1+c_{\mathcal{T}})\left(\sqrt{\alpha\beta\eta^2 - 2\alpha\eta + 1} + \sqrt{1-\alpha_0}\right)$, $\lambda = (1+c_{\mathcal{T}})(2\sqrt{\alpha\beta\eta^2 - 2\alpha\eta + 1})$, $\gamma = (1+c_{\mathcal{T}})(\frac{\beta_0}{\alpha_0} + \frac{\alpha_0\beta_0}{\sqrt{1-\alpha_0{}^2}} + \eta)$.

Theorem 2 shows that our algorithm achieves linear convergence using stochastic variance-reduced gradients, with error decreasing as iterations increase. This aligns with our experiments, where more iterations reduce errors. We further derive a corollary that supports the convergence and specifies the range for η.

Corollary 2.1. *To ensure convergence of our algorithm, the learning rate* η*, which is a constant, should be chosen within the range* $(\frac{2\alpha - \sqrt{4\alpha^2 - 3.75\alpha\beta}}{2\alpha\beta}, \frac{2\alpha + \sqrt{4\alpha^2 - 3.75\alpha\beta}}{2\alpha\beta})$*. For this range to be valid, the following inequality must hold:* $\frac{\delta}{1-\lambda} < 1$.

Corollary 2.1 is the cornerstone of Theorem 2. It ensures that the upper bound for the estimation error does not blow up to infinity, and provides a constant value for the finite series. Similarly, it also ensures that the upper bound will decay more after performing more iterations. Corollary 2.1 also provides a range of η, which is smaller than the one given by GraphSto-IHT. This way we can find η such that the algorithm will always converge.

5 Experiments

5.1 Experimental setup

We perform multiple experiments to compare our proposed algorithms with baseline methods. For our experiments, we consider the residual norm of the loss function, $\|\mathbf{A}x^{t+1} - y\|$ as the number of epochs increases. Due to the non-convex nature of the problem, there are several local minima and the algorithm may not approach the global minimum, x^*. Additionally, in real-world applications, x^* is often unknown. Therefore, we use the residual norm as a measurement of convergence as opposed to the distance from the final iterate to the target vector, $\|x^{t+1} - x^*\|$. All experiments are tested on a Ubuntu 22.04.4 server with 256 AMD EPYC 9554 64-core processors and with 1.6 TB RAM. All codes are written in Python[1].

[1]All code is available at https://github.com/Derek-Fox/graph-scsg-iht

5.2 Synthetic Dataset

We first tested our methods on synthetic datasets to determine the optimal parameters. For a fair comparison, we followed the exact settings used in GraphSto-IHT, conducting multiple experiments using a grid graph with a dimension of 256 and unit-weight edges.

Choice of η. To study the effect of the learning rate on the performance of our algorithms, we varied η across {0.1, 0.01, 0.001} and tested these rates in various sparsity cases, with sparsity values chosen from {256, 128, 64, 32}. By varying the learning rate, we aimed to understand the convergence behaviour of our algorithms. As shown in Figure 1, our experiments reveal that, as expected, with a larger learning rate, all methods converge quickly, while with a smaller learning rate, the steps take longer to achieve zero residual loss. Additionally, both our proposed methods and the baseline method converge stably, and regardless of the learning rate setting, GraphSVRG-IHT consistently performs the best.

Choice of sparsity. Studying the setting of sparsity, s, is crucial for understanding the performance of our algorithms in graph sparsification optimization. To examine the effect of sparsity, we compared our methods, GraphSVRG-IHT and GraphSCSG-IHT against the baseline algorithm GraphSto-IHT. Using the experimental settings from GraphSto-IHT we employed a grid graph of dimension 256, fixed the learning rate $\eta = 0.01$ and set the batch size equal to the sparsity parameter $B = s$. We varied the sparsity parameter s to observe the behavior of all algorithms, as shown in Figure 2. Figure 2 demonstrates that as the sparsity parameter s decreases, GraphSVRG-IHT outperforms the other algorithms in minimizing the residual norm $\|\mathbf{A}x^{t+1} - y\|$ over epochs. Another interesting finding is that we observed that GraphSto-IHT and GraphSCSG-IHT display almost identical behavior in this parameter setting.

