mathematics

MDPI

Article

A Semismooth Newton-Based Augmented Lagrangian Algorithm for the Generalized Convex Nearly Isotonic Regression Problem

Yanmei Xu [1], Lanyu Lin [1,*] and Yong-Jin Liu [2]

[1] School of Mathematics and Statistics, Fuzhou University, Fuzhou 350108, China; 220320053@fzu.edu.cn
[2] Center for Applied Mathematics of Fujian Province, School of Mathematics and Statistics, Fuzhou University, Fuzhou 350108, China; yjliu@fzu.edu.cn
* Correspondence: lylin@fzu.edu.cn

Abstract: The generalized convex nearly isotonic regression problem addresses a least squares regression model that incorporates both sparsity and monotonicity constraints on the regression coefficients. In this paper, we introduce an efficient semismooth Newton-based augmented Lagrangian (SSNAL) algorithm to solve this problem. We demonstrate that, under reasonable assumptions, the SSNAL algorithm achieves global convergence and exhibits a linear convergence rate. Computationally, we derive the generalized Jacobian matrix associated with the proximal mapping of the generalized convex nearly isotonic regression regularizer and leverage the second-order sparsity when applying the semismooth Newton method to the subproblems in the SSNAL algorithm. Numerical experiments conducted on both synthetic and real datasets clearly demonstrate that our algorithm significantly outperforms first-order methods in terms of efficiency and robustness.

Keywords: generalized convex nearly isotonic regression; augmented Lagrangian algorithm; semismooth Newton method

MSC: 90C06; 90C25; 90C90

Academic Editor: Hsien-Chung Wu

Received: 9 January 2025
Revised: 28 January 2025
Accepted: 29 January 2025
Published: 2 February 2025

Citation: Xu, Y.; Lin, L.; Liu, Y.-J. A Semismooth Newton-Based Augmented Lagrangian Algorithm for the Generalized Convex Nearly Isotonic Regression Problem. *Mathematics* **2025**, *13*, 501. https://doi.org/10.3390/math13030501

1. Introduction

Data generated in various fields often exhibit clear monotonicity, as seen in meteorological climate data [1,2], economic demand/supply curves [3], and biological growth curves [4]. Thus, this paper focuses on statistical models under order constraints. Specifically, suppose that we have m observations $(\boldsymbol{a_i}, b_i)$ for $i = 1, \ldots, m$, where $\boldsymbol{a_i} = (a_{i1}, \ldots, a_{in})$ is a vector with n features and b_i is a response value. We concentrate on addressing the following optimization problem:

$$\min_{z \in \mathbb{R}^n} \left\{ \frac{1}{2}\|Az - b\|^2 + \lambda\|z\|_1 + \tau \sum_{i=1}^{n-1} (z_i - z_{i+1})_+ \right\}, \tag{1}$$

where $A = (\boldsymbol{a_1}, \ldots, \boldsymbol{a_m})$ is an $m \times n$ data matrix, $b = (b_1, \ldots, b_m)^\top$ is the response vector, λ, $\tau \geq 0$ are given regularization parameters, and $(z)_+ = \max(z, 0)$. In high-dimensional statistical regression, it is common for the number of features to exceed the number of samples. Therefore, in our paper, we assume that $m \leq n$. The penalty term is composed of two components: the first enforces sparsity in the coefficient estimates by incorporating prior knowledge, and the second penalizes violations of monotonicity among adjacent pairs.

Problem (1) is a generalization of a wide range of ordered convex problems, including an isotonic regression model [5], nearly isotonic regression model [6], and ordered lasso

problem [7]. The isotonic regression problem involves determining a vector $z \in \mathbb{R}^n$ that approximates a given vector $y \in \mathbb{R}^n$ while ensuring that z exhibits a non-decreasing (or non-increasing) sequence, i.e.,

$$\min_{z\in\mathbb{R}^n} \left\{ \sum_{i=1}^{n} (y_i - z_i)^2 \mid z_1 \le z_2 \le \ldots \le z_n \right\}, \tag{2}$$

where $z = (z_1, z_2, \ldots, z_n)^\top$ and $y = (y_1, y_2, \ldots, y_n)^\top$. Since the restricted monotonicity constraint may lead to a model too rigid and make it difficult to adapt to complex data structures, Tibshirani et al. [6] relaxed this monotonicity constraint and considered the following nearly isotonic regression model:

$$\min_{z\in\mathbb{R}^n} \left\{ \frac{1}{2} \sum_{i=1}^{n} (y_i - z_i)^2 + \tau \sum_{i=1}^{n-1} (z_i - z_{i+1})_+ \right\}, \tag{3}$$

where $\tau \ge 0$ is a given parameter. It is evident that problem (1) is a generalization of problem (3), as it addresses general regression issues and incorporates sparsity constraints on the coefficients. We refer to problem (1) as the generalized convex nearly isotonic regression (**GCNIR**) problem.

In addition, problem (1) can also be regarded as a generalization of the following ordered lasso problem [7]:

$$\begin{aligned} \min_{z\in\mathbb{R}^n} \quad & \frac{1}{2}\|Az - b\|^2 + \lambda\|z\|_1 \\ \text{s.t.} \quad & |z_1| \ge |z_2| \ge \cdots \ge |z_n|. \end{aligned} \tag{4}$$

Clearly, problem (4) extends the lasso problem by incorporating a monotonicity constraint on the absolute values of the coefficients. Like problem (2), this approach can lead to an overly rigid model (4). However, the **GCNIR** problem (1) mitigates the stringent monotonicity requirement of the ordered lasso, transforming it into a convex problem that is more flexible and tractable.

The **GCNIR** problem (1) can be reformulated into a convex quadratic programming (QP) problem by introducing new variables:

$$\begin{aligned} \min_{s\in\mathbb{R}^{4n}} \quad & \frac{1}{2} s^\top H s + \langle v, s \rangle \\ \text{s.t.} \quad & Ws = 0, \\ & s \ge 0, \end{aligned} \tag{5}$$

where

$$v = \begin{bmatrix} \lambda \mathbf{1}_n - A^\top b \\ \lambda \mathbf{1}_n + A^\top b \\ \tau \mathbf{1}_n \\ 0_{n\times 1} \end{bmatrix}, \quad H = \begin{bmatrix} A^\top A & -A^\top A & 0 & 0 \\ -A^\top A & A^\top A & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}_{4n\times 4n},$$

$$W = \begin{bmatrix} S, -S, -I_n, I_n \end{bmatrix}, \quad S = \begin{bmatrix} 1 & -1 & 0 & \cdots & 0 \\ 0 & 1 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & -1 \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix}_{n\times n},$$

$\mathbf{1}_n$ and I_n denote the all-ones column vector and $n \times n$ identity matrix, respectively. This implies that one can utilize the QP function "quadprog" provided by MATLAB or well-developed QP solvers, such as Gurobi and CPLEX [8], to compute reformulation (5) and thus solve problem (1). However, since $A^\top A$ has a size of $n \times n$, the computational cost of solving and storing $A^\top A$ becomes prohibitive, making it challenging to apply the aforementioned methods to large-scale problems.

Due to the challenges in solving the QP reformulation (5), it is logical to adapt the methods used for the previously discussed problems to address problem (1). The pool adjacent violators algorithm (PAVA) [9] is a cornerstone method for tackling shape-constrained statistical regression problems, as discussed in [10]. Initially developed for the isotonic regression model (2), PAVA has been extended to accommodate the nearly isotonic regression model (3), with adaptations such as the modified PAVA (MPAVA) [6] and the generalized PAVA (GPAVA) [2]. Despite its broad application, there is no theoretical guarantee that PAVA can be modified to tackle convex nonseparable minimization problems. Additionally, other approaches, such as the Generalized Proximal Gradient algorithm [7] and the alternating direction method of multipliers (ADMM) [11], have been proposed for solving the ordered lasso problem (4). To our knowledge, most current techniques for dealing with ordered models rely primarily on first-order information from the associated nonsmooth optimization framework. Consequently, we aim to develop a customized algorithm that utilizes second-order information to address the **GCNIR** problem more effectively.

This paper aims to develop a semismooth Newton-based augmented Lagrangian (SSNAL) algorithm to address the **GCNIR** problem (1) from a dual viewpoint. The SSNAL algorithm's primary benefits include its superior convergence characteristics and reduced computational demands, which are achieved by exploiting second-order sparsity and employing efficient strategies within the semismooth Newton (SSN) algorithm. Furthermore, the SSNAL algorithm has demonstrated its effectiveness in handling large-scale sparse convex models, as evidenced by its performance in applications such as Lasso [12], group Lasso [13], fused Lasso [14], clustered Lasso [15,16], multi-task Lasso [17], ℓ_1 trend filtering [18], density matrix least squares problems [19], the Dantzig selector [20], and others [21–24]. Building on these successes, we propose to apply the SSNAL algorithm to solve problem (1).

The primary contributions of this paper are as follows. First, we calculate the proximal mapping related to the **GCNIR** regularizer and its generalized Jacobian. Second, we utilize the SSNAL algorithm to address the **GCNIR** problem from a dual perspective. Furthermore, by capitalizing on the low-rank properties and second-order sparsity inherent in the **GCNIR** problem, we significantly reduce the computational cost associated with the SSN algorithm when solving the subproblems. Lastly, we perform a numerical analysis comparing our algorithm with first-order methods, including ADMM and the Accelerated Proximal Gradient (APG) method, demonstrating the efficiency and robustness of our approach.

