

Fast First-Order Methods for Composite Convex Optimization with Backtracking

Katya Scheinberg · Donald Goldfarb · Xi Bai

Received: 13 April 2011 / Revised: 3 January 2013 / Accepted: 10 December 2013 /
Published online: 26 March 2014
© SFoCM 2014

Abstract We propose new versions of accelerated first-order methods for convex composite optimization, where the prox parameter is allowed to increase from one iteration to the next. In particular, we show that a full backtracking strategy can be used within the FISTA and FALM algorithms while preserving their worst-case iteration complexities of $O(\sqrt{L(f)}/\epsilon)$. In the original versions of FISTA and FALM the prox parameter value on each iteration must be bounded from above by its value on prior iterations. The complexity of the algorithm then depends on the smallest value of the prox parameter encountered by the algorithm. The theoretical implications of using full backtracking in the framework of accelerated first-order and alternating linearization methods allow for better complexity estimates that depend on the “average” prox parameter value. Moreover, we show that in the case of compressed sensing problem and Lasso, the additional cost of the new backtracking strategy is negligible compared to the cost of the original FISTA iteration. Our computational results show the benefit of the new algorithm.

Keywords Convex optimization · Composite optimization · Accelerated first-order methods · Convergence rates

Mathematics Subject Classification 90C25

Communicated by Yurii Nesterov.

K. Scheinberg (✉) · X. Bai
Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA 18015, USA
e-mail: katas@lehigh.edu

X. Bai
e-mail: xib210@lehigh.edu

D. Goldfarb
Department of Industrial Engineering and Operations Research, Columbia University,
New York, NY 10027, USA
e-mail: goldfarb@columbia.edu

1 Introduction

First-order methods for convex optimization have become a focus of active research during the past several years. Their low per-iteration complexity and improved convergence rates have made them a viable alternative to the higher-order, but expensive, interior-point methods. The rapid growth of convex optimization applications in signal processing and machine learning has further increased the popularity of first-order methods.

We are interested in this paper in the following so-called composite unconstrained convex programming problem:

$$(P) \quad \min\{F(x) \equiv f(x) + g(x) : x \in \mathbf{R}^n\}, \quad (1.1)$$

where $f : \mathbf{R}^n \rightarrow \mathbf{R}$ and $g : \mathbf{R}^n \rightarrow \mathbf{R}$ are both convex functions and only f is required to be smooth. Many problems that arise in, for example, compressed sensing and statistical learning are of this form. For instance, in compressed sensing $g(x) = \|x\|_1$. This form also accommodates constrained problems with a simple constraint set X by setting $g(x)$ to be an indicator function of $x \in X$.

A class of first-order methods, pioneered by Nesterov [10] and referred to as accelerated gradient methods, enjoys an improved convergence rate compared to the classical gradient method. This class of methods has recently become exceptionally popular, and several variants have been developed (e.g., [1, 11, 12], [14]). In particular, we note that the so-called fast iterative shrinkage/thresholding algorithm (FISTA) of Beck and Teboulle [1], which enjoys the same improved convergence rate as Nesterov's optimal gradient method for convex composite objective functions [9], is designed to solve problems of the form (1.1).

Like the first-order algorithms proposed in [9], FISTA computes an ϵ -optimal solution in $O(\sqrt{L(f)}/\epsilon)$ steps, where $L(f)$ is a bound on the Lipschitz constant for $\nabla f(x)$. Hence, it is an *optimal gradient* method since this is the best complexity bound that one can obtain using only first-order information [8, 11]. In these accelerated gradient methods, a combination of previous points is used to compute the new point at each iteration. In [12, 14], these techniques together with smoothing techniques were applied to nonsmooth problems yielding optimal complexity results for such problems.

FISTA requires one gradient computation per iteration, unlike the method in [9]. Although the method in [9] employs a more general framework than FISTA, FISTA's convergence rate does not follow from the analysis in [9]. On the other hand, FISTA and its analysis in [1] are very simple and intuitive, which has led to its being used in many applications. The principal drawback of FISTA compared to the method in [9] is the requirement that the estimates of the Lipschitz constant (or the prox parameters) that are used must be nondecreasing from one iteration to the next. This can substantially limit the performance of FISTA when a large Lipschitz constant estimate is encountered early in the algorithm since this causes the sizes of the steps taken at that point, and at all subsequent iterates, to be very small.