Choice of batch size. After exploring the choice of η and s, we varied the batch size B on a grid graph with a dimension of 256 to demonstrate the advantages of GraphSCSG-IHT. Here we fixed $\eta = 0.01$, $s = 32$. For fair comparison, we considered the number of data points instead of the number of epochs to estimate the run time of the algorithms, which is a common practice in optimization.

Since gradient calculations are computationally expensive, using fewer data points would result in faster run times. Figure 3 shows three different scenarios with varying batch sizes B. When B equals to the dimension, GraphSCSG-IHT degrades to GraphSVRG-IHT resulting in similar performance. With smaller B (meaning fewer data points are used in the full gradient calculation), GraphSCSG-IHT consistently outperforms GraphSVRG-IHT. However, as B continues to decrease, GraphSCSG-IHT method initially outperforms GraphSVRG-IHT but was eventually surpassed by GraphSVRG-IHT slightly. This occurs because smaller batch size increase gradient variance, making it harder for GraphSCSG-IHT to maintain its initial advantage. Furthermore, the interaction between batch size and mini-batch size becomes more complex, ultimately affecting the final performance.

Therefore, we also conducted numerous experiments to study the effects of mini-batch size in conjunction with batch size. Additionally, to better understand the graph-structure patterns, we have explored how varying the number of connected components in subgraphs impacts final convergence performance. Additional results are provided in the Appendix.

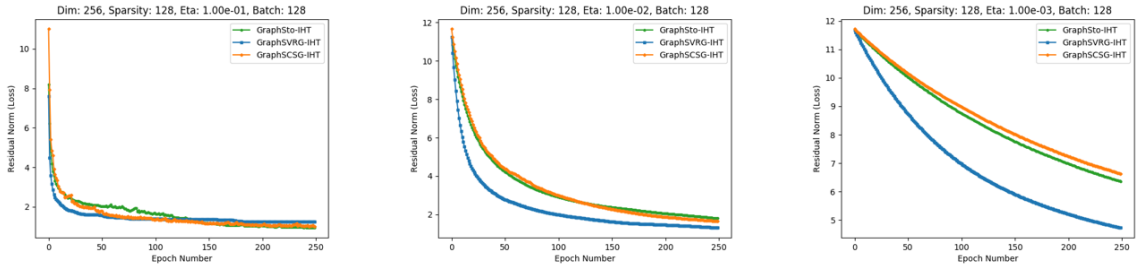

Figure 1: Comparison of methods with different learning rate.

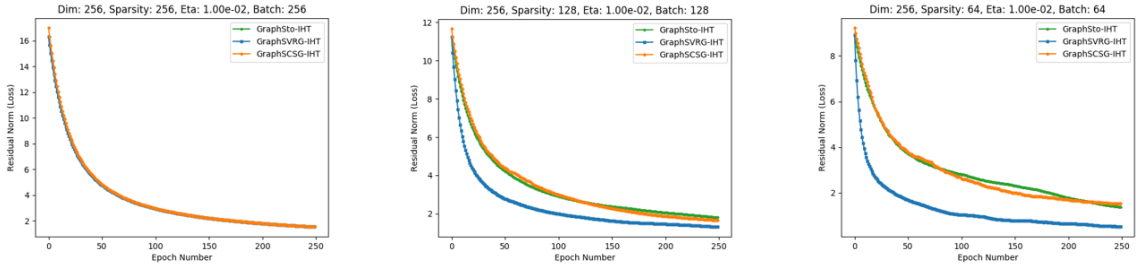

Figure 2: Comparison of methods with different sparsities.