The remaining sections of this paper are organized as follows. Section 2 delves into the analysis of the proximal mapping associated with the **GCNIR** regularizer and its generalized Jacobian. Section 3 outlines the framework of the SSNAL algorithm and discusses its convergence properties when applied to the dual formulation of the **GCNIR** problem (1). In Section 4, we evaluate the performance of the SSNAL algorithm through numerical experiments. Finally, we conclude the paper in Section 5.

Notation. For any $z \in \mathbb{R}^n$, "Diag(z)" represents a diagonal matrix with z_i in its i-th diagonal component. "$|z|$" refers to an absolute vector, where each entry i is $|z_i|$. "sign(z)" indicates the sign vector, i.e., sign$(z_i) = 1$ when $z_i > 0$, sign$(z_i) = -1$ when $z_i < 0$, and sign$(z_i) = 0$ when z_i is equal to zero. Additionally, the notation "Supp(z)" refers to

the support of the element z, specifically the collection of indices for which z_i is not equal to zero. For any positive integer n, $e_1 = (1, 0, 0, \ldots, 0)^\top$ and $e_n = (0, 0, \ldots, 0, 1)^\top$ are the $n \times 1$ unit column vectors. $I_n = (e_1, e_2, \ldots, e_n) \in \mathbb{R}^{n\times n}$, while $\mathbf{1}_n = (1, 1, \ldots, 1) \in \mathbb{R}^n$. $D^\dagger$ denotes the Moore–Penrose pseudoinverse of the matrix $D \in \mathbb{R}^{m\times n}$. Typically, h^* denotes the Fenchel conjugate of a given function h.

2. The Proximal Mapping of the GCNIR Regularizer and Its Generalized Jacobian

In this section, we shall present some results concerning the proximal mapping linked to the **GCNIR** regularizer along with its generalized Jacobian, which are necessary for later analysis.

Given any scalar $\kappa > 0$, for any proper closed convex function $p : \mathbb{R}^n \to (-\infty, \infty]$, the proximal mapping and Moreau envelope [25] of p is defined by

$$\mathrm{Prox}_p(w) := \underset{z}{\mathrm{argmin}}\left\{\frac{1}{2}\|z - w\|^2 + p(z)\right\},\ \forall w \in \mathbb{R}^n,$$
$$M_p^\kappa(w) := \min_z\left\{\frac{1}{2\kappa}\|z - w\|^2 + p(z)\right\},\ \forall w \in \mathbb{R}^n.$$

The Moreau identity [26] holds, i.e.,

$$\mathrm{Prox}_{\kappa p}(w) + \kappa \mathrm{Prox}_{p^*/\kappa}(w/\kappa) = w,\ \forall w \in \mathbb{R}^n.$$

According to [27], $M_p^\kappa(\cdot)$ is convex and continuously differentiable, and

$$\nabla M_p^\kappa(w) = (w - \mathrm{Prox}_{\kappa p}(w))/\kappa,\ \forall w \in \mathbb{R}^n.$$

Let φ be the **GCNIR** regularizer in (1), i.e.,

$$\varphi(z) = \lambda\|z\|_1 + g(z),\ \forall z \in \mathbb{R}^n,$$

where $g(z) = \tau\sum_{i=1}^{n-1}(z_i - z_{i+1})_+,\ \forall z \in \mathbb{R}^n$.

Before diving into the proximal mapping associated with the **GCNIR** regularizer φ, we briefly introduce $\sum_{i=1}^{n-1}|z_i - z_{i+1}|$ and relevant results, which are discussed in [14].

Define $q(z) = \sum_{i=1}^{n-1}|z_i - z_{i+1}| = \|Bz\|_1$, where B is defined by

$$B = \begin{pmatrix} 1 & -1 & 0 & \cdots & 0 \\ 0 & 1 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & -1 \end{pmatrix}_{(n-1)\times n}.$$

The proximal mapping $Z_\tau(\cdot)$ with respect to τq is given as

$$Z_\tau(w) := \underset{z}{\mathrm{argmin}}\left\{\frac{1}{2}\|z - w\|^2 + \tau q(z)\right\},\ \forall w \in \mathbb{R}^n.$$

Lemma 1. (See [14], Lemma 1). *For any given $\tau \geq 0$, we have that $Z_\tau(w) = w - B^\top S_\tau(Bw)$, where*

$$S_\tau(w) := \underset{s}{\mathrm{argmin}}\left\{\frac{1}{2}\|B^\top s\|^2 - \langle s, w\rangle \mid \|s\|_\infty \leq \tau\right\},\ \forall w \in \mathbb{R}^{n-1}. \tag{6}$$

On the basis of the above lemma, we can now explicitly calculate $\mathrm{Prox}_p(\cdot)$. For later convenience, we define $w_\tau = w - \tau(e_1 - e_n)/2$.

Proposition 1. *For any given λ, $\tau \geq 0$, it follows that for any $w \in \mathbb{R}^n$,*

$$\mathrm{Prox}_{\varphi}(w) = \mathrm{Prox}_{\lambda\|\cdot\|_1}(Z_{\tau/2}(w_\tau)) = \mathrm{sign}(Z_{\tau/2}(w_\tau)) \circ \max(|Z_{\tau/2}(w_\tau)| - \lambda, 0).$$

Proof. According to the definition of the proximal mapping, it holds that for any $w \in \mathbb{R}^n$,

$$\begin{aligned}
\mathrm{Prox}_{\varphi}(w) &= \underset{z}{\mathrm{argmin}}\left\{\frac{1}{2}\|z-w\|^2 + \varphi(z)\right\} \\
&= \underset{z}{\mathrm{argmin}}\left\{\frac{1}{2}\|z-w\|^2 + \lambda\|z\|_1 + \tau\sum_{i=1}^{n-1}(z_i - z_{i+1})_+\right\} \\
&= \underset{z}{\mathrm{argmin}}\left\{\frac{1}{2}\|z-w\|^2 + \lambda\|z\|_1 + \frac{\tau}{2}\sum_{i=1}^{n-1}|z_i - z_{i+1}| + \frac{\tau}{2}\langle z, e_1 - e_n\rangle\right\} \\
&= \underset{z}{\mathrm{argmin}}\left\{\frac{1}{2}\left\|z - \left(w - \frac{\tau}{2}(e_1 - e_n)\right)\right\|^2 + \lambda\|z\|_1 + \frac{\tau}{2}\sum_{i=1}^{n-1}|z_i - z_{i+1}|\right\} \\
&= \mathrm{Prox}_{\lambda\|\cdot\|_1 + \frac{\tau}{2}q}\left(w - \frac{\tau}{2}(e_1 - e_n)\right).
\end{aligned}$$

It follows from ([28], Corollary 4) that for any $w \in \mathbb{R}^n$,

$$\begin{aligned}
\mathrm{Prox}_{\lambda\|\cdot\|_1 + \frac{\tau}{2}q}\left(w - \frac{\tau}{2}(e_1 - e_n)\right) &= \mathrm{Prox}_{\lambda\|\cdot\|_1}(Z_{\tau/2}(w_\tau)) \\
&= \mathrm{sign}(Z_{\tau/2}(w_\tau)) \circ \max(|Z_{\tau/2}(w_\tau)| - \lambda, 0).
\end{aligned}$$

This completes the proof. □

Next, we analyze the generalized Jacobian of $\mathrm{Prox}_{\varphi}(\cdot)$, which is crucial for leveraging computational efficiency. We begin with presenting some findings concerning the generalized HS-Jacobian for $S_\tau(\cdot)$, according to [14,29].

As noted in [14], the generalized HS-Jacobian for $S_{\tau/2}$ at Bw_τ is given by

$$\mathcal{Q}(w_\tau) = \left\{Q \in \mathbb{R}^{(n-1)\times(n-1)} \mid Q = (\Sigma_\gamma B B^\top \Sigma_\gamma)^\dagger, \gamma \in \Gamma(w_\tau)\right\},$$

where $\Sigma_\gamma = \mathrm{Diag}(\sigma_\gamma) \in \mathbb{R}^{(n-1)\times(n-1)}$ with

$$(\sigma_\gamma)_i = \begin{cases} 0 & \text{if } i \in \Gamma, \\ 1 & \text{otherwise}, \end{cases} \quad i = 1, 2, \ldots, n-1,$$

and $\Gamma(w_\tau) = \{\gamma \subseteq \{1, 2, \ldots, n-1\} \mid \mathrm{Supp}(\xi) \subseteq \gamma \subseteq \mathcal{I}(w_\tau)\}$, where $\xi = B(Z_{\tau/2}(w_\tau))$ is an optimal Lagrangian multiplier for the constraint $\|s\|_\infty \leq \tau/2$ and

$$\mathcal{I}(w_\tau) = \left\{i \subseteq \{1, 2, \ldots, n-1\} \;\middle|\; |S_{\tau/2}(B(w_\tau))|_i = \tau/2\right\} \tag{7}$$

is an active index set.