Another very popular class of first-order methods for problems of the form (1.1) is the class of alternating direction methods (ADMs or ADMMs) [3]. This class of

methods has been shown to be very effective in practice for many statistical learning applications, but for most of its variants the convergence rates are either unknown or worse than those of the accelerated gradient methods such as FISTA. A recently proposed fast alternating linearization method (FALM) [7] is an accelerated version of an ADM and was shown to have the same complexity estimates as FISTA under similar assumptions. It is applied to problems for which both f and g have a special structure that allows the proximity operators $p_\mu^f(y)$ and $p_\mu^g(y)$ (defined subsequently) to be easily computed. Here we introduce the backtracking (BKTR) variant FALM, which allows for large steps and hence has better practical behavior. Moreover, considering backtracking within the FALM framework allows us to generalize the convergence rate results and substantially simplify the proofs in [7].

In addition, we discuss the concept of a *local composite Lipschitz constant* and show that with the use of a backtracking strategy the constants in complexity bounds for the methods in [1], [7], and [9] can be improved to depend on the average local composite Lipschitz constant for $\nabla f(x)$ rather than the worst-case Lipschitz constant $L(f)$, as in all previously derived bounds for accelerated first-order schemes. This result is potentially important in cases where the Lipschitz constant is very large at (or near) the solution but is moderate almost everywhere else. Important examples of such functions include smooth approximations of nonsmooth convex functions.

The paper is organized as follows. In the next section we introduce some preliminary results and definitions. In Sect. 3 we introduce and analyze the FISTA algorithm with full line search. In Sect. 3.1 we consider the more restricted line search scheme proposed in [9] and show that it can be applied to FISTA. We then discuss the complexity implications of this scheme. In Sect. 4 we extend our line search approach and our complexity analysis to the FALM method. We conclude in Sect. 5 with some computational results.

2 Preliminary Results

The following assumption is made throughout the paper:

Assumption 2.1 $f : \mathbf{R}^n \rightarrow \mathbf{R}$ is a smooth convex function of the type $C^{1,1}$, i.e., continuously differentiable with Lipschitz continuous gradient $L(f)$:

$$\|\nabla f(x) - \nabla f(y)\| \leq L(f)\|x - y\|, \forall x, y \in \mathbf{R}^n, \tag{2.1}$$

where $\|\cdot\|$ stands for standard Euclidean norm, unless specified otherwise.

We define the following notation.

Definition 2.1

$$Q_\mu(u, v) := f(v) + \langle u - v, \nabla f(v) \rangle + \frac{1}{2\mu} \|u - v\|^2 + g(u), \tag{2.2}$$

$$p_\mu(v) := \arg \min_u Q_\mu(u, v). \tag{2.3}$$

The following lemma from [1] plays a key role in our analysis.

Lemma 2.1 For any $y, x \in \mathbf{R}^n$ and $\mu > 0$, if

$$F(p_\mu(y)) \leq Q_\mu(p_\mu(y), y), \tag{2.4}$$

where $p_\mu(y)$ is given in Definition 2.1, then

$$2\mu(F(x) - F(p_\mu(y))) \geq \|p_\mu(y) - y\|^2 + 2\langle y - x, p_\mu(y) - y \rangle = \|p_\mu(y) - x\|^2 - \|y - x\|^2. \tag{2.5}$$

Proof The proof can be found in [1]. □

The following fact is well known and will be used in our theoretical analysis: if $\mu \leq 1/L(f)$, then (2.4) holds for any y , and hence (2.5) holds for any x and y .

2.1 Local Composite Lipschitz Constants

The lower bound on the value of the prox parameter μ is a key ingredient in the complexity bound for the algorithms that we consider in this paper. As we will see, the actual value of μ at each iteration is determined via a backtracking line search so that condition (2.4) is satisfied. A lower bound on μ can be derived from the facts that μ is reduced by a constant factor at each line search step and as soon as $\mu \leq 1/L(f)$, condition (2.4) is satisfied and the line search terminates. On the other hand condition (2.4) may be satisfied with a larger value of μ . As we will see from our results, increasing the lower bound on μ at any of the iterations improves the complexity bound on the algorithms. In an effort to improve this lower bound we now define a *local, composite* Lipschitz constant $L(f, g, y)$ whose value never exceeds that of $L(f)$ but is often smaller.

