SIAM J. OPTIM.
Vol. 27, No. 3, pp. 1583–1610

A REDUCED-SPACE ALGORITHM FOR MINIMIZING ℓ_1-REGULARIZED CONVEX FUNCTIONS[*]

TIANYI CHEN[†], FRANK E. CURTIS[‡], AND DANIEL P. ROBINSON[†]

Abstract. We present a new method for minimizing the sum of a differentiable convex function and an ℓ_1-norm regularizer. The main features of the new method include: (i) an evolving set of indices corresponding to variables that are predicted to be nonzero at a solution (i.e., the support); (ii) a reduced-space subproblem defined in terms of the predicted support; (iii) conditions that determine how accurately each subproblem must be solved, which allow for Newton, linear conjugate gradient, and coordinate-descent techniques to be employed; (iv) a computationally practical condition that determines when the predicted support should be updated; and (v) a reduced proximal gradient step that ensures sufficient decrease in the objective function when it is decided that variables should be added to the predicted support. We prove a convergence guarantee for our method and demonstrate its efficiency on a large set of model prediction problems.

Key words. nonlinear optimization, convex optimization, sparse optimization, active-set methods, reduced-space methods, subspace minimization, model prediction

AMS subject classifications. 90C06, 90C25, 90C30, 90C55, 90C90, 49J52, 49M37, 62–07, 62M20, 65K05

DOI. 10.1137/16M1062259

1. Introduction.

In this paper, we propose, analyze, and provide the results of numerical experiments for a new method for solving ℓ_1-norm regularized convex optimization problems of the form

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ F(x), \quad \text{where } F(x) := f(x) + \lambda\|x\|_1, \tag{1}$$

$f : \mathbb{R}^n \to \mathbb{R}$ is a twice continuously differentiable convex function, and $\lambda > 0$ is a weighting parameter. A necessary and sufficient optimality condition for (1) is

$$0 \in \partial F(x) = \nabla f(x) + \lambda\partial\|x\|_1 \tag{2}$$

with ∂F and $\partial\|\cdot\|_1$ denoting the subdifferentials of F and $\|\cdot\|_1$, respectively. Our method for solving (1) generates a sequence of iterates such that any limit point of the sequence satisfies (2). It is applicable when only first-order derivative information is computed but is most effective when one can at least approximate second-order derivative matrices, e.g., using limited-memory quasi-Newton techniques.

Problems of the form (1) arise in statistics, signal processing, and machine learning applications and are often associated with data fitting or maximum likelihood estimation. A popular setting is binary classification using logistic regression (f is a

[*]Received by the editors February 19, 2016; accepted for publication (in revised form) March 9, 2017; published electronically August 8, 2017.
http://www.siam.org/journals/siopt/27-3/M106225.html
Funding: The first and third authors were supported by IDIES Seed Funding at Johns Hopkins University. The second author was supported by the U.S. Department of Energy, Office of Science, Applied Mathematics, Early Career Research Program under Award Number DE–SC0010615, and by the U.S. National Science Foundation, Division of Mathematical Sciences, Computational Mathematics Program under Award Number DMS–1016291.

[†]Department of Applied Mathematics and Statistics, Johns Hopkins University, Baltimore, MD 21218 (tchen59@jhu.edu, daniel.p.robinson@gmail.com).
[‡]Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA 18015 (frank.e.curtis@gmail.com).

logistic cost function), although instances of such problems also arise when performing multiclass logistic regression, for example. Instances of (1) also surface when using LASSO or elastic-net formulations to perform data analysis and discovery, e.g., in the clustering of data drawn from a union of subspaces [10, 30, 31, 32].

1.1. Literature review and our key contributions. Popular first-order optimization methods for solving (1) include ISTA, FISTA, and SpaRSA [2, 29]. Second-order methods have also been proposed, which can roughly be split into three classes: continuously differentiable bound-constrained reformulations [4, 6, 15, 19, 21, 22, 23, 24], proximal-Newton methods [5, 16, 20, 25, 33], and orthant-based methods [1, 3, 18]. Continuously differentiable bound-constrained reformulations transform the unconstrained nondifferentiable problem (1) into a larger but equivalent bound-constrained continuously differentiable optimization problem (equivalent in the sense that a solution to the bound-constrained formulation directly emits a solution to (1)), which may be solved using standard bound-constrained optimization solvers such as those cited above. The other two classes more directly attack problem (1). In particular, proximal-Newton methods solve problem (1) by minimizing a sequence of subproblems formed as the sum of a quadratic approximation to f and the nonsmooth ℓ_1-norm regularizer. For example, the state-of-the-art software LIBLINEAR, which implements newGLMNET [33], uses a coordinate descent algorithm to approximately minimize each piecewise quadratic subproblem. Orthant-based methods, on the other hand, minimize smooth quadratic approximations to (1) over a sequence of orthants in $\mathbb{R}^n$ until a solution is found. Of particular interest is the recently proposed orthant-based method OBA [18] in which every iteration consists of a corrective cycle of orthant predictions and subspace minimization steps. OBA was shown to be slower than LIBLINEAR when the Hessian matrices were diagonally dominant but faster otherwise, at least on the collection of test problems considered in [18].

Since LIBLINEAR and OBA are the most relevant to the algorithm described in this paper, let us discuss their respective advantages and disadvantages in more detail. The key advantage of LIBLINEAR is its use of a coordinate descent (CD) algorithm to approximately minimize the piecewise quadratic subproblem. The use of CD means that one should expect excellent performance on problems whose Hessian matrices are strongly diagonally dominant. This expectation was confirmed, as mentioned above, by the OBA paper [18]. For some problems encountered in model prediction, e.g., when using logistic regression to perform classification, the Hessians are often strongly diagonally dominant, at least after certain data scaling techniques are used. However, not all prediction problems have such nice diagonal dominance properties, and in some instances the user would prefer to avoid discovering a proper scaling for their data. In these latter cases, the OBA method is typically superior.

Another potential advantage of the OBA method is its use of an active-set strategy that uses quadratic subproblems that are smaller in dimension than the ambient space. For many ℓ_1-norm regularized prediction problems, the number of nonzero components in a solution is a small percentage of the ambient dimension, and thus OBA spends most of its time solving small dimensional problems. This is an advantage, at least when the zero and nonzero structure of the solution is quickly identified.

We have the perspective that both LIBLINEAR and OBA are valuable state-of-the-art algorithms that complement each other. Our *fast reduced space algorithm* (FaRSA) is designed to capitalize on the advantages of both while avoiding their disadvantages. The following bulleted points summarize our key contributions.

(i) We present a new active-set line search method that utilizes reduced-space quadratic subproblems, approximate solutions of which can be computed efficiently.

Although similar subproblems are used by OBA, the precise manner in which they are formulated as well as the conditions used to terminate their solution are different.

(ii) Unlike the active-set OBA method, our method does not require the computation of an ISTA step during each iteration to ensure convergence. We achieve convergence by combining a new projected backtracking line search procedure, an approximate subspace minimization scheme, and a mechanism for determining when the support of the solution estimate should be updated.

(iii) Our framework is flexible. In particular, we introduce a new set of conditions that signal how accurately each quadratic subproblem should be solved and allow for various subproblem solvers to be used. In so doing, our method easily accommodates a CG subproblem solver as in OBA and a CD solver as in LIBLINEAR. Interestingly, this allows for multiple subproblem solvers to be used in parallel, thus allowing for numerical performance that can be as good as either LIBLINEAR and OBA regardless of whether the problem Hessians are strongly diagonally dominant.

(iv) As demonstrated in the numerical experiments described in this paper, a basic MATLAB implementation of FaRSA has a practical performance that is often better than OBA in terms of computational time on model prediction problems.

We remark that our proposed algorithm has similarities with the iterative method that one would obtain using the following procedure: (i) at a given iterate x_k, construct a quadratic model of f and recast the minimization of this model plus the regularization term $\lambda\|x\|_1$ into a bound-constrained quadratic optimization problem (similarly to the procedure in SpaRSA); (ii) approximately solve this subproblem using the techniques in [7, 8, 9] (see also [28]); and (iii) translate the resulting solution back into the space of x variables to produce a trial step from x_k, call it d_k. Indeed, our initial developments were based on these ideas. However, the algorithm proposed in this paper involves some deviations and enhancements from this starting point.

We end this review by noting that the use of subspace minimization is certainly not new. Serafini, Zanghirati, and Zanni [26, 27, 34, 35] showed that subspace minimization can be very effective for solving support vector machine optimization problems. In these works, which were based on the ideas first proposed by Joachims [17], the subspace selection procedure involves the solution of an optimization problem that includes a constraint that bounds the size of the chosen subspace. The subspace selection procedure was further refined and generalized to a broader class of problems by Gonzalez-Lima, Hager, and Zhang [14]. The improved subspace selection was the result of adding a simple quadratic term to the objective function of the optimization problem used to choose the subspace. Finally, the philosophy behind our subspace procedure is similar in spirit to that developed for strictly convex bound-constrained quadratic problems (BCQP) [7], convex BCQP [11, 12, 13], and nonconvex BCQP [22].

1.2. Notation. Let $\mathcal{I} \subseteq \{1, 2, \ldots, n\}$ denote an index set of variables. For any $v \in \mathbb{R}^n$, we let $[v]_\mathcal{I}$ denote the subvector of v consisting of elements of v with indices in $\mathcal{I}$. Similarly, for any symmetric matrix $M \in \mathbb{R}^{n\times n}$, we let $[M]_{\mathcal{I},\mathcal{I}}$ denote the submatrix of M consisting of the rows and columns of M that correspond to the index set $\mathcal{I}$. If, in addition, the index set $\mathcal{I}$ satisfies $[x]_i \neq 0$ for all $i \in \mathcal{I}$, then we let $\nabla_\mathcal{I} F(x)$ and $\nabla^2_{\mathcal{I}\mathcal{I}} F(x)$ denote the vector of first derivatives and matrix of second derivatives of F at x, respectively, corresponding to the elements in $\mathcal{I}$. For any vector v, we let $\operatorname{sgn}(v)$ denote the vector of the same length as v whose ith component is 0 when $[v]_i = 0$, is 1 when $[v]_i > 0$, and is -1 when $[v]_i < 0$. For any vector v, we let $\|v\|_1$ and $\|v\|$ denote its ℓ_1-norm and ℓ_2-norm, respectively.

2. Algorithm FaRSA.

Crucial to our algorithm is the manner in which we handle the zero and nonzero components of a solution estimate. In order to describe the details of our approach, we first define the index sets

$$\mathcal{I}^0(x) := \{i : [x]_i = 0\}, \quad \mathcal{I}^+(x) := \{i : [x]_i > 0\}, \quad \text{and} \quad \mathcal{I}^-(x) := \{i : [x]_i < 0\}.$$

We call $\mathcal{I}^0(x)$ the set of *zero variables*, $\mathcal{I}^+(x)$ the set of *positive variables*, $\mathcal{I}^-(x)$ the set of *negative variables*, and the union of $\mathcal{I}^-(x)$ and $\mathcal{I}^+(x)$ the set of *nonzero variables* at x. We use these sets to define measures of optimality corresponding to the zero and nonzero variables at x. Respectively, these measures are as follows:

$$[\beta(x)]_i := \begin{cases} \nabla_i f(x) + \lambda & \text{if } i \in \mathcal{I}^0(x) \text{ and } \nabla_i f(x) + \lambda < 0, \\ \nabla_i f(x) - \lambda & \text{if } i \in \mathcal{I}^0(x) \text{ and } \nabla_i f(x) - \lambda > 0, \\ 0 & \text{otherwise;} \end{cases}$$

$$[\phi(x)]_i :=$$
$$\begin{cases} 0 & \text{if } i \in \mathcal{I}^0(x), \\ \min\{\nabla_i f(x) + \lambda, \max\{[x]_i, \nabla_i f(x) - \lambda\}\} & \text{if } i \in \mathcal{I}^+(x) \text{ and } \nabla_i f(x) + \lambda > 0, \\ \max\{\nabla_i f(x) - \lambda, \min\{[x]_i, \nabla_i f(x) + \lambda\}\} & \text{if } i \in \mathcal{I}^-(x) \text{ and } \nabla_i f(x) - \lambda < 0, \\ \nabla_i f(x) + \lambda \cdot \operatorname{sgn}([x]_i) & \text{otherwise.} \end{cases}$$

If x was a solution to problem (1), then for any $i \in \mathcal{I}^0(x)$, it holds from the optimality conditions for problem (1) that $|\nabla_i f(x)| \leq \lambda$. Therefore, the size of $[\beta(x)]_i$ indicates how far is that *zero* variable from being optimal. In short, the size of the vector $\beta(x)$ is a measure of optimality for the zero variables, i.e., for those in $\mathcal{I}^0(x)$. On the other hand, for any $i \in \mathcal{I}^+(x) \cup \mathcal{I}^-(x)$, it holds from the optimality conditions that $\nabla_i f(x) + \operatorname{sgn}([x]_i)\lambda = 0$. Therefore, the size of $[\phi(x)]_i$ indicates how far that *nonzero* variable is from being optimal, although we note that its definition also takes into account the distance the nonzero variable can move before becoming zero, i.e., before switching orthants. In short, the size of the vector $\phi(x)$ is a measure of optimality for the nonzero variables, i.e., for those in $\mathcal{I}^+(x) \cup \mathcal{I}^-(x)$. (See Lemma A.1 in Appendix A for an explanation of how β and ϕ are related to the well-known ISTA iteration.)

The following result shows that the functions β and ϕ together correspond to a valid optimality measure for problem (1).