The multifunction $\mathcal{H} : \mathbb{R}^n \rightrightarrows \mathbb{R}^{n\times n}$ is given by

$$\mathcal{H}(w_\tau) = \left\{H \in \mathbb{R}^{n\times n} \mid H = I_n - B^\top Q B, Q \in \mathcal{Q}(w_\tau)\right\}.$$

The subsequent proposition demonstrates that $\mathcal{Q}(w_\tau)$ and $\mathcal{H}(w_\tau)$ may be regarded as the generalized HS-Jacobian for $S_{\tau/2}$ at Bw_τ, $Z_{\tau/2}$ at w_τ, respectively.

Proposition 2. *For all $w \in \mathbb{R}^n$, there exists a neighborhood $\mathcal{X}$ of w such that for any $x \in \mathcal{X}$,*

$$\begin{cases} \Gamma(x_\tau) \subseteq \Gamma(w_\tau),\ \mathcal{Q}(x_\tau) \subseteq \mathcal{Q}(w_\tau),\ \mathcal{H}(x_\tau) \subseteq \mathcal{H}(w_\tau), \\ S_{\tau/2}(Bx_\tau) = S_{\tau/2}(Bw_\tau) + QB(x-w),\ \forall Q \in \mathcal{Q}(x_\tau), \\ Z_{\tau/2}(x_\tau) = Z_{\tau/2}(w_\tau) + H(x-w),\ \forall H \in \mathcal{H}(x_\tau), \end{cases}$$

where $x_\tau = x - \tau(e_1 - e_n)/2$.

Proof. The results are derived from [14] (Proposition 2) and [29] (Lemma 2.1) with minor revisions. □

The multifunction $\mathcal{V} : \mathbb{R}^n \rightrightarrows \mathbb{R}^{n\times n}$ is defined as

$$\mathcal{V}(w) := \left\{ V \in \mathbb{R}^{n\times n} \;\middle|\; V = \Theta H, \quad \begin{array}{l} \Theta \in \partial_B \mathrm{Prox}_{\lambda\|\cdot\|_1}(Z_{\tau/2}(w_\tau)), \\ H \in \mathcal{H}(w_\tau) \end{array} \right\}, \tag{8}$$

where

$$\partial_B \mathrm{Prox}_{\lambda\|\cdot\|_1}(\eta) = \left\{ \mathrm{Diag}(\omega) \;\middle|\; \begin{array}{ll} \omega_i = 0 & |\eta_i| < \lambda, \\ \omega_i \in \{0,1\} & |\eta_i| = \lambda, \\ \omega_i = 1 & |\eta_i| > \lambda, \end{array} \right\}.$$

The mapping $\mathcal{V}(w)$ essentially acts as the generalized Jacobian for Prox_φ at w, which can be derived using the change-of-variables technique from previous work in [14] (Theorem 2).

Theorem 1. *Let λ and τ be non-negative real numbers, and let w be an element of $\mathbb{R}^n$. The set-valued function $\mathcal{V}$ has the following properties: it is compact-valued, nonempty, and upper semi-continuous. For each $V \in \mathcal{V}(w)$, it can be concluded that the matrix V is symmetric and positive semidefinite. Furthermore, there exists a neighborhood $\mathcal{X}$ of w that for any $x \in \mathcal{X}$,*

$$\mathrm{Prox}_\varphi(x) - \mathrm{Prox}_\varphi(w) - V(x-w) = 0,\ \ \forall V \in \mathcal{V}(x).$$

3. A Semismooth Newton-Based Augmented Lagrangian Algorithm

This section introduces the semismooth Newton-based augmented Lagrangian (SsNAL) algorithm, an efficient approach for solving the **GCNIR** problem in the high-dimension low-sample setting, i.e., the case of $m \ll n$. Directly applying the SsNAL algorithm to the primal problem would require solving a linear system of $n \times n$ dimension, leading to significant computational costs, particularly for large-scale problems. To overcome this, we solve the **GCNIR** problem from the dual perspective.

We can reformulate the **GCNIR** problem (1) as

$$\min_{z\in\mathbb{R}^n,\ v\in\mathbb{R}^m} \left\{ \frac{1}{2}\|v\|^2 + \varphi(z) \;\middle|\; Az - v = b \right\}. \tag{P}$$

The dual problem of (P) takes the following minimization form:

$$\min_{s\in\mathbb{R}^m,\ \zeta\in\mathbb{R}^n} \left\{ \frac{1}{2}\|s\|^2 + \langle b, s\rangle + \varphi^*(\zeta) \;\middle|\; A^\top s + \zeta = 0 \right\}. \tag{D}$$

The Lagrangian function of (D) is

$$l(s,\zeta;z) = \frac{1}{2}\|s\|^2 + \langle b, s\rangle + \varphi^*(\zeta) - \langle z, A^\top s + \zeta\rangle.$$

Additionally, given $\sigma > 0$, the augmented Lagrangian function of (D) takes the form:

$$\mathcal{L}_\sigma(s, \zeta; z) = l(s, \zeta; z) + \frac{\sigma}{2}\|A^\top s + \zeta\|^2.$$

3.1. The Framework of the SSNA Algorithm

Below is the outline of the framework for an SSNAL algorithm designed to solve problem (D) (Algorithm 1).

Algorithm 1 (SSNAL) A semismooth Newton-based augmented Lagrangian algorithm for (D)

Initialization: $\sigma_0 > 0$, $s^0 \in \mathbb{R}^m$, $\zeta^0 \in \mathbb{R}^n$, $z^0 \in \mathbb{R}^n$, $j = 0$.

1: Compute
$$s^{j+1} \approx \underset{s \in \mathbb{R}^m}{\operatorname{argmin}} \{\Phi_j(s)\}. \tag{9}$$
2: Compute $\zeta^{j+1} = \left(z^j - \sigma_j A^\top s^{j+1} - \mathrm{Prox}_{\sigma_j \varphi}\left(z^j - \sigma_j A^\top s^{j+1}\right)\right)/\sigma_j$.
3: Update $z^{j+1} = z^j - \sigma_j(A^\top s^{j+1} + \zeta^{j+1})$.
4: Update $\sigma_{j+1} \uparrow \sigma_\infty \leq \infty$, set $j \leftarrow j + 1$, and go to Step 1.

The stopping criteria, which have been studied in [30,31] for approximately solving the solution to (9), can be stated as:

$$\|\nabla \Phi_j(s^{j+1})\| \leq \epsilon_j / \sqrt{\sigma_j}, \ \sum_{j=0}^{\infty} \epsilon_j < \infty, \ \epsilon_j > 0, \tag{C1}$$

$$\|\nabla \Phi_j(s^{j+1})\| \leq \rho_j \sqrt{\sigma_j} \|A^\top s^{j+1} + \zeta^{j+1}\|, \ \sum_{j=0}^{\infty} \rho_j < \infty, \ \rho_j > 0, \tag{C2}$$

$$\|\nabla \Phi_j(s^{j+1})\| \leq \rho_j' \|A^\top s^{j+1} + \zeta^{j+1}\|, \ 0 \leq \rho_j' \to 0. \tag{C3}$$

Next, we present the convergence results of the SSNAL algorithm, addressing both global and local convergence. Since problem (P) is feasible, ref. [30] (Theorem 4) establishes that satisfying the stopping criterion (C1) guarantees the global convergence of the SSNAL algorithm for problem (D).

Theorem 2. *(Global convergence.) The infinite sequence $(s^j, \zeta^j; z^j)$ generated by the Algorithm 1, in accordance with the stopping criteria specified in (C1), ensures that the sequence z^j converges to an optimal solution of problem (P) and the sequence (s^j, ζ^j) converges to the unique optimal solution of problem (D).*

Let the objective function of problem (P) be denoted by $f(z) := \frac{1}{2}\|Az - b\|^2 + \varphi(z)$. It is clear that $T_f(z) := \partial f(z)$ and $T_l(s, \zeta; z) := \{(s', \zeta'; z') \mid (s', \zeta'; -z') \in \partial l(s, \zeta; z)\}$ are piecewise polyhedral multifunctions as described in [32]. This implies that both $\mathcal{T}_f$ and $\mathcal{T}_l$ satisfy error bound conditions at the original point with positive moduli c_f and c_l, respectively, as characterized in [33].

Following the analysis presented, we establish the local convergence of Algorithm 1, supported by the results in [12,30,31,34]. The proofs are analogous to those in [12] (Theorems 3.3). Therefore, we omit the detailed proofs here.

Theorem 3. *(Local convergence.) Given the conditions specified in (C1) and (C2), the sequence (s^j, ζ^j, z^j) generated by the Algorithm 1 converges to (s^*, ζ^*, z^*), and for sufficiently large values of j, the following holds:*

$$\|z^{j+1} - z^*\| \leq \delta_j \|z^j - z^*\|,$$

where $\delta_j = (2\rho_j + c_f/\sqrt{c_f^2+\sigma_j^2})/(1-\rho_j)$ *and* $\lim_{j\to\infty}\delta_j = c_f/\sqrt{c_f^2+\sigma_\infty^2} < 1$. *Furthermore, if the condition (C3) is also applied, then for sufficiently large values of* j,

$$\|(s^{j+1},\zeta^{j+1}) - (s^*,\zeta^*)\| \le \delta_j'\|z^{j+1}-z^j\|,$$

where $\delta_j' = c_l(1+\rho_j')/\sigma_j \to c_l/\sigma_\infty$ *when* $j\to\infty$.