As a first step consider any two vectors $u, v \in \mathbf{R}^n$, and let $[u, v]$ denote the set of points on a segment between u and v , in other words, $[u, v] = \{x : x = \lambda u + (1 - \lambda)v, 0 \leq \lambda \leq 1\}$. Let $L_{[u,v]}(f)$ be the Lipschitz constant of $\nabla f(x)$ restricted to $[u, v]$, i.e.,

$$\|\nabla f(x) - \nabla f(y)\| \leq L_{[u,v]}(f)\|x - y\|, \forall x, y \in [u, v].$$

From simple calculus it follows that

$$f(u) \leq f(v) + \nabla f(v)^\top(u - v) + \frac{L_{[u,v]}(f)}{2}\|u - v\|^2. \tag{2.6}$$

Note that the roles of u and v are interchangeable.

Definition 2.2 We call $L(f, g, y)$ a local composite Lipschitz constant for $\nabla f(x)$ at y if

$$\|\nabla f(x_1) - \nabla f(x_2)\| \leq L(f, g, y)\|x_1 - x_2\|, \forall x_1, x_2 \in [p_\mu(y), y], \forall \mu \leq 1/L(f, g, y).$$

In other words, $L(f, g, y)$ is a local composite Lipschitz constant if it is a Lipschitz constant for $\nabla f(x)$ restricted to the interval $[p_\mu(y), y]$ for any $\mu \leq 1/L(f, g, y)$.

The dependence of $L(f, g, y)$ on g arises from the dependence of $p_\mu(y)$ on g ; hence we use the term *composite* to emphasize this dependence. If $g(x) \equiv 0$, then $p_\mu(y) = y - \mu \nabla f(y)$ for any μ and, hence, $L(f, g, y)$ is a Lipschitz constant of $\nabla f(x)$ restricted to an interval $[y, y - \frac{1}{L(f,g,y)} \nabla f(x)]$.

For the purposes of this paper we point to two key observations:

- $L(f, g, y) \leq L(f)$ for all y and
- From Definition 2.2 it follows that for any μ and y , such that $\mu \leq 1/L(f, g, y)$, (2.4) holds [by (2.6)], and hence (2.5) holds for the given y and any x .

Let us now illustrate why $1/L(f, g, y)$ may be a better estimate for the prox parameter μ than $1/L(f)$.

Example 2.1 Consider a compressed sensing or Lasso setting:

$$(P) \quad \min\{F(x) \equiv \|Ax - b\|^2 + \rho\|x\|_1 : x \in \mathbf{R}^n\}. \tag{2.7}$$

In this case $f(x) = \|Ax - b\|^2$ and $L(f) = \|AA^\top\|_2$. Now consider a sparse vector \bar{y} ; without loss of generality assume that $\bar{y} = (\bar{y}_1, \bar{y}_2)$ with $\bar{y}_2 = 0$. Also, consider the gradient vector $z = A^\top(A\bar{y} - b)$, and assume that $\|z_2\|_\infty \leq \rho$ (z_2 is the subvector of z that corresponds to the subvector \bar{y}_2). In this case it is easy to see from the properties of the shrinkage operator that $p_\mu(\bar{y})_2 = 0$ for all $\mu > 0$. This implies that for any $x \in [p_\mu(\bar{y}), \bar{y}]$ $x_2 = 0$, and hence $L(f, g, \bar{y}) = \|A_1 A_1^\top\|_2$, which is clearly smaller than $L(f) = \|AA^\top\|_2$, where A_1 is the subset of columns of A that correspond to the subvector y_1 .

Since it may be difficult, in general, to compute the local composite Lipschitz constant accurately, we may consider estimating it via an upper bound that is still lower than $L(f)$. For instance, assume that for a given f, g , and y ,

$$\|\nabla f(x) - \nabla f(y)\| \leq L(f, g, y)\|x - y\|, \forall x : \|x - y\| \leq \|p_{1/L(f,g,y)}(y) - y\|, \tag{2.8}$$

in other words, $L(f, g, y)$ is a Lipschitz constant of $\nabla f(x)$ restricted on the ball around y of radius $\|p_{1/L(f,g,y)}(y) - y\|$. Then $L(f, g, y)$ is a local composite Lipschitz constant for these f, g , and y , and hence for any $\mu < 1/L(f, g, y)$, (2.4) holds.