LEMMA 2.1. *Let $\mathcal{S}$ be an infinite set of positive integers such that $\{x_k\}_{k\in\mathcal{S}} \to x_*$. Then, the x_* is an optimal solution to* (1) *if and only if $\{\beta(x_k)\}_{k\in\mathcal{S}} \to 0$ and $\{\phi(x_k)\}_{k\in\mathcal{S}} \to 0$. Consequently, x_* is an optimal solution to* (1) *if and only if $\|\beta(x_*)\| = \|\phi(x_*)\| = 0$.*

Proof. Suppose $\{\beta(x_k)\}_{k\in\mathcal{S}} \to 0$ and $\{\phi(x_k)\}_{k\in\mathcal{S}} \to 0$. Then, first, consider any i such that $[x_*]_i > 0$, which means that $[x_k]_i > 0$ for all sufficiently large $k \in \mathcal{S}$. We now consider two subcases. If $\nabla_i f(x_k) + \lambda \leq 0$ for infinitely many $k \in \mathcal{S}$, then it follows from the definition of $\phi(x_k)$, $\{\phi(x_k)\}_{k\in\mathcal{S}} \to 0$, and continuity of ∇f that $\nabla_i f(x_*) + \lambda = 0$. On the other hand, if $\nabla_i f(x_k) + \lambda > 0$ for infinitely many $k \in \mathcal{S}$, then it follows from the definition of $\phi(x_k)$, $\{\phi(x_k)\}_{k\in\mathcal{S}} \to 0$, $[x_*]_i > 0$, and continuity of ∇f that $\nabla_i f(x_*) + \lambda = 0$. By combining both cases, we have established that $\nabla_i f(x_*) + \lambda = 0$, so that the ith component satisfies the optimality conditions (2). A similar argument may be used for the case when one considers i such that $[x_*]_i < 0$ to show that $\nabla_i f(x_*) - \lambda = 0$.

It remains to consider i such that $[x_*]_i = 0$. We have four subcases to consider. First, if infinitely many $k \in \mathcal{S}$ satisfy $[x_k]_i = 0$ and $\nabla_i f(x_k) + \lambda < 0$, then it follows from the definition of $\beta(x_k)$, $\{\beta(x_k)\}_{k\in\mathcal{S}} \to 0$, and continuity of ∇f that $\nabla_i f(x_*) + \lambda = 0$; a similar argument shows that if infinitely many $k \in \mathcal{S}$ satisfy $[x_k]_i = 0$ and $\nabla_i f(x_k) - \lambda > 0$, then $\nabla_i f(x_*) - \lambda = 0$. Second, if infinitely many $k \in \mathcal{S}$ satisfy $[x_k]_i = 0$ and $|\nabla_i f(x_k)| < \lambda$, then, trivially, $|\nabla_i f(x_*)| \le \lambda$. Third, if infinitely many $k \in \mathcal{S}$ satisfy $[x_k]_i > 0$ and $\nabla_i f(x_k) + \lambda \le 0$, then it follows from the definition of $\phi(x_k)$, $\{\phi(x_k)\}_{k\in\mathcal{S}} \to 0$, and continuity of ∇f that $\nabla_i f(x_*) + \lambda = 0$; a similar argument shows that if infinitely many $k \in \mathcal{S}$ satisfy $[x_k]_i < 0$ and $\nabla_i f(x_k) - \lambda \ge 0$, then $\nabla_i f(x_*) - \lambda = 0$. Fourth, if infinitely many $k \in \mathcal{S}$ satisfy $[x_k]_i > 0$ and $\nabla_i f(x_k) + \lambda > 0$, then it follows from the definition of $\phi(x_k)$, $\{\phi(x_k)\}_{k\in\mathcal{S}} \to 0$, and continuity of ∇f that $|\nabla_i f(x_*)| \le \lambda$; a similar argument shows that if infinitely many $k \in \mathcal{S}$ satisfy $[x_k]_i < 0$ and $\nabla_i f(x_k) - \lambda < 0$, then $|\nabla_i f(x_*)| \le \lambda$. By combining these subcases, we conclude that $|\nabla_i f(x_*)| \le \lambda$, so the ith component satisfies the optimality condition (2).

To prove the reverse implication, suppose that x_* is a solution to problem (1). If $[x_*]_i > 0$, then $[\beta(x_k)]_i = 0$ for all sufficiently large $k \in \mathcal{S}$ and $\{[\phi(x_k)]_i\}_{k\in\mathcal{S}} \to 0$ since $\nabla_i f(x_*) + \lambda = 0$ and ∇f is continuous. If $[x_*]_i < 0$, then $[\beta(x_k)]_i = 0$ for all sufficiently large $k \in \mathcal{S}$ and $\{[\phi(x_k)]_i\}_{k\in\mathcal{S}} \to 0$ since $\nabla_i f(x_*) - \lambda = 0$ and ∇f is continuous. Finally, if $[x_*]_i = 0$, then $|\nabla_i f(x_*)| \le \lambda$ and continuity of ∇f imply that $\{[\beta(x_k)]_i\}_{k\in\mathcal{S}} \to 0$ and $\{[\phi(x_k)]_i\}_{k\in\mathcal{S}} \to 0$. This completes the proof. $\square$

We now state our proposed method, FaRSA, as Algorithm 1. When considering a reduced-space subproblem defined by a chosen index set $\mathcal{I}_k$ (see lines 7 and 14), the algorithm makes use of a quadratic model of the objective of the form (see line 10)

$$m_k(d) := g_k^T d + \tfrac{1}{2} d^T H_k d.$$

FaRSA also makes use of two line search subroutines, stated as Algorithms 2 and 3, the former of which employs the following projection operator dependent on x_k:

$$[\mathrm{Proj}(y\,; x_k)]_i := \begin{cases} \max\{0, [y]_i\} & \text{if } \mathcal{I}^+(x_k), \\ \min\{0, [y]_i\} & \text{if } \mathcal{I}^-(x_k), \\ 0 & \text{if } \mathcal{I}^0(x_k). \end{cases}$$

FaRSA computes a sequence of iterates $\{x_k\}$. During each iteration, the sets $\mathcal{I}^0(x_k)$, $\mathcal{I}^+(x_k)$, and $\mathcal{I}^-(x_k)$ are identified, which are used to define $\beta(x_k)$ and $\phi(x_k)$. We can see in line 4 of Algorithm 1 that when both $\|\beta(x_k)\|$ and $\|\phi(x_k)\|$ are less than a prescribed tolerance $\epsilon > 0$, it returns x_k as an approximate solution to (1); this is justified by Lemma 2.1. Otherwise, it proceeds in one of two ways depending on the relative sizes of $\|\beta(x_k)\|$ and $\|\phi(x_k)\|$. We describe these two cases next.

(i) The relationship $\|\beta(x_k)\| \le \gamma\|\phi(x_k)\|$ indicates that significant progress toward optimality can still be achieved by reducing F over the current set of nonzero variables at x_k; lines 7–12 are designed for this purpose. In line 7, a subset $\mathcal{I}_k$ of variables are chosen such that the norm of $\phi(x_k)$ over that subset of variables is at least proportional to the norm of $\phi(x_k)$ over the full set of variables. This allows control over the size of the subproblem, which may be as small as one-dimensional. Note that for $i \in \mathcal{I}_k$, it must hold that $[\phi(x_k)]_i \ne 0$, which in turn means that $i \notin \mathcal{I}^0(x_k)$; i.e., the ith variable is nonzero. This means that the reduced space subproblem to minimize $m_k(d)$ over $d \in \mathbb{R}^{|\mathcal{I}_k|}$ is aimed at minimizing F over the variables in $\mathcal{I}_k$. Our analysis does not require an exact minimizer of m_k. Rather, we allow for the computation of any direction $\bar{d}_k$ that satisfies the conditions in line 10, namely, $g_k^T \bar{d}_k \le g_k^T d_k^R$ and

Algorithm 1. FaRSA for solving problem (1).

```
1:  **Input:** $x_0$
2:  **Constants:** $\{\eta_\phi, \eta_\beta\} \subset (0,1]$, $, \xi \in (0,1)$, $\eta \in (0,1/2]$, and $\{\gamma, \epsilon\} \subset (0,\infty)$
3:  **for** $k = 0,1,2,\ldots$ **do**
4:      **if** $\max\{\|\beta(x_k)\|, \|\phi(x_k)\|\} \le \epsilon$ **then**
5:          **Return** the (approximate) solution $x_k$ of problem (1).
6:      **if** $\|\beta(x_k)\| \le \gamma\|\phi(x_k)\|$ **then**                    $[k \in \mathcal{S}_\phi]$
7:          Choose any $\mathcal{I}_k \subseteq \{i : [\phi(x_k)]_i \neq 0\}$ such that $\|[\phi(x_k)]_{\mathcal{I}_k}\| \ge \eta_\phi\|\phi(x_k)\|$.
8:          Set $H_k \leftarrow \nabla^2_{\mathcal{I}_k\mathcal{I}_k} F(x_k)$ and $g_k \leftarrow \nabla_{\mathcal{I}_k} F(x_k)$.
9:          Compute the reference direction

                $d_k^R \leftarrow -\alpha_k g_k$,  where  $\alpha_k \leftarrow \|g_k\|^2/(g_k^T H_k g_k)$.

10:         Compute any $\bar{d}_k \approx \text{argmin}\ m_k(d)$ such that the following inequalities hold:

                $g_k^T \bar{d}_k \le g_k^T d_k^R$  and  $m_k(\bar{d}_k) \le m_k(0)$.

11:         Set $[d_k]_{\mathcal{I}_k} \leftarrow \bar{d}_k$ and $[d_k]_i \leftarrow 0$ for $i \notin \mathcal{I}_k$.
12:         Use Algorithm 2 to compute $x_{k+1} \leftarrow \text{LINESEARCH\_}\phi(x_k, d_k, \mathcal{I}_k, \eta, \xi)$.
13:     **else**                                                               $[k \in \mathcal{S}_\beta]$
14:         Choose any $\mathcal{I}_k \subseteq \{i : [\beta(x_k)]_i \neq 0\}$ such that $\|[\beta(x_k)]_{\mathcal{I}_k}\| \ge \eta_\beta\|[\beta(x_k)\|$.
15:         Set $[d_k]_{\mathcal{I}_k} \leftarrow -[\beta(x_k)]_{\mathcal{I}_k}$ and $[d_k]_i \leftarrow 0$ for $i \notin \mathcal{I}_k$.
16:         Use Algorithm 3 to compute $x_{k+1} \leftarrow \text{LINESEARCH\_}\beta(x_k, d_k, \eta, \xi)$.
```

Algorithm 2. A line search procedure for computing x_{k+1} when $k \in \mathcal{S}_\phi$.

```
1:  **procedure** $x_{k+1} = \text{LINESEARCH\_}\phi(x_k, d_k, \mathcal{I}_k, \eta, \xi)$
2:      Set $j \leftarrow 0$ and $y_0 \leftarrow \text{Proj}(x_k + d_k\,; x_k)$.
3:      **while** $\text{sgn}(y_j) \neq \text{sgn}(x_k)$ **do**
4:          **if** $F(y_j) \le F(x_k)$ **then**
5:              **return** $x_{k+1} \leftarrow y_j$.                                $[k \in \mathcal{S}_\phi^{\text{ADD}}]$
6:          Set $j \leftarrow j+1$ and then $y_j \leftarrow \text{Proj}(x_k + \xi^j d_k\,; x_k)$.
7:      **if** $j \neq 0$ **then**
8:          Set $\alpha_B \leftarrow \text{argsup}\,\{\alpha > 0 : \text{sgn}(x_k + \alpha d_k) = \text{sgn}(x_k)\}$.
9:          Set $y_j \leftarrow x_k + \alpha_B d_k$.
10:         **if** $F(y_j) \le F(x_k) + \eta\alpha_B \nabla_{\mathcal{I}_k} F(x_k)^T [d_k]_{\mathcal{I}_k}$ **then**
11:             **return** $x_{k+1} \leftarrow y_j$.                                $[k \in \mathcal{S}_\phi^{\text{ADD}}]$
12:     **loop**
13:         **if** $F(y_j) \le F(x_k) + \eta\xi^j \nabla_{\mathcal{I}_k} F(x_k)^T [d_k]_{\mathcal{I}_k}$ **then**
14:             **return** $x_{k+1} \leftarrow y_j$.                                $[k \in \mathcal{S}_\phi^{\text{SD}}]$
15:         Set $j \leftarrow j+1$ and then $y_j \leftarrow x_k + \xi^j d_k$.
```

$m_k(\bar{d}_k) \le m_k(0)$, where the reference direction d_k^R is computed in line 9 by minimizing m_k along the steepest descent direction. The first condition imposes how much descent is required by the search direction $\bar{d}_k$, while the second condition ensures that the model is reduced at least as much as a zero step. It will be shown (see Lemma 3.9) that the first condition guarantees that the decrease in F for any iteration $k \in \mathcal{S}_\phi^{\text{SD}}$ is bounded below by a positive constant factor of $\|g_k\|_2^2$ (see (15)). It will be shown

Algorithm 3. A line search procedure for computing x_{k+1} when $k \in \mathcal{S}_\beta$.

```
1: procedure $x_{k+1}$ = LINESEARCH_$\beta(x_k, d_k, \eta, \xi)$
2:     Set $j \leftarrow 0$ and $y_0 \leftarrow x_k + d_k$.
3:     while $F\big(y_j\big) > F(x_k) - \eta\xi^j\|d_k\|^2$ do
4:         Set $j \leftarrow j + 1$ and then $y_j \leftarrow x_k + \xi^j d_k$.
5:     return $x_{k+1} \leftarrow y_j$.
```

(see Lemma 3.8) that the second condition ensures that $\bar{d}_k$ is bounded by a multiple of $\|g_k\|$. Such conditions are satisfied by a Newton step, by any CG iterate, and asymptotically by CD iterates. Once $\bar{d}_k$ is obtained, the search direction d_k in the full space is obtained by filling its elements that correspond to the index set $\mathcal{I}_k$ with the elements from $\bar{d}_k$ and setting the complementary set of variables to zero (see line 11). With the search direction d_k computed, we call Algorithm 2 in line 12, which performs a (nonstandard) backtracking projected line search. This line search procedure makes use of the projection operator $\text{Proj}(\cdot\,; x_k)$. This operator projects vectors onto the orthant inhabited by x_k, a feature shared by OBA. The while-loop that starts in line 3 of Algorithm 2 checks whether the trial point y_j decreases the objective function F relative to its value at x_k when $\text{sgn}(y_j) \neq \text{sgn}(x_k)$. If the line search terminates in this while-loop, then this implies that at least one component of x_k that was nonzero has become zero for $x_{k+1} = y_j$. Since the dimension of the reduced space will therefore be reduced during the next iteration (provided line 6 of Algorithm 1 tests true), the procedure only requires $F(x_{k+1}) \leq F(x_k)$ instead of a more traditional sufficient decrease condition, e.g., one based on the Armijo condition. If line 7 of Algorithm 2 is reached, then the current trial iterate y_j satisfies $\text{sgn}(y_j) = \text{sgn}(x_k)$; i.e., the trial iterate has entered the same orthant as that inhabited by x_k. Once this has occurred, the method could then perform a standard backtracking Armijo line search as stipulated in the loop starting at line 12. For the purpose of guaranteeing convergence, however, the method first checks whether the largest step along d_k that stays in the same orthant as x_k (see lines 8 and 9) satisfies the Armijo sufficient decrease condition (see line 10). (This aspect makes our procedure different from a standard backtracking scheme.) If Algorithm 2 terminates in line 5 or 11, then at least one nonzero variable at x_k will have become zero at x_{k+1}, which we indicate by saying $k \in \mathcal{S}_\phi^{\text{ADD}} \subseteq \mathcal{S}_\phi$. Otherwise, if Algorithm 2 terminates in line 14, then x_{k+1} and x_k are housed in the same orthant and sufficient decrease in F was achieved (i.e., the Armijo condition in line 13 was satisfied). Since sufficient decrease has been achieved in this case, we say that $k \in \mathcal{S}_\phi^{\text{SD}} \subseteq \mathcal{S}_\phi$.

