A SUBSPACE ACCELERATION METHOD FOR MINIMIZATION
INVOLVING A GROUP SPARSITY-INDUCING REGULARIZER∗

FRANK E. CURTIS, YUTONG DAI, AND DANIEL P. ROBINSON†

Abstract. We consider the problem of minimizing an objective function that is the sum of a
convex function and a group sparsity-inducing regularizer. Problems that integrate such regularizers
arise in modern machine learning applications, often for the purpose of obtaining models that are
easier to interpret and that have higher predictive accuracy. We present a new method for solving such
problems that utilize subspace acceleration, domain decomposition, and support identification. Our
analysis shows, under common assumptions, that the iterate sequence generated by our framework
is globally convergent, converges to an ϵ-approximate solution in at most \( O(\epsilon^{-(1+p)}) \) (respectively,
\( O(\epsilon^{-(2+p)}) \)) iterations for all \( \epsilon \) bounded above and large enough (respectively, all \( \epsilon \) bounded above)
where \( p > 0 \) is an algorithm parameter, and exhibits superlinear local convergence. Preliminary
numerical results for the task of binary classification based on regularized logistic regression show
that our approach is efficient and robust, with the ability to outperform a state-of-the-art method.

Key words. nonlinear optimization, convex optimization, worst-case iteration complexity, reg-
ularization methods, group regularizer, sparsity, logistic regression, subspace acceleration

AMS subject classifications. 49M37, 65K05, 65K10, 65Y20, 68Q25, 90C30, 90C60

1. Introduction. We consider the minimization of a function that may be writ-
ten as the sum of a convex function and a nonoverlapping group sparsity-inducing
regularizer. Specifically, given a convex and twice continuously differentiable function
\( f : \mathbb{R}^n \to \mathbb{R}, \) a collection of \( n_G > 0 \) nonoverlapping groups \( G := \{G_i\}_{i=1}^{n_G} \) that forms a
partition of \( \{1, 2, \ldots, n\} \) (i.e., \( G_i \cap G_j = \emptyset \) for all \( i \neq j \) and \( \bigcup_{i=1}^{n_G} G_i = \{1, 2, \ldots, n\} \)),
and group-wise weighting parameters \( \{\lambda_i\}_{i=1}^{n_G} > 0 \), our algorithm solves the problem

\[
\min_{x \in \mathbb{R}^n} \{f(x) + r(x)\}, \quad \text{where } r(x) := \sum_{i=1}^{n_G} \lambda_i \|x|_{G_i}\|_2
\]

and \( |x|_{G_i} \) is the subvector of \( x \) corresponding to elements in \( G_i \). The regularizer \( r \) gen-
eralizes the \( \ell_1 \)-norm, which is recovered by choosing \( G_i = \{i\} \) for all \( i \in \{1, 2, \ldots, n\} \).

Despite the successes of \( \ell_1 \)-norm regularization, its inadequacy in the context of
many modern machine learning applications has been noticed by researchers, and is
one motivation for the use of group regularization. In some machine learning appli-
cations the covariates come in groups (e.g., genes that regulate hormone levels
in microarray data [23]), in which case one may wish to select them jointly. Also,
integrating group information into the modeling process can improve both the in-
terpretability and accuracy [35] of the resulting model. Yuan and Lin [34] observed
that in the multi-factor analysis-of-variance problem, where each factor is expressed
through a set of dummy variables, deleting an irrelevant factor is equivalent to deleting
a group of dummy variables; the \( \ell_1 \)-norm regularizer fails to achieve this goal.

1.1. State-of-the-art methods. There is a long history of algorithms for solv-
ing regularized problems of the form (1.1) (see [1] and the references therein). Here,
we review some of the state-of-the-art approaches for solving sparsity-promoting prob-
lems that are most closely related to our proposed approach.

∗This material is based upon work supported by the U.S. National Science Foundation under the
Division of Computing and Communication Foundations (Award Number CCF-1740796).
†Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, USA;
E-mail: frank.e.curtis@gmail.com, yud319@lehigh.edu, daniel.p.robinson@gmail.com

This manuscript is for review purposes only.
First-order methods. Proximal methods are designed to solve problems of the form (1.1) and have received attention in the machine learning community [3, 7, 31]. A well-known example for $\ell_1$-norm regularized problems is the iterative shrinkage-thresholding algorithm (ISTA), which is obtained by applying a proximal gradient (PG) iteration to minimize a smooth function plus the $\ell_1$-norm regularizer [10, 12]. Under certain assumptions, one can prove a worst-case complexity bound on the number of iterations required by the PG method before it correctly identifies the support of the optimal solution [28]. Combined with the acceleration technique proposed by Nesterov [26, 27], one obtains the algorithm FISTA [3]. One obtains a related, but distinct approach from ISTA by posing an equivalent smooth reformulation of the problem—separating the positive and negative parts of the variables—and applying a gradient projection method to the resulting formulation [13, 15]. All of these approaches have been shown to work well in practice, at least compared to other first-order methods such as the subgradient algorithm. However, these algorithms are often inferior in practice compared to alternative approaches that employ space decomposition techniques and/or second-order derivatives [5, 6, 18].

As an alternative to PG and gradient projection techniques, researchers have considered (block) coordinate descent for solving $\ell_1$-norm regularized problems. Such a strategy is appealing, since when minimizing an $\ell_1$-norm regularized objective along coordinate directions, it is common that the objective is minimized with variables being zero. These approaches are also easy to implement to exploit parallel computing; see, e.g., the accelerated randomized proximal coordinate gradient method in [20], the parallel coordinate descent methods in [29], and the asynchronous coordinate descent technique in [22]. A downside of these approaches is that the space decomposition is performed in a prescribed manner, rather than in an adaptive way that can benefit from information acquired during the solution process. Also, these approaches do not effectively exploit second-order derivative information and require exact minimization along coordinate directions. An exception to this latter criticism is the inexact coordinate descent algorithm from [30], although this approach does not effectively exploit second-order derivatives and uses a prescribed space decomposition strategy.

Various other approaches have been proposed for solving problems involving specific regularizers. In [21], the authors discuss various methods for sparse learning that make use of projection techniques. A well-known package is GLMNET [16], which is designed for solving problems with the elastic-net regularization. Finally, let us mention the work in [32], which proposes and tests a groupwise-majorization-descent algorithm (called gglasso) for solving problems involving the group-$\ell_1$-norm regularizer. A potential downside of this approach is that it updates variables by groups in a cycle, rather than by using an adaptive space decomposition technique.

Second-order methods. Relatively few second-order methods have been proposed for minimizing sparsity-promoting objective functions. In [17], an accelerated regularized Newton scheme is proposed. A similar proximal-Newton method is proposed in [19], which under some assumptions can be shown to converge locally superlinearly. These approaches can be effective in practice, although they appear to lack good worst-case guarantees in terms of identification of the optimal solution support. Other approaches, such as the orthant-based method in [18], can predict the solution support, but in practice are often outperformed by a closely related method called FarsA [5, 6]. As for publicly available solvers based on second-order methods, most have been designed for specific loss functions and regularizers. For example, newGLMNET in [33] is designed for $\ell_1$-regularized logistic regression and the method in [14] is designed for regularized logistic regression and support vector machines.
1.2. Notation and assumptions. Let $\mathbb{R}$ denote the set of real numbers, $\mathbb{R}^n$ denote the set of $n$-dimensional real vectors, and $\mathbb{R}^{m \times n}$ denote the set of $m$-by-$n$-dimensional real matrices. The set of natural numbers is denoted as $\mathbb{N} := \{0, 1, 2, \ldots\}$.

For any set $\mathcal{I} \subseteq \{1, 2, \ldots, n\}$, we define the projection of $x \in \mathbb{R}^n$ onto the subspace spanned by the coordinate vectors indexed by the entries of $\mathcal{I}$ as $P_\mathcal{I}(x)$, so that

$$
[P_\mathcal{I}(x)]_i := \begin{cases} x_i & \text{if } i \in \mathcal{I}, \\ 0 & \text{if } i \notin \mathcal{I}. \end{cases}
$$

For a function $h : \mathbb{R}^n \to \mathbb{R}$, vector $x \in \mathbb{R}^n$, and direction $d \in \mathbb{R}^n$, the directional derivative of $h$ at $x$ in the direction $d$ is defined as the following limit:

$$
D_h(x;d) := \lim_{t \to 0} \frac{h(x + td) - h(x)}{t}.
$$

The following assumption is assumed to hold throughout the paper.

**Assumption 1.1.** The function $f : \mathbb{R}^n \to \mathbb{R}$ used in the definition of the objective function of problem (1.1) is convex and continuously differentiable. It follows that there exists a constant $L_f$ such that $\|\nabla f(x)\|_2 \leq L_f$ for all $x \in \mathcal{L} := \{x \in \mathbb{R}^n : f(x) + r(x) \leq f(x_0) + r(x_0)\}$ for any initial estimate $x_0$ of a solution to problem (1.1).

The objective function $f + r$ is bounded below and the gradient function $\nabla f$ is Lipschitz continuous on $\mathcal{L}$ with Lipschitz constant $L_g$.

1.3. Organization. In Section 2, we present preliminary results related to PG calculations. In Section 3, by using PG-calculations as a starting point, we propose a reduced-space second-order domain decomposition algorithm for solving problem (1.1). The algorithm is analyzed in Section 4 and numerical results are presented in Section 5. Finally, in Section 6, we provide concluding remarks.

2. Preliminaries. In this section, we discuss preliminary material related to the objective function $f + r$ and its associated PG calculations. (All proofs may be found in Appendix A.) For any $\bar{x} \in \mathbb{R}^n$ and $\bar{\sigma} > 0$, we define the PG update as

$$
T(\bar{x}, \bar{\sigma}) := \arg\min_{x \in \mathbb{R}^n} \left\{ \frac{1}{\sigma} \|x - (\bar{x} - \bar{\sigma}\nabla f(\bar{x}))\|_2^2 + r(x) \right\}
$$

and the associated PG step as

$$
s(\bar{x}, \bar{\sigma}) := T(\bar{x}, \bar{\sigma}) - \bar{x}.
$$

The next result shows that the directional derivative of $f + r$ along the PG step is negative with magnitude proportional to the squared norm of the PG direction.

**Lemma 2.1.** For any $\bar{x} \in \mathbb{R}^n$ and $\bar{\sigma} > 0$, the PG step $s(\bar{x}, \bar{\sigma})$ in (2.2) satisfies

$$
D_{f+r}(\bar{x}, s(\bar{x}, \bar{\sigma})) \leq -\frac{1}{\bar{\sigma}}\|s(\bar{x}, \bar{\sigma})\|_2^2.
$$

The PG update defined in (2.1) can be computed group-wise for each $G_i \in G$ by

$$
[T(\bar{x}, \bar{\sigma})]_i = \left[ \arg\min_{x \in \mathbb{R}^n} \left\{ \frac{1}{\sigma} \|x - (\bar{x} - \bar{\sigma}\nabla f(\bar{x}))\|_2^2 + \sum_{i=1}^n \lambda_i \|x_i\|_2 \right\} \right]_i
$$

$$
= \max \left\{ 1 - \frac{\bar{\sigma} \lambda_i}{\|\bar{x}_i - \bar{\sigma}\nabla \bar{\sigma}_i, f(\bar{x})\|_2}, 0 \right\} \left( [\bar{x}]_i - \bar{\sigma}\nabla \bar{\sigma}_i, f(\bar{x}) \right).
$$

Combining this observation with Lemma 2.1 leads to the following corollary, which will be relevant to the manner in which we design the algorithm we propose in Section 3.
Lemma 2.2. For any $\varpi \in \mathbb{R}^n$, $\bar{\varpi} > 0$, and set $\mathcal{I}$ equal to the union of a subset of $\{G_i\}_{i=1}^{m}$, the PG step $s(\varpi, \bar{\varpi})$ defined in (2.2) satisfies

$$D_{f+r}(\varpi; P_c(s(\varpi, \bar{\varpi})) \leq -\frac{1}{2} \| P_c(s(\varpi, \bar{\varpi})) \|^2$$

where the projection operator $P_c$ is defined through (1.2).

Our next result quantifies the decrease in $f + r$ that one can expect to obtain by taking a PG step $s(\varpi, \bar{\varpi})$, provided the PG parameter $\bar{\varpi}$ is sufficiently small.

Lemma 2.3. For any $\varpi \in \mathbb{R}^n$, $\bar{\varpi} \in (0, 2/L)$, and $\mathcal{I}$ equal to the union of a subset of $\{G_i\}_{i=1}^{m}$, the objective function decrease satisfies

$$f(\varpi + P_c(\varpi, \bar{\varpi}) + r(\varpi + P_c(\varpi, \bar{\varpi})) \leq f(\varpi) + r(\varpi) - (\frac{1}{2} - \frac{\bar{\varpi}}{2})\| P_c(s(\varpi, \bar{\varpi})) \|^2.$$ 

The next result shows that, when restricted to certain groups, the size of the PG step is bounded above by the gradient of the objective function.

Lemma 2.4. If the pair $(\varpi, \bar{\varpi})$ and group $G_i$ satisfy $\bar{\varpi} \in (0, 1]$, $|\varpi|_{G_i} \neq 0$, and $|\varpi + s(\varpi, \bar{\varpi})|_{G_i} \neq 0$, where $s(\varpi, \bar{\varpi})$ is defined in (2.2), then

$$\| \nabla_{\varpi_i} (f + r)(\varpi) \|_2 \geq || |s(\varpi, \bar{\varpi})|_{G_i} \|^2.$$ 

With the preliminaries now completed, we can propose our new algorithm.

3. Proposed Algorithm Framework. We propose Algorithm 3.1, which we call FaRSA-Group (Fast Reduced-Space Algorithm for Group sparsity-inducing regularization) for solving problem (1.1) that uses ideas related to domain decomposition, subspace acceleration, and support identification. An overview of the algorithm is described in Section 3.1. During each iteration of our method, at least one of three subroutines is called. The three subroutines are described in Sections 3.2–3.4.

3.1. Main algorithm (Algorithm 3.1). Our main algorithm is formally stated as Algorithm 3.1. At the beginning of the $k$th iteration, $x_k$ and $\alpha_k > 0$ denote the current solution estimate for problem (1.1) and the PG parameter, respectively. We then compute $s_k$ in Line 5 as the PG step associated with problem (1.1), namely,

$$s_k := s(x_k, \alpha_k) \text{ with } s(x_k, \alpha_k) \text{ defined in (2.2)}$$

Although the repeated computation of PG steps is the basis for a first-order method, here we primarily use it to predict the zero/nonzero structure of a solution and to formulate optimality measures. Specifically, in Line 6 we compute the index set

$$\mathcal{I}_k^c := \{ j \in G_i : [x_k]_{G_i} \neq 0, \ [x_k + s_k]_{G_i} \neq 0, \text{ and } \| [x_k]_{G_i} \|_2 \geq \kappa_1 \| \nabla_{\varpi_i} (f + r)(x_k) \|_2 \}$$

for some $\kappa_1 \in (0, \infty)$. The groups of variables that compose $\mathcal{I}_k^c$ are candidates for use in a Newton-type calculation aimed to accelerated convergence. Before using them, however, we first check to see if each candidate block is sufficiently far from zero, and those that are not are removed. Specifically, we first define

$$\mathcal{I}_k^\text{small} := \{ j \in G_i : G_i \subseteq \mathcal{I}_k^c \text{ and } \| [x_k]_{G_i} \|_2 < \kappa_2 \| \nabla_{\varpi_i}^c (f + r)(x_k) \|^2 \}$$

for some $\{\kappa_2, p\} \subset (0, \infty)$, and then define in Line 7 the sets and optimality measures

$$\mathcal{I}_k^c :\mathcal{I}_k^c \setminus \mathcal{I}_k^\text{small} \quad \text{and} \quad \lambda_k^c := \| [s_k]_{\mathcal{I}_k^c} \|^2 \quad \text{and} \quad \lambda_k^p := \| [s_k]_{\mathcal{I}_k^p} \|^2$$

This manuscript is for review purposes only.
where by convention $\| \cdot \|_2 = 0$. (See Lemma 4.1 for a justification that these sets together represent a measure of optimality.) This construction of sets also ensures that the subvector of $x_k$ that corresponds to $G_i$ for each $G_i \subseteq T_k^G$ is at least a distance away from zero (see Lemma 4.5(i)), which is crucial in our analysis.