To prove this, it is sufficient to verify that $\|p_\mu(y) - y\| \leq \|p_{1/L(f,g,y)}(y) - y\|$ for any $\mu < 1/L(f, g, y)$.

Lemma 2.2 *Suppose that $p_\mu(y) := \arg \min_x Q_\mu(x, y)$ for a given y , where $Q_\mu(x, y)$ is defined as in (2.1). If $\mu_1 \leq \mu_2$, then $\|p_{\mu_1}(y) - y\| \leq \|p_{\mu_2}(y) - y\|$.*

Proof Assume $\mu_1 > \mu_2$ ($\mu_1 = \mu_2$ is trivial); then $\alpha \equiv \frac{1}{2\mu_1} - \frac{1}{2\mu_2} > 0$. Suppose that $\|p_{\mu_1}(y) - y\| > \|p_{\mu_2}(y) - y\|$. Then

$$\begin{aligned} Q_{\mu_1}(p_{\mu_1}(y), y) - Q_{\mu_2}(p_{\mu_1}(y), y) &= \alpha \|p_{\mu_1}(y) - y\|^2 > \alpha \|p_{\mu_2}(y) - y\|^2 \\ &= Q_{\mu_1}(p_{\mu_2}(y), y) - Q_{\mu_2}(p_{\mu_2}(y), y). \end{aligned}$$

Hence, by applying the fact that $p_\mu(y)$ is a minimizer of $Q_\mu(x, y)$ in x for $\mu = \mu_1$ and $\mu = \mu_2$ we obtain

$$0 > Q_{\mu_1}(p_{\mu_1}(y), y) - Q_{\mu_1}(p_{\mu_2}(y), y) > Q_{\mu_2}(p_{\mu_1}(y), y) - Q_{\mu_2}(p_{\mu_2}(y), y) > 0,$$

which implies a contradiction. □

Example 2.2 To illustrate the case where an upper bound of the local composite Lipschitz constant can be estimated to be lower than the global Lipschitz constant, consider the following simple setting: let $f(x) = H_\nu \|Ax - b\|_2$ be the smooth Huber function approximation of the nonsmooth function $\|Ax - b\|_2$, and let $g(x) \equiv 0$ for simplicity of the derivations. In this case,

$$\nabla f(x) = \begin{cases} \frac{A^\top(Ax-b)}{\nu} & \text{if } \|Ax - b\| \leq \nu, \\ \frac{A^\top(Ax-b)}{\|Ax-b\|_2} & \text{otherwise.} \end{cases}$$

$L(f)$ is then equal to $\|A^\top A\|_2/\nu$, which can be very large if ν is small.

Assume now that a particular vector \bar{y} is given for which

$$\|A\bar{y} - b\| \geq 2\lambda$$

for some $\lambda \gg \nu$. Since the function $\|Ax - b\|$ is Lipschitz continuous with constant $\|A\|_2$, then for any x such that $\|x - \bar{y}\| \leq \lambda/\|A\|_2$ we have

$$\|A\bar{y} - b\| - \|Ax - b\| \leq \|A\|_2 \|x - \bar{y}\| \leq \lambda,$$

and therefore

$$\|Ax - b\| \geq \lambda.$$

From the properties of the Huber function $\|\nabla^2 f(x)\| \leq \|A^\top A\|_2/\lambda$. Now consider $\mu \leq \lambda/\|A^\top A\|_2$, since $p_\mu(\bar{y}) = \bar{y} - \mu \nabla f(\bar{y})$; then

$$\|p_\mu(\bar{y}) - \bar{y}\| \leq \mu \|A\|_2 \leq \lambda/\|A\|_2.$$

This implies that (2.8) holds for $L(f, g, \bar{y}) = \|A^\top A\|_2/\lambda$; in other words, the local composite Lipschitz constant for this case is bounded from above by $\|A^\top A\|_2/\lambda$, which is significantly smaller than the worst-case Lipschitz constant $\|A^\top A\|_2/\nu$. We can conclude that

$$L(f, g, y) \leq \frac{\|A^\top A\|_2}{2\|Ay - b\|}, \quad \forall y : \|Ay - b\| \geq 2\nu. \tag{2.9}$$

The preceding definition of local composite Lipschitz constants is provided here to motivate the results that follow. In the following sections we will show that the complexity bounds of line search variants of FISTA and FALM depend on the *average* value of the prox parameter rather than the worst-case lower bound. Hence, if a sequence of prox parameters can be bounded from below by a sequence of local estimates, for instance by inverses of local composite Lipschitz constants, then this can potentially provide significantly better complexity bounds for FISTA and FALM for certain problem classes.