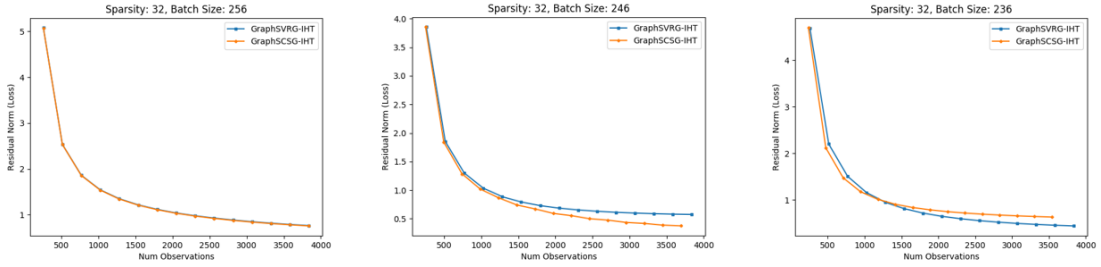

Figure 3: Number of data points vs. residual loss value with different batch size.

5.3 Real-world Dataset

To test our algorithms on a real-world dataset, we use a large breast cancer dataset [27]. This dataset contains 295 training samples with 8,141 genes dimensions, including 78 positive (metastatic) and 217 negative (non-metastatic) samples. Following the experimental setting in [30], we use a Protein-Protein Interaction network with 637 pathways from [15], the dataset is folded into 5 subfolds, and 20 trials are conducted.

Table 1 shows the results from the gene identification task on the breast cancer dataset. GraphSCSG-IHT identifies 40% of the 25 genes highly correlated with breast cancer, as gathered by [30]. GraphSto-IHT and GraphSVRG-IHT both identify 24% of these genes, consistent with the findings in [30]. All the graph-structured methods greatly outperform the IHT and StoIHT methods which only identify 8% of the cancer-related genes. These results demonstrate the promise of variance-reduction techniques for stochastic gradient descent in the setting of graph sparsity optimization.

In summary, GraphSVRG-IHT is more stable while GraphSCSG-IHT shows excellent performance under appropriate settings due to the introduction of two additional batch size. This allows for greater flexibility and control in the gradient estimation process. Overall, our two new algorithms demonstrate both efficiency and effectiveness by incorporating variance reduction techniques in graph sparsity problems, making them suitable for a wide range of applications. However, our empirical evaluation has some limitations, as it mainly focused on synthetic and medical datasets, which may not represent the full diversity of real-world scenarios. Future work will explore the performance of our algorithms on a wider range of data types and application contexts.

Table 1: Identified genes from breast cancer dataset

Method	Number	Genes
IHT	2	NAT1 TOP2A
StoIHT	2	NAT1 TOP2A
GraphSto-IHT	6	AR ATM BRCA2 CCND2 CDKN1A TOP2A
GraphSVRG-IHT	6	AR ATM BRCA2 CCND2 CDKN1A TOP2A
GraphSCSG-IHT	10	AR ATM BRCA1 BRCA2 CCND2 CCND3 CDKN1A CHEK2 FBP1 TOP2A

6 Conclusion

We have proposed two algorithms to utilize variance reduction techniques in the setting of graph sparse optimization. The proposed algorithms can significantly improve the stability and efficiency in machine learning and other applications. Theoretically, we present a proof framework demonstrating linear convergence, though the analysis assumes certain conditions, such as the sparsity and structure of the underlying graph, which may not always be present in practice. Empirically, our algorithms are competitive in minimizing the objective loss function compared to their predecessors in various experimental settings with a synthetic dataset. Additionally, testing on a large-scale medical dataset demonstrated superior performance in identifying cancer-related genes. Future work should include testing our algorithms on more larger real-world datasets. By employing graph-structured hard thresholding, we can uncover more underlying subgraphs and related patterns, with significant implications in fields such as medical research and social network analysis. This approach can enhance our understanding of complex data structures and lead to more effective solutions.

Acknowledgments

This work was completed at the University of North Carolina at Greensboro, funded by NSF REU program 2349369.