3.2. SSN Algorithm for Subproblem (9)

We shall present a highly effective semismooth Newton (SSN) algorithm designed to tackle problem (9) in this subsection.

For any given $\sigma > 0$ and a fixed $\tilde{z} \in \mathbb{R}^n$, our goal is to solve the following problem:

$$\min_{s\in\mathbb{R}^m}\left\{\Phi(s) := \inf_{\zeta}\ \mathcal{L}_\sigma(s,\zeta;\tilde{z})\right\}, \tag{11}$$

where the objective function of problem (11) is given by

$$\Phi(s) = M_{\varphi^*}^{1/\sigma}\left(\frac{\tilde{z}}{\sigma} - A^\top s\right) - \frac{1}{2\sigma}\|\tilde{z}\|^2 + \langle s,b\rangle + \frac{1}{2}\|s\|^2.$$

It is well known that $\Phi(\cdot)$ is continuously differentiable with

$$\nabla\Phi(s) = -A\mathrm{Prox}_{\sigma\varphi}(\tilde{z}-\sigma A^\top s) + b + s = -\sigma A\mathrm{Prox}_{\varphi}\left(\frac{\tilde{z}}{\sigma} - A^\top s\right) + b + s.$$

Since $\Phi(\cdot)$ is strongly convex, we can obtain the unique solution of problem (11) via solving the following nonsmooth equations:

$$\nabla\Phi(s) = 0. \tag{12}$$

For any $s\in\mathbb{R}^m$, the multifunction is well defined and can be expressed as

$$\mathcal{N}(s) := \left\{N\in\mathbb{R}^{m\times m}\ \middle|\ N = I_m + \sigma AVA^\top, V\in\mathcal{V}\left(\frac{\tilde{z}}{\sigma} - A^\top s\right)\right\},$$

where the operator $\mathcal{V}$ is defined in (8).

Remark 1. *Based on Theorem 1, we can deduce that* $\mathcal{V}$ *is not only nonempty but also has the properties of being compact-valued and upper semicontinuous. In addition, each component of the function* $\mathcal{N}(\cdot)$ *is symmetric and positive definite.*

Before delving into the SSN algorithm, we introduce the following lemma, the proof of which is analogous to the one presented in [15] (Remark 2.12).

Lemma 2. *For any positive constant* r, *the proximal mapping* $\mathrm{Prox}_\varphi(\cdot)$ *is* r*-order semismooth at* $w\in\mathbb{R}^n$ *with respect to the multifunction* $\mathcal{V}$. *Similarly, the gradient* $\nabla\Phi$ *is* r*-order semismooth at* $s\in\mathbb{R}^m$ *with respect to the multifunction* $\mathcal{N}$.

Proof. Given that $S_{\tau/2}(\cdot)$ and $\mathrm{Prox}_{\lambda\|\cdot\|_1}(\cdot)$ are piecewise affine functions as shown in [14], it follows that $Z_{\tau/2}(\cdot)$ is Lipschitz continuous. Consequently, the proximal mapping $\mathrm{Prox}_\varphi(\cdot)$ is also a piecewise affine function that maintains Lipschitz continuity. As established in [35], $\mathrm{Prox}_\varphi(\cdot)$ is directionally differentiable at every point. Combining this with Theorem 1 and the definition of semismoothness [36–39], we can conclude that $\mathrm{Prox}_\varphi(\cdot)$ exhibits r-order

semismoothness on $\mathbb{R}^n$. Similarly, it can be inferred that $\nabla\Phi$ also demonstrates r-order semismoothness on $\mathbb{R}^m$. This completes the proof. □

Given that the gradient $\nabla\Phi(\cdot)$ is nonsmooth, it is appropriate to employ the semismooth Newton (SSN) algorithm instead of the standard Newton method to solve Equations (12). Building on the analysis provided, we are now ready to proceed with the development of an SSN algorithm to solve (12) (Algorithm 2).

Algorithm 2 (SSN) A semismooth Newton algorithm for (12)

Initialization: $s^0 \in \mathbb{R}^n$, $\alpha \in (0, 1/2)$, $\beta \in (0, 1)$. Set $t = 0$.

1: Select $N_t \in \mathcal{N}(s^t)$. Solve the following linear system

$$N_t d = -\nabla\Phi(s^t). \tag{13}$$

2: Set $\mu_t = \beta^{m_t}$, where m_t is the first nonnegative integer m such that

$$\Phi(s^t + \beta^m d^t) \leq \Phi(s^t) + \alpha\beta^m \langle \nabla\Phi(s^t), d^t \rangle.$$

3: Update $s^{t+1} = s^t + \mu_t d^t$, $t \leftarrow t + 1$, and go to step 1.

For problem (13), we can employ the conjugate gradient algorithm to obtain d^t such that

$$\|N_t d^t + \nabla\Phi(s^t)\| \leq \min\{\varrho, \|\nabla\Phi(s^t)\|^{1+\xi}\}, \tag{14}$$

where $\varrho \in (0, 1)$, $\xi \in (0, 1]$. Regarding the convergence of Algorithm 2, it is confirmed in the work of Li [14] (Theorem 3), and we now present their result directly as follows.

Theorem 4. *The infinite sequence s^t, generated by the Algorithm 2, converges to the unique optimal solution $\bar{s}$ to problem (11). Additionally, it follows that*

$$\|s^{t+1} - \bar{s}\| = O(\|s^t - \bar{s}\|^{1+\xi}),$$

where ξ is given in solving problem (14).

When applying Algorithm 2 to solve problem (12), the most computationally intensive step involves determining the search direction, which is derived from solving the linear system (13), i.e.,

$$(I_m + \sigma AVA^\top)d = -\nabla\Phi(s). \tag{15}$$

Therefore, we aim to investigate the second-order information of the matrix V to reduce computation time. Firstly, let $\Theta = \mathrm{Diag}(\theta)$ with

$$\theta_i = \begin{cases} 0 & \text{if } |Z_{\tau/2}(w_\tau)|_i \leq \lambda \\ 1 & \text{otherwise} \end{cases}, \ i = 1, 2, \ldots, n,$$

and denote $\Sigma = \mathrm{Diag}(\sigma) \in \mathbb{R}^{(n-1)\times(n-1)}$ with

$$\sigma_i = \begin{cases} 0 & \text{if } i \in \mathcal{I}(w_\tau) \\ 1 & \text{otherwise} \end{cases}, \ i = 1, 2, \ldots, n-1,$$

where $\mathcal{I}(w_\tau)$ is given as in (7). It is derived from [14] (Propositions 2 and 3) that $V_0 \in \mathcal{V}(w)$, i.e.,

$$V_0 = \Theta H = \Theta(I_n - B^\top(\Sigma BB^\top\Sigma)^\dagger B),$$

where $H \in \mathcal{H}(w_\tau)$. Additionally, we can restructure H as a block diagonal matrix and exploit the generalized Jacobian's sparse low-rank structure to substantially reduce computation time. Moreover, we have several options for solving the linear system (15), such as the Cholesky factorization, Sherman–Morrison–Woodbury (SMW) formula, or preconditioned conjugate gradient (PCG) method. These techniques further contribute to improving computational efficiency. Specific details can be found in [14,40], which we do not repeat here.

4. Numerical Experiments

In this section, we evaluate the efficiency of the SSNAL algorithm for solving the **GCNIR** problem by comparing it with the ADMM and APG algorithms using both synthetic and real datasets. The computational results were achieved by using MATLAB R2022b on a Dell desktop equipped with an Intel(R) Core(TM) i7-11700 CPU running at 2.50 GHz, along with 8 GB of RAM.

*4.1. Some First-Order Methods for the **GCNIR** Problem*

We provide a brief overview of the framework of ADMM and APG.

ADMM is recognized as a representative algorithm for addressing convex optimization problems [41], including the problem presented in (D). Here is a summary of the framework for the ADMM algorithm (see Algorithm 3):

Algorithm 3 ADMM for the dual problem (D)

Initialization: Set $\sigma > 0$, $c \in (0, (1+\sqrt{5})/2)$, $\zeta^0 \in \mathbb{R}^n$, $z^0 \in \mathbb{R}^n$, and initialize $j = 0$.

1: Address

$$s^{j+1} \approx \underset{s\in\mathbb{R}^m}{\operatorname{argmin}} \ \mathcal{L}_\sigma(s, \zeta^j; z^j). \tag{16}$$

by utilizing either direct solvers or iterative methods like preconditioned conjugate gradient (PCG) method for solving this linear system to obtain s^{j+1}:

$$(I_m + \sigma A A^\top)s = A(z^j - \sigma\zeta^j) - b.$$

2: Calculate $\zeta^{j+1} = \underset{\zeta\in\mathbb{R}^n}{\operatorname{argmin}} \ \mathcal{L}_\sigma(s^{j+1}, \zeta; z^j) = \operatorname{Prox}_{\varphi^*/\sigma}\left(\frac{z^j}{\sigma} - A^\top s^{j+1}\right)$.

3: Update $z^{j+1} = z^j - c\sigma(A^\top s^{j+1} + \zeta^{j+1})$.

4: Set $j \leftarrow j+1$, and go to Step 1.