3 Fast Iterative Shrinkage/Thresholding Method with Variable Step Sizes

In this section we develop an extension of the fast iterative shrinkage/thresholding algorithm (FISTA) of [1] that computes the step size via backtracking starting from any value. We first consider the following generalization (Algorithm 1) of FISTA. In FISTA $\theta_k = 1$, for all k , we will explain why this choice is imposed and how it can be generalized.

Algorithm 1

FISTA

0. Set $t_1 = 1$, $0 < \beta < 1$ and $y^1 = x^0$, $\mu_0 > 0$;

1. for $k = 1, 2, \dots$ do

(1) Choose $0 < \bar{\mu}_k < \mu_0$, $\theta_k > 0$

(2) Find the smallest $i_k \geq 0$ such that $\mu_k = \beta^{i_k} \bar{\mu}_k$ and $F(p_{\mu_k}(y^k)) \leq Q_{\mu_k}(p_{\mu_k}(y^k), y^k)$

(3) $x^k := p_{\mu^k}(y^k)$

$$t_{k+1} := \left(1 + \sqrt{1 + 4\theta_k t_k^2}\right) / 2$$

$$y^{k+1} := x^k + \frac{t_k - 1}{t_{k+1}} [x^k - x^{k-1}]$$

According to Step 2 of this algorithm, if μ_0 and $\bar{\mu}_k$ are chosen so that $\mu_0 > 1/L(f)$ and $\mu_k \geq \mu_{k-1}$, then $\mu_k \geq \beta/L(f)$, $\forall k$, since, as already noted, $F(p_{\mu_k}(y^k)) \leq Q_{\mu_k}(p_{\mu_k}(y^k), y^k)$ holds for any y_k if $\mu_k < 1/L(f)$. The following lemma is an immediate consequence of this.

Lemma 3.1 *Let $\mu_0 > 1/L(f)$; then at each iteration of Algorithm 1 there are at most $\log_{\frac{1}{\beta}}(\mu_0 L(f)) + 1$ line search backtracking steps and, hence, computations of $p_{\mu_k}(y_k)$.*

If for a given y_k we define a local composite local Lipschitz constant $L_k (= L(f, g, y_k))$, as discussed in Sect. 2.1, then $\mu_k \geq \beta/L_k$, and the number of line search steps at iteration k is at most $\log_{\frac{1}{\beta}}(\mu_0 L_k) + 1$.

The complexity of Algorithm 1 for $\theta_k \equiv 1$ is analyzed in [1]. The proof in [1] relies on the fact that $\bar{\mu}_{k+1}$ is chosen so that $\mu_{k+1} \leq \mu_k$ for all $k \geq 1$, i.e., the step size μ_k is monotonically nonincreasing. Here we will use an analysis similar to that in

[1] but that allows for increases in the step size in Algorithm 1 while preserving the algorithm’s $O(\sqrt{L(f)/\epsilon})$ complexity. As we will show, we can accomplish this by choosing appropriate values of θ_k .

We will first need an auxiliary lemma, which is analogous to a part of a theorem in [1].

Lemma 3.2 *Assume that on each step of Algorithm 1, $\mu_k t_k^2 \geq \mu_{k+1} t_{k+1} (t_{k+1} - 1)$. Then for $v_k = F(x^k) - F(x^*)$ and $\mathbf{u}_k = t_k x_k - (t_k - 1)x_{k-1} - x^*$ we have for all $k \geq 1$*

$$2\mu_k t_k^2 v_k + \|\mathbf{u}_k\|^2 \geq 2\mu_{k+1} t_{k+1}^2 v_{k+1} + \|\mathbf{u}_{k+1}\|^2. \tag{3.1}$$