References

[1] Cem Aksoylar, Lorenzo Orecchia, and Venkatesh Saligrama. 2017. Connected subgraph detection with mirror descent on SDPs. In *International Conference on Machine Learning*. PMLR, 51–59.
[2] Ery Arias-Castro, Emmanuel J Candes, and Arnaud Durand. 2011. Detection of an anomalous cluster in a network. *The Annals of Statistics* (2011), 278–304.
[3] Francis Bach, Rodolphe Jenatton, Julien Mairal, and Guillaume Obozinski. 2012. Structured sparsity through convex optimization. *Statist. Sci.* 27, 4 (2012), 450–468.
[4] Francis Bach, Rodolphe Jenatton, Julien Mairal, Guillaume Obozinski, et al. 2012. Optimization with sparsity-inducing penalties. *Foundations and Trends® in Machine Learning* 4, 1 (2012), 1–106.
[5] Sohail Bahmani, Bhiksha Raj, and Petros T Boufounos. 2013. Greedy sparsity-constrained optimization. *Journal of Machine Learning Research* 14, Mar (2013), 807–841.
[6] Richard G Baraniuk, Volkan Cevher, Marco F Duarte, and Chinmay Hegde. 2010. Model-based compressive sensing. *IEEE Transactions on information theory* 56, 4 (2010), 1982–2001.
[7] Thomas Blumensath and Mike E Davies. 2009. Iterative hard thresholding for compressed sensing. *Applied and computational harmonic analysis* 27, 3 (2009), 265–274.
[8] Scott Shaobing Chen, David L Donoho, and Michael A Saunders. 2001. Atomic decomposition by basis pursuit. *SIAM review* 43, 1 (2001), 129–159.
[9] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. 2014. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in neural information processing systems*. 1646–1654.
[10] Simon Foucart. 2011. Hard thresholding pursuit: an algorithm for compressive sensing. *SIAM J. Numer. Anal.* 49, 6 (2011), 2543–2563.
[11] Chinmay Hegde, Piotr Indyk, and Ludwig Schmidt. 2014. A fast approximation algorithm for tree-sparse recovery. In *2014 IEEE International Symposium on Information Theory*. IEEE, 1842–1846.
[12] Chinmay Hegde, Piotr Indyk, and Ludwig Schmidt. 2015. Approximation algorithms for model-based compressive sensing. *IEEE Transactions on Information Theory* 61, 9 (2015), 5129–5147.
[13] Chinmay Hegde, Piotr Indyk, and Ludwig Schmidt. 2015. A nearly-linear time framework for graph-structured sparsity. In *International Conference on Machine Learning*. PMLR, 928–937.
[14] Chinmay Hegde, Piotr Indyk, and Ludwig Schmidt. 2016. Fast recovery from a union of subspaces. *Advances in Neural Information Processing Systems* 29 (2016).
[15] Laurent Jacob, Guillaume Obozinski, and Jean-Philippe Vert. 2009. Group lasso with overlap and graph lasso. In *Proceedings of the 26th annual international conference on machine learning*. 433–440.
[16] Prateek Jain, Ambuj Tewari, and Purushottam Kar. 2014. On iterative hard thresholding methods for high-dimensional m-estimation. In *Advances in Neural Information Processing Systems*. 685–693.
[17] Rie Johnson and Tong Zhang. 2013. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*. 315–323.
[18] Lihua Lei and Michael Jordan. 2017. Less than a single pass: Stochastically controlled stochastic gradient. In *Artificial Intelligence and Statistics*. 148–156.
[19] Xingguo Li, Tuo Zhao, Raman Arora, Han Liu, and Jarvis Haupt. 2016. Stochastic variance reduced optimization for nonconvex sparse learning. In *International Conference on Machine Learning*. 917–925.
[20] Guannan Liang, Qianqian Tong, Chunjiang Zhu, and Jinbo Bi. 2020. An effective hard thresholding method based on stochastic variance reduction for nonconvex sparse learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34(02). 1585–1592.
[21] Nam Nguyen, Deanna Needell, and Tina Woolf. 2017. Linear convergence of stochastic iterative greedy algorithms with sparse constraints. *IEEE Transactions on Information Theory* 63, 11 (2017), 6869–6895.
[22] Jing Qian, Venkatesh Saligrama, and Yuting Chen. 2014. Connected sub-graph detection. In *Artificial Intelligence and Statistics*. PMLR, 796–804.
[23] Polina Rozenshtein, Aris Anagnostopoulos, Aristides Gionis, and Nikolaj Tatti. 2014. Event detection in activity networks. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1176–1185.
[24] Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)* 58, 1 (1996), 267–288.
[25] Berwin A Turlach, William N Venables, and Stephen J Wright. 2005. Simultaneous variable selection. *Technometrics* 47, 3 (2005), 349–363.
[26] Sara A Van de Geer et al. 2008. High-dimensional generalized linear models and the lasso. *The Annals of Statistics* 36, 2 (2008), 614–645.
[27] Marc J Van De Vijver, Yudong D He, Laura J Van't Veer, Hongyue Dai, Augustinus AM Hart, Dorien W Voskuil, George J Schreiber, Johannes L Peterse, Chris Roberts, Matthew J Marton, et al. 2002. A gene-expression signature as a predictor of survival in breast cancer. *New England Journal of Medicine* 347, 25 (2002), 1999–2009.
[28] Ming Yuan and Yi Lin. 2006. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society Series B: Statistical Methodology* 68, 1 (2006), 49–67.
[29] Xiaotong Yuan, Ping Li, and Tong Zhang. 2014. Gradient hard thresholding pursuit for sparsity-constrained optimization. In *International Conference on Machine Learning*. 127–135.
[30] Baojian Zhou, Feng Chen, and Yiming Ying. 2019. Stochastic iterative hard thresholding for graph-structured sparsity optimization. In *International Conference on Machine Learning*. PMLR, 7563–7573.
[31] Pan Zhou, Xiaotong Yuan, and Jiashi Feng. 2018. Efficient stochastic gradient hard thresholding. In *Advances in Neural Information Processing Systems*. 1988–1997.