Armed with $\chi_k^{pg}$ and $\chi_k^{cs}$, Algorithm 3.1 seeks decrease in the objective function in a subspace that is likely to allow for significant progress. We consider two cases.

**Case 1: the condition $\chi_k^{pg} \leq \chi_k^{cs}$ checked in Line 8 holds.** In this case, the inequality $\chi_k^{pg} \leq \chi_k^{cs}$ indicates that significant reduction in the objective function can be achieved by focusing on variables in the set $T_k^G$. Therefore, in Line 9 we choose any index set $\mathcal{I}_k$ that is (i) a subset of $T_k^G$, (ii) equal to the union of some subset of groups from $G$, and (iii) the size of the PG step restricted to the index set $\mathcal{I}_k$ is at least a fraction of the size of the PG step when restricted to the index set $T_k^G$. The easiest choice that satisfies these conditions is $\mathcal{I}_k \equiv T_k^G$, but for large-scale problems it may be beneficial to restrict $|\mathcal{I}_k|$. The opposite extreme choice is selecting $\mathcal{I}_k$ as the group $G_i$ contained in $T_k^{cs}$ with largest associated PG step, in which case one would choose $\varphi = 1/\sqrt{m_k}$ for the user-defined parameter in Line 9. Once $\mathcal{I}_k$ has been selected, a reduced-space gradient $g_k$ and reduced-space positive-definite matrix $H_k$ is computed in Line 10, where the derivatives are taken with respect to variables in $\mathcal{I}_k$. (In practice, $H_k$ could be selected based on $\nabla^2_{\mathcal{I}_k \mathcal{I}_k} (f + r)(x_k)$ to ensure a fast local convergence rate.) Note that such derivatives exist since by construction $\mathcal{T}_k \subseteq \mathcal{T}_k^{cs} \subseteq \mathcal{T}_k^{pg}$, and from (3.2) the objective function $f + r$ is differentiable with respect to groups of variables in $\mathcal{T}_k^{cs}$. Next, $g_k$ and $H_k$ are used to compute a direction $d_k$ of sufficient descent for $f + r$ by calling the subroutine $CG\_DIRECTION$ (see Section 3.2). Once a full-space vector $d_k$ is obtained by padding $d_k$ with zeros in Line 12, a projected line search is performed by calling subroutine $UPDATE\_CG$ in Line 13 (see Section 3.3).

**Case 2: the condition $\chi_k^{pg} > \chi_k^{cs}$ checked in Line 8 does not hold.** In this case, the inequality $\chi_k^{pg} > \chi_k^{cs}$ indicates that significant reduction in the objective function can be achieved by focusing on variables in the set $T_k^{pg}$. Therefore, in Line 16, we choose any index set $\mathcal{I}_k$ that is (i) a subset of $T_k^{pg}$, (ii) equal to the union of some subset of groups from $G$, and (iii) the size of the PG step restricted to the index set $\mathcal{I}_k$ is at least a fraction of the size of the PG step restricted to the index set $T_k^{pg}$. The easiest choice that satisfies these conditions is $\mathcal{I}_k \equiv T_k^{pg}$. Once $\mathcal{I}_k$ has been chosen, the next iterate is obtained by performing a line search along the PG direction in Line 17 by calling the subroutine $UPDATE\_PG$ (for details, see Section 3.4). If the subroutine returns flag$^{pg} = \text{decrease}_{\alpha}$, the PG parameter is decreased for the next iteration.

**3.2. Computing a CG direction (Algorithm 3.2).** This subroutine returns a reduced-space direction $d_k$ that satisfies conditions (3.7)–(3.9). We call it a reduced-space vector because the inputs $g_k$ and $H_k$ are elements in $\mathbb{R}^{\mathcal{I}_k}$ and $\mathbb{R}^{\mathcal{I}_k \times \mathcal{I}_k}$, respectively, where $\mathcal{I}_k$ is computed in Line 9 of Algorithm 3.1. Condition (3.7) ensures that $d_k$ is a descent direction for the objective function as a consequence of how the reference direction $d_k^{ref}$ is computed in Line 25. Condition (3.8) ensures that $d_k$ reduces the model $m_k$ at least as much as a zero step. Finally, condition (3.9) promotes fast local convergence of the iterate sequence $\{x_k\}$ (see Section 4.2), but its enforcement (or lack of enforcement) is irrelevant with respect to the complexity result that we prove in Section 4.1. The subroutine name $CG\_DIRECTION$ indicates our intent to use the linear CG algorithm in our implementation, although other possible options...
include a block-wise coordinate descent method applied to the model $m_k$ in (3.6). In particular, the direction associated with every iteration of the CG algorithm satisfies conditions (3.7)–(3.8), and condition (3.9) is satisfied by all sufficiently large CG iterations. Thus, the requirements of this subroutine can always be met.

### 3.3. Reduced-space search using the CG direction (Algorithm 3.3)

This subroutine performs a search using the direction $d_k$ returned by the subroutine `cg_direction` in Line 11 of Algorithm 3.1. For an illustration of this search, which incorporates projections, see Figure 3.1. The approach uses the direction $d_k$, without modification, for each block of variables $G_i$ such that the ray $\{x_k + \tau d_k | G_i : \tau \geq 0\}$ does not intersect the ball centered at zero of radius $\bar{\rho}_{k,i} = \min\{\rho_{k,i}, \sin(\theta) \|x_k| G_i\|_2\}$, where $\rho_{k,i}$ is defined in (3.5) and $\theta \in (0, \pi/2)$ is a user-defined parameter. When they do intersect, we first compute $\tau_{k,i}$ as the smallest step along the Newton direction (restricted to block $G_i$) that intersects the ball. Then, during the search that follows, anytime the trial step size $\xi_j$ is larger than $\tau_{k,i}$, the trial step for block $G_i$ is set to zero; otherwise, the Newton direction is used so that the trial step (with respect to block $G_i$) is $[x_k + \xi d_k | G_i]$ (see Line 41). If termination occurs in Line 42, then a new block of variables will become zero, in which case we require the objective function not to increase (see Line 43). On the other hand, if termination occurs in Line 48, then it indicates that the objective function has been sufficiently reduced (see Line 47) and no new groups of zeros have been formed.

### 3.4. Reduced-space line search along a PG step (Algorithm 3.4)

This subroutine performs a line search along the PG direction $P_T(s_k)$. The search ensures that the next iterate yields decrease in the objective of size at least $(\eta \xi_j / \alpha_k) \|P_T(s_k)\|_2^2$ for some positive integer $j$ computed within the while loop in Line 53. Once the while loop terminates, the update flag$^{\text{PG}}_{k} \leftarrow \text{same} \alpha$ is made if $j = 0$, and set as flag$^{\text{PG}}_{k} \leftarrow \text{decrease} \alpha$ otherwise. The motivation for this update is Lemma 2.3, which shows that the while loop in Line 53 will terminate with $j = 0$ if the PG parameter $\alpha_k$ is sufficiently small. Therefore, anytime $j > 0$, Algorithm 3.4 returns flag$^{\text{PG}}_{k} \leftarrow \text{decrease} \alpha$ to Algorithm 3.1 in Line 17 so that the PG parameter value for the next iteration is reduced by a factor of $\xi \in (0, 1)$ in Line 19.

---

**Fig. 3.1.** The reduced-space projected search based on the Newton-CG direction $d_k$ described in Section 3.3. In the figure on the left, the direction $d_k$ does not intersect the ball of radius $\bar{\rho}_{k,i}$. In this case, standard backtracking is used, as indicated by the solid green dots. In the figure on the right, the direction $d_k$ does intersect the ball of radius $\bar{\rho}_{k,i}$. In this case, all points after the first point of intersection (indicated by hollow green circles) are projected to zero. Once the backtracking points leave the ball of radius $\bar{\rho}_{k,i}$ (indicated as solid green dots), standard backtracking is resumed.
Algorithm 3.1 FaRSA-Group for solving problem (1.1).

1. **Input:** $x_0$
2. **Constants:** \{ϕ, ξ, η, ζ\} ⊂ (0, 1), \{κ_1, κ_2, p\} ⊂ (0, ∞), \θ ∈ (0, π/2), and q ∈ [1, 2].
3. Choose any initial PG parameter $\alpha_0 ∈ (0, 1]$.
4. **for** $k = 0, 1, 2, \ldots$ **do**
5. Compute the step $s_k$ from (3.1).
6. Compute the set $\bar{I}_k^{cg}$ from (3.2).
7. Compute $\bar{I}_k^{cg}$ and $\bar{I}_k^{pg}$ and their optimality measures $\chi_{k}^{cg}$ and $\chi_{k}^{pg}$ from (3.4).
8. **if** $\chi_{k}^{pg} ≤ \chi_{k}^{cg}$ **then**
9. Choose any $I_k ⊆ I_k^{cg}$ such that

\[
||s_k||_2 ≥ ϕ||s_k||_{I_k^{cg}}||_2 ≡ ϕχ_{k}^{cg}
\]

and $I_k$ is the union of some \{G_j\}.

10. Set $g_k ← \nabla f(x_k)$ and pick a positive-definite $H_k ∈ ℝ_{|I_k^c|×|I_k^c|}$.
11. Call Algorithm 3.2 to obtain $\bar{d}_k ← \text{CG}_\text{DIRECTION}(g_k, H_k)$.
12. Set $[d_k]_{I_k} ← \bar{d}_k$ and $[d_k]_{I_k^c} ← 0$.
13. Call Algorithm 3.3 to obtain $(x_{k+1}, \text{flag}_{k}^{cg}) ← \text{UPDATE}_\text{CG}(x_k, d_k, I_k)$.
14. Set $\alpha_{k+1} ← \alpha_k$.
15. **else**
16. Choose any $I_k ⊆ I_k^{cg}$ such that

\[
||s_k||_{I_k}||_2 ≥ ϕ||s_k||_{I_k^{cg}}||_2 ≡ ϕχ_{k}^{pg}
\]

and $I_k$ is the union of some \{G_j\}.

17. Call Algorithm 3.4 to obtain $(x_{k+1}, \text{flag}_{k}^{pg}) ← \text{UPDATE}_\text{PG}(x_k, s_k, \alpha_k, I_k)$.
18. **if** $\text{flag}_{k}^{pg} = \text{decrease}$ **then**
19. $\alpha_{k+1} ← ζ\alpha_k$.
20. **else**
21. $\alpha_{k+1} ← \alpha_k$.

**Algorithm 3.2** Computing $\bar{d}_k$ in Line 11 of Algorithm 3.1.

22. **procedure** $\bar{d}_k ← \text{CG}_\text{DIRECTION}(g_k, H_k)$
23. **Constant:** $q$ is provided by Algorithm 3.1.
24. Define the model

\[
m_k(d) := g_k^T d + \frac{1}{2} d^T H_k d.
\]
25. Compute the reference direction (an approximate minimizer of $m_k$) as

\[
d_k^R ← −\beta_k g_k, \text{ where } \beta_k ← ||g_k||^2_2/(g_k^T H_k g_k).
\]
26. Choose $\mu_k ∈ (0, 1]$ and then compute any $\bar{d}_k ≈ \text{argmin}_d m_k(d)$ that satisfies

\[
g_k^T \bar{d}_k ≤ g_k^T d_k^R,
\]
\[
m_k(\bar{d}_k) ≤ m_k(0), \text{ and }
\]
\[
||H_k \bar{d}_k + g_k||_2 ≤ \mu_k ||g_k||^2_2.
\]
27. **return** $\bar{d}_k$
Algorithm 3.3 Computing $x_{k+1}$ in Line 13 of Algorithm 3.1.

| Line | Code |
|------|------|
| 28:  | `procedure (x_{k+1}, flag^c_k) = UPDATE.CG(x_k, d_k, \bar{I}_k)` |
| 29:  | **Constants:** $\eta$, $\xi$, and $\theta$ provided by Algorithm 3.1. |
| 30:  | **for each** $i$ **such that** $G_i \subseteq I_k$ **do** |
| 31:  | Compute $\bar{p}_{k,i}$ as defined in (3.5). |
| 32:  | Set $\tilde{p}_{k,i} \leftarrow \min\{\bar{p}_{k,i}, \sin(\theta)\|x_k g_i\|_2\}$. |
| 33:  | if $\{[x_k + \tau d_k g_i : \tau > 0] \cap \{x \in \mathbb{R}^{\|g_i\|} : \|x\|_2 \leq \tilde{p}_{k,i}\} = \emptyset$ then |
| 34:  | Set $\tau_{k,i} \leftarrow \infty$. |
| 35:  | **else** |
| 36:  | Set $\tau_{k,i}$ as the smallest positive root of $\|x_k + \tau d_k g_i\|_2 = \tilde{p}_{k,i}$. |
| 37:  | Set $j \leftarrow 0$ and $\tau_k := \min_i \{\tau_{k,i} : G_i \subseteq I_k\}$. |
| 38:  | **while** $\xi^j \geq \tau_k$ **do** |
| 39:  | **for each** $i$ **such that** $G_i \in I_k$ **do** |
| 40:  | Set $[y_j]_k \leftarrow [x_k]_I\xi^j$. |
| 41:  | if $f(y_j) + r(y_j) \leq f(x_k) + r(x_k)$ then |
| 42:  | return $x_{k+1} \leftarrow y_j$ and $\text{flag}^c_k \leftarrow \text{new_zero}$ |
| 43:  | Set $j \leftarrow j + 1$. |
| 44:  | **loop** |
| 45:  | Set $y_j \leftarrow x_k + \xi^j d_k$. |
| 46:  | if $f(y_j) + r(y_j) \leq f(x_k) + r(x_k) + \eta \xi^j \nabla x_k (f + r)(x_k) + \eta \xi^j \nabla x_k (f + r)(x_k)^T d_k \rightleftharpoons I_k$ then |
| 47:  | return $x_{k+1} \leftarrow y_j$ and $\text{flag}^c_k \leftarrow \text{suff.descent}$ |
| 48:  | Set $j \leftarrow j + 1$. |

Algorithm 3.4 Computing $x_{k+1}$ in Line 17 of Algorithm 3.1.

| Line | Code |
|------|------|
| 50:  | `procedure (x_{k+1}, flag^p_k) = UPDATE.PG(x_k, s_k, \alpha_k, \bar{I}_k)` |
| 51:  | **Constants:** $\eta$ and $\xi$ provided by Algorithm 3.1. |
| 52:  | Set $j \leftarrow 0$ and $y_0 \leftarrow x_k + P_{\bar{I}_k}(s_k)$. |
| 53:  | **while** $f(y_j) + r(y_j) > f(x_k) + r(x_k) + \eta \xi^j \frac{1}{\alpha} \|P_{\bar{I}_k}(s_k)\|_2^2$ **do** |
| 54:  | Set $j \leftarrow j + 1$ and then $y_j \leftarrow x_k + \xi^j P_{\bar{I}_k}(s_k)$. |
| 55:  | if $j = 0$ then |
| 56:  | return $x_{k+1} \leftarrow y_j$ and $\text{flag}^p_k \leftarrow \text{same._alpha}$ |
| 57:  | else |
| 58:  | return $x_{k+1} \leftarrow y_j$ and $\text{flag}^p_k \leftarrow \text{decrease._alpha}$ |

4. Analysis. Our analysis considers worst-case complexity (Section 4.1) and local convergence (Section 4.2) properties of Algorithm 3.1. To identify an approximate solution to problem (1.1), we use the measure $\max\{\lambda_k^{pg}, \lambda_k^{cg}\}$, as we now justify.