(ii) When $\|\beta(x_k)\| > \gamma\|\phi(x_k)\|$, progress toward optimality is best achieved by freeing at least one variable that is currently set to zero; lines 14–16 are designed for this purpose. Since $\|\beta(x_k)\|$ is relatively large, in line 14 of Algorithm 1 a subset $\mathcal{I}_k$ of variables is chosen such that the norm of $\beta(x_k)$ over that subset of variables is at least proportional to the norm of $\beta(x_k)$ over the full set of variables. Similar to the previous case, this allows control over the size of the subproblem, which in the extreme case may be one-dimensional. If $i \in \mathcal{I}_k$, then $[\beta(x_k)]_i \neq 0$, which in turn means that $i \in \mathcal{I}^0(x_k)$; i.e., the ith variable has the value zero. The components of $\beta(x_k)$ that correspond to $\mathcal{I}_k$ are then used to define the search direction d_k in line 15. With the search direction d_k computed, Algorithm 3 is called in line 16, which performs a standard backtracking Armijo line search to obtain x_{k+1}. If a unit step length is taken, i.e., if $x_{k+1} = x_k + d_k$, then x_{k+1} can be interpreted as the iterate that would be obtained by taking a *reduced ISTA step* in the space of variables indexed by $\mathcal{I}_k$. (For additional details, see Lemma A.1 in Appendix A.)

3. Convergence Analysis.

Our analysis uses the following assumption that is assumed to hold throughout this section.

Assumption 3.1. The function $f : \mathbb{R}^n \to \mathbb{R}$ is convex, twice continuously differentiable, and bounded below on the level set $\mathcal{L} := \{x \in \mathbb{R}^n : F(x) \leq F(x_0)\}$. The gradient function $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$ is Lipschitz continuous on $\mathcal{L}$ with Lipschitz constant L. The Hessian function $\nabla^2 f : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ is uniformly positive definite and bounded on $\mathcal{L}$; i.e., there exist positive constants $\theta_{\min}$ and $\theta_{\max}$ such that

$$\theta_{\min}\|v\|^2 \leq v^T H(x) v \leq \theta_{\max}\|v\|^2 \text{ for all } \{x, v\} \subset \mathbb{R}^n.$$

The following remark concerning Assumption 3.1 is important.

Remark 3.2. The assumption that the Hessian function $\nabla^2 f$ is uniformly positive definite and bounded on $\mathcal{L}$ in Assumption 3.1 has been made for simplicity. Such an assumption can easily be replaced by the requirement that the sequence of matrices $\{H_k\}$ used in line 8 of Algorithm 1 be chosen as any symmetric matrices with eigenvalues that are uniformly bounded above and away from zero. In this case, the convergence results presented in this paper for Algorithm 1 also apply when f is merely convex.

Our analysis uses the index sets (already shown in Algorithms 1–2)

$$\begin{aligned}
\mathcal{S}_\phi &:= \{k : \text{lines 7–12 in Algorithm 1 are performed during iteration } k\};\\
\mathcal{S}_\phi^{\text{ADD}} &:= \{k \in \mathcal{S}_\phi : \operatorname{sgn}(x_{k+1}) \neq \operatorname{sgn}(x_k)\};\\
\mathcal{S}_\phi^{\text{SD}} &:= \{k \in \mathcal{S}_\phi : \operatorname{sgn}(x_{k+1}) = \operatorname{sgn}(x_k)\}; \text{and}\\
\mathcal{S}_\beta &:= \{k : \text{lines 14–16 in Algorithm 1 are performed during iteration } k\}.
\end{aligned}$$

We start with a lemma that establishes an important identity for iterations in $\mathcal{S}_\beta$.

LEMMA 3.3. *If $k \in \mathcal{S}_\beta$, then $(\mathcal{I}_k, d_k)$ in lines 14 and 15 of Algorithm 1 yield*

$$[d_k]_{\mathcal{I}_k} = -[\nabla f(x_k) + \lambda \cdot \operatorname{sgn}(x_k + \xi^j d_k)]_{\mathcal{I}_k} \text{ for any integer } j. \tag{3}$$

Consequently, the right-hand side of (3) *has the same value for any integer j.*

Proof. We prove that (3) holds for an arbitrary element of $\mathcal{I}_k$. To this end, let j be any integer and $i \in \mathcal{I}_k \subseteq \{\ell : [\beta(x_k)]_\ell \neq 0]\}$, where $\mathcal{I}_k$ is defined in line 14. It follows from the definition of $\mathcal{I}_k$, the definition of d_k in line 15, and $i \in \mathcal{I}_k$ that

$$[d_k]_i = \begin{cases} -\big(\nabla_i f(x_k) + \lambda\big) & \text{if } \nabla_i f(x_k) + \lambda < 0,\\ -\big(\nabla_i f(x_k) - \lambda\big) & \text{if } \nabla_i f(x_k) - \lambda > 0,\end{cases} \tag{4}$$

so that $[d_k]_i \neq 0$. Also, since $[x_k]_i = 0$ for $i \in \mathcal{I}_k$, we know that $[x_k + \xi^j d_k]_i \neq 0$. Thus, we need only consider the following two cases.

Case 1. Suppose $[x_k + \xi^j d_k]_i > 0$. In this case, the right-hand side of (3) is equal to $-(\nabla_i f(x_k) + \lambda)$. As for the left-hand side, since $[x_k]_i = 0$ and $[x_k + \xi^j d_k]_i > 0$, we have $0 < [x_k + \xi^j d_k]_i = \xi^j [d_k]_i$, which combined with $\xi^j > 0$ means that $[d_k]_i > 0$. This fact and (4) gives $[d_k]_i = -(\nabla_i f(x_k) + \lambda)$, so (3) holds.

Case 2. Suppose $[x_k + \xi^j d_k]_i < 0$. In this case, the right-hand side of (3) is equal to $-(\nabla_i f(x_k) - \lambda)$. As for the left-hand side, since $[x_k]_i = 0$ and $[x_k + \xi^j d_k]_i < 0$, we have $0 > [x_k + \xi^j d_k]_i = \xi^j [d_k]_i$, which when combined with $\xi^j > 0$ means that $[d_k]_i < 0$. This fact and (4) gives $[d_k]_i = -(\nabla_i f(x_k) - \lambda)$, so (3) holds. ∎

We can now establish a bound for a decrease in the objective when $k \in \mathcal{S}_\beta$.

LEMMA 3.4. *If $k \in \mathcal{S}_\beta$, then d_k in line 15 of Algorithm 1 yields*

$$F(x_k + \xi^j d_k) \le F(x_k) - \frac{\xi^j}{2}\|d_k\|^2 \quad \textit{for any integer } j \textit{ with } 0 \le \xi^j \le \frac{1}{L}.$$

Proof. Let j be any integer with $0 \le \xi^j \le \frac{1}{L}$ and let $y_j := x_k + \xi^j d_k$. By Lipschitz continuity of the gradient function ∇f, we have

$$\begin{aligned} f(y_j) &\le f(x_k) + \xi^j \nabla f(x_k)^T d_k + \frac{L}{2}\|\xi^j d_k\|^2 \\ &\le f(x_k) + \xi^j \nabla f(x_k)^T d_k + \frac{\xi^j}{2}\|d_k\|^2. \end{aligned} \tag{5}$$

It then follows from (5), convexity of both f and $\lambda\|\cdot\|_1$, the fact that $\operatorname{sgn}(y_j) \in \partial\|y_j\|_1$, the definition of d_k (in particular that $[d_k]_i = 0$ for $i \notin \mathcal{I}_k$), and Lemma 3.3 that the following holds for all $z \in \mathbb{R}^n$:

$$\begin{aligned} F(y_j) &= f(y_j) + \lambda\|y_j\|_1 \\ &\le f(x_k) + \xi^j \nabla f(x_k)^T d_k + \frac{\xi^j}{2}\|d_k\|^2 + \lambda\|y_j\|_1 \\ &\le f(z) + \nabla f(x_k)^T(x_k - z) + \xi^j \nabla f(x_k)^T d_k + \frac{\xi^j}{2}\|d_k\|^2 \\ &\quad + \lambda\|z\|_1 + \lambda \cdot \operatorname{sgn}(y_j)^T(y_j - z) \\ &\le F(z) + [\nabla f(x_k) + \lambda \cdot \operatorname{sgn}(y_j)]^T(x_k - z) \\ &\quad + \xi^j[\nabla f(x_k) + \lambda \cdot \operatorname{sgn}(y_j)]^T d_k + \frac{\xi^j}{2}\|d_k\|^2 \\ &= F(z) + [\nabla f(x_k) + \lambda \cdot \operatorname{sgn}(y_j)]^T(x_k - z) - \xi^j\|d_k\|^2 + \frac{\xi^j}{2}\|d_k\|^2 \\ &= F(z) + [\nabla f(x_k) + \lambda \cdot \operatorname{sgn}(y_j)]^T(x_k - z) - \frac{\xi^j}{2}\|d_k\|^2. \end{aligned} \tag{6}$$

The desired result follows by considering $z = x_k$ in (6). ☐

We now show that Algorithm 3 called in line 16 of Algorithm 1 is well defined and that it returns x_{k+1} yielding sufficient decrease in the objective function.

LEMMA 3.5. *If $k \in \mathcal{S}_\beta$, then x_{k+1} satisfies*

$$F(x_{k+1}) \le F(x_k) - \kappa_\beta \max\{\|\beta(x_k)\|^2, \gamma^2\|\phi(x_k)\|^2\}, \tag{7}$$

where $\kappa_\beta := \eta\eta_\beta \min\{1, \xi/L\}$.

Proof. Let j be any integer with $0 \le \xi^j \le \frac{1}{L}$ and let $y_j := x_k + \xi^j d_k$. It follows from Lemma 3.4 and the fact that $\eta \in (0, 1/2]$ in Algorithm 1 that

$$F\big(y_j\big) \le F(x_k) - \frac{\xi^j}{2}\|d_k\|^2 \le F(x_k) - \eta\xi^j\|d_k\|^2.$$

It follows from this inequality that Algorithm 3 will return the vector $x_{k+1} = x_k + \xi^{j_*} d_k$ with $\xi^{j_*} \ge \min\{1, \xi/L\}$ when called in line 16 of Algorithm 1. Using this bound, line 3 of Algorithm 3, and lines 15 and 14 of Algorithm 1, we have

$$\begin{aligned} F(x_{k+1}) &\le F(x_k) - \eta \xi^{j_*} \|d_k\|^2 \le F(x_k) - \eta \min\{1, \xi/L\} \|d_k\|^2 \\ &= F(x_k) - \eta \min\{1, \xi/L\} \|[\beta(x_k)]_{\mathcal{I}_k}\|^2 \le F(x_k) - \eta \eta_\beta \min\{1, \xi/L\} \|\beta(x_k)\|^2. \end{aligned}$$

The inequality (7) follows from the definition of κ_β, the previous inequality, and the fact that the inequality in line 6 of Algorithm 1 must not hold since line 16 is assumed to be reached. ∎

We now show that the index set $\mathcal{S}_\beta$ must be finite.

LEMMA 3.6. *The index set $\mathcal{S}_\beta$ must be finite, i.e., $|\mathcal{S}_\beta| < \infty$.*

Proof. To derive a contradiction, suppose that $|\mathcal{S}_\beta| = \infty$, which also means that Algorithm 1 does not terminate finitely. Since Algorithm 1 does not terminate finitely, we know from line 4 of Algorithm 1 that $\max\{\|\beta(x_k)\|, \|\phi(x_k)\|\} > \epsilon$ for all $k \ge 0$. Combining this inequality with Lemma 3.5 and the fact that $F(x_{k+1}) \le F(x_k)$ for all $k \notin \mathcal{S}_\beta$ (as a result of Algorithm 2 called in line 12 of Algorithm 1), we may conclude for any nonnegative integer ℓ and $\kappa_\beta > 0$ defined in Lemma 3.5 that

$$\begin{aligned} F(x_0) - F(x_{\ell+1}) &= \sum_{k=0}^{\ell} \big[F(x_k) - F(x_{k+1})\big] \\ &\ge \sum_{k \in \mathcal{S}_\beta, k \le \ell} \big[F(x_k) - F(x_{k+1})\big] \\ &\ge \sum_{k \in \mathcal{S}_\beta, k \le \ell} \kappa_\beta \max\{\|\beta(x_k)\|^2, \gamma^2 \|\phi(x_k)\|^2\} \\ &\ge \sum_{k \in \mathcal{S}_\beta, k \le \ell} \kappa_\beta \min\{1, \gamma^2\} \epsilon^2. \end{aligned}$$

Rearranging the previous inequality shows that

$$\begin{aligned} \lim_{l \to \infty} F(x_{\ell+1}) &\le \lim_{\ell \to \infty} \left[F(x_0) - \sum_{k \in \mathcal{S}_\beta, k \le \ell} \kappa_\beta \min\{1, \gamma^2\} \epsilon^2 \right] \\ &= F(x_0) - \sum_{k \in \mathcal{S}_\beta} \kappa_\beta \min\{1, \gamma^2\} \epsilon^2 = -\infty, \end{aligned}$$

which contradicts Assumption 3.1. Thus, we conclude that $|\mathcal{S}_\beta| < \infty$. ∎

To prove that Algorithm 1 terminates finitely with an approximate solution to problem (1), all that remains is to prove that the set $\mathcal{S}_\phi$ is finite. To establish that $\mathcal{S}_\phi \equiv \mathcal{S}_\phi^{\text{ADD}} \cup \mathcal{S}_\phi^{\text{SD}}$ is finite, we proceed by showing individually that both $\mathcal{S}_\phi^{\text{ADD}}$ and $\mathcal{S}_\phi^{\text{SD}}$ are finite. We begin with the set $\mathcal{S}_\phi^{\text{ADD}}$.