A Parameters Table

Table: Constraints for Parameters

Notation	Name	Constraint
η	Step size	Fixed
F()	Function	Minimized
$\mathbb{M}$	Set of Supports	
$\mathbb{M}_{\mathcal{H}}$	Set of Supports	
$\mathbb{M}_{\mathcal{T}}$	Set of Supports	
B	Big batch	
I^j	Big batch sample	$I^j \subset [n]$ and $\lvert I^j \rvert = B$
b	Small batch	Subset of B
I_k^j	Small batch sample	$I_k^j \subset [n]$ and $\lvert I_k^j \rvert = b$
$\mathcal{J}$	Number of outer loops	
j	Outer loop index	
$\mathcal{K}$	Number of inner loops	
k	Inner loop index	
$\tilde{x}^1$	Starting Position	$supp(\tilde{x}^1) \in \mathbb{M}$
s	Number of non-zero entries	Sparsity Constraint
$P(x, \mathbb{M}, \mathbb{M}_{\mathcal{H}})$	Head Projection	
$P(x, \mathbb{M}, \mathbb{M}_{\mathcal{T}})$	Tail Projection	
$\tilde{v}^j$	SVRG Gradient	
$\tilde{\mu}^j$	SCSG Gradient	
$\tilde{x}^j$	Current Position	
v_k^j	Reduced Variance Gradient	
μ_k^j	Reduced Variance Gradient	
τ_k	Sparsified Gradient	
g	Number of Connected Components	