**Lemma 4.1.** Let $K \subseteq \mathbb{N}$ be such that $\lim_{k \in K} x_k = x_*$ and $\lim_{k \in K} \alpha_k = \alpha_*, \alpha > 0$. Then, $x_*$ is a solution to problem (1.1) if and only if $\lim_{k \in K} \max\{\lambda_k^{pg}, \lambda_k^{cg}\} = 0$.

**Proof.** First, we may apply [8, Theorem 3.2.8], with the choice $y = (\bar{x}, \bar{\alpha})$ and the set map $\mathcal{C}(y) = \mathbb{R}^n$, to the objective function appearing in (2.1) to conclude that $T(\bar{x}, \bar{\alpha})$ is continuous on $\mathbb{R}^n \times (0, \infty)$. Combining this property with the definition of $T$ in (2.1) and the assumption that $\lim_{k \in K} (x_k, \alpha_k) = (x_*, \alpha_*)$ with $\alpha_*, \alpha > 0$ shows
that \( \lim_{k \in K} s_k = \lim_{k \in K} (T(x_k, \alpha_k) - x_k) = T(x_*, \alpha_*) - x_* \). It follows from this limit and the fact that Assumption 1.1 and [2, Theorem 10.7] together show that \( x_* \) is a solution to problem (1.1) if and only if \( T(x_*, \alpha_*) = x_* \).

If \( \max\{\chi_k^{\text{cr}}, \chi_k^{\text{pg}}\} = 0 \) for some \( k \in \mathbb{N} \), then Lemma 4.1 implies that \( x_k \) is a solution to problem (1.1). Hence, all that remains is to consider the behavior of Algorithm 3.1 when an infinite number of iterations is performed. To focus on this case, we make the following assumption, which is assumed to hold throughout the rest of this section.

**Assumption 4.1.** For all iterations \( k \in \mathbb{N} \), it holds that \( \max\{\chi_k^{\text{cr}}, \chi_k^{\text{pg}}\} > 0 \).

Since our analysis considers the properties of the sequence of iterates, it is convenient to define the following partition of iterations performed by Algorithm 3.1:

\[
K^{\text{cr}} := \{ k \in \mathbb{N} : \text{Line 13 is reached during the } k\text{th iteration} \},
\]

\[
K^{\text{cr}}_0 := \{ k \in K^{\text{cr}} : \text{subroutine UPDATE\_CG returns flag}_{k}^{\text{cr}} = \text{new\_zero} \text{ in Line 13} \},
\]

\[
K^{\text{cr}}_{\text{sd}} := \{ k \in K^{\text{cr}} : \text{subroutine UPDATE\_CG returns flag}_{k}^{\text{cr}} = \text{suff\_descent} \text{ in Line 13} \},
\]

\[
K^{\text{pg}} := \{ k \in \mathbb{N} : \text{Line 17 is reached during the } k\text{th iteration} \},
\]

\[
K^{\text{pg}}_{\text{st}} := \{ k \in K^{\text{pg}} : \text{subroutine UPDATE\_PG returns flag}_{k}^{\text{pg}} = \text{same\_\alpha} \text{ in Line 17} \}, \quad \text{and}
\]

\[
K^{\text{pg}}_{\text{decrease\_\alpha}} := \{ k \in K^{\text{pg}} : \text{subroutine UPDATE\_PG returns flag}_{k}^{\text{pg}} = \text{decrease\_\alpha} \text{ in Line 17} \},
\]

so that \( K^{\text{cr}} = K^{\text{cr}}_0 \cup K^{\text{cr}}_{\text{sd}} \), \( K^{\text{pg}} = K^{\text{pg}}_{\text{st}} \cup K^{\text{pg}}_{\text{decrease\_\alpha}} \), and \( \mathbb{N} = K^{\text{cr}} \cup K^{\text{pg}} \).

Finally, we assume that the symmetric and positive-definite matrices required in Line 10 are chosen to be bounded and uniformly positive definite.

**Assumption 4.2.** The matrix sequence \( \{H_k\}_{k \in K^{\text{cr}}} \) chosen in Line 10 is bounded and uniformly positive definite. That is, there exist constants \( 0 < \mu_{\min} \leq \mu_{\max} < \infty \) such that \( \mu_{\min} \|v\|^2 \leq v^T H_k v \leq \mu_{\max} \|v\|^2 \) for all \( k \in K^{\text{cr}} \) and \( v \in \mathbb{R}^{|I_k^*|} \).

**4.1. Complexity result.** We first focus our attention on iterations in \( K^{\text{pg}} \). The next result shows that Algorithm 3.4 is well posed and that the new iterate that it produces satisfies a decrease property that will be useful for our complexity analysis.

**Lemma 4.2.** For each \( k \in K^{\text{pg}} \), Algorithm 3.4 is called in Line 17 and successfully returns \( x_{k+1} \) and \( \text{flag}_{k}^{\text{pg}} \). Moreover, the value of \( \text{flag}_{k}^{\text{pg}} \) indicates whether \( k \in K^{\text{pg}}_{\text{st}} \) or \( k \in K^{\text{pg}}_{\text{decrease\_\alpha}} \), and for these respective cases the following properties hold:

(i) If \( k \in K^{\text{pg}}_{\text{st}} \), then \( \alpha_{k+1} = \alpha_k \) and

\[
(4.1) \quad f(x_{k+1}) + r(x_{k+1}) \leq f(x_k) + r(x_k) - \frac{\eta \varphi}{\alpha_k}(\chi_k^{\text{pg}})^2.
\]

(ii) If \( k \in K^{\text{pg}}_{\text{decrease\_\alpha}} \), then \( \alpha_{k+1} = \xi \alpha_k \) and \( f(x_{k+1}) + r(x_{k+1}) < f(x_k) + r(x_k) \).

**Proof.** Since \( k \in K^{\text{pg}} \), we know that the condition tested in Line 8 of Algorithm 3.1 must not hold, meaning that \( \chi_k^{\text{pg}} > \chi_k^{\text{cr}} \). Combining this observation with Line 16 of Algorithm 3.1 shows that the set \( I_k^* \) defined in Line 16 satisfies

\[
(4.2) \quad \|P_{I_k^*}(s_k)\|_2 = \|s_k|_{I_k^*}\|_2 \geq \varphi \chi_k^{\text{pg}} > 0.
\]

Combining this result with Lemma 2.2 (using \( I = I_k^* \), \( x = x_k \), and \( \alpha = \alpha_k \)) yields

\[
(4.3) \quad D_{f+r}(x_k; P_{I_k^*}(s_k)) \leq - \frac{1}{\eta \varphi} \|P_{I_k^*}(s_k)\|_2^2 < 0.
\]

It is possible that Algorithm 3.4 terminates in Line 56 because the inequality in Line 53 does not hold for \( j = 0 \). In this case, Algorithm 3.4 successfully returns
\( x_{k+1} = y_0 = x_k + P_{\mathcal{L}_k}(s_k) \) and \( \text{flag}^{\text{PG}}_{\alpha} = \text{same}_\alpha \), also indicating that \( k \in \mathcal{K}^{\text{PG}} \). Since the while-loop in Line 53 terminates with \( j = 0 \), we can conclude that 

\[
(4.4) \quad f(x_{k+1}) + r(x_{k+1}) = f(y_0) + r(y_0) \leq f(x_k) + r(x_k) - \frac{\eta}{\eta_k} \|P_{\mathcal{L}_k}(s_k)\|^2.
\]

Combining this inequality with (4.2) shows that (4.1) holds. Finally, since \( \text{flag}^{\text{PG}}_{\alpha} = \text{same}_\alpha \), it follows from Line 21 that \( \alpha_{k+1} = \alpha_k \), completing the proof in this case.

It remains to consider the case when Algorithm 3.4 is unable to terminate in Line 56 because the inequality in Line 53 holds for \( j = 0 \). In this case, it follows from (4.3) and standard results for a backtracking Armijo line search that, for all sufficiently large \( j \), the vector \( y_j \leftarrow x_k + \xi^j P_{\mathcal{L}_k}(s_k) \) defined in Line 54 of Algorithm 3.4 satisfies 

\[
(4.5) \quad f(y_j) + r(y_j) \leq f(x_k) + r(x_k) + \eta \xi^j D_{f+r}(x_k; P_{\mathcal{L}_k}(s_k))
\]

\[
\leq f(x_k) + r(x_k) - \eta \xi^j \frac{1}{\eta_k} \|P_{\mathcal{L}_k}(s_k)\|^2.
\]

This inequality shows that the while loop starting in Line 53 of Algorithm 3.4 will terminate finitely, and thus Algorithm 3.4 successfully returns \( x_{k+1} = y_j = x_k + \xi^j P_{\mathcal{L}_k}(s_j) \) for some \( j > 0 \) and \( \text{flag}^{\text{PG}}_{\alpha} = \text{decrease}_\alpha \), also indicating that \( k \in \mathcal{K}^{\text{PG}}_{\xi} \).

Combining (4.5), \( y_j = x_{k+1} \), and (4.3) proves that \( f(x_{k+1}) + r(x_{k+1}) < f(x_k) + r(x_k) \), as claimed. Finally, since \( \text{flag}^{\text{PG}}_{\alpha} = \text{decrease}_\alpha \), we see in Line 19 that \( \alpha_{k+1} = \xi \alpha_k \).

Next, we prove that the PG parameter remains bounded away from zero.

**Lemma 4.3.** The PG parameter sequence generated by Algorithm 3.1 satisfies 

\[
(4.6) \quad 1 \geq \alpha_k \geq \alpha_{\min} := \min \{ \alpha_0, \frac{2 \xi (1-\eta)}{L} \} > 0 \quad \text{for all} \ k \in \mathbb{N}.
\]

Moreover, a bound on the number of times the PG parameter is decreased is given by 

\[
(4.7) \quad |\mathcal{K}^{\text{PG}}_{\xi}| \leq \xi^{\alpha_0} := \max \left\{ 0, \left\lfloor \log \left( \frac{\alpha_0 L}{2 \xi (1-\eta)} \right)/ \log(\xi^{-1}) \right\rfloor \right\}.
\]

**Proof.** We first prove (4.6). Since \( \alpha_0 \in (0,1] \) in Line 3 and \( \alpha_{k+1} \leq \alpha_k \) for all \( k \in \mathbb{N} \), we need only prove the lower bound on \( \alpha_k \) in (4.6). With that goal in mind, for the purpose of obtaining a contradiction, suppose that there exists an iteration \( k \) satisfying \( \alpha_k \leq 2(1-\eta)/L < 2/L \), with the latter inequality holding since \( \eta \in (0,1) \).

First suppose that \( k \in \mathcal{K}^{\text{PG}} \). With \( y_0 = x_k + P_{\mathcal{L}_k}(s_k) \) as defined in Line 52 of Algorithm 3.4, it follows from Lemma 2.3 with \( \bar{x} = x_k, \bar{\alpha} = \alpha_k \), and \( s(\bar{x}, \bar{\alpha}) = s_k \) that 

\[
(4.8) \quad f(y_0) + r(y_0) \leq f(x_k) + r(x_k) - \frac{\eta}{\eta_k} \|P_{\mathcal{L}_k}(s_k)\|^2
\]

\[
\leq f(x_k) + r(x_k) - \frac{1}{\eta_k} \frac{2(1-\eta)}{2 \xi L} \|P_{\mathcal{L}_k}(s_k)\|^2
\]

\[
= f(x_k) + r(x_k) - \frac{\eta}{\eta_k} \|P_{\mathcal{L}_k}(s_k)\|^2.
\]

This inequality implies that the condition checked in Line 53 for \( j = 0 \) will not hold, meaning that \( j = 0 \) when Line 55 is reached so that \( \text{flag}^{\text{PG}}_{\alpha} \leftarrow \text{same}_\alpha \) in Line 56. Thus, when Line 18 in Algorithm 3.1 is reached, the update \( \alpha_{k+1} \leftarrow \alpha_k \) will take place. Second, if \( k \in \mathcal{K}^{\text{PG}} \), then Algorithm 3.1 sets \( \alpha_{k+1} \leftarrow \alpha_k \). To summarize, anytime \( \alpha_k \leq 2(1-\eta)/L \), the update \( \alpha_{k+1} \leftarrow \alpha_k \) takes place. Combining this property with the fact that when the PG parameter is decreased the update \( \alpha_{k+1} \leftarrow \xi \alpha_k \) is used (see Line 19 in Algorithm 3.1), shows that (4.6) holds.

We now prove (4.7). Let us observe from the first paragraph in this proof that if \( \alpha_0 \leq 2(1-\eta)/L \) then \( |\mathcal{K}^{\text{PG}}_{\xi}| = 0 \), which verifies that (4.7) holds. Therefore, for \( \alpha_0 \in (0,1\) and \( \alpha_k \leq 2(1-\eta)/L \), the update \( \alpha_{k+1} \leftarrow \xi \alpha_k \) takes place (see Line 19 in Algorithm 3.1), also indicating that \( k \in \mathcal{K}^{\text{PG}}_{\xi} \).
the remainder of the proof, suppose that \( \alpha_0 > 2(1 - \eta)/L \). Combining this bound with the fact that when the PG parameter is decreased the update \( \alpha_{k+1} \leftarrow \xi \alpha_k \) is used, we can see that an upper bound on \( |K_{\text{PG}}| \) is the smallest integer \( \ell \) such that \( \alpha_0 \xi^\ell \leq 2(1 - \eta)/L \). Solving this inequality for \( \ell \) shows that the result in (4.7) holds. \( \Box \)

We now switch our attention to iterations in \( K_{\text{CG}} \). The next result establishes that Algorithm 3.2 is well posed, and that the direction \( d_k \) that results from it when called by Algorithm 3.1 satisfies a certain descent property.

**Lemma 4.4.** For each \( k \in K_{\text{CG}} \), Algorithm 3.2 is well posed. Moreover, the resulting direction \( d_k \), which is used to compute \( d_k \) in Line 12, guarantees that \( d_k \) satisfies

(i) \( \nabla I_k (f + r)(x_k)^T [d_k] I_k \leq -\frac{1}{\mu_{\max}} \| \nabla I_k (f + r)(x_k) \|^2_2 < 0 \), and

(ii) \( \| d_k \|_2 \leq (2/\mu_{\min}) \| \nabla I_k (f + r)(x_k) \|_2 \)

where \( I_k \subseteq I_{k_{\text{CG}}} \) is the set in Line 9 used as an input to Algorithm 3.2 in Line 13.

**Proof.** Since \( k \in K_{\text{CG}} \), Algorithm 3.2 is called in Line 11 with input \( I_k \) defined in Line 9. We first prove that \( g_k = \nabla I_k (f + r)(x_k) \), as defined in Line 10, is nonzero. For a proof by contradiction, suppose that \( g_k = 0 \) so that \( \nabla I_k (f + r)(x_k) = 0 \) for all \( i \) such that \( G_i \subseteq I_k \). Consider arbitrary such \( i \). Note that \( [x_k]_{G_i} \neq 0 \) and \( [x_k + s_k]_{G_i} \neq 0 \) since \( G_i \subseteq I_k \subseteq \bar{T}_{k_{\text{CG}}} \) (see Line 9) and by how \( \bar{T}_{k_{\text{CG}}} \) is defined. This allows us to conclude from Lemma 2.4 that \( [s_k]_{G_i} = 0 \), i.e., that \( [s_k]_{I_k} = 0 \) since \( i \) with \( G_i \subseteq I_k \) was arbitrary. This fact and Line 9 yields \( \chi_k^p = 0 \), but since the inequality in Line 8 must hold, we also have \( \chi_k^p = 0 \). This contradicts Assumption 4.1, thus establishing that \( g_k \neq 0 \).