Let $L = \lambda_{\max}(A^\top A)$, representing the Lipschitz constant for function $\|Az - b\|^2/2$ in the primal problem (P). Here is a summary of the framework for the APG algorithm (see Algorithm 4):

Algorithm 4 APG for the primal problem (P)

Initialization: Choose $\epsilon > 0$, set $x^0 = z^0 \in \mathbb{R}^n$, $r_0 = 1$, and initialize $j = 0$.

1: Calculate

$$z^{j+1} = \operatorname{Prox}_{\varphi/L}\left(x^j - \frac{A^\top(Ax^j - b))}{L}\right).$$

2: Compute $r_{j+1} = (1 + \sqrt{1 + 4r_j^2})/2$.

3: Update

$$x^{j+1} = z^{j+1} + \frac{r_j - 1}{r_{j+1}}(z^{j+1} - z^j).$$

4: Set $j \leftarrow j+1$, and go to Step 1.

4.2. Stopping Criteria

Utilizing the KKT conditions of problem (P) and (D), we can obtain the following relative KKT residual:

$$E_P := \frac{\|Az - b - s\|}{1 + \|b\|}, E_D := \frac{\|A^\top s + \zeta\|}{1 + \|\zeta\|}, E_K := \frac{\|z - \mathrm{Prox}_\varphi(z - A^\top(Az - b))\|}{1 + \|z\| + \|A^\top(Az - b)\|}.$$

In addition, let $\mathrm{obj_P}$ and $\mathrm{obj_D}$ represent the objective values of (P) and (D), respectively, i.e.,

$$\mathrm{obj_P} := \frac{1}{2}\|Az - b\|^2 + \lambda\|z\|_1 + \tau\sum_{i=1}^{n-1}(z_i - z_{i+1})_+, \ \mathrm{obj_D} := -\frac{1}{2}\|s\|^2 - \langle b, s\rangle,$$

Then, the relative dual gap is defined by

$$E_G := \frac{|\mathrm{obj_P} - \mathrm{obj_D}|}{1 + |\mathrm{obj_P}| + |\mathrm{obj_D}|}.$$

In later experiments, we start the SSNAL algorithm with the parameters $(s^0, \zeta^0, z^0) = (0, 0, 0)$ and terminate it when Res $:= \max\{E_P, E_D, E_K, E_G\} \leq$ tol with a given error tolerance "to". To enhance convergence speed, it is essential to dynamically modify the penalty parameter σ in the SSNAL algorithm. Specifically, we initially set $\sigma_0 = \min(1, 1/\sqrt{\lambda_{\max}(AA^\top)})$ and tune σ_{j+1} every three steps, i.e.,

$$\sigma_{j+1} = \begin{cases} \max\{0.001, 0.002\sigma_j\}, & E_D^j < 0.1E_P^j, \\ \min\{5000, 50\sigma_j\}, & E_D^j > 10E_P^j, \\ \sigma_j, & \text{otherwise}, \end{cases}$$

where σ_j, E_P^j, and E_D^j represent the values of σ, E_P, and E_D during the j-th iteration, respectively.

In the case of the ADMM algorithm, we set the initial point $(s^0, \zeta^0, z^0) = (0, 0, 0)$ and terminate the algorithm when Res $= E_K \leq$ tol. For the APG algorithm, we start the initial point $(z^0, x^0) = (0, 0)$ and terminate the algorithm when Res $= E_K \leq$ tol.

Additionally, we set a tolerance level of tol $= 10^{-6}$. The tested algorithms will terminate under two conditions: either upon reaching their maximum number of iterations (100 for SSNAL; 30,000 for ADMM and APG) or if their running time exceeds 3 h.

In the following tables, "nnz" and "mon" represent the counts of non-zero elements in z and Bz, as derived from SSNAL through these estimations:

$$\mathrm{nnz} := \min\left\{ j \mid \sum_{k=1}^{j} |z|_{(k)} \geq 0.999\|z\|_1 \right\},$$
$$\mathrm{mon} := \min\left\{ j \mid \sum_{k=1}^{j} |(Bz)_+|_{(k)} \geq 0.999\|(Bz)_+\|_1 \right\},$$

where $|z|_{(k)}$ represents the k-th largest component in $|z|$, ordered as $|z|_{(1)} \geq |z|_{(2)} \geq \cdots \geq |z|_{(n)}$. Time is shown in seconds.

4.3. Results on Synthetic Data

In this subsection, we evaluate the performance of three algorithms: SSNAL, ADMM, and APG on synthetic data.

In later experiments, we test five instances: $(m;n) = (300k;8000k)$, $k = 1,\ldots,5$. To create synthetic data, we employ the model

$$b = Az + \tilde{\xi}\epsilon,\ \epsilon \sim N(0,I),$$

where $A \in \mathbb{R}^{m\times n}$ is drawn from a Gaussian distribution $N(0,I_n)$ and $z \in \mathbb{R}^n$ is generated by a distributed random number. Following the way provided in [15], we set $\tilde{\xi}$ to be $0.1\|Az\|/\|\epsilon\|$ for each case. The regularization parameters for the **GCNIR** problem (1) are specified as follows:

$$\lambda = 0.5\varpi\|A^\top b\|_\infty, \tau = \lambda, \tag{P1}$$

where $0 < \varpi < 2$. In total, we test 20 instances.

Table 1 presents a comparative analysis of the performance of three algorithms on synthetic datasets ranging from small to large scales. It is evident that the SSNAL algorithm exhibits both efficiency and robustness across a variety of parameter settings. Although all algorithms are capable of solving the problem with the required level of accuracy, SSNAL consistently outperforms the other two methods in terms of computational time. For instance, in the case of instance 5 with $\varpi = 0.6$, SSNAL completes the task in a mere 3 s, whereas ADMM and APG take over 60 and 80 s, respectively.

Table 1. The performances of SSNAL, ADMM, and APG when applied to synthetic data.

i	$\lambda_{\max}(AA^\top)$	ϖ	nnz; mon	Time	Res
				SSNAL | ADMM | APG	SSNAL | ADMM | APG
1	1.13×10^4	0.6	93; 69	0.26 | 0.40 | 1.56	1.85×10^{-7} | 9.42×10^{-7} | 9.69×10^{-7}
		0.7	57; 45	0.19 | 0.32 | 1.29	7.75×10^{-8} | 9.21×10^{-7} | 9.99×10^{-7}
		0.8	25; 20	0.11 | 0.29 | 1.12	2.02×10^{-8} | 8.89×10^{-7} | 9.40×10^{-7}
		0.9	9; 8	0.07 | 0.29 | 0.66	5.60×10^{-7} | 8.38×10^{-7} | 9.41×10^{-7}
2	2.28×10^4	0.6	144; 100	0.53 | 4.25 | 13.38	3.13×10^{-8} | 9.97×10^{-7} | 9.90×10^{-7}
		0.7	73; 53	0.48 | 4.20 | 9.85	7.98×10^{-9} | 9.30×10^{-7} | 9.69×10^{-7}
		0.8	23; 16	0.29 | 3.85 | 7.11	3.87×10^{-7} | 9.36×10^{-7} | 9.57×10^{-7}
		0.9	8; 5	0.16 | 3.72 | 3.64	1.07×10^{-7} | 9.72×10^{-7} | 9.77×10^{-7}
3	3.43×10^4	0.6	251; 170	1.57 | 16.54 | 41.17	2.56×10^{-8} | 9.91×10^{-7} | 9.92×10^{-7}
		0.7	134; 93	1.02 | 16.24 | 30.42	6.80×10^{-7} | 9.82×10^{-7} | 9.66×10^{-7}
		0.8	54; 37	0.78 | 15.50 | 23.71	3.08×10^{-7} | 9.29×10^{-7} | 9.98×10^{-7}
		0.9	14; 8	0.42 | 14.51 | 19.73	1.09×10^{-7} | 9.21×10^{-7} | 9.83×10^{-7}
4	4.55×10^4	0.6	286; 205	2.01 | 35.65 | 63.67	9.47×10^{-7} | 9.52×10^{-7} | 9.94×10^{-7}
		0.7	145; 102	1.77 | 33.64 | 49.07	3.31×10^{-7} | 9.77×10^{-7} | 9.94×10^{-7}
		0.8	58; 40	1.12 | 31.12 | 39.93	1.46×10^{-7} | 9.60×10^{-7} | 9.50×10^{-7}
		0.9	12; 9	0.64 | 28.96 | 34.60	7.32×10^{-8} | 9.66×10^{-7} | 9.47×10^{-7}
5	5.69×10^4	0.6	189; 133	2.62 | 61.07 | 83.39	3.50×10^{-7} | 9.68×10^{-7} | 9.98×10^{-7}
		0.7	69; 47	1.50 | 58.16 | 63.74	1.24×10^{-7} | 9.44×10^{-7} | 9.59×10^{-7}
		0.8	10; 9	1.10 | 52.05 | 53.83	6.25×10^{-8} | 9.16×10^{-7} | 9.41×10^{-7}
		0.9	4; 3	0.83 | 50.61 | 42.12	2.10×10^{-8} | 8.82×10^{-7} | 9.61×10^{-7}

4.4. Results on Real Datasets

We evaluate the performances of three methods on three datasets: 10-K Corpus, StatLib, and UCI, which have been sourced from the LIBSVM datasets [42].