LEMMA 3.7. *The set $\mathcal{S}_\phi^{\text{ADD}}$ is finite, i.e., $|\mathcal{S}_\phi^{\text{ADD}}| < \infty$.*

Proof. To derive a contradiction, suppose that $|\mathcal{S}_\phi^{\text{ADD}}| = \infty$, which in particular means that Algorithm 1 does not terminate finitely. Since Lemma 3.6 shows that $\mathcal{S}_\beta$ is finite, we may also conclude that there exists an iteration k_1 such that $k \in \mathcal{S}_\phi = \mathcal{S}_\phi^{\text{ADD}} \cup \mathcal{S}_\phi^{\text{SD}}$ for all $k \ge k_1$.

We proceed by making two observations. First, if the ith component of x_k becomes zero for some iteration $k \ge k_1$, it will remain zero for the remainder of the iterations. This can be seen by using lines 11 and 7 of Algorithm 1 and the definition of $\phi(x_k)$ to deduce that if $[d_k]_i \ne 0$, then $i \in \mathcal{I}_k \subseteq \{\ell : [\phi(x_k)]_\ell \ne 0\} \subseteq \mathcal{I}^+(x_k) \cup \mathcal{I}^-(x_k)$

for all $k \ge k_1$; equivalently, if $i \in \mathcal{I}^0(x_k)$, then $[d_k]_i = 0$. The second observation is that at least one nonzero component of x_k becomes zero at x_{k+1} for each $k \in \mathcal{S}_\phi^{\text{ADD}}$. This can be seen by construction of Algorithm 2 when it is called in line 12 of Algorithm 1. Together, these observations contradict $|\mathcal{S}_\phi^{\text{ADD}}| = \infty$ since at most n variables may become zero. Thus, we must conclude that $|\mathcal{S}_\phi^{\text{ADD}}| < \infty$. ∎

To establish that $\mathcal{S}_\phi^{\text{SD}}$ is finite, we require the following two lemmas. The first lemma gives a bound on the size of $\bar{d}_k$ that holds whenever $k \in \mathcal{S}_\phi$.

LEMMA 3.8. *If $k \in \mathcal{S}_\phi$, then $\|\bar{d}_k\| \le (2/\theta_{\min})\|g_k\|$, where $\theta_{\min} > 0$ is defined in Assumption* 3.1.

Proof. Let $k \in \mathcal{S}_\phi$ so that $\bar{d}_k$ is computed in line 10 of Algorithm 1 and let d_k^{N} be the Newton step satisfying $H_k d_k^{\text{N}} = -g_k$ with H_k and g_k defined in line 8 of Algorithm 1. It follows that

$$\|d_k^{\text{N}}\| \le \|H_k^{-1}\|\|g_k\|. \tag{8}$$

Let us also define the quadratic function $\bar{m}_k(d) := m_k(d_k^{\text{N}} + d)$ and the associated level set $\mathcal{L}_k := \{d : \bar{m}_k(d) \le 0\}$. We then see that

$$(\bar{d}_k - d_k^{\text{N}}) \in \mathcal{L}_k \tag{9}$$

since $\bar{m}_k(\bar{d}_k - d_k^{\text{N}}) = m_k(\bar{d}_k) \le m_k(0) = 0$, where we have used the condition $m_k(\bar{d}_k) \le m_k(0)$ that is required to hold in line 10 of Algorithm 1.

We are now interested in finding a point in $\mathcal{L}_k$ with largest norm. To characterize such a point, we consider the optimization problem

$$\underset{d \in \mathbb{R}^n}{\text{maximize}}\ \tfrac{1}{2}\|d\|^2 \ \text{subject to}\ d \in \mathcal{L}_k. \tag{10}$$

It is not difficult to prove that a global maximizer of problem (10) is $d_* := \alpha_* v$ with $\alpha_*^2 := (-g_k^T d_k^{\text{N}})/\theta$, where (v, θ) with $\|v\| = 1$ is an eigenpair corresponding to the left-most eigenvalue $\theta \ge \theta_{\min}$ of H_k. (This can also be seen to hold since the level curves of $\bar{m}_k$ are ellipses, $d = 0$ is the minimizer of $\bar{m}_k$, and the eigenvector corresponding to the left-most eigenvalue of H_k is the direction of least positive curvature.) Thus, it follows that $\|d\|^2 \le \|d_*\|^2$ for all $d \in \mathcal{L}_k$. Combining this with (9), the definition of d_*, and (8) shows that

$$\|\bar{d}_k - d_k^{\text{N}}\|^2 \le \|d_*\|^2 = \alpha_*^2\|v\|^2 = \frac{-g_k^T d_k^{\text{N}}}{\theta} \le \frac{\|g_k\|\|d_k^{\text{N}}\|}{\theta} \le \frac{\|H_k^{-1}\|\|g_k\|^2}{\theta} = \left(\frac{\|g_k\|}{\theta}\right)^2.$$

By combining the previous inequality with the triangle inequality and (8), we obtain

$$\|\bar{d}_k\| \le \|\bar{d}_k - d_k^{\text{N}}\| + \|d_k^{\text{N}}\| \le \frac{\|g_k\|}{\theta} + \frac{\|g_k\|}{\theta} = \frac{2\|g_k\|}{\theta} \le \frac{2\|g_k\|}{\theta_{\min}},$$

which complete the proof. ∎

The next result establishes a bound on the decrease in F when $k \in \mathcal{S}_\phi^{\text{SD}}$.

LEMMA 3.9. *If $k \in \mathcal{S}_\phi^{\text{SD}}$, then x_{k+1} satisfies*

$$F(x_{k+1}) \le F(x_k) - \kappa_\phi \max\{\gamma^{-2}\|\beta(x_k)\|^2, \|\phi(x_k)\|^2\}, \tag{11}$$

where $\kappa_\phi := \eta_\phi^2 \min\left\{\frac{\eta}{\theta_{\max}}, \frac{\eta\xi(1-\eta)\theta_{\min}^2}{2\theta_{\max}^3}\right\} > 0$.

Proof. Let $k \in \mathcal{S}_\phi^{\text{SD}}$. We consider two cases. First, suppose that $j = 0$ when line 7 in Algorithm 2 is reached. In this case, it follows by construction of Algorithm 2 that $\text{sgn}(y_0) = \text{sgn}(x_k + d_k) = \text{sgn}(x_k)$; i.e., the full step d_k and the vector x_k are contained in the same orthant. Consequently, the loop that starts in line 12 is simply a backtracking Armijo line search. Thus, if

$$\xi^j \in \left(0, \frac{2(\eta-1)\nabla_{\mathcal{I}_k} F(x_k)^T [d_k]_{\mathcal{I}_k}}{\theta_{\max}\|[d_k]_{\mathcal{I}_k}\|^2}\right] \equiv \left(0, \frac{2(\eta-1)g_k^T \bar{d}_k}{\theta_{\max}\|\bar{d}_k\|^2}\right], \tag{12}$$

then, by well-known properties of twice continuously differentiable functions with Lipschitz continuous gradients, we have that

$$\begin{aligned} F(x_k + \xi^j d_k) &\le F(x_k) + \xi^j \nabla_{\mathcal{I}_k} F(x_k)^T [d_k]_{\mathcal{I}_k} + \tfrac{1}{2}\xi^{2j}\theta_{\max}\|[d_k]_{\mathcal{I}_k}\|^2 \\ &\le F(x_k) + \xi^j \nabla_{\mathcal{I}_k} F(x_k)^T [d_k]_{\mathcal{I}_k} + \xi^j(\eta-1)\nabla_{\mathcal{I}_k} F(x_k)^T [d_k]_{\mathcal{I}_k} \\ &= F(x_k) + \eta\xi^j \nabla_{\mathcal{I}_k} F(x_k)^T [d_k]_{\mathcal{I}_k}; \end{aligned}$$

i.e., the inequality in line 13 will hold whenever (12) holds. On the other hand, suppose that $j > 0$ when line 7 in Algorithm 2 is reached. Then, since $k \in \mathcal{S}_\phi^{\text{SD}}$, we may conclude that

$$F(x_k + \alpha_B d_k) > F(x_k) + \eta\alpha_B \nabla_{\mathcal{I}_k} F(x_k)^T [d_k]_{\mathcal{I}_k} = F(x_k) + \eta\alpha_B g_k^T \bar{d}_k \tag{13}$$

in line 10 because otherwise we would have $k \in \mathcal{S}_\phi^{\text{ADD}} = \mathcal{S}_\phi \setminus \mathcal{S}_\phi^{\text{SD}}$. Since no points of nondifferentiability of $\|\cdot\|_1$ exist on the line segment connecting x_k to $x_k + \alpha_B d_k$ (which follows by the definition of α_B in line 8 of Algorithm 2), we can conclude for the same reason that we acquired (12) that (13) implies

$$\alpha_B > \frac{2(1-\eta)|g_k^T \bar{d}_k|}{\theta_{\max}\|\bar{d}_k\|^2}.$$

Combining these two cases, we have that the line search procedure in Algorithm 2 will terminate with $x_{k+1} = x_k + \xi^j d_k$, where

$$\xi^j \ge \min\left\{1, \frac{2\xi(1-\eta)|g_k^T \bar{d}_k|}{\theta_{\max}\|\bar{d}_k\|^2}\right\} \quad \text{and} \quad F(x_{k+1}) \le F(x_k) + \eta\xi^j g_k^T \bar{d}_k. \tag{14}$$

Let us now consider two cases. First, suppose that $\xi^j = 1$ is returned from the line search, i.e., $j = 0$. Then, it follows from (14), lines 10 and 9 of Algorithm 1, the Cauchy-Schwarz inequality, and Assumption 3.1 that

$$\begin{aligned} F(x_k) - F(x_{k+1}) &\ge -\eta\xi^j g_k^T \bar{d}_k = \eta|g_k^T \bar{d}_k| \ge \eta|g_k^T d_k^R| \\ &= \eta\alpha_k\|g_k\|^2 = \eta\frac{\|g_k\|^4}{g_k^T H_k g_k} \ge \frac{\eta}{\theta_{\max}}\|g_k\|^2. \end{aligned} \tag{15}$$

Now suppose that $\xi^j < 1$. Then, it follows from (14), the inequality $|g_k^T \bar{d}_k| \ge \|g_k\|^2/\theta_{\max}$ established while deriving (15), and Lemma 3.8 that

$$\begin{aligned} F(x_k) - F(x_{k+1}) &\ge -\eta\xi^j g_k^T \bar{d}_k = \eta\xi^j|g_k^T \bar{d}_k| \ge \frac{2\eta\xi(1-\eta)|g_k^T \bar{d}_k|^2}{\theta_{\max}\|\bar{d}_k\|^2} \\ &\ge \frac{2\eta\xi(1-\eta)\theta_{\min}^2\|g_k\|^4}{4\theta_{\max}^3\|g_k\|^2} = \left(\frac{\eta\xi(1-\eta)\theta_{\min}^2}{2\theta_{\max}^3}\right)\|g_k\|^2. \end{aligned} \tag{16}$$

Combining (15) and (16) for the two cases establishes that

$$\begin{aligned}
F(x_k) - F(x_{k+1}) &\geq \min\left\{\frac{\eta}{\theta_{\max}}, \frac{\eta\xi(1-\eta)\theta_{\min}^2}{2\theta_{\max}^3}\right\}\|g_k\|^2 \\
&= \min\left\{\frac{\eta}{\theta_{\max}}, \frac{\eta\xi(1-\eta)\theta_{\min}^2}{2\theta_{\max}^3}\right\}\|\nabla_{\mathcal{I}_k} F(x_k)\|^2 \\
&\geq \min\left\{\frac{\eta}{\theta_{\max}}, \frac{\eta\xi(1-\eta)\theta_{\min}^2}{2\theta_{\max}^3}\right\}\|[\phi(x_k)]_{\mathcal{I}_k}\|^2 \\
&\geq \eta_\phi^2 \min\left\{\frac{\eta}{\theta_{\max}}, \frac{\eta\xi(1-\eta)\theta_{\min}^2}{2\theta_{\max}^3}\right\}\|\phi(x_k)\|^2 \quad \text{for } k \in \mathcal{S}_\phi^{\text{SD}},
\end{aligned}$$

where we have also used the condition in line 7 of Algorithm 1 and the definition of $\phi(x_k)$. The inequality (11) follows from the previous inequality and the fact that $\|\beta(x_k)\| \leq \gamma\|\phi(x_k)\|$ for all $k \in \mathcal{S}_\phi \subseteq \mathcal{S}_\phi^{\text{SD}}$ as can be seen by line 6 of Algorithm 1. ∎

We may now establish finiteness of the index set $\mathcal{S}_\phi^{\text{SD}}$.

LEMMA 3.10. *The index set* $\mathcal{S}_\phi^{\text{SD}}$ *is finite, i.e.,* $|\mathcal{S}_\phi^{\text{SD}}| < \infty$.