Now, it follows from Lines 10, 12, 26, and 25, \( g_k \neq 0 \), and Assumption 4.2 that

\[
\nabla I_k (f + r)(x_k)^T [d_k] I_k \equiv g_k^T d_k \leq g_k^T d_k^{R_{\text{CG}}} = -\beta_k \| g_k \|^2_2
\]

\[
= -\| g_k \|^2_2/(g_k^T H_k g_k) \leq -\frac{1}{\mu_{\max}} \| g_k \|^2_2.
\]

The result in (i) follows from this inequality and \( g_k = \nabla I_k (f + r)(x_k) \neq 0 \).

Part (ii) is precisely [5, Lemma 3.8] under our Assumption 4.2 since our conditions placed upon the step \( d_k \) are exactly the same as those used in [5]. \( \Box \)

The next lemma shows that, for \( k \in K_{\text{CG}} \), a local Lipschitz property holds along a certain portion of the search path defined by the reduced-space Newton-CG direction.

**Lemma 4.5.** Let \( k \in K_{\text{CG}} \) so that \( I_k \) is computed in Line 9. The following hold:

(i) The constant \( \theta \in (0, \pi/2) \) and index set \( I_k \) passed into Algorithm 3.3 satisfy, for each \( i \) such that \( G_i \subseteq I_k \) with \( \rho_{k,i} \) computed in (3.5) and \( \bar{\rho}_{k,i} \) computed in Line 32, the following conditions:

(a) \( \| [x_k + s_k]_{G_i} \|_2 \neq 0 \),

(b) \( \| [x_k]_{G_i} \|_2 \geq [\rho_{k,i}] \geq \bar{\rho}_{k,i} \geq \sin(\theta) \rho_{k,i} > 0 \), and

(c) \( \| [x_k]_{G_i} \|_2 - \bar{\rho}_{k,i} \geq \kappa_2 (1 - \sin(\theta)) \| \nabla I_k (f + r)(x_k) \|^2_2 \).

(ii) For all step sizes \( \beta \in [0, \tau_k] \) with \( \tau_k \) computed in Line 37, it holds, with

(4.8) \( \lambda_{\max} := \max \{ \lambda_1, \lambda_2, \ldots, \lambda_{n_{\text{CG}}} \} \) and \( \rho_{k,\min} := \min_i \{ \rho_{k,i} : G_i \subseteq I_k \} \)

that \( \| \nabla I_k (f + r)(x_k) - \nabla I_k (f + r)(x_k + \beta d_k) \|_2 \leq \beta(L + \lambda_{\max}/\rho_{k,\min}) \| [d_k] I_k \|_2 \).

**Proof.** We first prove part (i). Consider arbitrary \( i \) with \( G_i \subseteq I_k \), where \( I_k \subseteq \bar{T}_{k_{\text{CG}}} \) is passed into Algorithm 3.3 and constructed to satisfy the condition in Line 9. Part (a) follows from \( \bar{T}_{k_{\text{CG}}} \subseteq \bar{T}_{k_{\text{CG}}} \) and the definition of \( \bar{T}_{k_{\text{CG}}} \) in (3.2). The first inequality in part (b) follows from \( \bar{T}_{k_{\text{CG}}} \subseteq \bar{T}_{k_{\text{CG}}} \), and how \( \bar{T}_{k_{\text{CG}}} = \bar{T}_{k_{\text{CG}}} \) and \( \bar{T}_{k_{\text{CG}}} \) are defined. The second inequality in (b) follows from how \( \bar{\rho}_{k,i} \) is defined in Line 32. The third inequality in
(b) follows from Line 32 and the first inequality in (b). To complete the proof for part (b), we must prove that $\rho_{k,i} > 0$. For a proof by contradiction, assume that $\rho_{k,i} = 0$, which by (3.5) means that $\|\nabla_{x_k}^G(f + r)(x_k)\|_2 = 0$. It follows from this fact that each $i$ with $G_i \subseteq I_k \subseteq I_k^c$ satisfies $\nabla_{x_k}^i(f + r)(x_k)\|_2 = 0$, which in light of Lemma 2.4 (using $\bar{x} = x_k$, $\bar{\alpha} = \alpha_k$, and $s(\bar{x}, \bar{\alpha}) = s_k$) and the definition of $I_k^c$ implies that $\|s_k|_{I_k}\|_2 = 0$ for each $G_i \subseteq I_k$, i.e., that $\|\nabla I_{x_k}(f + r)(x_k)\|_2 = 0$. It now follows from Line 9 that $\chi^k_c = 0$, which combined with the inequality in Line 8 shows that $\chi^k_{c^r} = 0$. Since we have reached a contradiction to Assumption 4.1, we must conclude that $\rho_{k,i} > 0$, as claimed. Finally, we aim to prove part (c). It follows from Line 32, $\theta \in (0, \pi/2)$, part (b), (3.5), and the fact that $I_k \subseteq I_k^c$ that

$$
\|x_k|_{G_i}\| - \bar{\rho}_{k,i} \geq \|x_k|_{\bar{G}_i}\| - \sin(\theta)\|x_k|_{\bar{G}_i}\| = (1 - \sin(\theta))\|x_k|_{\bar{G}_i}\|
$$

which completes the proof of part (c).

To prove part (ii), let $\beta \in (0, \tau_k)$. It follows from part (i) and the definition of $\tau_k$ in Line 37 that every point on the segment that connects $[x_k|_{\bar{G}_i}]$ to $[x_k + \beta d_k|_{\bar{G}_i}]$ is outside of the ball in $R|_{\bar{G}_i}$ centered at zero of radius $\bar{\rho}_{k,i} > 0$. This means that both $\|x_k|_{\bar{G}_i}\| \geq \bar{\rho}_{k,i}$ and $\|x_k + \beta d_k|_{\bar{G}_i}\| \geq \bar{\rho}_{k,i}$. It now follows that

$$
\|\nabla_{x_k} G_i r(x_k) - \nabla_{x_k} G_i r(x_k + \beta d_k)\|_2
$$

(4.9)

$$
= \lambda_i \|x_k|_{G_i}\|_2 - \|x_k + \beta d_k|_{G_i}\|_2 = \lambda_i \|x_k|_{\bar{G}_i}\|_2 - \|x_k + \beta d_k|_{\bar{G}_i}\|_2 \leq \lambda_i \|d_k|_{G_i}\|_2,
$$

where the (only) inequality follows from the nonexpansive property of the projection (of $\|x_k|_{\bar{G}_i}$ and $\|x_k + \beta d_k|_{\bar{G}_i}\|$) onto the ball of radius $\bar{\rho}_{k,i}$. From (4.9) we have

$$
\|\nabla_{x_k} G_i r(x_k) - \nabla_{x_k} G_i r(x_k + \beta d_k)\|_2
$$

(4.10)

$$
\leq \sum_{i \bar{G}_i \subseteq I_k} \|x_k|_{\bar{G}_i}\|_2 \leq \beta^2 \sum_{i \bar{G}_i \subseteq I_k} \|d_k|_{G_i}\|_2
$$

It follows from Assumption 1.1, $d_k|_{I_k} = 0$, the triangle inequality, and (4.10) that

$$
\|\nabla_{x_k} G_i r(x_k) - \nabla_{x_k} G_i r(x_k + \beta d_k)\|_2
$$

(4.11)

$$
\leq \|\nabla_{x_k} G_i r(x_k) - \nabla_{x_k} G_i r(x_k + \beta d_k)\|_2 + \|\nabla_{x_k} G_i r(x_k + \beta d_k)\|_2
$$

$$
\leq \sum_{i \bar{G}_i \subseteq I_k} \|d_k|_{G_i}\|_2 = \beta \sum_{i \bar{G}_i \subseteq I_k} \|d_k|_{I_k}\|_2
$$

which completes the proof. \qed

We now show that Algorithm 3.4 is well posed and that the new iterate it produces satisfies a decrease property that will be used in the final complexity result.

Lemma 4.6. For each $k \in K^c$, Algorithm 3.3 is called in Line 13 and successfully returns $x_{k+1}$ and $\text{flag}_{k+1}^c$. Moreover, the value of $\text{flag}_{k+1}^c$ indicates whether $k \in K^c = 0$ or $k \in K^c = 0$, and for these respective cases the following properties hold:
(i) If $k \in \mathcal{K}_0^{cg}$, then $f(x_{k+1}) + r(x_{k+1}) \leq f(x_k) + r(x_k)$ and $x_{k+1}$ has at least one additional block of zeros compared to $x_k$.

(ii) If $k \in \mathcal{K}_d^{cg}$, then

\begin{equation}
4.11 \quad f(x_{k+1}) + r(x_{k+1}) \leq f(x_k) + r(x_k) - \min\{c_1(\lambda_k^{cg})^{1+p}, c_2(\lambda_k^{cg})^{2+p}\}
\end{equation}

where

\begin{align}
4.12 \quad c_1 &:= \frac{\eta \xi \mu_{\text{min}} \kappa_2 (1 - \sin(\theta)) \varphi^{1+p}}{2 \mu_{\text{max}}} > 0 \quad \text{and} \\
4.12 \quad c_2 &:= \frac{\kappa_2 \mu_{\text{min}}^2 \xi (1 - \eta) \varphi^{2+p}}{2 \mu_{\text{max}}^2} \left(L \kappa_2 (L_f + \lambda_{\text{max}} \sqrt{\kappa_0})^p + \lambda_{\text{max}}\right) > 0.
\end{align}

Proof. Throughout, we use $F := f + r$. It is possible that Algorithm 3.3 successfully terminates in Line 43, in which case it follows from Line 43 and Line 42 that the returned $x_{k+1}$ and flag$^{cg}_k$ satisfy $F(x_{k+1}) \leq F(x_k)$ and flag$^{cg}_k = \text{new\_zero}$, indicating that $k \in \mathcal{K}_d^{cg}$. Moreover, upon termination, the value $j$ satisfies $\xi^j \geq \tau_k$ (see Line 38), which combined with Line 41 shows that at least one additional group of variables has become zero at $x_{k+1}$. This proves that part (i) holds.

Next, suppose that Algorithm 3.3 does not terminate in Line 43. Observe from the definition of $\tau_k$ in Line 37 that $\tau_k > 0$ (this follows from Lemma 4.5(i) and the definition of $\rho_k$). Therefore, it follows that the while loop starting in Line 38 will terminate with the smallest nonnegative integer $j$ such that $\xi^j < \tau_k$, and the loop in Line 45 will begin with $j = j$. We now claim that the condition in Line 47 used to determine termination of the loop is satisfied for all $j \geq j$ such that

\begin{equation}
4.13 \quad \xi^j \in \left[0, \frac{2(\eta - 1) \nabla I_k F(x_k)^T[d_k]_{I_k}}{L + \lambda_{\text{max}}/\rho_{k, \text{min}}} \right] \subset [0, \tau_k).
\end{equation}

To see that this claim holds, we can use the integral form of Taylor’s Theorem and Lemma 4.5(ii) (using the fact that $\gamma \xi^j \in [0, \tau_k]$ for all $\gamma \in [0, 1]$) to obtain

\begin{align}
4.14 \quad |F(x_k + \xi^j d_k) - F(x_k) - \xi^j \nabla I_k F(x_k)^T[d_k]_{I_k}| &
\leq \left| \int_0^1 \xi^j [d_k]_{I_k}^T \left( \nabla I_k F(x_k + \gamma \xi^j d_k) - \nabla I_k F(x_k) \right) \gamma d\gamma \right|
\leq \xi^j \int_0^1 \left\| [d_k]_{I_k} \right\|_2 \left\| \nabla I_k F(x_k + \gamma \xi^j d_k) - \nabla I_k F(x_k) \right\|_2 d\gamma
\leq \xi^j (L + \lambda_{\text{max}}/\rho_{k, \text{min}}) \left\| [d_k]_{I_k} \right\|_2 \int_0^1 \gamma d\gamma = \frac{1}{2} \xi^j (L + \lambda_{\text{max}}/\rho_{k, \text{min}}) \left\| [d_k]_{I_k} \right\|_2^2.
\end{align}

Combining this inequality with (4.13) yields

\begin{align}
4.15 \quad F(x_k + \xi^j d_k) &\leq F(x_k) + \xi^j \nabla I_k F(x_k)^T[d_k]_{I_k} + \frac{1}{2} \xi^j (L + \lambda_{\text{max}}/\rho_{k, \text{min}}) \left\| [d_k]_{I_k} \right\|_2^2
\quad = F(x_k) + \xi^j \nabla I_k F(x_k)^T[d_k]_{I_k} + \xi^j (\eta - 1) \nabla I_k F(x_k)^T[d_k]_{I_k}
\quad = F(x_k) + \eta \xi^j \nabla I_k F(x_k)^T[d_k]_{I_k},
\end{align}

which establishes our claim that the inequality in Line 47 holds for all $j \geq j$ such that $\xi^j$ satisfies (4.13). This shows that the loop will successfully terminate with flag$^{cg}_k = \text{suff\_descent}$ (thus indicating that $k \in \mathcal{K}_d^{cg}$) and $x_{k+1}$ satisfying

\begin{equation}
4.16 \quad F(x_{k+1}) \leq F(x_k) + \eta \xi^j \nabla I_k F(x_k)^T[d_k]_{I_k}
\end{equation}
for some \( \hat{j} \) satisfying
\[
x_{\hat{j}} \geq \min \left\{ \xi_{\hat{j}} \frac{2 \xi (\eta - 1) \nabla I_k F(x_k)^T [d_k] I_k}{(L + \lambda_{\max}/\rho_{k,\min}) \| [d_k] I_k \|^2_2} \right\}
\]
(4.15)
\[
\geq \min \left\{ \xi \tau_k, \frac{2 \xi (\eta - 1) \nabla I_k F(x_k)^T [d_k] I_k}{(L + \lambda_{\max}/\rho_{k,\min}) \| [d_k] I_k \|^2_2} \right\}
\]
where the second inequality follows from the fact that \( \tilde{\mathcal{J}} \) is the smallest nonnegative integer such that \( \xi_{\tilde{\mathcal{J}}} < \tau_k \). We now consider two cases.

\textbf{Case 1:} the minimum in (4.15) is \( \xi \tau_k \), from which we may conclude that \( \tau_k < \infty \).