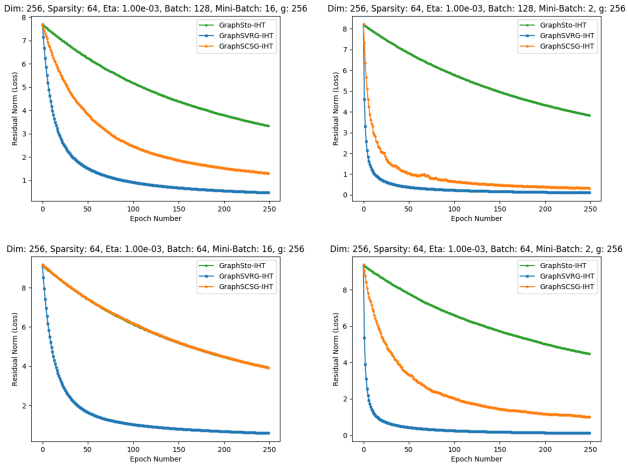

Figure 4: Various choices for B and b

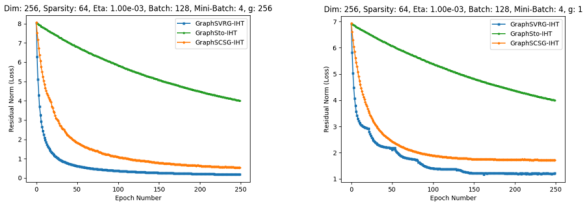

Figure 5: Different number of Connected Components

B More Results

B.1 Simulation

As pointed out in our main paper, we also explored the effects of mini-batch size in conjunction with batch size. Figure 4 demonstrates the generality of GraphSCSG-IHT. With different settings of B and b GraphSCSG-IHT can range in behavior between GraphSto-IHT and GraphSVRG-IHT. Not included in the figure is the case when B equals the dimension of the data and $b = 1$, in which case GraphSCSG-IHT and GraphSVRG-IHT are only distinguished by the number of inner loops (which could be parameterized itself to make the two algorithms identical).

We vary the number of connected components, controlled by parameter g, in Figure 5. We observe that when g is low (in this case $g = 1$), the loss function curve does not smoothly decrease but frequently levels off before stepping downward. This is due to the algorithm intermittently finding a more suitable support set, which allows it to more rapidly decrease the objective loss. The restriction of the algorithm to only one connected component makes this phenomenon much more apparent.

We vary the number of connected components, controlled by parameter g, in Figure 5. We observe that when g is low (in this case $g = 1$), the loss function curve does not smoothly decrease but frequently levels off before stepping downward. This is due to the algorithm intermittently finding a more suitable support set, which allows it to more rapidly decrease the objective loss. The restriction of the algorithm to only one connected component makes this phenomenon much more apparent.

B.2 Breast cancer dataset

All parameters are tuned using 5-fold cross validation, following the experimental settings of [30]. The sparsity s is tuned from $[10, 20, \ldots 100]$ and the block size is tuned from $[n, n/2]$, where n is the number of samples. The task is to find a single connected component, so g is set to 1 for the head and tail projections. The learning rate is tuned using backtracking line search. We record the Balanced Classification Error and Area Under Curve (AUC) scores for each method over the 20 trials.

Table 2: Balanced Classification Error ± Std. Dev. on Breast Cancer Dataset

	IHT	StoIHT	GraphSto-IHT	GraphSVRG-IHT	GraphSCSG-IHT
Trial 1	0.352±0.079	0.355±0.083	0.367±0.091	**0.335**±0.048	0.474±0.048
Trial 2	0.350±0.047	0.346±0.045	0.395±0.074	**0.340**±0.077	0.496±0.015
Trial 3	0.354±0.060	0.370±0.061	0.365±0.107	**0.318**±0.061	0.478±0.061
Trial 4	0.351±0.046	**0.350**±0.066	0.365±0.032	0.365±0.032	0.459±0.050
Trial 5	**0.357**±0.064	0.365±0.063	0.366±0.065	0.368±0.082	0.418±0.076
Trial 6	0.334±0.050	**0.327**±0.046	0.330±0.078	0.352±0.090	0.447±0.071
Trial 7	0.343±0.050	0.402±0.085	**0.321**±0.027	**0.321**±0.027	0.499±0.004
Trial 8	0.352±0.059	0.402±0.103	**0.338**±0.021	0.349±0.030	0.489±0.013
Trial 9	0.340±0.082	0.357±0.058	**0.335**±0.110	0.343±0.040	0.452±0.053
Trial 10	0.344±0.068	0.351±0.065	0.349±0.050	**0.331**±0.061	0.502±0.005
Trial 11	0.339±0.056	**0.318**±0.061	0.353±0.076	0.357±0.054	0.473±0.054
Trial 12	0.355±0.061	0.394±0.053	0.349±0.088	**0.329**±0.082	0.463±0.072
Trial 13	0.327±0.053	0.317±0.040	**0.312**±0.065	0.323±0.060	0.457±0.051
Trial 14	0.362±0.058	0.340±0.071	0.348±0.019	**0.330**±0.054	0.453±0.045
Trial 15	0.381±0.048	0.346±0.051	**0.297**±0.059	**0.297**±0.059	0.416±0.069
Trial 16	0.330±0.094	**0.320**±0.089	0.330±0.074	0.339±0.064	0.488±0.055
Trial 17	0.349±0.074	0.332±0.067	0.353±0.078	**0.330**±0.056	0.437±0.055
Trial 18	0.331±0.096	0.331±0.096	0.292±0.067	**0.290**±0.069	0.462±0.053
Trial 19	0.327±0.052	0.333±0.054	0.311±0.021	**0.306**±0.023	0.476±0.017
Trial 20	**0.333**±0.038	0.338±0.026	0.352±0.064	0.376±0.036	0.427±0.052
Averaged	0.345±0.065	0.350±0.072	0.341±0.073	**0.335**±0.062	0.463±0.057