Proof. To derive a contradiction, suppose that $|\mathcal{S}_\phi^{\text{SD}}| = \infty$, which means that Algorithm 1 does not terminate finitely. Thus, it follows from line 4 of Algorithm 1 that $\max\{\|\beta(x_k)\|, \|\phi(x_k)\|\} > \epsilon$ for all $k \geq 0$. Also, it follows from Lemmas 3.6 and 3.7 that there exists an iteration number k_1 such that $k \in \mathcal{S}_\phi^{\text{SD}}$ for all $k \geq k_1$. Thus, with Lemma 3.9, we have for all $\ell \geq k_1$ that

$$\begin{aligned}
F(x_{k_1}) - F(x_{\ell+1}) &= \sum_{k=k_1}^{\ell}\left[F(x_k) - F(x_{k+1})\right] \\
&= \sum_{k\in\mathcal{S}_\phi^{\text{SD}}, k_1\leq k\leq \ell}\left[F(x_k) - F(x_{k+1})\right] \\
&\geq \sum_{k\in\mathcal{S}_\phi^{\text{SD}}, k_1\leq k\leq \ell} \kappa_\phi \max\{\gamma^{-2}\|\beta(x_k)\|^2, \|\phi(x_k)\|^2\} \\
&\geq \sum_{k\in\mathcal{S}_\phi^{\text{SD}}, k_1\leq k\leq \ell} \kappa_\phi \min\{\gamma^{-2}, 1\}\epsilon^2.
\end{aligned}$$

Rearranging the previous inequality shows that

$$\begin{aligned}
\lim_{l\to\infty} F(x_{\ell+1}) &\leq \lim_{\ell\to\infty}\left[F(x_{k_1}) - \sum_{k\in\mathcal{S}_\phi^{\text{SD}}, k_1\leq k\leq \ell} \kappa_\phi \min\{\gamma^{-2}, 1\}\epsilon^2\right] \\
&= F(x_{k_1}) - \sum_{k\in\mathcal{S}_\phi^{\text{SD}}, k_1\leq k} \kappa_\phi \min\{\gamma^{-2}, 1\}\epsilon^2 = -\infty,
\end{aligned}$$

which contradicts Assumption 3.1. Thus, we conclude that $|\mathcal{S}_\phi^{\text{SD}}| < \infty$. ∎

We now prove our first main convergence result.

THEOREM 3.11. *Algorithm* 1 *terminates finitely.*

Proof. Since each iteration number k generated in the algorithm is an element of $\mathcal{S}_\beta \cup \mathcal{S}_\phi^{\text{ADD}} \cup \mathcal{S}_\phi^{\text{SD}}$, the result follows by Lemmas 3.6, 3.7, and 3.10. ∎

Our final convergence result states what happens when the finite termination criterion is removed from Algorithm 1.

THEOREM 3.12. *Let x_* be the unique solution to problem* (1)*. If ϵ in the finite termination condition in line* 4 *of Algorithm* 1 *is replaced by zero, then either:*

(i) *there exists an iteration k such that $x_k = x_*$; or*

(ii) *infinitely many iterations $\{x_k\}$ are computed and they satisfy*

$$\lim_{k\to\infty} x_k = x_*, \quad \lim_{k\to\infty} \varphi(x_k) = 0, \quad \textit{and} \quad \lim_{k\to\infty} \beta(x_k) = 0.$$

Proof. If case (i) occurs, then there is nothing left to prove. Thus, for the remainder of the proof, we assume that case (i) does not occur. Since case (i) does not occur, we know that Algorithm 1 performs an infinite sequence of iterations. Let us then define the set $\mathcal{S} := \mathcal{S}_\beta \cup \mathcal{S}_\phi^{\text{SD}}$, which must be infinite (since any consecutive subsequence of iterations in $\mathcal{S}_\phi^{\text{ADD}}$ must be finite by the finiteness of n). It follows from (7) for $k \in \mathcal{S}_\beta$, (11) for $k \in \mathcal{S}_\phi^{\text{SD}}$, and Assumption 3.1 (specifically, the assumption that f is bounded below over $\mathcal{L}$) that

$$\lim_{k\in\mathcal{S}} \max\{\|\beta(x_k)\|, \|\varphi(x_k)\|\} = 0.$$

Combining this with Assumption 3.1 and Lemma 2.1 gives

$$\lim_{k\in\mathcal{S}} x_k = x_*. \tag{17}$$

Now, we claim that the previous limit holds over all iterations. To prove this by contradiction, suppose that there exists an infinite $\mathcal{K} \subseteq \mathcal{S}_\varphi^{\text{ADD}}$ and a scalar $\varepsilon > 0$ with

$$\|x_k - x_*\| \geq \varepsilon \quad \text{for all } k \in \mathcal{K}. \tag{18}$$

From Assumption 3.1, we conclude that there exists $\delta > 0$ such that

$$\text{if } F(x) \leq F(x_*) + \delta, \text{ then } \|x - x_*\| < \varepsilon. \tag{19}$$

Moreover, from (17) and Assumption 3.1, there exists a smallest $k_S \in \mathcal{S}$ such that

$$F(x_{k_S}) \leq F(x_*) + \delta. \tag{20}$$

There then exists a smallest $k_K \in \mathcal{K}$ such that $k_K > k_S$. Since, by construction, $\{F(x_k)\}_{k\geq 0}$ is monotonically decreasing, we may conclude with (20) that

$$F(x_{k_K}) \leq F(x_{k_S}) \leq F(x_*) + \delta. \tag{21}$$

Combining (21) and (19), we deduce that $\|x_{k_K} - x_*\| < \epsilon$, which contradicts (18) since $k_K \in \mathcal{K}$. This completes the proof. ∎

4. Numerical Results.

In this section, we present results when employing an implementation of FaRSA to solve a collection of ℓ_1-norm regularized logistic regression problems that take the form

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \; \frac{1}{N} \sum_{i=1}^{N} \log\left(1 + e^{-y_i x^T d_i}\right) + \lambda \|x\|_1, \tag{22}$$

where $d_i \in \mathbb{R}^n$ is the ith data vector, N is the number of data vectors in the data set, $y_i \in \{-1, 1\}$ is the class label for the ith data vector, and $\lambda = 1/N$ is the weighting parameter; we refer to $D \in \mathbb{R}^{N\times n}$, whose ith row is d_i^T, as the data set matrix. Such problems routinely arise in the context of model prediction, making the design of advanced optimization algorithms that efficiently and reliably solve them paramount in big data applications. We first describe the data sets considered in our experiments, then describe some details of our implementation (henceforth simply referred to as FaRSA), and finally present the results of our experiments.

TABLE 1
Data sets.

Data set	N	n	Unscaled	Scaling used
fourclass	862	2	✓	into [−1,1]
svmguide1	3089	4	✓	into [−1,1]
cod-rna	59535	8		
breast-cancer	683	10		
australian	690	14		
skin-nonskin	245057	3	✓	into [−1,1]
SUSY	5000000	18	✓	into [−1,1]
splice	1000	60		
heart	270	13		
german.numer	1000	24	✓	into [−1,1]
diabetes	768	8	✓	into [−1,1]
madelon	2000	500	✓	into [−1,1]
w8a	49749	300		
a9a	32561	123		
mnist	30001	784	✓	into [0,1)
liver-disorders	345	6	✓	into [−1,1]
sonar	208	60		
ijcnn1	49990	22		
svmguide3	1243	22		
synthetic	5000	5000		
gisette	6000	5000		
pathway-ad	278	71		
real-sim	72309	20958		
mushrooms	8124	112		
covtype.binary	581012	54		
rcv1.binary	20242	47236		
colon-cancer	62	2000	✓	into [−1,1]
leukemia	34	7129		
duke-breast-cancer	38	7129	✓	into [−1,1]
gene-ad	71	17375	✓	into [−1,1]
news20	19996	1355191		

4.1. Data sets. We tested FaRSA on the ℓ_1-norm regularized logistic regression problem (22) using 31 data sets (see Table 1), 19 of which are available only after standard scaling practices have been applied. For the remaining 12 data sets, we considered both unscaled and scaled versions, where, for each, the scaling technique employed is described in the last column of Table 1. A check mark in the "Unscaled" column indicates that we were able to obtain an unscaled version of that data set.

Most of the data sets in Table 1 can be obtained from the LIBSVM repository.[1] From this repository, we excluded all regression and multiple-class (greater than two) instances, except for mnist since it is such a commonly used data set. Since mnist is for digit classification, we transformed it for binary classification by assigning the digits 0–4 to the label -1 and the digits 5–9 to the label 1. The remaining data sets were binary classification examples from which we removed HIGGS, kdd2010(algebra), kdd2010(bridge to algebra), epsilon, url, and webspam since insufficient computer memory was available. (All experiments were conducted on a 64-bit machine with an Intel I7 4.0-GHz CPU and 16 GB of main memory.) Finally, for the adult data (a1a–a9a) and webpage data (w1a–w8a), we used only the largest instances, namely, problems a9a and w8a. This left us with our final subset of data sets from LIBSVM.

In addition, we also tested FaRSA on three other data sets: synthetic, gene-ad, and pathway-ad. The synthetic set is a randomly generated nondiagonally dominant

[1]https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.

data set created by the authors of OBA. The sets gene-ad and pathway-ad are data sets related to Alzheimer's disease. They were obtained by preprocessing the sets GSE4226[2] and GSE4227[3] using the method presented in [36] and merging the results into the single data set: gene-ad. The gene data (gene-ad) was converted to pathway data (pathway-ad) using the ideas described in [36]. The union of these three data sets and those from the LIBSVM repository comprised our complete test set.

For the unscaled data sets (see columns 4 and 5 in Table 1), we adopted standard scaling techniques. For problems scaled into $[-1,1]$, we used the normalization

$$d_i^c \leftarrow \frac{d_i^c - \operatorname{mean}(d_i^c)\cdot e}{\|d_i^c - \operatorname{mean}(d_i^c)\cdot e\|_\infty} \quad \text{for all } i \in \{1,2,\cdots,n\}, \tag{23}$$

where d_i^c denotes the ith column of the data set matrix D and $e \in \mathbb{R}^N$ is a vector of all ones. For problem mnist, which was scaled into $[0,1)$, we used a common converting method in image processing. Specifically, we defined

$$I(i,j) = \frac{P(i,j)}{2^b}, \tag{24}$$

where $P(i,j)$ is the given unscaled integer pixel value satisfying

$$P(i,j) \in \{0,1,2,\ldots,2^b-1\},$$

b is the intensity resolution ($b = 8$ for the mnist data set), and (i,j) range over the size of the image. The scaled pixel values are then given by the values $I(i,j) \in [0,1)$.

4.2. Implementation details. We developed a preliminary MATLAB implementation of FaRSA that we are happy to provide on request. In this section, we describe the algorithm-specific choices made to obtain the results that we present.

For determining the iteration type, we chose $\gamma = 1$ in line 6 of Algorithm 1 so that no preference was given to iterations being in either $\mathcal{S}_\phi$ or $\mathcal{S}_\beta$. For any $k \in \mathcal{S}_\phi$, we made the simple choice of $\mathcal{I}_k = \{i : [\phi(x_k)]_i \neq 0\}$. This made the inequality in line 7 satisfied for any $\eta_\phi \in (0,1]$, making the choice of this parameter irrelevant. (In a more sophisticated implementation, one might consider other choices of $\mathcal{I}_k$, say, to adaptively control $|\mathcal{I}_k|$, to improve efficiency.) With this choice for $\mathcal{I}_k$ made, Algorithm 1 allows for great flexibility in obtaining a search direction that satisfies the conditions in line 10 (see (iii) in section 1.1 for additional comments). For our tests, we applied the CG method to the system $H_k d = -g_k$ defined by the terms constructed in line 8, except that we added a diagonal matrix with entries 10^{-8} to H_k (an approach also adopted by OBA and LIBLINEAR). As discussed in section 2, the conditions that are required to be satisfied by the trial step will hold if CG is terminated during any iteration. To help limit the number of backtracking steps required by the subsequent backtracking line search, we terminated CG as soon as one of three conditions was satisfied. To describe these conditions, we let d_j denote the jth CG iteration, $r_j = \|H_k d_j + g_k\|$ denote the jth CG residual, and v_j denote the number of components in $x_k + d_j$ that fall into a different orthant than x_k. With these definitions, we terminated CG as soon as one of the following was satisfied:

$$\begin{aligned} r_j &\le \max\{10^{-1} r_0, 10^{-12}\}; \\ v_j &\ge \max\{10^3, 10^{-1}|\mathcal{I}_k|\}; \text{ or} \\ \|d_j\| &\ge \delta_{k,\phi} := \max\{10^{-3}, \min\{10^3, 10\|x_{k_\phi(k)+1} - x_{k_\phi(k)}\|\}\}, \end{aligned}$$

[2]http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE4226.
[3]http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE4227.

where $k_\phi(k) := \max\{\bar{k} : \bar{k} \in \mathcal{S}_\phi \text{ and } \bar{k} < k\}$. This first condition is a standard requirement of asking the residual to be reduced by a fraction of the initial residual. We used the second condition to trigger termination when a CG iterate predicted that "too many" of the variables at $x_k + d_j$ are in the "wrong" orthant. Finally, the third condition ensured that the size of the trial step was moderate, thus functioning as an implicit trust-region constraint; this condition was motivated by the well-known fact that CG iterations $\{d_j\}$ are monotonically increasing in norm.

When $k \in \mathcal{S}_\beta$, we made the simplest choice of $\mathcal{I}_k = \{i : [\beta(x_k)]_i \neq 0\}$, making the choice of $\eta_\beta \in (0, 1]$ irrelevant in our tests (though adaptive choices of $\mathcal{I}_k$ might be worthwhile in a more sophisticated implementation). Since there is no natural scaling for the direction $\beta(x_k)$ because it is based on first-derivative information only, it is important from a practical perspective to adaptively scale the direction. Therefore, in line 15, we used the alternative safeguarded direction defined by

$$[d_k]_{\mathcal{I}_k} = -\delta_{k,\beta} \frac{[\beta(x_k)]_{\mathcal{I}_k}}{\|[\beta(x_k)]_{\mathcal{I}_k}\|}, \tag{25}$$

where $\delta_{k,\beta} := \max\{10^{-5}, \min\{1, \|x_{k_\beta(k)+1} - x_{k_\beta(k)}\|\}\}$ with $k_\beta(k) := \max\{\bar{k} : \bar{k} \in \mathcal{S}_\beta \text{ and } \bar{k} < k\}$. Since this is a safeguarded scaling of the d_k defined in line 15, it is fully covered by the theory that we developed in section 3.