Using (4.14) and Lemma 4.4(i) we have that
\[
(4.16) \quad F(x_{k+1}) \leq F(x_k) + \eta \xi \nabla I_k F(x_k)^T [d_k] I_k \leq F(x_k) - \frac{\eta \xi}{\mu_{\max}} \tau_k \| \nabla I_k F(x_k) \|^2_2.
\]
We now seek a lower bound on \( \tau_k \). Consider \( i \) such that \( \tau_{k,i} < \infty \) when computed in Algorithm 3.3. The triangle inequality gives \( \tilde{\rho}_{k,i} = \| x_k + \tau_{k,i} d_k \|_2 \geq \| x_k \|_2 - \tau_{k,i} \| d_k \|_2 \), which together with Lemma 4.5(i)(c) and Lemma 4.4(ii) shows that
\[
\tau_{k,i} \geq \frac{\mu_{\min} \kappa_2 (1 - \sin(\theta)) \| \nabla I_k F(x_k) \|^p_2}{2 \| \nabla I_k F(x_k) \|^2_2} = \frac{1}{2} \mu_{\min} \kappa_2 (1 - \sin(\theta)) \| \nabla I_k F(x_k) \|^p_2 - 1.
\]
From this, it follows that \( \tau_k \geq \frac{1}{2} \mu_{\min} \kappa_2 (1 - \sin(\theta)) \| \nabla I_k F(x_k) \|^p_2 - 1 \). Using this inequality with (4.16), Lemma 2.4, and the set \( \mathcal{I}_k \) from Line 9 shows that
\[
F(x_{k+1}) \leq F(x_k) - \frac{\eta \xi \mu_{\min} \kappa_2 (1 - \sin(\theta)) \| \nabla I_k F(x_k) \|^1_2 + p}{2 \mu_{\max}}
\]
(4.17)
\[
\leq F(x_k) - \frac{\eta \xi \mu_{\min} \kappa_2 (1 - \sin(\theta)) \| s_k \|_1^2 + p}{2 \mu_{\max}}
\]
\[
\leq F(x_k) - \frac{\eta \xi \mu_{\min} \kappa_2 (1 - \sin(\theta)) 2^{1+p} \phi^{1+p}}{2 \mu_{\max}} (\lambda_{k}^{\infty})^{1+p},
\]
thus completing the proof for this case.

\textbf{Case 2:} the minimum in (4.15) is \( \frac{2 \xi (\eta - 1) \nabla I_k F(x_k)^T [d_k] I_k}{(L + \lambda_{\max}/\rho_{k,\min}) \| [d_k] I_k \|^2_2} \). Combining this fact with (4.14), (4.15), Lemma 4.4(i), and Lemma 4.4(ii) shows that
\[
F(x_{k+1}) \leq F(x_k) + \frac{2 \xi (1 - \eta) \| \nabla I_k F(x_k) \|_2^4}{\mu_{\max}^2 (L + \lambda_{\max}/\rho_{k,\min}) \| [d_k] I_k \|^2_2}
\]
\[
\leq F(x_k) + \frac{4 \mu_{\max}^2 \xi (1 - \eta) \| \nabla I_k F(x_k) \|_2^4}{4 \mu_{\max}^2 (L + \lambda_{\max}/\rho_{k,\min}) \| \nabla I_k F(x_k) \|^2_2}
\]
\[
= F(x_k) + \frac{\mu_{\min}^2 \xi (1 - \eta) \| \nabla I_k F(x_k) \|^2_2}{2 \mu_{\max}^2 (L + \lambda_{\max}/\rho_{k,\min})}.
\]
It follows from (4.8), (3.5), and \( \mathcal{I}_k \subseteq \mathcal{I}_k^{\infty} \) that \( \rho_{k,\min} \geq \kappa_2 \| \nabla I_k F(x_k) \|^2_2 \). Combining
this bound with (4.17) shows that
\[
F(x_{k+1}) \leq F(x_k) - \frac{\mu_{\min}^2 \xi \eta (1 - \eta) \|
abla_{\mathcal{I}_k} F(x_k)\|^2}{2 \mu_{\max}^2 (L + \lambda_{\max}/\rho_{\min})} \\
\leq F(x_k) - \frac{\mu_{\min}^2 \xi \eta (1 - \eta) \|
abla_{\mathcal{I}_k} F(x_k)\|^2}{2 \mu_{\max}^2 (L + \lambda_{\max}/(\kappa_2 \|
abla_{\mathcal{I}_k} F(x_k)\|^2))} \\
= F(x_k) - \frac{\kappa_2 \mu_{\min}^2 \xi \eta (1 - \eta) \|
abla_{\mathcal{I}_k} F(x_k)\|_{2+p}}{2 \mu_{\max}^2 (L \kappa_2 \|
abla_{\mathcal{I}_k} F(x_k)\|_2^p + \lambda_{\max})}.
\]

Next, we know from Lemma 4.2, Lemma 4.6(i), and equations (4.16) and (4.18) that
\[
F(x_k) \leq F(x_0) \quad \text{for all } k \in \mathbb{N}, \text{ i.e., } x_k \in \mathcal{L} \text{ for all } k \in \mathbb{N}.\]
Combining this fact with the triangle inequality, Assumption 1.1, the definition of \( r \), and (4.8) gives
\[
\|
abla_{\mathcal{I}_k} F(x_k)\|_2 \leq \|
abla_{\mathcal{I}_k} f(x_k)\|_2 + \|
abla_{\mathcal{I}_k} r(x_k)\|_2 \\
= \|
abla_{\mathcal{I}_k} f(x_k)\|_2 + \left( \sum_{i : G_i \subseteq \mathcal{I}_k} \|
abla_{G_i} r(x_k)\|_2^2 \right)^{1/2} \\
\leq L_f + \left( \sum_{i : G_i \subseteq \mathcal{I}_k} \lambda_i^2 \right)^{1/2} \leq L_f + \left( \sum_{i : G_i \subseteq \mathcal{I}_k} \lambda_i^2 \right)^{1/2} \leq L_f + \lambda_{\max}\sqrt{\kappa}. \\
\]
Combining this with (4.18) gives
\[
F(x_{k+1}) \leq F(x_k) - \left( \frac{\kappa_2 \mu_{\min}^2 \xi \eta (1 - \eta)}{2 \mu_{\max}^2 (L \kappa_2 (L_f + \lambda_{\max}\sqrt{\kappa})^p + \lambda_{\max})} \right) \|
abla_{\mathcal{I}_k} F(x_k)\|_{2+p}, \\
\]
which combined with Lemma 2.4 and how the index set \( \mathcal{I}_k \) in Line 9 is defined gives
\[
F(x_{k+1}) \leq F(x_k) - \left( \frac{\kappa_2 \mu_{\min}^2 \xi \eta (1 - \eta)}{2 \mu_{\max}^2 (L \kappa_2 (L_f + \lambda_{\max}\sqrt{\kappa})^p + \lambda_{\max})} \right) \|s_k\|_{\mathcal{I}_k}^{2+p} \\
\leq F(x_k) - \left( \frac{\kappa_2 \mu_{\min}^2 \xi \eta (1 - \eta) 2^{2+p}}{2 \mu_{\max}^2 (L \kappa_2 (L_f + \lambda_{\max}\sqrt{\kappa})^p + \lambda_{\max})} \right) (\lambda_k^{c_k})^{2+p}, \\
\]
thus completing the proof. \( \square \)

The result in (4.11) motivates us to define the following subsets of \( \mathcal{K}_{\mathcal{I}_k}^{\mathcal{S}} \):
\[
(4.19) \quad \mathcal{K}_{\mathcal{I}_k}^{\mathcal{S}} := \{ k \in \mathcal{K}_{\mathcal{I}_k}^{\mathcal{S}} : \lambda_k^{c_k} \geq c_1/c_2 \} \quad \text{and} \quad \mathcal{K}_{\mathcal{I}_k}^{\mathcal{S}} := \mathcal{K}_{\mathcal{I}_k}^{\mathcal{S}} \setminus \mathcal{K}_{\mathcal{I}_k}^{\mathcal{S}}, \\
\]
This distinction plays a role in our complexity result. First, we require a lemma.

**Lemma 4.7.** The objective function \( f + r \) is monotonically decreasing over the sequence of iterates \( \{x_k\} \) and \( \lim_{k \to \infty} (f(x_k) + r(x_k)) =: F_{\min} \geq -\infty. \)
\[ \]
**Proof.** It follows from Lemma 4.2 and Lemma 4.6 that the objective function is monotonically decreasing over the iterate sequence. The remaining conclusion of the lemma follows from the monotonicity property and Assumption 1.1. \( \square \)

The main theorem can now be stated. It gives an upper bound on the number of iterations performed by Algorithm 3.1 before an approximate solution is obtained.
Theorem 4.8. Let $c_1$ and $c_2$ be the constants defined in (4.12) and let us define $c_3 := \eta \varphi^2 / \alpha_0 > 0$. For any $\epsilon > 0$, define $\mathcal{K}_\epsilon := \{ k \in \mathbb{N} : \max \{ \chi^c_k, \chi^p_k \} > \epsilon \}$. Then,

$$|\mathcal{K}_\epsilon^c \cap \mathcal{K}_\epsilon| \leq c_{pg} \epsilon^{-2} + 1,$$

$$|\mathcal{K}_\epsilon^c \cap \mathcal{K}_\epsilon| \leq c_{big} \epsilon^{-(1+p)} + 1, \text{ and}$$

$$|\mathcal{K}_\epsilon^c \cap \mathcal{K}_\epsilon| \leq c_{small} \epsilon^{-(2+p)} + 1$$

where the constants $c_{pg}$, $c_{big}$, and $c_{small}$ are given, respectively, by

$$c_{pg} := (f(x_0) + r(x_0) - F_{min}) / c_3,$$

$$c_{big} := (f(x_0) + r(x_0) - F_{min}) / c_1, \text{ and}$$

$$c_{small} := (f(x_0) + r(x_0) - F_{min}) / c_2.$$

Therefore, if $\epsilon \geq c_1 / c_2$, then

$$|\mathcal{K}_\epsilon| \leq (c^c_1 \epsilon^{-2} + c_{pg} \epsilon^{-(1+p)} + 2)(1 + n_G) + n_G$$

where $c^c_1$ is defined in (4.7); otherwise, i.e., if $\epsilon < c_1 / c_2$, then

$$|\mathcal{K}_\epsilon| \leq (c^c_1 \epsilon^{-2} + c_{big} \epsilon^{-(1+p)} + c_{small} \epsilon^{-(2+p)} + 3)(1 + n_G) + n_G.$$

Proof. Note that the definitions of $\mathcal{K}_\epsilon^c$ and $\mathcal{K}_\epsilon^p$ together with Line 8 show that

$$\chi^{pg}_k \geq \chi^{cg}_k \text{ for } k \in \mathcal{K}_\epsilon^c \text{ and } \chi^{pg}_k > \chi^{cg}_k \text{ for } k \in \mathcal{K}_\epsilon^p.$$

Define $\Delta_k := f(x_k) + r(x_k) - (f(x_{k+1}) + r(x_{k+1}))$, and $m_k := \max \{ \chi^p_k, \chi^c_k \}$. Using Lemma 4.2(i), Lemma 4.3, Lemma 4.6(ii), the definitions of $c_3$ and $\mathcal{K}_\epsilon$ in the statement of the theorem, and (4.24) shows for arbitrary $k \in \mathbb{N}$ that

$$f(x_0) + r(x_0) - (f(x_{k+1}) + r(x_{k+1})) = \sum_{0 \leq k \leq k} \Delta_k$$

$$\geq \sum_{k \in \mathcal{K}_\epsilon^p \cap \mathcal{K}_\epsilon} \Delta_k + \sum_{k \in \mathcal{K}_\epsilon^c \cap \mathcal{K}_\epsilon} \Delta_k + \sum_{k \in \mathcal{K}_\epsilon^c \cap \mathcal{K}_\epsilon} \Delta_k$$

$$\geq \sum_{k \in \mathcal{K}_\epsilon^p \cap \mathcal{K}_\epsilon} c_3 (\chi^c_k)^2 + \sum_{k \in \mathcal{K}_\epsilon^c \cap \mathcal{K}_\epsilon} c_1 (\chi^c_k)^{1+p} + \sum_{k \in \mathcal{K}_\epsilon^c \cap \mathcal{K}_\epsilon} c_2 (\chi^c_k)^{2+p}$$

From this inequality, Lemma 4.2, and (4.21) one finds that (4.20) follows.

Next, suppose that $\epsilon \geq c_1 / c_2$. It then follows from (4.19) and (4.24) that $\chi^{cg}_k = \max \{ \chi^p_k, \chi^c_k \} > \epsilon \geq c_1 / c_2$ for all $k \in \mathcal{K}_\epsilon^c$, which implies that $\mathcal{K}_\epsilon^c \cap \mathcal{K}_\epsilon = \emptyset$. The result in (4.22) follows from this observation, (4.20), (4.7), and since (by Lemma 4.6(i)) at most $n_G$ iterations in $\mathcal{K}^c_0$ can occur before the first, after the last, or between any two iterations in $\mathcal{K}_i^c \cup \mathcal{K}_f^c \cup \mathcal{K}_s^c$.

The final result (4.23) follows using the same argument as in the previous paragraph, except now $\mathcal{K}_s^c \cap \mathcal{K}_\epsilon$ is no longer necessarily empty. □
We see from (4.23) that, for all sufficiently small $\epsilon$, the worst case complexity result for Algorithm 3.1 is $\epsilon^{-2+2p}$, which is worse than the $\epsilon^{-2}$ result that holds for the PG method. If one is concerned with such a result, the difference can be made arbitrarily small (for a range of $\epsilon$ values typically used in practice) by choosing $p$ sufficiently small. However, as is typical with well-designed second-derivative methods, although the complexity bound is worse, it typically performs better (see Section 5).

4.2. Local convergence. We now consider the local convergence rate of the iterates generated by Algorithm 3.1. Our analysis is performed under the following additional assumption that will be assumed to hold throughout this section.

Assumption 4.3. The function $f$ is twice continuously differentiable and strongly convex. It follows that that there exists a unique solution $x_{*}$ to the optimization problem (1.1) with optimal support $S_{*} := \{ i : [x_{*}]_{i} \neq 0 \}$. Moreover, we assume that $\nabla^{2} f : \mathbb{R}^{n} \to \mathbb{R}^{n \times n}$ is Lipschitz continuous in a neighborhood of the solution $x_{*}$, and that $f + r$ is nondegenerate at $x_{*}$ in the sense that $\| \nabla f(x_{*}) \|_{2} < \lambda_{i}$ for all $i \notin S_{*}$.

Optimality conditions for problem (1.1) imply that $\| \nabla f(x_{*}) \|_{2} \leq \lambda_{i}$ for all $i \notin S_{*}$. Thus, the final condition in Assumption 4.3 is a strengthening of this fact.

Assumption 4.4. The following algorithmic choices are made in Algorithm 3.1:

(i) The backtracking parameter is chosen to satisfy $\eta \in (0, 1/2)$.

(ii) For all sufficiently large $k \in \mathbb{N}$, $I_{k}$ in Line 9/16 is chosen as

$$I_{k} = \begin{cases} I_{k}^{CG} & \text{if } k \in K^{CG}, \\ I_{k}^{PG} & \text{if } k \in K^{PG}. \end{cases}$$

(iii) For all sufficiently large $k \in K^{CG}$, $H_{k} = \nabla^{2} f(x_{k}) (f + r)(x_{k})$ is chosen in Line 10.

The next result establishes that the iterate sequence converges to $x_{*}$.

Theorem 4.9. The iterate sequence $\{ x_{k} \}$ generated by Algorithm 3.1 satisfies

$$\lim_{k \to \infty} x_{k} = x_{*} \quad \text{and} \quad \lim_{k \to \infty} \max \{ \lambda_{k}^{CG}, \chi_{k}^{CG} \} = 0.$$

Proof. Theorem 4.8 gives $\lim_{k \to \infty} \max \{ \lambda_{k}^{PG}, \chi_{k}^{CG} \} = 0$. Since $\{ x_{k} \}$ is bounded due to monotonicity of $\{ f(x_{k}) + r(x_{k}) \}$ (see Lemma 4.7) and Assumption 4.3, there exists an infinite $K \subseteq \mathbb{N}$ and $\hat{x}$ so that $\lim_{k \in K, k \to \infty} x_{k} = \hat{x}$. It follows from Lemma 4.1 and Lemma 4.3 that $\hat{x}$ is a solution to problem (1.1), but with Assumption 4.3 this means that $\hat{x} = x_{*}$, so $\lim_{k \in K, k \to \infty} x_{k} = x_{*}$. The fact that the entire sequence $\{ x_{k} \}$ converges to $x_{*}$ follows from this fact, Assumption 4.3, and monotonicity of $\{ f(x_{k}) + r(x_{k}) \}$. \[ \square \]

We now show for groups whose variables are all equal to zero at the solution that the PG step will eventually predict them to be zero.