In our experiments with real datasets, we follow the methodology in [12] and utilize polynomial basis functions [43] to expand the original features across twelve datasets. For example, the number "10" in "mgscale10" indicates that we use a tenth-order polynomial to generate the basis functions. Furthermore, Table 2 provides statistical details for the

twelve datasets under consideration, where "m" refers to the sample size and "n" denotes the number of features. For the **GCNIR** problem, the regularization parameters are set as the following three different strategies [13,14]: (P1) given in the previous section and

$$\lambda = 0.5\varpi\|A^\top b\|_\infty,\ \tau = 9\lambda; \tag{P2}$$

$$\lambda = 0.5\varpi\|A^\top b\|_\infty,\ \tau = \lambda/\sqrt{n}. \tag{P3}$$

In our experiments, we use two values of ϖ for all datasets.

Table 2. Summary of tested data sets.

No.	Proname	$m; n$	$\lambda_{\max}(AA^\top)$
N1	E2006.train	16,087;150,360	1.91×10^5
N2	E2006.test	3308; 150,358	4.79×10^4
N3	pyrim5	74; 201,376	1.22×10^6
N4	triazines4	186; 635,376	2.07×10^7
N5	abalone7	4177; 6435	5.21×10^5
N6	bodyfat7	252; 116,280	5.29×10^4
N7	housing7	506; 77,520	3.28×10^5
N8	mpg7	392; 3432	1.28×10^4
N9	space9	3107; 5005	4.01×10^3
N10	mg10	1385; 8008	5.11×10^3
N11	eunite2001_train6	336; 74,613	3.54×10^4
N12	eunite2001_test6	31; 74,613	2.14×10^3

Tables 3–5 display the comparative results of the three algorithms across parameter sets (P1), (P2), and (P3), respectively. An analysis of these tables reveals that the SSNAL algorithm was able to solve all 72 test cases within a 70 s timeframe, with the majority of these cases being resolved in under 20 s. In comparison, ADMM encountered failures in 23 cases, while APG failed in 50 instances. These results underscore the superior performance of SSNAL, which not only consistently outperformed ADMM and APG in terms of speed but also demonstrated a higher success rate.

Table 3. The performances of SSNAL, ADMM, and APG when applied to real datasets with the regularization parameters defined as in (P1).

No.	ϖ	nnz; mon	Time	Res
			SSNAL | ADMM | APG	**SSNAL | ADMM | APG**
N1	1×10^{-5}	3; 2	1.48 | 918.82 | 394.89	2.11×10^{-7} | 9.97×10^{-7} | 5.89×10^{-7}
	1×10^{-6}	30; 27	2.95 | 10,425.34 | 3574.08	3.13×10^{-7} | **$1.64 * \times 10^{-7}$** | **6.47×10^{-7}**
N2	1×10^{-5}	1; 1	0.87 | 5.35 | 2.69	1.78×10^{-7} | 7.19×10^{-7} | 1.44×10^{-10}
	1×10^{-6}	53; 51	1.40 | 1415.89 | 1147.08	3.39×10^{-7} | **4.12×10^{-4}** | **2.79×10^{-4}**
N3	1×10^{-3}	188; 62	9.35 | 1218.89 | 1769.23	2.19×10^{-9} | 9.99×10^{-7} | **1.14×10^{-3}**
	1×10^{-4}	229; 72	12.64 | 1844.98 | 1759.28	1.99×10^{-8} | **1.19×10^{-4}** | **3.58×10^{-3}**
N4	1×10^{-3}	813; 144	44.33 | 10,800.13 | 10,800.24	7.25×10^{-7} | **8.61×10^{-4}** | **2.67×10^{-3}**
	1×10^{-4}	932; 191	66.75 | 10,800.39 | 10,800.20	5.07×10^{-8} | **4.20×10^{-4}** | **1.51×10^{-2}**
N5	1×10^{-3}	31; 16	3.74 | 203.73 | 3056.66	2.16×10^{-8} | 9.95×10^{-7} | **2.08×10^{-5}**
	1×10^{-4}	76; 57	8.13 | 2050.13 | 3028.76	6.65×10^{-8} | 1.00×10^{-6} | **8.40×10^{-4}**
N6	1×10^{-3}	2; 2	4.03 | 1718.95 | 613.07	1.01×10^{-9} | 1.00×10^{-6} | 6.60×10^{-7}
	1×10^{-4}	4; 3	4.78 | 1087.70 | 2832.63	1.10×10^{-9} | 1.00×10^{-6} | 9.89×10^{-7}

Table 3. *Cont.*

No.	ϖ	nnz; mon	Time	Res
			SSNAL | ADMM | APG	SSNAL | ADMM | APG
N7	1×10^{-3}	192; 121	8.07 | 1391.03 | 3812.60	2.83×10^{-8} | 1.00×10^{-6} | **5.54×10^{-5}**
	1×10^{-4}	337; 232	10.96 | 4666.00 | 4022.00	9.20×10^{-8} | 1.00×10^{-6} | **2.20×10^{-3}**
N8	1×10^{-3}	57; 33	0.35 | 0.96 | 15.52	4.17×10^{-9} | 1.00×10^{-6} | **1.68×10^{-6}**
	1×10^{-4}	169; 99	0.48 | 27.31 | 20.70	9.73×10^{-7} | **1.59×10^{-5}** | **5.51×10^{-5}**
N9	1×10^{-3}	23; 12	0.92 | 105.33 | 272.03	3.45×10^{-7} | 9.97×10^{-7} | 8.64×10^{-7}
	1×10^{-4}	47; 31	1.38 | 1057.48 | 1602.03	1.26×10^{-8} | 1.00×10^{-6} | **5.78×10^{-6}**
N10	1×10^{-3}	28; 21	0.66 | 37.79 | 268.66	1.26×10^{-7} | 9.98×10^{-7} | 9.92×10^{-7}
	1×10^{-4}	87; 60	1.44 | 505.11 | 1062.37	2.73×10^{-8} | 1.00×10^{-6} | **2.98×10^{-6}**
N11	1×10^{-3}	22; 7	3.17 | 204.05 | 2806.48	2.80×10^{-8} | 9.96×10^{-7} | **1.94×10^{-5}**
	1×10^{-4}	142; 26	3.27 | 321.88 | 2704.28	9.52×10^{-8} | 9.99×10^{-7} | **4.34×10^{-5}**
N12	1×10^{-3}	19; 5	0.75 | 27.07 | 241.41	2.25×10^{-8} | 9.98×10^{-7} | **5.09×10^{-6}**
	1×10^{-4}	102; 27	0.88 | 252.12 | 248.56	3.96×10^{-7} | **9.87×10^{-6}** | **7.27×10^{-6}**

* The bolded data indicate that the corresponding value did not meet the precision requirements.

Table 4. The performances of SSNAL, ADMM, and APG when applied to real datasets with the regularization parameters defined as in (P2).

No.	ϖ	nnz; mon	Time	Res
			SSNAL | ADMM | APG	SSNAL | ADMM | APG
N1	1×10^{-5}	1; 1	1.03 | 289.40 | 2.51	9.13×10^{-9} | 9.63×10^{-7} | 5.50×10^{-9}
	1×10^{-6}	3; 2	2.16 | 1142.71 | 2192.27	5.88×10^{-8} | 9.99×10^{-7} | 2.36×10^{-7}
N2	1×10^{-5}	1; 1	0.33 | 5.97 | 0.70	1.12×10^{-7} | 7.18×10^{-7} | 2.78×10^{-9}
	1×10^{-6}	5; 4	0.64 | 274.56 | 1128.41	1.92×10^{-7} | 9.99×10^{-7} | **$2.16 * \times 10^{-5}$**
N3	1×10^{-3}	716; 46	5.70 | 574.10 | 1760.94	1.00×10^{-8} | 9.99×10^{-7} | **5.91×10^{-4}**
	1×10^{-4}	776; 61	7.90 | 1785.58 | 1728.24	8.79×10^{-7} | **9.31×10^{-5}** | **5.62×10^{-3}**
N4	1×10^{-3}	2475; 119	26.30 | 10,800.38 | 10,800.40	4.09×10^{-8} | **2.41×10^{-3}** | **9.35×10^{-4}**
	1×10^{-4}	3525; 161	50.08 | 10,800.09 | 10,800.13	2.56×10^{-7} | **4.60×10^{-5}** | **8.83×10^{-3}**
N5	1×10^{-3}	96; 9	2.18 | 232.40 | 2917.46	6.92×10^{-9} | 9.95×10^{-7} | **4.89×10^{-6}**
	1×10^{-4}	82; 21	4.39 | 202.29 | 2925.83	6.72×10^{-9} | 9.97×10^{-7} | **1.76×10^{-4}**
N6	1×10^{-3}	9; 3	1.98 | 1187.52 | 1422.26	6.39×10^{-7} | 1.00×10^{-6} | 9.71×10^{-7}
	1×10^{-4}	8; 3	3.30 | 1064.66 | 2015.64	3.20×10^{-7} | 1.00×10^{-6} | 9.62×10^{-7}
N7	1×10^{-3}	347; 52	5.07 | 711.36 | 4618.18	5.92×10^{-8} | 1.00×10^{-6} | **1.75×10^{-5}**
	1×10^{-4}	711; 125	8.65 | 3054.37 | 4570.98	1.61×10^{-7} | 1.00×10^{-6} | **2.74×10^{-4}**
N8	1×10^{-3}	53; 13	0.24 | 1.15 | 13.51	4.70×10^{-7} | 1.00×10^{-6} | 9.96×10^{-7}
	1×10^{-4}	214; 51	0.52 | 11.98 | 19.47	7.79×10^{-8} | 1.00×10^{-6} | **1.09×10^{-5}**
N9	1×10^{-3}	16; 8	0.74 | 6.79 | 44.64	1.10×10^{-9} | 8.36×10^{-7} | 9.60×10^{-7}
	1×10^{-4}	70; 18	1.27 | 553.37 | 1597.02	3.19×10^{-8} | 1.00×10^{-6} | 9.91×10^{-7}
N10	1×10^{-3}	34; 12	0.73 | 5.65 | 100.67	3.00×10^{-7} | 9.32×10^{-7} | 9.97×10^{-7}
	1×10^{-4}	151; 33	0.96 | 219.02 | 701.33	6.53×10^{-8} | 9.99×10^{-7} | 1.00×10^{-6}
N11	1×10^{-3}	110; 3	1.89 | 304.06 | 874.43	9.99×10^{-9} | 9.99×10^{-7} | 9.53×10^{-7}
	1×10^{-4}	811; 19	3.24 | 180.17 | 2900.86	8.66×10^{-8} | 9.99×10^{-7} | **4.28×10^{-5}**
N12	1×10^{-3}	218; 3	0.56 | 10.26 | 52.18	1.81×10^{-7} | 1.00×10^{-6} | 9.41×10^{-7}
	1×10^{-4}	491; 15	0.84 | 266.41 | 254.19	3.78×10^{-7} | **2.06×10^{-6}** | **3.66×10^{-6}**