The values $\eta = 10^{-2}$ and $\xi = 0.5$ were used in the line search regardless of whether it was the line search performed by Algorithm 2 when called by Algorithm 1 (line 12) or the line search performed by Algorithm 3 when called by Algorithm 1 (line 16). The starting point x_0 was chosen as the zero vector for all problems, and the termination tolerance, maximum allowed number of iterations, and maximum allowed time limit values were chosen to be $\epsilon = 10^{-6}$, 1000, and 10 minutes, respectively.

4.3. Test results. The output from FaRSA for the problems corresponding to the scaled and unscaled data sets in our experiments are summarized in Tables 2 and 3 and Tables 4 and 5, respectively. Tables 2 and 4 focus on the computational time in seconds (Time) and sparsity (% of zeros) required to obtain the computed solutions, while Tables 3 and 5 provide the number of required objective function evaluations (Fevals), Hessian-vector products (Hv), and iterations (Iterations); iterations for FaRSA are in the format ($|\mathcal{S}_\phi|$, $|\mathcal{S}_\beta|$), so that the total iterations are $|\mathcal{S}_\phi| + |\mathcal{S}_\beta|$. For comparison purposes, we also provide the output from the OBA solver whose MATLAB implementation was graciously provided by the authors. For a fair comparison, we used the same stopping tolerance value of $\epsilon = 10^{-6}$ for OBA and made no modifications to their code, but we mention that the stopping criteria used in their software is

$$\|\max\{\min\{\nabla f(x_k) + \lambda e, x_k\}, \nabla f(x_k) - \lambda e\}\|_\infty \leq \epsilon,$$

where $e \in \mathbb{R}^n$ is a vector of all ones. The run time reported for each problem (named according to the corresponding data set) are the averages from running each problem instance 10 times. We do not provide the final objective values since they were the same to at least 5 digits of accuracy for FaRSA and OBA on all problems that were successfully solved by both algorithms.

In the tables we use "max iter" to denote that the maximum number of iterations 1000 was reached, "max time" to denote that the maximum time limit of 10 minutes was reached, "Inf" to denote that MATLAB returned an infinite objective function value during the solution process, and "ascent" to mean that the objective

TABLE 2
CPU time and sparsity for FaRSA and OBA on scaled problem variants.

	Time (seconds)			% of zeros	
Problems	FaRSA	OBA	OBA/FaRSA	FaRSA	OBA
fourclass	0.00326	0.00705	2.1626	0	0
svmguide1	0.0384	0.06457	1.6815	0	0
cod-rna	0.48762	0.18618	0.3818	0	0
breast-cancer	0.0089	0.0174	1.9674	0	0
australian	0.01443	0.03769	2.6119	0	0
skin-nonskin	3.20594	ascent	Inf	0	—
SUSY	241.2437	205.1242	0.8502	0	0
splice	0.0101	0.01982	1.9624	5	5
heart	0.00706	0.01357	1.9221	7.7	7.7
german.numer	0.01159	0.02111	1.8214	8.3	8.3
diabetes	0.00581	0.00979	1.6850	12.5	12.5
madelon	0.26604	0.37497	1.4094	19.8	19.8
w8a	0.97079	0.99154	1.0214	21.3	18.7
a9a	0.78203	3.26994	4.1813	22.0	20.3
mnist	18.78034	54.432	3.0545	37.7	37.8
liver-disorders	0.016955	0.078115	4.6221	40.0	40.0
sonar	0.02012	0.02938	1.4602	41.7	41.7
ijcnn1	0.06153	0.08178	1.3291	45.5	45.5
svmguide3	0.01856	0.03478	1.8739	45.5	45.5
synthetic	45.82688	20.42464	0.4457	57.4	49.5
gisette	13.30533	28.22136	2.1211	84.6	84.6
pathway-ad	0.16054	1.30585	8.1341	87.4	87.4
real-sim	2.3221	2.43764	1.0214	91.9	91.8
mushrooms	0.03089	0.05815	1.8825	97.3	97.3
covtype.binary	1.04162	ascent	Inf	98.2	—
rcv1.binary	0.39186	0.80563	1.7427	98.8	98.8
colon-cancer	0.04069	0.03905	0.9597	99.0	99.0
leukemia	0.09151	0.12086	1.3207	99.7	99.7
duke-breast-cancer	0.06227	0.10628	1.7068	99.7	99.7
gene-ad	0.21525	0.15943	0.7407	99.8	99.8
news20	6.09086	19.77945	3.2474	99.9	99.9

function increased. In theory, ascent is possible for OBA only when their estimate (10^8 in their code) of the Lipschitz constant for the gradient of f is not large enough. Although simple adaptive strategies could be used to avoid such issues, we made no such attempts because we did not want to make any edits to their code.

Table 2 shows that FaRSA performed better than OBA in terms of computational time on 26 of the 31 (83.87%) scaled test problems. OBA is faster than FaRSA only on problems cod-rna, SUSY, synthetic, gene-ad, and colon-cancer. However, FaRSA is between 3 and 8 times faster than OBA on problems a9a, mnist, pathway-ad, liver-disorders, and news20 and between 1 and 3 times faster than OBA on the remaining 21 problems. In terms of sparsity, the two algorithms are comparable. Table 3 shows that OBA usually requires fewer objective function evaluations but a greater number of Hessian-vector products, which explains OBA's higher run time. Also, since $|\mathcal{S}_\beta|$ is consistently small in the iteration column, we know that FaRSA quickly identifies the orthant that contains the optimal solution.

By turning our attention to Tables 4 and 5, we see that the performance of both FaRSA and OBA deteriorates when the problems are unscaled. OBA fails on problems skin-nonskin, gene-ad, and mnist because it generates iterates that increase the objective function. Overall, FaRSA was able to solve 10 of the 12 unscaled problems, and OBA performed better than FaRSA in terms of run time on only a single test

TABLE 3
Number of objective function evaluations (Fevals), Hessian-vector products (Hv), and iterations (Iterations) for FaRSA and OBA on scaled problem variants. Iterations for FaRSA are in the format ($|\mathcal{S}_\phi|$, $|\mathcal{S}_\beta|$); the total iterations are $|\mathcal{S}_\phi| + |\mathcal{S}_\beta|$.

	Fevals		Hv		Iterations	
Problems	FaRSA	OBA	FaRSA	OBA	FaRSA	OBA
fourclass	6	6	7	12	(4, 1)	5
svmguide1	7	7	17	23	(5, 1)	6
cod-rna	43	16	90	67	(30, 3)	15
breast-cancer	14	11	36	44	(8, 3)	9
australian	14	11	62	63	(8, 3)	9
skin-nonskin	364	—	531	—	(180, 2)	—
SUSY	54	35	661	482	(48, 5)	34
splice	14	10	24	36	(6, 3)	9
heart	9	9	40	49	(6, 2)	8
german.numer	8	9	45	65	(5, 2)	8
diabetes	8	7	19	23	(5, 2)	6
madelon	36	14	82	171	(10, 4)	13
w8a	41	15	516	455	(19, 4)	14
a9a	65	26	483	2237	(28, 6)	25
mnist	50	44	1597	4536	(33, 6)	43
liver-disorders	6	8	10	22	(4, 1)	6
sonar	27	12	140	149	(13, 4)	11
ijcnn1	10	11	43	38	(7, 2)	8
svmguide3	25	19	75	122	(11, 4)	16
synthetic	717	31	290	510	(241, 7)	30
gisette	185	74	933	55798	(78, 10)	69
pathway-ad	479	89	654	12806	(128, 19)	88
real-sim	26	11	248	293	(14, 4)	10
mushrooms	27	16	23	110	(13, 1)	15
covtype.binary	40	—	8	—	(4, 1)	—
rcv1.binary	17	7	102	189	(13, 3)	6
colon-cancer	80	10	128	192	(21, 3)	9
leukemia	100	16	153	346	(26, 3)	15
duke-breast-cancer	70	11	107	356	(20, 3)	11
gene-ad	97	10	155	369	(29, 3)	9
news20	23	9	133	403	(14, 4)	8

problem (colon-cancer). Finally, the trend that OBA performs better in terms of the number of required objective function evaluations but worse in terms of the number of needed Hessian-vector products still holds.

The previous tables show that FaRSA efficiently and reliably obtains solutions that satisfy the stopping tolerance value of $\epsilon = 10^{-6}$. In practice, one sometimes only requests a low accuracy solution, often motivated by problems that may arise due to overfitting. To explore the performance of FaRSA for various stopping tolerance levels, we created Figures 1–3 in Appendix B. Each plot shows the run time (y-axis) required to achieve the desired optimality accuracy (x-axis) for the stated problem. These figures show that the superior performance, measured in terms of computational time, previously displayed by FaRSA for the stopping tolerance 10^{-6} also generally holds for larger stopping tolerances.

Finally, we note that additional test results, which compare the performance of FaRSA to the highly successful bound-constrained solver ASA-CG [15] when applied to a bound-constrained reformulation of problem (1), may be found in Appendix C.

5. Conclusion. We presented a new reduced-space algorithm, FaRSA, for minimizing an ℓ_1-norm regularized convex function. The method uses an adaptive

Table 4
CPU time and sparsity for FaRSA and OBA on unscaled problem variants.

	Time (seconds)			% of zeros	
Problems	FaRSA	OBA	OBA/FaRSA	FaRSA	OBA
fourclass	0.00486	0.00775	1.5946	0	0
svmguide1	0.05641	0.13327	2.3625	0	0
diabetes	0.01964	0.02159	1.0993	0	0
german.numer	0.03168	0.0564	1.7803	0	0
skin-nonskin	0.11661	ascent	Inf	0	—
liver-disorders	0.00571	0.03277	5.7391	0	0
madelon	6.55674	41.9519	6.3983	8	8
colon-cancer	0.08429	0.05364	0.6364	98.7	98.7
duke-breast-cancer	0.08936	0.12487	1.3973	99.7	99.7
gene-ad	4.62839	ascent	Inf	99.8	—
mnist	max time	ascent	—	—	—
SUSY	max iter	max iter	—	—	—

Table 5
Number of objective function evaluations (Fevals), Hessian-vector products (Hv), and iterations (Iterations) for FaRSA and OBA on unscaled problem variants. Iterations for FaRSA are in the format ($|\mathcal{S}_\phi|$, $|\mathcal{S}_\beta|$); the total iterations are $|\mathcal{S}_\phi| + |\mathcal{S}_\beta|$.

	Fevals		Hv		Iterations	
Problems	FaRSA	OBA	FaRSA	OBA	FaRSA	OBA
fourclass	15	6	10	14	(5, 2)	5
svmguide1	15	11	25	39	(8, 2)	10
diabetes	37	10	72	67	(17, 5)	9
german.numer	31	18	140	175	(18, 5)	17
skin-nonskin	15	—	18	—	(7, 1)	—
liver-disorders	15	11	24	38	(7, 2)	8
madelon	255	140	2624	18418	(155, 7)	139
colon-cancer	165	11	168	304	(36, 12)	10
duke-breast-cancer	95	16	108	424	(25, 3)	15
gene-ad	3909	—	2525	—	(444, 37)	—
mnist	—	—	—	—	—	—
SUSY	—	—	—	—	—	—

condition to determine when the current reduced-space should be updated, which is itself based on measures of optimality in the current reduced space and its complement. Global convergence was established for our method, and numerical experiments on ℓ_1-norm regularized logistic problems exhibited its practical performance. In particular, the experiments showed that FaRSA was generally better in terms of computational time than a recently proposed reduced-space orthant-based algorithm called OBA, regardless of the solution accuracy requested. FaRSA was also shown to be better than OBA in terms of requiring fewer Hessian-vector products but worse in terms of requiring a greater number of objective function evaluations. Since OBA was shown in [18] to be better than the state-of-the-art solver used in LIBLINEAR when the second-derivative matrices were not diagonally dominant, we expect that FaRSA will serve as a valuable data analysis tool. OBA and our preliminary implementation of FaRSA will often be outperformed by LIBLINEAR when the second-derivative matrices are diagonally dominant. However, FaRSA was designed with great flexibility in how the subproblem solutions are obtained. Although our preliminary implementation invoked linear-CG as the subproblem solver, our framework also allows for coordinate-descent–based algorithms to be used, such as those used in LIBLINEAR. We expect to provide such options as well as include features that control the subproblem size in a future release of our solver. We believe that once these enhancements have been made, FaRSA will be competitive with LIBLINEAR on all classes of problems and superior when the second-derivative matrices are not diagonally dominant.

Appendix A. Relationship between FaRSA and ISTA.

The step in line 16 of Algorithm 1 may be interpreted as a *reduced* ISTA [2] step. The next lemma makes this relationship precise.

LEMMA A.1. *For any k, let s_k be the* full *ISTA step defined by*

$$s_k := \operatorname{shrink}\big(x_k - \nabla f(x_k)\big) - x_k, \text{ where}$$

$$[\operatorname{shrink}\big(x_k - \nabla f(x_k)\big) - x_k]_i := \begin{cases} -[\nabla f(x_k)]_i + \lambda & \text{if } [x_k - \nabla f(x_k)]_i < -\lambda, \\ -[x_k]_i & \text{if } [x_k - \nabla f(x_k)]_i \in [-\lambda, \lambda], \\ -[\nabla f(x_k)]_i - \lambda & \text{if } [x_k - \nabla f(x_k)]_i > \lambda. \end{cases}$$

Then, $s_k = -(\beta(x_k) + \phi(x_k))$.

Proof. Recall the definitions of the components of $\beta(x_k)$ and $\phi(x_k)$, which may be rewritten in a slightly more convenient form as follows:

$$[\beta(x_k)]_i := \begin{cases} \nabla_i f(x_k) + \lambda & \text{if } [x_k]_i = 0 \text{ and } \nabla_i f(x_k) + \lambda < 0, \\ \nabla_i f(x_k) - \lambda & \text{if } [x_k]_i = 0 \text{ and } \nabla_i f(x_k) - \lambda > 0, \\ 0 & \text{otherwise,} \end{cases}$$

$$[\phi(x_k)]_i := \begin{cases} 0 & \text{if } [x_k]_i = 0, \\ \min\{\nabla_i f(x_k) + \lambda, \max\{[x_k]_i, \nabla_i f(x_k) - \lambda\}\} & \text{if } [x_k]_i > 0 \text{ and } \nabla_i f(x_k) + \lambda > 0, \\ \max\{\nabla_i f(x_k) - \lambda, \min\{[x_k]_i, \nabla_i f(x_k) + \lambda\}\} & \text{if } [x_k]_i < 0 \text{ and } \nabla_i f(x_k) - \lambda < 0, \\ \nabla_i f(x_k) + \lambda \cdot \operatorname{sgn}([x_k]_i) & \text{otherwise.} \end{cases}$$

For any component i, we proceed by considering various cases and subcases.