Lemma 4.10. For all $i \notin S_{*}$ and sufficiently large $k$, it holds that $[x_{k} + s_{k}]_{i} = 0$.

Proof. First note that Lemma 4.3 and the update strategy for $\{ \alpha_{k} \}$ in Algorithm 3.1 ensure that there exists $\tilde{k}_{1}$ such that $\alpha_{k} = \alpha_{*} > 0$ for all $k \geq \tilde{k}_{1}$. Now, let $i \notin S_{*}$ so that $[x_{*}]_{i} = 0$. It follows from Assumption 4.3 that

$$\frac{\alpha_{*} \lambda_{i}}{\| [x_{*} - \alpha_{*} \nabla f(x_{*})]_{i} \|_{2}} = \frac{\lambda_{i}}{\| \nabla f(x_{*}) \|_{2}} > 1.$$

Combining this with Theorem 4.9, $\alpha_{k} = \alpha_{*} > 0$ for all $k \geq \tilde{k}_{1}$, and Assumption 1.1 shows that there exists $\tilde{k}_{2} \geq \tilde{k}_{1}$ such that $1 - \alpha_{k} \lambda_{i}/\| [x_{k} - \alpha_{k} \nabla f(x_{k})]_{i} \|_{2} < 0$ for all $k \geq \tilde{k}_{2}$. Using this fact with (2.2) and (2.3) shows that $[x_{k} + s_{k}]_{i} = 0$ for all $k \geq \tilde{k}_{2}$. This completes the proof since the choice $i \notin S_{*}$ was arbitrary and $n_{G}$ is finite. \[ \square \]
We now show that, eventually, the set \( S_* \) determines the sets \( T_k^{ps} \) and \( T_k^{gr} \).

**Lemma 4.11.** For all sufficiently large \( k \), it holds that

\[
T_k^{ps} = \{ j \in G_i : i \notin S_* \} \quad \text{and} \quad T_k^{gr} = \{ j \in G_i : i \in S_* \}
\]

where the sets \( T_k^{ps} \) and \( T_k^{gr} \) are defined in (3.4).

**Proof.** Let \( \bar{k}_1 \) be large enough so that the conclusion of Lemma 4.10 holds, i.e., if \( k \geq \bar{k}_1 \) and \( i \notin S_* \), then \( [x_k + s_k]_{G_i} = 0 \). Together with (3.2), this shows that for all sufficiently large \( k \geq \bar{k}_1 \) and \( i \notin S_* \), and thus \( G_i \subset T_k^{ps} \) (see (3.4)) for all \( k \geq \bar{k}_1 \) and \( i \notin S_* \). In other words, it holds that \( \{ j \in G_i : i \notin S_* \} \subset T_k^{ps} \) for all \( k \geq \bar{k}_1 \).

Next, we prove that there exists \( \bar{k}_2 \) such that \( T_k^{ps} \) is not empty for all sufficiently large \( k \) and \( \bar{k}_2 \). For a proof by contradiction, suppose that there exists an infinite subsequence \( K \subseteq \mathbb{N} \)

and group index \( i \) such that \( G_i \subset T_k^{ps} \) and \( i \notin S_* \) for all \( k \in K \). Since \( G_i \subset T_k^{ps} \) for all \( k \in K \), it follows from (3.2), (3.3), and (3.4) that at least one of

\[
(x_k)_{G_i} = 0, \quad [x_k + s_k]_{G_i} = 0, \quad \| [x_k]_{G_i} \|_2 < \kappa_1 \| \nabla G_i (f + r)(x_k) \|_2 \quad \text{or}
\]

\[
(x_k)_{G_i} = 0, \quad \| [x_k]_{G_i} \|_2 < \kappa_2 \| \nabla T_k^{gr} (f + r)(x_k) \|_2^p
\]

holds for all \( k \in K \). However, since \( i \in S_* \), it follows from Theorem 4.9 that the first condition in (4.26) does not hold for all sufficiently large \( k \in K \). Also, it follows from Theorem 4.9, the facts that \( \chi_k^{ps} = \| s_k \|_{T_k^{ps}} \|_2 \) and \( \chi_k^{gr} = \| s_k \|_{T_k^{gr}} \|_2 \), and the fact that \( T_k^{ps} \cup T_k^{gr} = \{1, \ldots, n\} \) that \( \lim_{k \to \infty} \| s_k \|_2 = 0 \), which combined with \( i \in S_* \) proves that \( [x_k + s_k]_{G_i} \neq 0 \) for all sufficiently large \( k \). Hence, the second condition in (4.26) does not hold for all sufficiently large \( k \in K \). Next, from the optimality conditions for problem (1.1), the fact that \( i \in S_* \), Theorem 4.9, Assumption 1.1, and the fact that \( f + r \) is differentiable over the variables in \( G_i \) for sufficiently large \( k \) that we have \( \lim_{k \to \infty} \| \nabla G_i (f + r)(x_k) \|_2 = 0 \). This limit, \( [x_k]_{G_i} \neq 0 \), and Theorem 4.9 show that \( \| [x_k]_{G_i} \|_2 \geq \kappa_1 \| \nabla G_i (f + r)(x_k) \|_2 \) for all sufficiently large \( k \), meaning that the third condition in (4.26) does not hold for all sufficiently large \( k \in K \). Therefore, we must conclude that the inequality in (4.27) holds for all sufficiently large \( k \in K \). Combining this with \( i \in S_* \), shows that there exists \( \epsilon > 0 \) such that

\[
\| \nabla T_k^{ps} (f + r)(x_k) \|_2 \geq \epsilon > 0 \quad \text{for all sufficiently large} \quad k \in K,
\]

which in particular shows that \( T_k^{ps} \neq \emptyset \) for all sufficiently large \( k \in K \). Since the optimality conditions for problem (1.1) together with Theorem 4.9, Assumption 1.1, and the fact that \( f + r \) is differentiable over the variables in \( G_i \) for sufficiently large \( k \) imply that \( \lim_{k \to \infty} \| \nabla G_i (f + r)(x_k) \|_2 = 0 \) for all \( i \in S_* \), we must conclude from (4.28) that, for all sufficiently large \( k \in K \), there exists an \( i_k \notin S_* \) such that \( G_{i_k} \subset T_k^{gr} \).

However, Lemma 4.10 yields \( [x_k + s_k]_{G_{i_k}} = 0 \) for all sufficiently large \( k \in K \), which together with (3.2) shows that \( G_{i_k} \subset T_k^{gr} \), which is a contradiction. Therefore, there exists \( \bar{k}_2 \) such that \( T_k^{ps} \subseteq \{ j \in G_i : i \notin S_* \} \) for all \( k \geq \bar{k}_2 \).

The conclusions of the two previous paragraphs yields \( T_k^{ps} \equiv \{ j \in G_i : i \notin S_* \} \) for all sufficiently large \( k \). The final assertion, namely that \( T_k^{gr} \equiv \{ j \in G_i : i \in S_* \} \), follows from the fact that \( T_k^{ps} \) and \( T_k^{gr} \) partition \( \{1, 2, \ldots, n\} \) for every iteration \( k \).

The next result shows that, for iterations \( k \) sufficiently large, the support of \( x_k \) agrees with the support of the solution \( x_* \).

**Lemma 4.12.** For all sufficiently large \( k \), it holds that

\[
[x_k]_{G_i} \neq 0 \quad \text{for all} \quad i \in S_* \quad \text{and} \quad [x_k]_{G_i} = 0 \quad \text{for all} \quad i \notin S_*.
\]
Proof. Theorem 4.9 shows that \( |x_k|_{\mathcal{G}_i} \neq 0 \) for all sufficiently large \( k \) and all \( i \in \mathcal{S}_s \), which is the first desired result. Hence, let us proceed by considering arbitrary \( i \notin \mathcal{S}_s \).

Assumption 4.4(ii), Lemma 4.10, Lemma 4.11, and Lemma 4.3 ensure the existence of an iteration \( \bar{\mathcal{F}} \) such that, for all \( k \geq \bar{\mathcal{F}} \), the following hold:

\[
\mathcal{G}_i \subseteq T_{k}^{\text{ps}}, \quad |x_k + s_k|_{\mathcal{G}_i} = 0, \quad \text{and} \quad \alpha_k = \alpha_{\bar{\mathcal{F}}}. \tag{4.29}
\]

We claim that the second desired result follows from (4.29) if there exists some sufficiently large \( k \geq \bar{\mathcal{F}} \) such that \( k \in \mathcal{K}^{\text{ps}} \) and \( |x_{k+1}|_{\mathcal{G}_i} = |x_k + s_k|_{\mathcal{G}_i} = 0 \). Indeed, since \( k \) is an arbitrary element from \( \{1, \ldots, n_{\mathcal{G}}\} \setminus \mathcal{S}_s \), \( n_{\mathcal{G}} \) is finite, and the second condition in (4.29) shows that values of the variables in \( \mathcal{G}_i \) can only be modified if \( k \in \mathcal{K}^{\text{ps}} \), the existence of such \( k \) along with (4.29) shows that iteration \( \bar{\mathcal{F}} \in \mathcal{K}^{\text{ps}} \) sets \( |x_{k+1}|_{\mathcal{G}_i} \) to zero, and these variables will remain zero for all future iterations.

Let us now show the existence of such \( \bar{\mathcal{F}} \geq \bar{\mathcal{F}} \). We claim that there exists \( k \geq \bar{\mathcal{F}} \) such that \( |x_k|_{\mathcal{G}_i} = 0 \). For a proof by contradiction, suppose that \( |x_k|_{\mathcal{G}_i} \neq 0 \) for all \( k \geq \bar{\mathcal{F}} \). Combining this with Theorem 4.9, \( i \notin \mathcal{S}_s \), and the fact that the variables in \( \mathcal{G}_i \) can have their values changed only if \( k \in \mathcal{K}^{\text{ps}} \) implies that there exists \( k \geq \bar{\mathcal{F}} \) such that \( k \in \mathcal{K}^{\text{ps}} \). Now, since \( k \in \mathcal{K}^{\text{ps}} \) and \( \alpha_k = \alpha_{\bar{\mathcal{F}}} \) for all \( k \geq \bar{\mathcal{F}} \), it follows from Algorithm 3.1 that flag_{\mathcal{K}^{\text{ps}}} = \text{same}_{\mathcal{G}} \) is returned in Line 17. Using this fact, the update used in Line 56, and (4.29) shows that \( |x_{k+1}|_{\mathcal{G}_i} = |x_k + s_k|_{\mathcal{G}_i} = 0 \). \( \square \)

We require one more lemma that shows that eventually all iterations are in \( \mathcal{K}^{\text{cg}} \).

Lemma 4.13. For all \( k \) sufficiently large, it holds that \( k \in \mathcal{K}^{\text{cg}} \).

Proof. We first show that all sufficiently large \( k \) are in \( \mathcal{K}^{\text{cg}} \). It follows from Lemma 4.11 that \( T_k^{\text{ps}} \equiv \{ j \in \mathcal{G}_i : i \notin \mathcal{S}_s \} \) for all sufficiently large \( k \). Combining this with Lemma 4.12 and Lemma 4.10 shows that there exists an iteration \( \bar{\mathcal{F}} \) such that \( |x_k|_{T_k^{\text{ps}}} = 0 \) and \( |x_k + s_k|_{T_k^{\text{ps}}} = 0 \) for all \( k \geq \bar{\mathcal{F}} \), which means that \( \chi_k^{\text{ps}} = ||s_k||_{T_k^{\text{ps}}}^2 = 0 \) for all \( k \geq \bar{\mathcal{F}} \). It follows from this fact, Line 8, and Assumption 4.1 that \( k \in \mathcal{K}^{\text{cg}} \) for all \( k \geq \bar{\mathcal{F}} \). Now, notice that at most \( n_{\mathcal{G}} - 1 \) iterations from \( \bar{\mathcal{F}} \) onward can be in \( \mathcal{K}^{\text{cg}} \) because of Lemma 4.6(i). (Every iteration \( k \in \mathcal{K}^{\text{cg}}_0 \) fixes at least one new group of variables to zero and if they ever all become zero so that \( T_k^{\text{cg}} = \emptyset \), then the contradiction \( k \in \mathcal{K}^{\text{ps}} \) is reached.) Therefore, it follows that all sufficiently large \( k \) must be in \( \mathcal{K}^{\text{cg}}_{\text{sd}} \). \( \square \)

We can now state our main local convergence result.

Theorem 4.14. If in Algorithm 3.2 we choose either \( q \in \{1, 2\} \), or \( q = 1 \) and \( \{\mu_k\} \to 0 \), then \( \{x_k\} \to x_* \) at a superlinear rate. In particular, if we choose \( q = 2 \), then the rate of convergence is quadratic.

Proof. It follows from Lemma 4.11, Lemma 4.12, and Lemma 4.13 that, for all sufficiently large \( k \), the iterates generated by Algorithm 3.1 satisfy the recurrence \( x_{k+1} = x_k + \xi k \cdot d_k \), where \( d_k \) is the result of the backtracking Armijo line search in Line 47, \( ||x_k||_{T_k^{\text{ps}}}^2 = ||d_k||_{T_k^{\text{ps}}}^2 = 0 \), and \( |x_k|_{T_k^{\text{ps}}} = d_k \) with \( d_k \) computed by Algorithm 3.2 to satisfy (3.9). In other words, for all sufficiently large \( k \), we have \( |x_k|_{T_k^{\text{ps}}} = |x_*|_{T_k^{\text{ps}}} = 0 \) and the values of the variables in \( T_k^{\text{ps}} \equiv \{ j \in \mathcal{G}_i : i \in \mathcal{S}_s \} \) are updated exactly as those of an inexact Newton method for computing a root of \( \nabla x_k^r (f + r) \). Since, by Theorem 4.9, we have \( \lim_{k \to \infty} x_k = x_* \), the desired conclusions follow under the stated conditions from [11, Theorem 3.3] and noting the well-known result that the unit step size \( \xi_k = 1 \) is accepted (asymptotically) by a backtracking Armijo line search when \( \eta \in (0, 1/2) \) (see Assumption 4.4) under our assumptions. \( \square \)

Theorem 4.14 states conditions under which Algorithm 3.1 yields a superlinear, or even quadratic, rate of local convergence. The neighborhood about \( x_* \) in which such
a rate will be achieved, and the explicit constants in the convergence rate that will be
achieved, depend as usual on magnitudes of a Lipschitz constant for \( \nabla I_k(f + r) \) and
an upper bound on a norm of the inverse of \( \nabla^2_\alpha(f + r) \), where \( I_k := \{ j \in G_i : i \in S_i \} \).

Due to the properties of the regularizer \( r \), the latter of these values may be inversely
proportional to the norms of the groups of variables in the support at the solution.

5. Numerical Results. In this section, we present the results of numerical
experiments with an implementation of FaRSA-Group (Algorithm 3.1) applied to solve
a collection of group sparse regularized logistic regression problems of the form

\[
\min_{x \in \mathbb{R}^n} \frac{1}{N} \sum_{i=1}^{N} \log \left( 1 + e^{-y_i x^T d_i} \right) + \sum_{i=1}^{n_G} \lambda_i \| x \|_2^2,
\]

where \( d_i \in \mathbb{R}^n \) is the \( i \)th data point, \( N \) is the number of data points in the data set,
\( y_i \in \{-1, 1\} \) is the class label for the \( i \)th data point, and \( \lambda_i \) is the weight parameter
for the \( i \)th group. We first describe details of our implementation, then describe the
data sets considered in our experiments, and finally present our experimental results.