* The bolded data indicate that the corresponding value did not meet the precision requirements.

From Table 3, SSNAL not only succeeds in achieving the required accuracy but also takes less time than ADMM and APG. For instance, in the case of N7 with $\varpi = 1 \times 10^{-4}$, the SSNAL algorithm takes 10.96 s to reach the high accuracy of Res $= 9.20 \times 10^{-8}$, while the ADMM algorithm needs 4666 s to achieve the lower accuracy of Res $= 1.00 \times 10^{-6}$, and the APG algorithm even requires 4022 s to achieve a relatively large error of Res $= 2.20 \times 10^{-3}$. Therefore, the SSNAL algorithm outperforms the other two algorithms in addressing the **GCNIR** problem.

Table 5. The performances of SSNAL, ADMM, and APG when applied to real datasets with the regularization parameters defined as in (P3).

No.	ϖ	nnz; mon	Time	Res
			SSNAL | ADMM | APG	**SSNAL | ADMM | APG**
N1	1×10^{-5}	2; 2	3.09 | 1863.19 | 2538.36	5.75×10^{-7} | 9.99×10^{-7} | 9.56×10^{-7}
	1×10^{-6}	95; 93	4.55 | 9785.88 | 3499.10	3.32×10^{-7} | **4.03 * × 10^{-4}** | **1.02 × 10^{-3}**
N2	1×10^{-5}	3; 3	1.97 | 180.83 | 1142.03	3.34×10^{-8} | 9.99×10^{-7} | **1.98 × 10^{-5}**
	1×10^{-6}	170; 168	7.19 | 1411.02 | 1103.22	6.32×10^{-7} | **4.20 × 10^{-4}** | **5.08 × 10^{-4}**
N3	1×10^{-3}	84; 83	4.39 | 1615.80 | 1548.70	3.38×10^{-7} | **3.10 × 10^{-5}** | **9.85 × 10^{-4}**
	1×10^{-4}	79; 78	4.62 | 1571.90 | 1546.56	9.00×10^{-7} | **9.94 × 10^{-4}** | **3.05 × 10^{-3}**
N4	1×10^{-3}	231; 168	33.43 | 10,800.08 | 10,800.05	7.56×10^{-7} | **3.58 × 10^{-4}** | **4.47 × 10^{-3}**
	1×10^{-4}	230; 192	52.89 | 10,800.22 | 10,800.35	3.57×10^{-7} | **3.72 × 10^{-2}** | **1.33 × 10^{-2}**
N5	1×10^{-3}	29; 28	4.48 | 972.66 | 3166.79	1.33×10^{-7} | 1.00×10^{-6} | **1.04 × 10^{-4}**
	1×10^{-4}	92; 86	16.39 | 3582.72 | 3013.39	1.35×10^{-7} | **1.37 × 10^{-5}** | **2.21 × 10^{-3}**
N6	1×10^{-3}	2; 2	3.31 | 1644.35 | 694.08	1.87×10^{-8} | 1.00×10^{-6} | 3.33×10^{-7}
	1×10^{-4}	8; 7	3.83 | 846.32 | 2765.49	5.32×10^{-8} | 1.00×10^{-6} | **4.51 × 10^{-5}**
N7	1×10^{-3}	195; 187	7.29 | 3279.67 | 4712.21	5.63×10^{-7} | 9.99×10^{-7} | **1.71 × 10^{-4}**
	1×10^{-4}	337; 331	7.50 | 4727.07 | 5025.99	9.51×10^{-7} | **1.94 × 10^{-5}** | **4.61 × 10^{-3}**
N8	1×10^{-3}	56; 54	0.28 | 5.28 | 16.26	2.66×10^{-8} | 1.00×10^{-6} | **6.37 × 10^{-6}**
	1×10^{-4}	169; 140	0.41 | 22.06 | 16.95	1.06×10^{-7} | **1.65 × 10^{-4}** | **1.28 × 10^{-4}**
N9	1×10^{-3}	18; 15	0.99 | 133.89 | 531.25	1.13×10^{-9} | 9.97×10^{-7} | 5.77×10^{-7}
	1×10^{-4}	50; 39	1.71 | 1936.72 | 1632.34	1.02×10^{-7} | **2.45 × 10^{-6}** | **6.39 × 10^{-6}**
N10	1×10^{-3}	32; 28	0.96 | 39.57 | 471.36	1.72×10^{-7} | 9.93×10^{-7} | 9.93×10^{-7}
	1×10^{-4}	92; 86	1.41 | 1105.05 | 1022.00	2.44×10^{-7} | **2.31 × 10^{-5}** | **1.10 × 10^{-5}**
N11	1×10^{-3}	18; 13	2.49 | 290.45 | 2284.04	1.04×10^{-8} | 9.97×10^{-7} | **7.13 × 10^{-6}**
	1×10^{-4}	100; 95	3.19 | 1436.32 | 2290.59	6.20×10^{-9} | 9.88×10^{-7} | **1.05 × 10^{-4}**
N12	1×10^{-3}	23; 21	0.63 | 110.50 | 235.08	1.74×10^{-7} | 9.97×10^{-7} | **1.17 × 10^{-5}**
	1×10^{-4}	47; 44	0.67 | 243.67 | 236.79	6.24×10^{-7} | **3.30 × 10^{-5}** | **1.41 × 10^{-5}**

* The bolded data indicate that the corresponding value did not meet the precision requirements.

Table 4 shows that the number of reversed order coefficients in z is almost the fewest among all tables. This is because the regularization parameter τ, which enforces monotonicity, is larger than the regularization parameter λ, which enforces sparsity in (P2). From Table 4, it is also evident that SSNAL demands considerably less time than the other two methods on twelve cases. Moreover, for more challenging tests, such as N3 with $\varpi = 1 \times 10^{-4}$, only SSNAL successfully solved this problem, while the other two algorithms did not meet the accuracy requirements. The results strongly indicate that our SSNAL algorithm can efficiently and reliably solve the **GCNIR** problem.

Table 5 further illustrates that SSNAL continues to outperform ADMM and APG by a significant margin. This advantage is particularly pronounced for large-scale problems.

In particular, for the case N4 with $\varpi = 1 \times 10^{-4}$, SSNAL solves it to the desired accuracy within 53 s, while ADMM and APG fail to solve it within 3 h.

Consequently, we can confidently state that our SSNAL algorithm can efficiently and robustly solve the **GCNIR** problem (1) on real datasets with high accuracy.

5. Conclusions

In this paper, we proposed a highly efficient semismooth Newton-based augmented Lagrangian method for solving the **GCNIR** problem from the dual perspective. The proximal mapping associated with the **GCNIR** regularizer and its generalized Jacobian have been derived, and we have utilized the second-order sparsity structure to achieve superior performance in solving the subproblem of the SSNAL algorithm. Numerical results have demonstrated the efficiency and robustness of our proposed algorithm compared to the widely used ADMM and APG methods on both the synthetic and real datasets. Looking ahead, we anticipate our algorithm to play a significant role in solving convex problems with the **GCNIR** regularizer, thereby facilitating data analysis in statistical learning.