Case 1. Suppose that

$$[x_k - \nabla f(x_k)]_i > \lambda, \text{ meaning that } [x_k]_i > \nabla_i f(x_k) + \lambda. \tag{26}$$

Subcase 1a. Suppose that $[x_k]_i > 0$ and $\nabla_i f(x_k) + \lambda > 0$, so $\nabla_i f(x_k) > -\lambda$. Then, $[\beta(x_k)]_i = 0$ and

$$[\phi(x_k)]_i = \min\{\nabla_i f(x_k) + \lambda, \max\{[x_k]_i, \nabla_i f(x_k) - \lambda\}\}. \tag{27}$$

By (26), it follows that $[x_k]_i > \nabla_i f(x_k) + \lambda > \nabla_i f(x_k) - \lambda$, which with $[x_k]_i > 0$ means the max in (27) evaluates as $[x_k]_i$. Then, again with (26), the min in (27) yields

$$[\phi(x_k)]_i = \nabla_i f(x_k) + \lambda = -[s_k]_i. \tag{28}$$

Subcase 1b. Suppose that $[x_k]_i > 0$ and $\nabla_i f(x_k) + \lambda \le 0$, so $\nabla_i f(x_k) \le -\lambda$. Then, $[\beta(x_k)]_i = 0$ and

$$[\phi(x_k)]_i = \nabla_i f(x_k) + \lambda = -[s_k]_i. \tag{29}$$

Subcase 1c. Suppose that $[x_k]_i = 0$ and $\nabla_i f(x_k) + \lambda \leq 0$, so $\nabla_i f(x_k) \leq -\lambda$. Then, $[\phi(x_k)]_i = 0$ and

$$[\beta(x_k)]_i = \nabla_i f(x_k) + \lambda = -[s_k]_i. \tag{30}$$

Subcase 1d. Suppose that $[x_k]_i < 0$ and $\nabla_i f(x_k) + \lambda < 0$, so $\nabla_i f(x_k) < -\lambda$. Then, $[\beta(x_k)]_i = 0$ and

$$[\phi(x_k)]_i = \max\{\nabla_i f(x_k) - \lambda, \min\{[x_k]_i, \nabla_i f(x_k) + \lambda\}\}. \tag{31}$$

By (26), it follows that $[x_k]_i > \nabla_i f(x_k) + \lambda$, which along with $\nabla_i f(x_k) + \lambda < 0$ means that the min in (31) evaluates as $\nabla_i f(x_k) + \lambda$. Then, since $\nabla_i f(x_k) + \lambda > \nabla_i f(x_k) - \lambda$, the max in (31) yields

$$[\phi(x_k)]_i = \nabla_i f(x_k) + \lambda = -[s_k]_i. \tag{32}$$

Since Subcases 1a–1d exhaust all possibilities under (26), we conclude from the results in (28), (29), (30), and (32) that for Case 1 we have $[s_k]_i = -[\beta(x_k) + \phi(x_k)]_i$.

Case 2. Suppose that

$$[x_k - \nabla f(x_k)]_i < -\lambda, \text{ meaning that } [x_k]_i < \nabla_i f(x_k) - \lambda. \tag{33}$$

We claim that the analysis for this case is symmetric to that in Case 1 above, from which we may conclude that for this case we again have $[s_k]_i = -[\beta(x_k) + \phi(x_k)]_i$.

Case 3. Suppose that

$$[x_k - \nabla f(x_k)]_i \in [-\lambda, \lambda], \text{ meaning that } [x_k]_i \in \nabla_i f(x_k) + [-\lambda, \lambda]. \tag{34}$$

Subcase 3a. Suppose that $[x_k]_i > 0$. Then, $[\beta(x_k)]_i = 0$ and, since $[x_k]_i > 0$ and (34) imply $\nabla_i f(x_k) > -\lambda$,

$$[\phi(x_k)]_i = \min\{\nabla_i f(x_k) + \lambda, \max\{[x_k]_i, \nabla_i f(x_k) - \lambda\}\}. \tag{35}$$

Since (34) also implies $[x_k]_i > \nabla_i f(x_k) - \lambda$, it follows along with $[x_k]_i > 0$ that the max in (35) evaluates as $[x_k]_i$. Then, since (34) implies $[x_k]_i < \nabla_i f(x_k) + \lambda$, the min in (35) yields

$$[\phi(x_k)]_i = [x_k]_i = -[s_k]_i. \tag{36}$$

Subcase 3b. Suppose that $[x_k]_i = 0$. Then, $[\phi(x_k)]_i = 0$ and, along under (34),

$$[\beta(x_k)]_i = -[s_k]_i = 0. \tag{37}$$

Subcase 3c. Suppose that $[x_k]_i < 0$. We claim that the analysis for this case is symmetric to that in Subcase 3a, from which we may conclude that for this subcase we again have

$$[\phi(x_k)]_i = [x_k]_i = -[s_k]_i. \tag{38}$$

Since Subcases 1a–1d exhaust all possibilities under (34), we conclude from the results in (36), (37), and (38) that for Case 3 we have $[s_k]_i = -[\beta(x_k) + \phi(x_k)]_i$. The result follows, as we have proved the desired result under all cases. $\square$

Appendix B. Additional numerical experiments for FaRSA and OBA. To explore the performance of FaRSA for various stopping tolerance levels, we created Figures 1–3. Each plot shows the run time (y-axis) required to achieve the desired optimality accuracy (x-axis) for the stated problem. These figures show that the superior performance of FaRSA displayed for the stopping tolerance 10^{-6} in section 4.3 also generally holds for larger stopping tolerances.

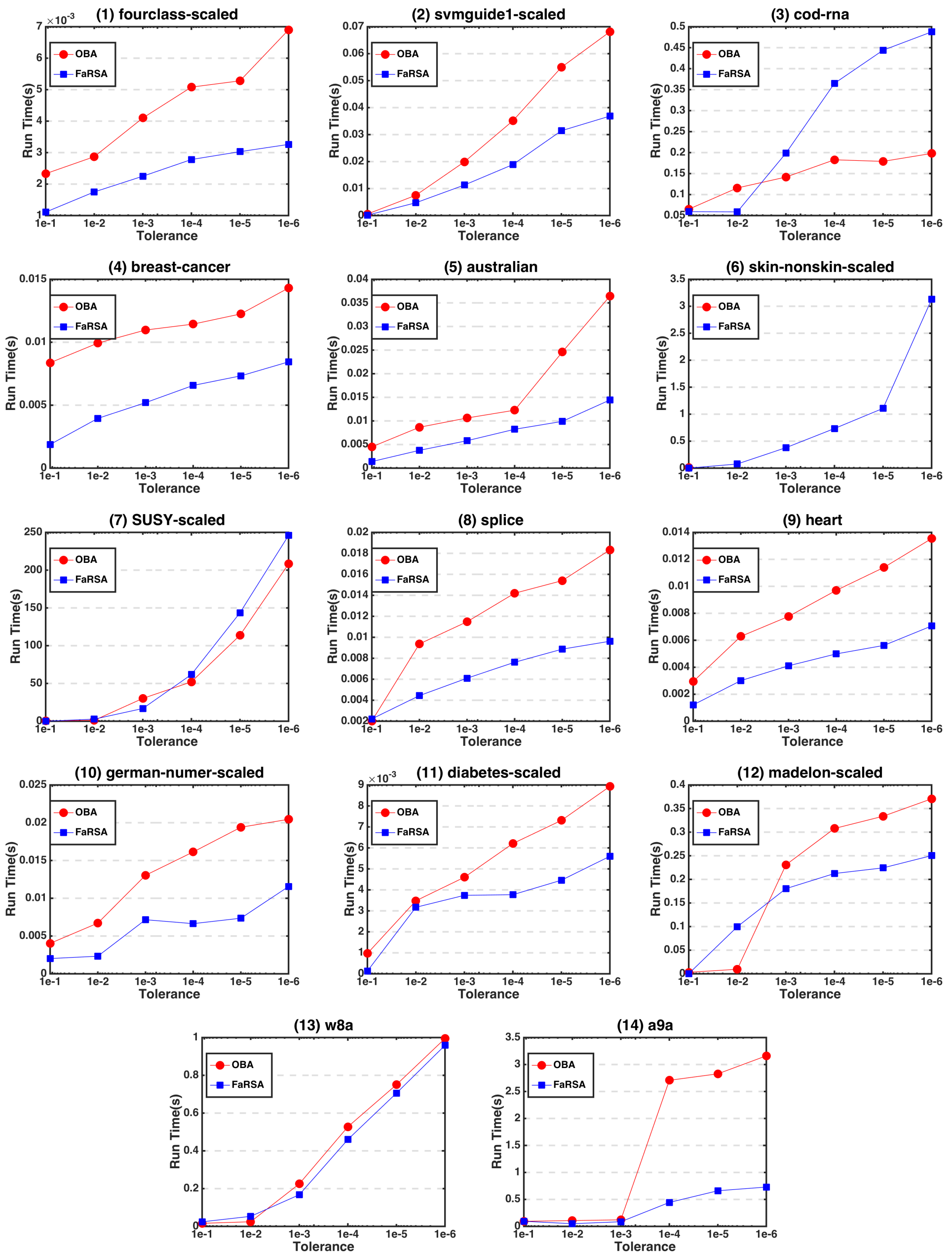

FIG. 1. *CPU time comparison between FaRSA and OBA for various stopping tolerances on problems* `fourclass` *through* `a9a` *with scaled data.*

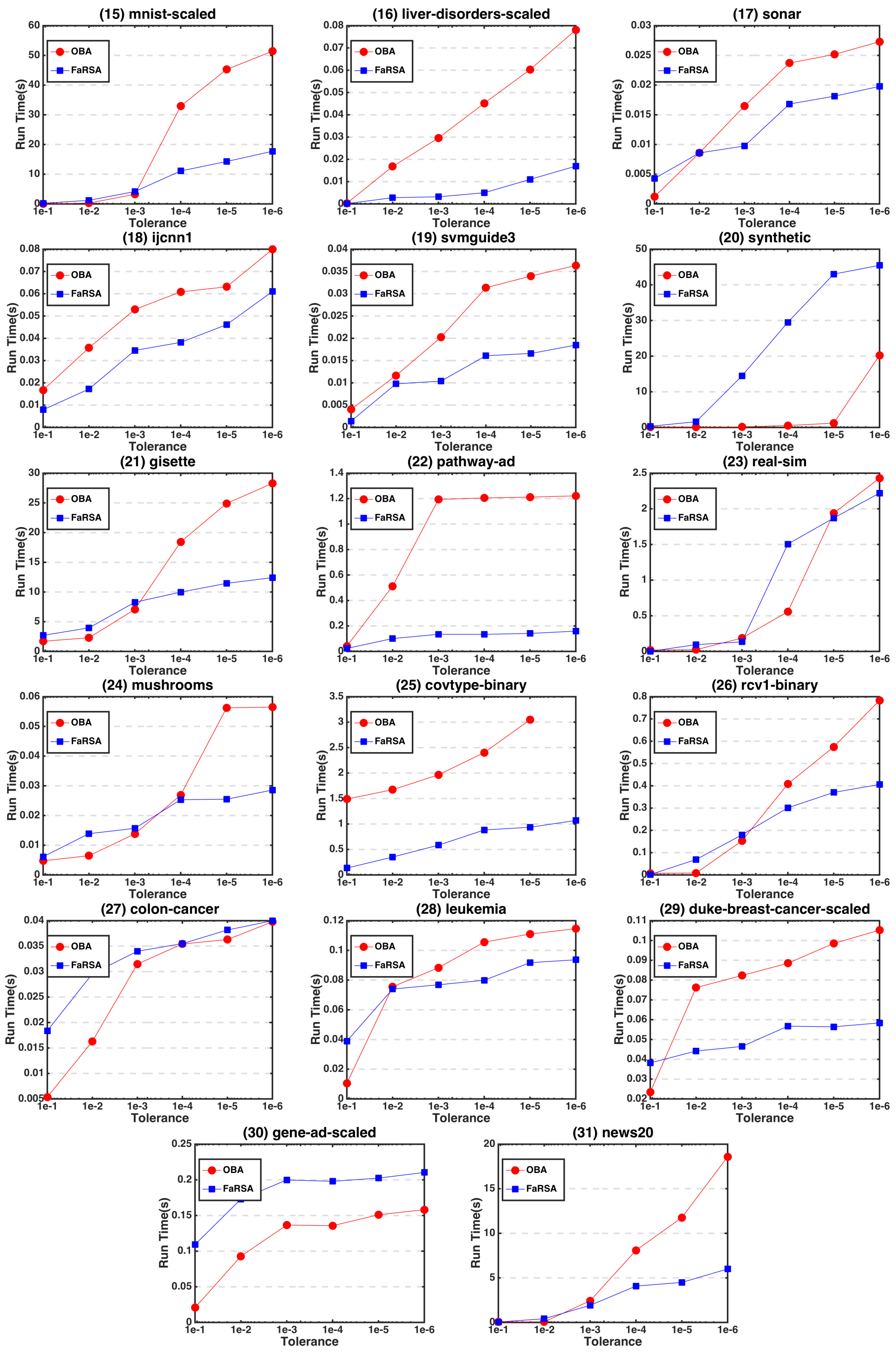

Fig. 2. *CPU time comparison between FaRSA and OBA for various stopping tolerances on problems* `mnist` *through* `news20` *with scaled data.*