Table 3: AUC score ± Std. Dev. on Breast Cancer Dataset

	IHT	StoIHT	GraphSto-IHT	GraphSVRG-IHT	GraphSCSG-IHT
Trial 1	0.726±0.067	**0.736**±0.054	0.720±0.030	0.729±0.039	0.713±0.023
Trial 2	0.683±0.067	0.692±0.064	0.697±0.044	**0.712**±0.062	0.696±0.101
Trial 3	0.716±0.062	**0.726**±0.077	**0.726**±0.064	0.715±0.065	0.703±0.027
Trial 4	0.695±0.059	**0.701**±0.061	0.687±0.041	0.687±0.041	0.693±0.033
Trial 5	0.699±0.042	0.700±0.067	**0.707**±0.040	0.701±0.028	0.701±0.045
Trial 6	0.677±0.071	0.673±0.074	**0.704**±0.051	0.669±0.071	0.640±0.101
Trial 7	0.712±0.069	0.719±0.066	**0.741**±0.055	**0.741**±0.055	0.701±0.069
Trial 8	0.711±0.081	0.707±0.055	**0.721**±0.062	0.717±0.055	0.657±0.075
Trial 9	0.717±0.066	0.718±0.068	**0.723**±0.044	0.718±0.026	0.719±0.058
Trial 10	0.710±0.060	**0.711**±0.060	0.691±0.052	0.702±0.062	0.655±0.103
Trial 11	0.708±0.082	0.719±0.086	**0.722**±0.085	0.711±0.074	0.693±0.142
Trial 12	0.713±0.025	0.711±0.045	**0.736**±0.055	0.730±0.060	0.664±0.057
Trial 13	0.714±0.058	**0.719**±0.058	0.718±0.073	0.703±0.071	0.704±0.074
Trial 14	0.700±0.058	0.705±0.067	0.711±0.064	**0.712**±0.046	0.665±0.061
Trial 15	0.696±0.033	0.687±0.040	**0.721**±0.058	**0.721**±0.058	0.715±0.040
Trial 16	**0.729**±0.081	0.725±0.083	0.723±0.072	0.725±0.073	0.734±0.049
Trial 17	0.720±0.055	**0.722**±0.062	0.718±0.037	0.717±0.037	0.681±0.061
Trial 18	0.712±0.039	0.712±0.039	**0.733**±0.036	0.730±0.034	0.679±0.056
Trial 19	0.721±0.067	0.724±0.068	0.736±0.056	**0.738**±0.054	0.701±0.057
Trial 20	**0.720**±0.031	0.706±0.027	0.717±0.045	0.705±0.033	0.703±0.056
Averaged	0.709±0.062	0.711±0.064	**0.717**±0.057	0.714±0.057	0.691±0.074