5.1. Implementation details. We have developed a Python implementation
of FaRSA-Group that is available upon request. The values of the input parameters
for Algorithm 3.1 and Algorithm 3.2 that we used are given in Figure 5.1 (with some
caveats that are mentioned in the following paragraph).

We initialized \( x_0 \) as the zero vector and \( \sigma_0 \)
as an estimate of the inverse of the Lipschitz constant of \( f \) at \( x_0 \). To be precise, our software randomly generated a vector \( y_0 \in \mathbb{R}^n \)
such that \( \|x_0 - y_0\|_2 = 10^{-8} \), and then set
\( \sigma_0 = \min\{1, \|x_0 - y_0\|_2/\|\nabla f(x_0) - \nabla f(y_0)\|_2\} \).

Since \( \varphi = 1 \), it follows from Algorithm 3.1
that (4.25) holds for all \( k \in \mathbb{N} \). (However,
for data sets with \( N < n \), we initially chose
\( \varphi = 0.8 \) and switched to \( \varphi = 1 \) when an iteration in \( K^{\infty} \) satisfied
\( f(x_k) - f(x_{k+1}) \leq 10^{-3} \). When \( N < n \), the matrix \( \nabla^2 f(x_k) \)
is singular, which in practice often led to large CG directions and multiple backtracks
in the line search. These ill effects were partly remedied by this scheme for updating \( \varphi \).
When defining the set \( I_k^{\text{split}} \) in (3.3), we used \( \tilde{\kappa}_{2,i} = \kappa_2 ||G_i||/||I_k^{\text{split}}|| \) in place of \( \kappa_2 \) for all \( i \) such that \( G_i \subseteq I_k^{\text{split}} \). This choice accounted for the fact that the two
different norms in (3.3) are associated with vectors of different dimension. Note that
since \( (1/n)\kappa_2 \leq \tilde{\kappa}_{2,i} \leq n\kappa_2 \), this choice is easily incorporated into the analysis in
Section 4. The choice of \( H_k \) in Line 10 was based on a regularization of the exact
second-derivatives of \( f \). In particular, for any scalar \( \delta \geq 0 \), consider

\[
\frac{1}{\delta} D^T \Sigma(x) D \approx \nabla^2 f(x)
\]

where \( D^T := [d_1, d_2, \ldots, d_N] \) and \( \Sigma(x) \) is the diagonal matrix with \( i \)th diagonal entry
\( [\Sigma(x)]_{ii} := \max\{\sigma_i(x) (1 - \sigma_i(x)), \delta\} \)
with \( \sigma_i(x) := \exp(y_i d_i^T x)/(1 + \exp(y_i d_i^T x)) \)
for all \( i \in \{1, 2, \ldots, N\} \). Notice that if \( \delta = 0 \), then \((1/N)D^T \Sigma(x) D \equiv \nabla^2 f(x) \).

In order to use a small amount of regularization in our tests, we chose \( \delta = 10^{-8} \). With
this choice of \( \delta \), our choice of \( H_k \) in Line 10 can now be written as

\[
H_k \leftarrow [\frac{1}{N} D^T \Sigma(x) D]_{I_k I_k} + \nabla^2_{I_k I_k} r(x_k),
\]

\[
\text{param. value} \quad \text{param. value}
\]
\[
\begin{array}{ccc}
\varphi & 1 & \kappa_1 & 0.1 \\
\xi & 0.5 & \kappa_2 & 10^{-2} \\
\eta & 10^{-3} & \theta & \pi/4 \\
\zeta & 0.8 & \xi & 1 \\
p & 2 & \mu_k & 1 \\
\end{array}
\]

Fig. 5.1. Parameter values used in our tests for Algorithm 3.1 and Algorithm 3.2.
where we remind the reader that $\nabla_{x_k}^2 r(x_k)$ is well defined because the construction of $\mathcal{Z}_k \subseteq T_{x_k}^* \mathbb{R}$ ensures that $[x_k]_{\mathcal{G}_k} \neq 0$ for all $\mathcal{G}_k \subseteq \mathcal{I}_k$.

In Algorithm 3.2, we applied the CG method to the system $H_k d = -g_k$ to approximately solve the optimization problem defined in Line 26. As pointed out in Section 3.2, the direction associated with every iteration of the CG algorithm satisfies condition (3.7) and condition (3.8), which were required to establish the complexity result in Theorem 4.8. To reduce the cost of the CG computation and limit the number of backtracking steps required by Algorithm 3.3, we terminated Algorithm 3.2 when at least one of three conditions was satisfied. To describe these conditions checked during the $k$th iteration, let $d_{j,k}$ denote the $j$th CG iterate and let $t_{j,k} := \|H_k d_{j,k} + g_k\|_2$ denote the $j$th CG residual. The three conditions are given by

\begin{align}
(5.2a) & \quad t_{j,k} \leq \max\{\min\{0.1 t_{0,k}, t_{0,k}^{1.5}\}, 10^{-10}\}, \\
(5.2b) & \quad \|d_{j,k}\| \geq 10^3 \min\{1, \|\nabla_{x_k} (f + r)(x_k)\|_2\}, \quad \text{and} \\
(5.2c) & \quad j = |\mathcal{I}_k|.
\end{align}

Outcome (5.2a) is the ideal termination condition since it indicates that the residual of the linear system has been sufficiently reduced (see (3.9)). Outcome (5.2b) serves as a trust-region constraint on the norm of the trial step $d_k$; in particular, when the inequality in (5.2b) holds, the size of the CG iterate $d_{j,k}$ is relatively large, indicating that $x_k$ is not close to an optimal solution. Therefore, we restrict its size with the intent of needing fewer backtracking steps during the subsequent line search. Outcome (5.2c) caps the number of CG iterations to $|\mathcal{I}_k|$ (the size of the reduced space) since, in exact arithmetic, CG converges to an exact solution in at most $|\mathcal{I}_k|$ iterations.

Algorithm 3.1 decreases the value of the PG parameter (see Line 19) for the next iteration using a simple multiplicative factor when $\text{flag}_{pg}$ = decrease.α. However, in practice, we found an adaptation of the approach in [9] to be more efficient. To describe this approach, let $\xi_j$ be the search direction and step size used to obtain $x_{k+1} = x_k + \xi_j d_k$. It is well known [2, Lemma 5.7] that if $\alpha \in (0,1/L_f]$, then $f(x_{k+1}) \leq f(x_k) + \xi_j \nabla f(x_k)^T d_k + \frac{1}{2\alpha} \|\xi_j d_k\|^2_2$. Setting this inequality to be an equality and then solving for $\alpha$, one obtains

$$\hat{\alpha}_k := \frac{\|\xi_j d_k\|^2_2}{2(f(x_{k+1}) - f(x_k) - \xi_j \nabla f(x_k)^T d_k)},$$

which can be viewed as a local Lipschitz constant estimate for $f$ at $x_k$. In our tests, we updated the PG parameter at the end of each iteration of Algorithm 3.1 as

$$\alpha_{k+1} \leftarrow \min\{1, \hat{\alpha}_k/2\}.$$  

Although this PG parameter update strategy worked better than the basic strategy in Algorithm 3.1 (see Line 19 and Line 21), it is not covered by our analysis in Section 4. However, a simple modification of our analysis would be to allow the update in (5.3) to increase the PG parameter at most a finite number of times, say 100 times, at which point the update $\alpha_{k+1} \leftarrow \min\{\alpha_k, \hat{\alpha}_k/2\} \leq \alpha_k$ would be used. This strategy is covered by our earlier analysis (with a larger constant in the complexity result).

We terminate our algorithm when $\max\{\lambda^c_k, \lambda^p_k\} \leq 10^{-6} \max\{\lambda^c_0, \lambda^p_0, 1\}$.

5.2. Data sets. We tested FaRSA-Group on problem (5.1) using data sets from the LIBSVM repository. From this repository, we excluded all regression instances.

1https://www.csie.ntu.edu.tw/cjlin/libsvmtools/datasets
and multiple-class (greater than two) classification instances. We compared the performance of our algorithm to the well-cited package \texttt{gglasso} \cite{wai}, which is a state-of-the-art group-wise majorization descent method. Since \texttt{gglasso} does not support sparse data matrix inputs, we excluded all data sets that were too large to be stored in memory (6GB). Finally, for the adult data (a1a–a9a) and webpage data (w1a–w8a), we used only the largest instances, namely a9a and w8a. This left us with our final subset of 25 data sets that can be found in Table 5.1.

Scaling of the data sets can be important. If the LIBSVM website indicated that a data set was already scaled, then we used the data set without modification. However, when the website did not indicate that scaling for a data set was used, we scaled each column of the feature data (i.e., feature-wise scaling) into the range $[-1, 1]$ by dividing each of its entries by the largest entry in absolute value. Labels for some data sets (e.g., breast-cancer, covtype, liver-disorders, mushrooms, phishing, skin-nonskin and svmguide1) do not take values in $\{-1, 1\}$, but rather in $\{0, 1\}$ or $\{1, 2\}$. For these data sets, we mapped the smaller label to $-1$ and the larger label to 1.

| data set    | N    | n    | scale | who     | used |
|-------------|------|------|-------|---------|------|
| a9a         | 32561| 123  | [0,1] | website | 8    |
| australian  | 690  | 140  | [-1,1] | website | 2    |
| breast-cancer | 683 | 10   | [-1,1] | website | 0    |
| cod-rna     | 50535| 8    | [-1,1] | us      | 8    |
| colon-cancer | 62  | 2000 | (μ, σ²) = (0, 1) | website | 8    |
| covtype.binary | 581012 | 54 | [0,1] | website | 8    |
| diabetes    | 768  | 8    | [-1,1] | website | 0    |
| duke breast-cancer | 44  | 7192 | (μ, σ²) = (0, 1) | website | 8    |
| fourclass   | 862  | 2    | [-1,1] | website | 0    |
| german-numer | 1000 | 24   | [-1,1] | website | 0    |
| gisette     | 6000 | 5000 | [-1,1] | website | 8    |
| heart       | 270  | 13   | [-1,1] | website | 2    |
| ijcnn1      | 49990| 22   | [-1.5, 1.5] | website | 8    |
| ionsphere   | 351  | 34   | [-1,1] | website | 0    |
| leukemia    | 38   | 7129 | (μ, σ²) = (0, 1) | website | 8    |
| liver-disorders | 145 | 5    | [-1,1] | website | 0    |
| madelon     | 2000 | 5000 | [-1,1] | us      | 8    |
| mushrooms   | 8124 | 112  | [0,1] | website | 6    |
| phishing    | 11055| 68   | [0,1] | website | 7    |
| skin-nonskin | 245057 | 3   | [-1,1] | us      | 8    |
| splice      | 1000 | 60   | [-1,1] | website | 0    |
| sonar       | 208  | 60   | [-1,1] | website | 4    |
| svmguide1   | 3089 | 4    | [-1,1] | us      | 0    |
| svmguide2   | 1243 | 21   | [-1,1] | website | 0    |
| w8a         | 49749| 300  | [0,1] | website | 8    |

\footnote{\url{https://cran.r-project.org/web/packages/gglasso}}
5.3. Experimental setup and test results. We tested FaRSA-Group and gglasso for solving problem (5.1) using the data sets in Table 5.1. All default settings for gglasso were used, including the same starting point $x_0 = 0$ used by FaRSA-Group. We considered four group structures and two different solution sparsity levels. Specifically, we considered the four different numbers of groups

$$\text{number of groups} \in \{[0.25n], [0.50n], [0.75n], n\},$$

where $n$ is the problem dimension; notice that the last setting recovers $\ell_1$-norm regularization. Then, for a given number of groups, the variables were sequentially distributed (as evenly as possible) to the groups; e.g., 10 variables among 3 groups would have been distributed as $\mathcal{G}_1 = \{1, 2, 3\}$, $\mathcal{G}_2 = \{4, 5, 6\}$, and $\mathcal{G}_3 = \{7, 8, 9, 10\}$. For the two different solution sparsity levels, we considered groups weights

$$\lambda_i = 0.1\lambda_{\min}\sqrt{|\mathcal{G}_i|} \quad \text{and} \quad \lambda_i = 0.01\lambda_{\min}\sqrt{|\mathcal{G}_i|}$$

where $\lambda_{\min} = \min\{\lambda \geq 0 : \text{the solution to (5.1) with } \lambda_i = \lambda\sqrt{|\mathcal{G}_i|} \text{ is } x = 0\}$ (see [32, equation (23)]). Since there were 25 data sets, a total of 200 problem instances were tested (each data set has 8 instances). The experiments were conducted using the cluster in the Computational Optimization Research Laboratory (COR@L) at Lehigh University with an AMD Opteron Processor 6128 2.0 GHz CPU. In the following paragraphs, we compared the performance of FaRSA-Group with that of gglasso with respect to CPU time (seconds), final objective value, and solution sparsity.

First consider the CPU time. For each problem instance, we allowed a maximum of 1000 seconds. If the CPU time in a run went above this limit, we terminated that run and considered the algorithm to have failed. Out of the 200 problem instances, FaRSA-Group failed 2 times and gglasso failed 7 times. Figure 5.2 illustrates a performance profile based on [24] for comparing the computing times on problem instances that FaRSA-Group and/or gglasso took at least 1 second to terminate; this resulted in 109 problem instances. The last column of Table 5.1 gives the number of instances for each data set used in this profile. Each bar in the plot corresponds to a problem instance, with the height of the bar given by

$$-\log_2\left(\frac{\text{time required by FaRSA-Group}}{\text{time required by gglasso}}\right).$$

Therefore, an upward pointing bar indicates that FaRSA-Group took less time to find the optimal solution for that problem instance and a downward pointing bar means that gglasso took less time, and in either case the size of the bar indicates the magnitude of the outperformance factor. A bar that reaches the $y$-axis limit of ±10 is used when indicating that an algorithm was successful when solving a problem instance while the competing algorithm was unsuccessful.

To compare final objective function values, let $F_{\text{FaRSA-Group}}$ and $F_{\text{gglasso}}$ denote (for a given problem instance) the objective values returned by FaRSA-Group and gglasso,
respectively. If \( F_{\text{glasso}} - F_{\text{FaRSA-Group}} > 10^{-8} \), then we considered FaRSA-Group to have obtained a lower objective function value; if \( F_{\text{FaRSA-Group}} - F_{\text{glasso}} > 10^{-8} \), then we considered \( \text{glasso} \) to have obtained a lower objective function value; and if \( |F_{\text{FaRSA-Group}} - F_{\text{glasso}}| \leq 10^{-8} \), then we considered them to have performed equally.

From the 109 problem instances that at least one algorithm took at least one second to terminate, FaRSA-Group outperformed \( \text{glasso} \) 95 times and \( \text{glasso} \) outperformed FaRSA-Group 7 times. From the entire 200 instances, FaRSA-Group outperformed \( \text{glasso} \) 153 times and \( \text{glasso} \) outperformed FaRSA-Group 35 times.

In terms of solution sparsity, we considered FaRSA-Group to have outperformed \( \text{glasso} \) if the following two conditions held: (i) all zero groups in the \( \text{glasso} \) solution were also zero groups in the FaRSA-Group solution, and (ii) the solution returned by FaRSA-Group had at least one zero group that was not a zero group in the \( \text{glasso} \) solution. A similar criterion was used to define when \( \text{glasso} \) was considered to have outperformed FaRSA-Group. From the 109 test instances, FaRSA-Group outperformed \( \text{glasso} \) in 30 cases and \( \text{glasso} \) outperformed FaRSA-Group in 7 cases. From the entire collection of 200 problem instances, FaRSA-Group outperformed \( \text{glasso} \) in 33 cases and \( \text{glasso} \) outperformed FaRSA-Group in 8 cases.