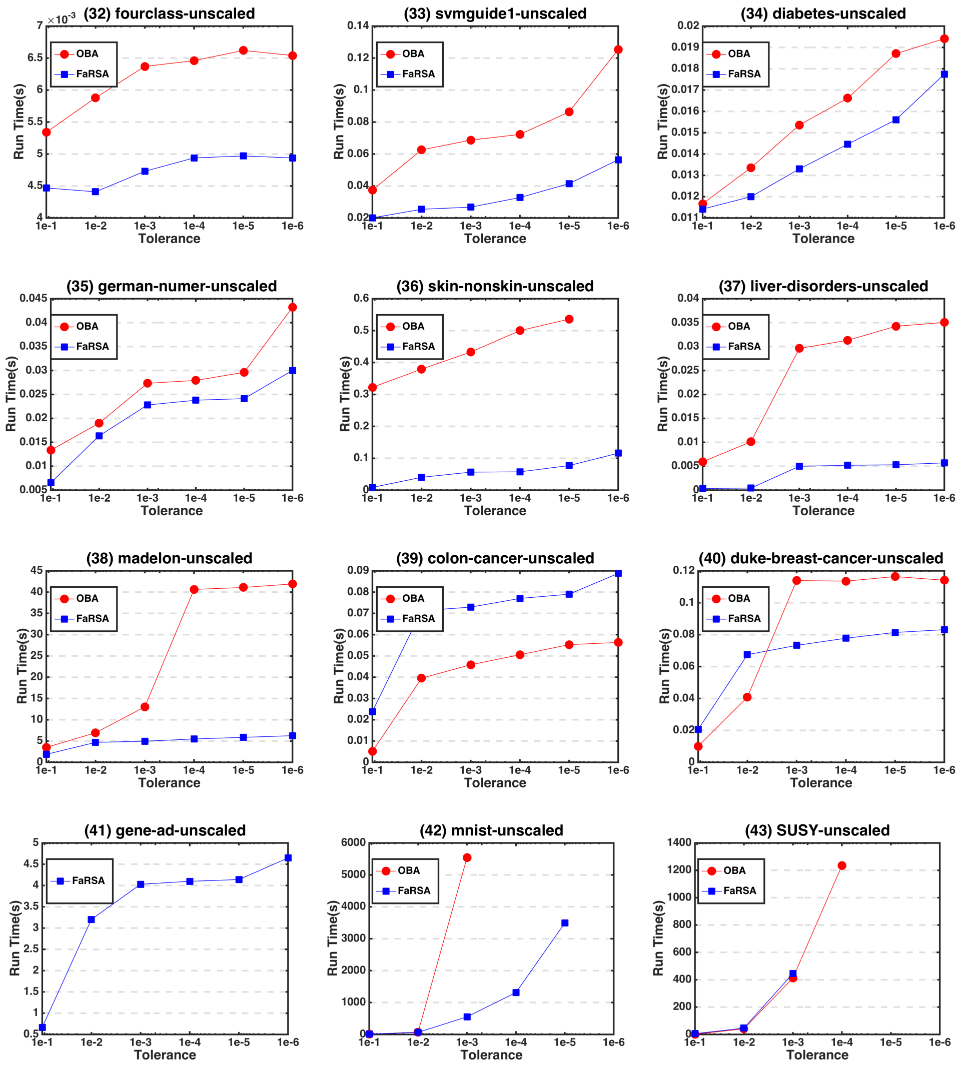

FIG. 3. *CPU time comparison between FaRSA and OBA for various stopping tolerances on the set of problems with unscaled data.*

Appendix C. Numerical experiments comparing FaRSA and ASA-CG. In this section, we compare the performance of FaRSA to the highly successful bound-constrained solver ASA-CG [15] when applied to a bound-constrained reformulation of problem (1). Specifically, problem (1) is equivalent to

$$\underset{u\in\mathbb{R}^n, v\in\mathbb{R}^n}{\text{minimize}}\ f(u-v)+\lambda e^T(u+v)\ \text{ subject to }\ u\geq 0,\ v\geq 0,$$

which has a continuously differentiable objective function and simple bound constraints. The results of solving this optimization problem, with f being the logistic function as described in the beginning of section 4, are given in Table 6. The problems tested include those from Table 1, except that problems mnist, synthetic, pathway-ad, and gene-ad were removed since they would require the creation of special subroutines to interface with the ASA-CG code because they are not part of the LIBSVM repos-

Table 6
Number of objective function evaluations (Fevals), object gradient evaluations (Gevals), and CPU time (Time) measured in seconds for FaRSA and ASA-CG.

		FaRSA			ASA-CG		
Problems	$N \times n$	Fevals	Gevals	Time	Fevals	Gevals	Time
fourclass	1.7e+3	15	8	0.0049	23	14	0.0033
liver-disorders	2.1e+3	15	9	0.0057	66	32	0.0037
heart	3.5e+3	9	9	0.0071	49	35	0.0025
diabetes	6.1e+3	37	23	0.0196	320	184	0.0424
breast-cancer	6.8e+3	14	12	0.0090	48	29	0.0047
australian	9.7e+3	14	12	0.0144	143	88	0.0172
svmguide1	1.2e+4	15	11	0.0564	148	76	0.0719
sonar	1.2e+4	27	18	0.0201	310	198	0.0171
german.numer	2.4e+4	31	24	0.0317	1122	624	0.2004
svmguide3	2.7e+4	25	16	0.0186	335	204	0.0778
splice	6.0e+4	14	10	0.0101	33	25	0.0088
colon-cancer	1.2e+5	165	49	0.0843	231	168	0.0504
leukemia	2.4e+5	100	30	0.0915	247	156	0.1189
duke-breast-cancer	2.7e+5	95	29	0.0894	205	151	0.1170
cod-rna	4.83+5	43	34	0.4876	254	143	2.2957
skin-nonskin	7.4e+5	15	9	0.1166	15	12	0.4279
mushrooms	9.1e+5	27	15	0.0309	87	61	0.1441
madelon	1.0e+6	255	163	6.5567	85545	49956	179.08
ijcnn1	1.1e+6	10	10	0.0615	238	139	2.1032
a9a	4.0e+6	65	35	0.7820	1616	952	9.6969
w8a	1.5e+7	41	24	0.9708	564	362	5.1881
gisette	3.0e+7	185	89	13.305	4585	2666	263.12
covtype.binary	3.1e+7	40	6	1.0416	51	31	4.1737
SUSY	9.0e+7	—	—	max iter	10	7	9.6931
rcv1.binary	9.6e+8	17	17	0.3919	269	166	2.3506
real-sim	1.5e+9	26	19	2.3221	271	145	5.8905
news20	2.7e+10	23	19	6.0909	652	376	35.004

itory. Here, we have placed the test problems in increasing order according to the value $N \times n$, i.e., the product of the data set size (N) with the number of features (n). We used the same experimental setup for FaRSA as described in section 4.2 and used the default setup provided by the ASA-CG software package.

We can clearly observe that FaRSA becomes better compared to ASA-CG as $N \times n$ increases in terms of function evaluations, gradient evaluations, and time. We suspect that an implementation of FaRSA in a computer language such as C (the same language that ASA-CG is written in) would allow FaRSA to outperform ASA-CG on every test problem except for problem SUSY, which FaRSA failed to solve. However, it is also important to remark that ASA-CG only uses first derivatives in its implementation.

Acknowledgments. We thank Nitish Keskar, Jorge Nocedal, Figen Öztoprak, and Andreas Wächter for providing the Matlab code of their OBA algorithm and for several discussions on their numerical experience with OBA. We also thank Qingsong Zhu for providing us the data sets gene-ad and pathway-ad used in section 4.1. Finally, we thank the referees for taking the time to read our manuscript and for their helpful comments and suggestions.

REFERENCES

[1] G. Andrew and J. Gao, *Scalable training of l_1-regularized log-linear models*, in Proceedings of the 24th International Conference on Machine Learning, 2007, pp. 33–40.

[2] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM J. Imaging Sci., 2 (2009), pp. 183–202.

[3] R. H. Byrd, G. M. Chin, J. Nocedal, and F. Oztoprak, *A family of second-order methods for convex ℓ_1-regularized optimization*, Math. Program., 159 (2016), pp. 435–467.

[4] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, *A limited memory algorithm for bound constrained optimization*, SIAM J. Sci. Comput., 16 (1995), pp. 1190–1208.

[5] R. H. Byrd, J. Nocedal, and F. Oztoprak, *An inexact successive quadratic approximation method for ℓ_1 regularized optimization*, Math. Program., 157 (2016), pp. 375–396.

[6] F. E. Curtis, Z. Han, and D. P. Robinson, *A globally convergent primal-dual active-set framework for large-scale convex quadratic optimization*, Comput. Oper. Appl., 60 (2015), pp. 311–341.

[7] Z. Dostál, *Box constrained quadratic programming with proportioning and projections*, SIAM J. Optim., 7 (1997), pp. 871–887, https://doi.org/10.1137/S1052623494266250.

[8] Z. Dostál, *A proportioning based algorithm with rate of convergence for bound constrained quadratic programming*, Numer. Algorithms, 34 (2003), pp. 293–302, https://doi.org/10.1023/B:NUMA.0000005347.98806.b2.

[9] Z. Dostál and J. Schoberl, *Minimizing quadratic functions subject to bound constraints with the rate of convergence and finite termination*, Comput. Oper. Appl., 30 (2005), pp. 23–43, https://doi.org/10.1023/B:COAP.0000049888.80264.25.

[10] E. Elhamifar and R. Vidal, *Sparse subspace clustering*, in IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2009, pp. 2790–2797.

[11] A. Friedlander, J. M. Martínez, and M. Raydon, *A new method for large-scale box constrained convex quadratic minimization problems*, Optim. Meth. Softw., 5 (1995), pp. 57–74.

[12] A. Friedlander and J. M. Martínez, *On the numerical solution of bound constrained optimization problems*, Revue française d'automatique, d'informatique et de recherche opérationnelle. Rech. Oper., 23 (1989), pp. 319–341.

[13] A. Friedlander and J. M. Martinez, *On the maximization of a concave quadratic function with box constraints*, SIAM J. Optim., 4 (1994), pp. 177–192.

[14] M. D. Gonzalez-Lima, W. W. Hager, and H. Zhang, *An affine-scaling interior-point method for continuous knapsack constraints*, SIAM J. Optim., 21 (2011), pp. 361–390.

[15] W. W. Hager and H. Zhang, *A new active set algorithm for box constrained optimization*, SIAM J. Optim., 17 (2006), pp. 526–557.

[16] C.-J. Hsieh, I. S. Dhillon, P. K. Ravikumar, and M. A. Sustik, *Sparse inverse covariance matrix estimation using quadratic approximation*, in Advances in Neural Information Processing Systems, 2011, pp. 2330–2338.

[17] T. Joachims, *Making large-scale support vector machine learning practical*, Advances in Kernel Methods: Support Vector Machines, 1999.

[18] N. Keskar, J. Nocedal, F. Öztoprak, and A. Wächter, *A second-order method for convex 1-regularized optimization with active-set prediction*, Optim. Meth. Softw., 31 (2016), pp. 605–621.

[19] M. Ko, J. Zowe, et al., *An iterative two-step algorithm for linear complementarity problems*, Numer. Math., 68 (1994), pp. 95–106.

[20] J. Lee, Y. Sun, and M. Saunders, *Proximal newton-type methods for convex optimization*, in Advances in Neural Information Processing Systems, 2012, pp. 836–844.

[21] C.-J. Lin and J. J. Moré, *Newton's method for large bound-constrained optimization problems*, SIAM J. Opti., 9 (1999), pp. 1100–1127.

[22] H. Mohy-ud-Din and D. P. Robinson, *A solver for nonconvex bound-constrained quadratic optimization*, SIAM J. Optim., 25 (2015), pp. 2385–2407, https://doi.org/10.1137/15M1022100.

[23] J. J. Moré and G. Toraldo, *On the solution of large quadratic programming problems with bound constraints*, SIAM J. Optim., 1 (1991), pp. 93–113.

[24] D. P. Robinson, L. Feng, J. M. Nocedal, and J.-S. Pang, *Subspace accelerated matrix splitting algorithms for asymmetric and symmetric linear complementarity problems*, SIAM J. Optim., 23 (2013), pp. 1371–1397.

[25] K. Scheinberg and X. Tang, *Practical inexact proximal quasi-newton method with global complexity analysis*, preprint arXiv:1311.6547 (2013).

[26] T. Serafini, G. Zanghirati, and L. Zanni, *Gradient projection methods for quadratic programs and applications in training support vector machines*, Optim. Meth. Softw., 20 (2005), pp. 353–378.

[27] T. Serafini and L. Zanni, *On the working set selection in gradient projection-based decomposition techniques for support vector machines*, Optim. Meth. Softw., 20 (2005), pp. 583–596.

[28] H. M. ud Din and D. P. Robinson, *A solver for nonconvex bound-constrained quadratic optimization*, SIAM J. Optim., 25 (2015), pp. 2385–2407, https://doi.org/10.1137/15M1022100.

[29] S. J. Wright, R. D. Nowak, and M. A. Figueiredo, *Sparse reconstruction by separable approximation*, IEEE Trans. Signal Process., 57 (2009), pp. 2479–2493.

[30] C. Yang, D. P. Robinson, and R. Vidal, *Sparse subspace clustering with missing entries*, in International Conference on Machine Learning, 2015, pp. 2463–2472.

[31] C. You, C.-G. Li, D. P. Robinson, and R. Vidal, *Oracle based active set algorithm for scalable elastic net subspace clustering*, in IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[32] C. You, D. P. Robinson, and R. Vidal, *Sparse subspace clustering by orthogonal matching pursuit*, in IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[33] G.-X. Yuan, C.-H. Ho, and C.-J. Lin, *An improved glmnet for l1-regularized logistic regression*, J. Mach. Learn. Res., 13 (2012), pp. 1999–2030.

[34] L. Zanni, *An improved gradient projection-based decomposition technique for support vector machines*, Comput. Manag. Sci., 3 (2006), pp. 131–145.

[35] L. Zanni, T. Serafini, and G. Zanghirati, *Parallel software for training large scale support vector machines on multiprocessor systems*, J. Mach. Learn. Res., 7 (2006), pp. 1467–1492.

[36] Q. Zhu, E. Izumchenko, A. M. Aliper, E. Makarev, K. Paz, A. A. Buzdin, A. A. Zhavoronkov, and D. Sidransky, *Pathway activation strength is a novel independent prognostic biomarker for cetuximab sensitivity in colorectal cancer patients*, Hum. Gen. Var., 2 (2015).