Author Contributions: Conceptualization, Y.-J.L.; methodology, Y.X.; software, Y.X.; validation, L.L. and Y.-J.L.; formal analysis, Y.X., L.L. and Y.-J.L.; investigation, Y.X.; resources, Y.-J.L.; data curation, Y.X.; writing—original draft preparation, Y.X.; writing—review and editing, L.L. and Y.-J.L.; visualization, Y.X.; supervision, L.L. and Y.-J.L.; project administration, L.L. and Y.-J.L.; funding acquisition, L.L. and Y.-J.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (Grant No. 12271097), the Key Program of National Science Foundation of Fujian Province of China (Grant No. 2023J02007), the Central Guidance on Local Science and Technology Development Fund of Fujian Province (Grant No. 2023L3003), and the Fujian Alliance of Mathematics (Grant No. 2023SXLMMS01, 2025SXLMQN01).

Data Availability Statement: All data generated or analyzed during this study are included in this article.

Acknowledgments: The authors would very much like to thank the reviewers for their helpful suggestions.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Matyasovszky, I. Estimating red noise spectra of climatological time series. *Időjárás Q. J. Hung. Meteorol. Serv.* **2013**, *117*, 187–200.
2. Yu, Y.L.; Xing, E. Exact algorithms for isotonic regression and related. *J. Phys.* **2016**, *699*, 012016. [CrossRef]
3. Matsuda, T.; Miyatake, Y. Generalized nearly isotonic regression. *arXiv* **2021**, arXiv:2108.13010.
4. Obozinski, G.; Lanckriet, G.; Grant, C.; Jordan, M.I.; Noble, W.S. Consistent probabilistic outputs for protein function prediction. *Genome Biol.* **2008**, *9*, 247–254. [CrossRef] [PubMed]
5. Barlow, R.E.; Brunk, H.D. The isotonic regression problem and its dual. *J. Am. Stat. Assoc.* **1972**, *67*, 140–147. [CrossRef]
6. Tibshirani, R.J.; Hoefling, H.; Tibshirani, R. Nearly-isotonic regression. *Technometrics* **2011**, *53*, 54–61. [CrossRef]
7. Tibshirani, R.; Suo, X. An Ordered Lasso and Sparse Time-Lagged Regression. *Technometrics* **2016**, *58*, 415–423. [CrossRef] [PubMed]
8. Lin, L.; Liu, Y.J. An Efficient Hessian Based Algorithm for Singly Linearly and Box Constrained Least Squares Regression. *J. Sci. Comput.* **2021**, *88*, 26. [CrossRef]
9. Ayer, M.; Brunk, H.D.; Ewing, G.M.; Reid, W.T.; Silverman, E. An empirical distribution function for sampling with incomplete information. *Ann. Math. Statist.* **1955**, *26*, 641–647. [CrossRef]
10. Yu, Z.S.; Chen, X.Y.; Li, X.D. A dynamic programming approach for generalized nearly isotonic optimization. *Math. Prog. Comp.* **2023**, *15*, 195–225. [CrossRef]
11. Brian R. Gaines, J.K.; Zhou, H. Algorithms for fitting the constrained Lasso. *J. Comput. Graph. Stat.* **2018**, *27*, 861–871.

12. Li, X.D.; Sun, D.F.; Toh, K.C. A highly efficient semismooth Newton augmented Lagrangian method for solving Lasso problems. *SIAM J. Optim.* **2018**, *28*, 433–458. [CrossRef]
13. Zhang, Y.; Zhang, N.; Sun, D.; Toh, K.C. An efficient Hessian based algorithm for solving large-scale sparse group Lasso problems. *Math. Program.* **2018**, *179*, 223–263. [CrossRef]
14. Li, X.; Sun, D.; Toh, K.C. On efficiently solving the subproblems of a level-set method for fused Lasso problems. *SIAM J. Optim.* **2018**, *28*, 1842–1866. [CrossRef]
15. Lin, M.X.; Liu, Y.J.; Sun, D.F.; Toh, K.C. Efficient sparse semismooth Newton methods for the clustered Lasso problem. *SIAM J. Optim.* **2019**, *29*, 2026–2052. [CrossRef]
16. Sun, D.; Toh, K.C.; Yuan, Y. Convex clustering: Model, theoretical guarantee and efficient algorithm. *J. Mach. Learn. Res.* **2021**, *22*, 1–32.
17. Lin, L.; Liu, Y.J. An inexact semismooth Newton-based augmented Lagrangian algorithm for multi-task Lasso problems. *Asia Pac. J. Oper. Res.* **2024**, *41*, 2350027. [CrossRef]
18. Liu, Y.J.; Zhang, T. Sparse Hessian based semismooth Newton augmented Lagrangian algorithm for general ℓ_1 trend filtering. *Pac. J. Optim.* **2023**, *19*, 187–204.
19. Liu, Y.J.; Yu, J. A semismooth Newton-based augmented Lagrangian algorithm for density matrix least squares problems. *J. Optim. Theory Appl.* **2022**, *195*, 749–779. [CrossRef]
20. Fang, S.; Liu, Y.J.; Xiong, X. Efficient Sparse Hessian-Based Semismooth Newton Algorithms for Dantzig Selector. *SIAM J. Sci. Comput.* **2021**, *43*, 4147–4171. [CrossRef]
21. Liu, Y.J.; Yu, J. A semismooth Newton based dual proximal point algorithm for maximum eigenvalue problem. *Comput. Optim. Appl.* **2023**, *85*, 547–582. [CrossRef]
22. Liu, Y.J.; Zhu, Q. A semismooth Newton based augmented Lagrangian algorithm for Weber problem. *Pac. J. Optim.* **2022**, *18*, 299–315.
23. Liu, Y.J.; Xu, J.J.; Lin, L.Y. An easily implementable algorithm for efficient projection onto the ordered weighted ℓ_1 norm ball. *J. Oper. Res. Soc. China* **2023**, *11*, 925–940. [CrossRef]
24. Liu, Y.J.; Wan, Y.; Lin, L. An efficient algorithm for Fantope-constrained sparse principal subspace estimation problem. *Appl. Math. Comput.* **2024**, *475*, 128708. [CrossRef]
25. Moreau, J. Proximité et dualité dans un espace hilbertien. *Bull. Société Mathématique Fr.* **1965**, *93*, 273–299. [CrossRef]
26. Rockafellar, R. *Convex Analysis*; Princeton University Press: Princeton, NJ, USA, 1970; pp. 338–339.
27. Lemaréchal, C.; Sagastizábal, C. Practical aspects of the Moreau–Yosida regularization: Theoretical preliminaries. *SIAM J. Optim.* **1997**, *7*, 367–385. [CrossRef]
28. Yu, Y. On decomposing the proximal map. In Proceedings of the 27th International Conference on Neural Information Processing Systems, New York, NY, USA, 5–10 December 2013; pp. 91–99.
29. Han, J.; Sun, D. Newton and quasi-Newton methods for normal maps with polyhedral sets. *J. Optim. Theory Appl.* **1997**, *94*, 659–676. [CrossRef]
30. Rockafellar, R.T. Augmented Lagrangians and applications of the proximal point algorithm in convex programming. *Math. Oper. Res.* **1976**, *1*, 97–116. [CrossRef]
31. Rockafellar, R.T. Monotone operators and the proximal point algorithm. *SIAM J. Control Optim.* **1976**, *14*, 877–898. [CrossRef]
32. Rockafellar, R.T.; Wets, R.J.B. *Variational Analysis*; Springer: Berlin, Germany, 1998; p. 550.
33. Robinson, S.M. Some continuity properties of polyhedral multifunctions. In *Mathematical Programming at Oberwolfach*; König, H., Korte, B., Ritter, K., Eds.; Springer: Berlin, Germany, 1981; pp. 206–214.
34. Luque, F.J. Asymptotic convergence analysis of the proximal point algorithm. *SIAM J. Control Optim.* **1984**, *22*, 277–293. [CrossRef]
35. Facchinei, F.; Pang, J.S. *Finite-Dimensional Variational Inequalities and Complementarity Problems*; Springer: New York, NY, USA, 2003; p. 345.
36. Mifflin, R. Semismooth and semiconvex functions in constrained optimization. *SIAM J. Control Optim.* **1977**, *15*, 959–972. [CrossRef]
37. Kummer, B., Newton's method for non-differentiable functions. In *Advances in Mathematical Optimization*; Guddat, J., Ed.; De Gruyter: Berlin, Germany, 1988; pp. 114–125.
38. Qi, L.; Sun, J. A nonsmooth version of Newton's method. *Math. Program.* **1993**, *58*, 353–367. [CrossRef]
39. Sun, D.; Sun, J. Semismooth matrix-valued functions. *Math. Oper. Res.* **2002**, *27*, 150–169. [CrossRef]
40. Luo, Z.; Sun, D.; Toh, K.C.; Xiu, N. Solving the OSCAR and SLOPE models using a semismooth Newton-based augmented Lagrangian method. *J. Mach. Learn. Res.* **2019**, *20*, 1–25.
41. Gabay, D.; Mercier, B. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Appl. Math. Comput.* **1976**, *2*, 17–40. [CrossRef]

42. LIBSVM—A Library for Support Vector Machines. Available online: https://www.csie.ntu.edu.tw/~cjlin/libsvm/ (accessed on 15 January 2024).
43. Huang, L.; Jia, J.; Yu, B.; Chun, B.G.; Maniatis, P.; Naik, M. Predicting execution time of computer programs using sparse polynomial regression. In Proceedings of the 24th International Conference on Neural Information Processing Systems, New York, NY, USA, 6–9 December 2010; pp. 883–891.