6. Conclusion. We presented a new framework for solving optimization problems that incorporate group sparsity-inducing regularization by using subspace acceleration, domain decomposition, and support identification. In terms of theory, we proved a complexity result on the maximum number of iterations before an \( \epsilon \)-approximate solution is computed (Theorem 4.8), and a local superlinear convergence rate (Theorem 4.14). The strong convergence theory was supported by experimental results for minimizing a group sparsity-regularized logistic function for the task of classification. In terms of robustness, computational time, final objective value obtained, and solution sparsity, the numerical results showed that our proposed FaRSA-Group framework outperformed a state-of-the-art method.

Appendix A. Proofs. In this appendix, for completeness, we provide detailed proofs of the results from Section 2 related to the PG computations.

Proof of Lemma 2.1. Let \( x_+ = T(x, \bar{\sigma}) \) denote the PG update in (2.1) so that \( x_+ = x + s(x, \bar{\sigma}) \) with \( s(x, \bar{\sigma}) \) defined in (2.2). It follows from the optimality conditions for the problem in (2.1) that there exists \( g_+ \in \partial r(x_+) \) such that

\[
(A.1) \quad x_+ - x - \bar{\sigma} \nabla f(x) + \bar{\sigma} g_+ = 0.
\]

Next, for an arbitrary \( g_{f+r} \in \partial (f + r)(x) \), it follows from Assumption 1.1 and [4, Proposition 5.4.6] that there exits \( g_r \in \partial r(x) \) satisfying \( g_{f+r} = \nabla f(x) + g_r \). From the definitions of \( g_r \) and \( g_+ \) and convexity of \( r \), it follows that

\[
(A.2) \quad r(x_+) \geq r(x) + g_r^T (x + x - x) \quad \text{and} \quad r(x) \geq r(x_+) + g_+^T (x - x_+).
\]

Adding the two equations in (A.2) together yields \( (g_r - g_+)^T (x_+ - x) \leq 0 \). Combining this with the definition of \( g_{f+r} \), (A.1), and the definition of \( x_+ \) that

\[
(A.3) \quad s(x, \bar{\sigma})^T g_{f+r} = (x_+ - x)^T (\nabla f(x) + g_r)
\]

\[
= \frac{1}{\bar{\sigma}} (x_+ - x)^T (x_+ - x - \bar{\sigma} g_+ + \bar{\sigma} g_r)
\]

\[
= -\frac{1}{\bar{\sigma}} \| x_+ - x \|^2 + (x_+ - x)^T (g_r - g_+) \leq -\frac{1}{\bar{\sigma}} \| s(x, \bar{\sigma}) \|^2.
\]

Since \( g_{f+r} \in \partial (f + r)(x) \) was arbitrary, the result [25, Theorem 2.87] and (A.3) yield

\[
D_{f+r}(x; s(x, \bar{\sigma})) = \sup_{g \in \partial (f + r)(x)} s(x, \bar{\sigma})^T g \leq -\frac{1}{\bar{\sigma}} \| s(x, \bar{\sigma}) \|^2,
\]

This manuscript is for review purposes only.
which is the desired result and completes the proof.

**Proof of Lemma 2.2.** The proof follows exactly as in the proof of Lemma 2.1 above, but where all calculations are restricted to groups in the set $\mathcal{I}$ (also see (2.3)).

**Proof of Lemma 2.3.** The result, for the case $\mathcal{I} = \{1, 2, \ldots, n\}$, can be found in [2, Lemma 10.4]. For the general case, i.e., when $\mathcal{I}$ is equal to the union of a subset of $\{G_i\}_{i=1}^{n_2}$, the result follows by using the same proof as for [2, Lemma 11.9].

**Proof of Lemma 2.4.** Denote $g_i := \nabla g_i f(x)$, $x_i = [x_i]_{G_i}$, and $s_i = [s(x, x)]_{G_i}$. Since $f + r$ is differentiable with respect to the variables in $G_i$ at $x$ since $[x]_{G_i} \neq 0$, we have

$$\|\nabla g_i (f + r)(x)\|^2 = \|g_i + \lambda_i x_i /\|x_i\|_2\|^2 = \|g_i\|^2 + 2\lambda_i \frac{g_i^T x_i}{\|x_i\|_2} + \lambda_i^2,$$

which means that it is sufficient to prove that

$$\|g_i\|^2 + 2\lambda_i \frac{g_i^T x_i}{\|x_i\|_2} + \lambda_i^2 \geq \|s_i\|^2.$$

Since $x_i + s_i \neq 0$ by assumption, we know that $s_i$ (see (2.3)) satisfies

$$s_i = \left(1 - \frac{\alpha \lambda_i}{\|x_i - \lambda g_i\|_2}\right)(x_i - \lambda g_i) - x_i = x_i - \alpha g_i - \frac{\alpha \lambda_i (x_i - \lambda g_i)}{\|x_i - \lambda g_i\|_2} - x_i = -\alpha \left(g_i + \frac{\alpha \lambda_i (x_i - \lambda g_i)}{\|x_i - \lambda g_i\|_2}\right).$$

so that

$$\|s_i\|^2 = \alpha^2 \left(\|g_i\|^2 + 2\alpha \lambda_i \frac{g_i^T (x_i - \lambda g_i)}{\|x_i - \lambda g_i\|_2} + \alpha^2 \lambda_i^2\right).$$

Thus, it is sufficient to prove that

$$\|g_i\|^2 + 2\lambda_i \frac{g_i^T x_i}{\|x_i\|_2} + \lambda_i^2 \geq \alpha^2 \left(\|g_i\|^2 + 2\alpha \lambda_i \frac{g_i^T (x_i - \lambda g_i)}{\|x_i - \lambda g_i\|_2} + \alpha^2 \lambda_i^2\right).$$

We consider two cases, and note that $x_i \neq 0$ by assumption and that $x_i - \lambda g_i \neq 0$ as a consequence of (2.3) and the assumption that $x_i + s_i \neq 0$.

**Case 1:** $\alpha = 1$. In this case, the desired inequality simplifies to

$$\frac{g_i^T x_i}{\|x_i\|_2} \geq \frac{g_i^T (x_i - g_i)}{\|x_i - g_i\|_2}.$$  

We now consider the following two subcases.

**Case 1a:** $g_i^T x_i \geq 0$. The desired inequality clearly holds if $g_i^T (x_i - g_i) \leq 0$. Thus, for the remainder of this subcase, we assume that $g_i^T (x_i - g_i) > 0$, which equivalently means that $g_i^T x_i > \|g_i\|^2$, which implies that $-2x_i g_i + \|g_i\|^2 < 0$. It follows from this inequality and the fact that $(g_i^T x_i)^2 \leq \|g_i\|^2 \|x_i\|_2^2$ (by Cauchy-Schwarz) that

$$(g_i^T x_i)^2 (2x_i g_i + \|g_i\|^2) \geq (2x_i g_i + \|g_i\|^2) \|g_i\|^2 \|x_i\|_2^2 = \|g_i\|^4 - 2g_i^T x_i \|g_i\|^2 \|x_i\|_2^2.$$

We can now add the term $(g_i^T x_i)^2 \|x_i\|_2^2$ to both sides to obtain

$$(g_i^T x_i)^2 (\|x_i\|_2^2 - 2x_i g_i + \|g_i\|^2) \geq ((g_i^T x_i)^2 + \|g_i\|^4 - 2g_i^T x_i \|g_i\|^2 \|x_i\|_2^2) \|x_i\|_2^2,$$
which can be written equivalently as
\[(g_i^T x_i)^2 \|x_i - g_i\|^2 \geq (g_i^T x_i - \|g_i\|^2)^2 \|x_i\|^2 = (g_i^T (x_i - g_i))^2 \|x_i\|^2.\]

After taking the square root of both sides, we obtain (A.4).

Case 1b: \(g_i^T x_i < 0\). Using \(g_i^T x_i < 0\) and \((g_i^T x_i)^2 \leq \|g_i\|^2 \|x_i\|^2\) (by Cauchy-Schwarz), we have
\[(g_i^T x_i^2)(-2x_i^T g_i + \|g_i\|^2) \leq (-2x_i^T g_i + \|g_i\|^2)\|g_i\|^2 \|x_i\|^2 = (\|g_i\|^4 - 2g_i^T x_i \|g_i\|^2) \|x_i\|^2.

We can now add the term \((g_i^T x_i)^2 \|x_i\|^2\) to both sides to obtain
\[\left(\frac{g_i^T x_i}{\|x_i\|^2}\right)^2 \geq \left( -2x_i^T g_i + \|g_i\|^2 \right) \leq \left( (g_i^T x_i)^2 + \|g_i\|^4 - 2g_i^T x_i \|g_i\|^2 \right) \|x_i\|^2,
\]
which can be written equivalently as
\[(g_i^T x_i)^2 \|x_i - g_i\|^2 \leq (g_i^T x_i - \|g_i\|^2)^2 \|x_i\|^2 = (g_i^T (x_i - g_i))^2 \|x_i\|^2.

After taking the square root of both sides and rearranging, we obtain
\[
\left|\frac{g_i^T x_i}{\|x_i\|^2}\right| \leq \left|\frac{g_i^T (x_i - g_i)}{\|x_i - g_i\|^2}\right|.
\]

Combining this result with \(0 > g_i^T x_i \geq g_i^T (x_i - g_i)\) gives (A.4), as claimed.

Case 2: \(\alpha \in (0, 1)\). The proof of follows from Case 1 and [2, Theorem 10.9], which in our notation from (2.2) proves that \(\|s(\bar{x}, \alpha)\|^2 \leq \|s(\bar{x}, 1)\|^2\) when \(\bar{x} \in (0, 1)\).

REFERENCES

1. F. Bach, R. Jenatton, J. Mairal, and G. Obozinski, Optimization with sparsity-inducing penalties, Foundations and Trends® in Machine Learning, 4 (2012), pp. 1–106.
2. A. Beck, First-order methods in optimization, vol. 25, SIAM, 2017.
3. A. Beck and M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, SIAM Journal on Imaging Sciences, 2 (2009), pp. 183–202.
4. D. P. Bertsekas, Convex optimization theory, Athena Scientific, Belmont, Ma., 2009.
5. T. Chen, F. E. Curtis, and D. P. Robinson, A reduced-space algorithm for minimizing \(\ell_1\)-regularized convex functions, SIAM Journal on Optimization, 27 (2017), pp. 1583–1610.
6. T. Chen, F. E. Curtis, and D. P. Robinson, FaRSA for \(\ell_1\)-regularized convex optimization: local convergence and numerical experience, 33 (2018), pp. 396–415.
7. P. Combettes and J.-C. Pesquet, Proximal splitting methods in signal processing, in Fixed-point Algorithms for Inverse Problems in Science and Eng., Springer, 2011, pp. 185–212.
8. A. R. Conn, N. I. M. Gould, and Ph. L. Toint, Trust-Region Methods, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000.
9. F. E. Curtis and D. P. Robinson, Exploiting negative curvature in deterministic and stochastic optimization, Mathematical Programming, 176 (2019), pp. 69–94.
10. I. Daubechies, M. Defrise, and C. Mol, An iterative thresholding algorithm for linear inverse problems with a sparsity constraint, Comm. Pure Appl. Math., 58 (2004), pp. 1413–1457.
11. R. S. Dembo, S. C. Eisenstat, and T. Steihaug, Inexact newton methods, SIAM Journal on Numerical Analysis, 19 (1982), pp. 400–408.
12. D. Donoho, Denoising by soft-thresholding, Trans. Inform. Theory, 41 (1995), pp. 613–627.
13. R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, Liblinear: A library for large linear classification, J. Mach. Learn. Res., 9 (2008), pp. 1871–1874.
14. R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, LIBLINEAR: A library for large linear classification, JMLR, 9 (2008), pp. 1871–1874.
15. M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright, Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems, IEEE J. Selected Topics Signal Process., 1 (2007), pp. 586–597.
J. Friedman, T. Hastie, and R. Tibshirani, *Regularization paths for generalized linear models via coordinate descent*, Journal of Statistical Software, 33 (2010), p. 1.

G. N. Grapiglia and Y. Nesterov, *Accelerated regularized newton methods for minimizing composite convex functions*, SIAM Journal on Optimization, 29 (2019), pp. 77–99.

N. Keskar, J. Nocedal, F. Oztoprak, and A. Wächter, *A second-order method for convex \(\ell_1\)-regularized optimization with active-set prediction*, Optimization Methods and Software, 31 (2016), pp. 605–621.

J. D. Lee, Y. Sun, and M. A. Saunders, *Proximal newton-type methods for minimizing composite functions*, SIAM Journal on Optimization, 24 (2014), pp. 1420–1443.

Q. Lin, Z. Lu, and L. Xiao, *An accelerated randomized proximal coordinate gradient method and its application to regularized empirical risk minimization*, SIAM Journal on Optimization, 25 (2015), pp. 2244–2273.

J. Liu, S. Ji, and J. Ye, *SLEP: Sparse Learning with Efficient Projections*, Arizona State University, 2009, http://www.public.asu.edu/~jye02/Software/SLEP.

J. Liu and S. J. Wright, *Asynchronous stochastic coordinate descent: Parallelism and convergence properties*, SIAM Journal on Optimization, 25 (2015), pp. 351–376.

S. Ma, X. Song, and J. Huang, *Supervised group Lasso with applications to microarray data analysis*, BMC bioinformatics, 8 (2007), p. 60.

J. L. Morales, *A numerical study of limited memory BFGS methods*, Applied Mathematics Letters, 15 (2002), pp. 481–487.

B. S. Mordukhovich and N. M. Nam, *An easy path to convex analysis and applications*, vol. 6, Morgan & Claypool Publishers, 2013.

Y. Nesterov, *A method of solving a convex programming problem with convergence rate \(O(1/k^2)\)*, Soviet Mathematics Doklady, 27(2) (1983), pp. 372–376.

Y. Nesterov, *Gradient methods for minimizing composite functions*, Mathematical Programming, 140 (2013), pp. 125–161.

J. Nutini, M. Schmidt, and W. Hare, *Active-set complexity of proximal gradient: How long does it take to find the sparsity pattern?*, Optimization Letters, 13 (2019), pp. 645–655.

P. Richtárik and M. Takáč, *Parallel coordinate descent methods for big data optimization*, Mathematical Programming, 156 (2016), pp. 433–484.

R. Tapprenen, P. Richtárik, and J. Gondzio, *Inexact coordinate descent: complexity and preconditioning*, J. of Optimization Theory and Applications, 170 (2016), pp. 144–176.

S. J. Wright, R. D. Nowak, and M. A. Figueiredo, *Sparse reconstruction by separable approximation*, IEEE Transactions on Signal Processing, 57 (2009), pp. 2479–2493.

Y. Yang and H. Zou, *A fast unified algorithm for solving group-lasso penalize learning problems*, Statistics and Computing, 25 (2015), pp. 1129–1141.

G.-X. Yuan, C.-H. Ho, and C.-J. Lin, *An improved GLMNET for \(\ell_1\)-regularized logistic regression*, Journal of Machine Learning Research, 13 (2012), pp. 1999–2030.

M. Yuan and Y. Lin, *Model selection and estimation in regression with grouped variables*, J. of the Royal Statistical Society: Series B (Statistical Methodology), 68 (2006), pp. 49–67.

Y. Zeng and P. Breheny, *Overlapping group logistic regression with applications to genetic pathway selection*, Cancer Informatics, 15 (2016), pp. CIN–S40043.