Proof Let $y = y^{k+1}$, $\mu = \mu_{k+1}$, and $p_\mu(y) = x^{k+1}$ in Lemma 2.1. Then from this lemma it follows that for $x = x^k$ we have

$$2\mu_{k+1}(v_k - v_{k+1}) \geq \|y^{k+1} - x^{k+1}\|^2 + 2\langle x^{k+1} - y^{k+1}, y^{k+1} - x^k \rangle, \tag{3.2}$$

and for $x = x^*$ we have

$$-2\mu_{k+1} v_{k+1} \geq \|y^{k+1} - x^{k+1}\|^2 + 2\langle x^{k+1} - y^{k+1}, y^{k+1} - x^* \rangle. \tag{3.3}$$

Multiplying the first inequality by $t_{k+1} - 1$ and adding it to the second inequality we have

$$2\mu_{k+1}((t_{k+1} - 1)v_k - t_{k+1}v_{k+1}) \geq t_{k+1}\|y^{k+1} - x^{k+1}\|^2 + 2\langle x^{k+1} - y^{k+1}, t_{k+1}y^{k+1} - (t_{k+1} - 1)x_k - x^* \rangle.$$

Then multiplying this inequality by t_{k+1} and rearranging the right-hand-side terms yields

$$2\mu_{k+1}((t_{k+1} - 1)t_{k+1}v_k - t_{k+1}^2 v_{k+1}) \geq \|t_{k+1}x^{k+1} - (t_{k+1} - 1)x_k - x^*\|^2 - \|t_{k+1}y^{k+1} - (t_{k+1} - 1)x_k - x^*\|^2.$$

Now from the definition of y^{k+1} in Algorithm 1 and \mathbf{u}_k ,

$$t_{k+1}y^{k+1} = t_{k+1}x_k + (t_k - 1)(x_k - x_{k-1}),$$

and it follows that

$$t_{k+1}y^{k+1} - (t_{k+1} - 1)x_k - x^* = t_k x_k - (t_k - 1)x_{k-1} - x^* = \mathbf{u}_k.$$

Hence,

$$2\mu_{k+1}((t_{k+1} - 1)t_{k+1}v_k - t_{k+1}^2 v_{k+1}) \geq \|\mathbf{u}_{k+1}\|^2 - \|\mathbf{u}_k\|^2. \tag{3.4}$$

Since $\mu_k t_k^2 \geq \mu_{k+1} t_{k+1} (t_{k+1} - 1)$, we then have

$$2(\mu_k t_k^2 v_k - \mu_{k+1} t_{k+1}^2 v_{k+1}) \geq \|\mathbf{u}_{k+1}\|^2 - \|\mathbf{u}_k\|^2, \tag{3.5}$$

which is equivalent to (3.1). □

Theorem 3.1 *Assume that on each step of Algorithm 1, $\mu_k t_k^2 \geq \mu_{k+1} t_{k+1} (t_{k+1} - 1)$. The sequence $\{x^k\}$ generated by Algorithm 1 satisfies*

$$F(x^k) - F(x^*) \leq \frac{\|x^0 - x^*\|^2}{2\mu_k t_k^2}. \tag{3.6}$$

Proof Since $\|\mathbf{u}_k\|^2 \geq 0$, $t_1 = 1$, and $\mathbf{u}_1 = x_1 - x^*$, it follows from Lemma 3.2 that

$$2\mu_k t_k^2 v_k \leq 2\mu_k t_k^2 v_k + \|\mathbf{u}_k\|^2 \leq 2\mu_1 t_1^2 v_1 + \|x_1 - x^*\|^2. \tag{3.7}$$

But from Lemma 2.1 with $x = x^*$, $y = y_1 = x_0$, $t_1 = 1$, and $\mu = \mu_1$, we have that

$$-2\mu_1 v_1 \geq \|x^1 - x^*\|^2 - \|x^0 - x^*\|^2. \tag{3.8}$$

Therefore, combining (3.7) and (3.8) we obtain that

$$2\mu_k t_k^2 v^k \leq \|x^0 - x^*\|^2,$$

which is equivalent to (3.6). □

The result we want to obtain is

$$F(x^k) - F(x^*) = v_k \leq \frac{\|x^0 - x^*\|^2}{\eta k^2} \tag{3.9}$$

for some fixed $\eta > 0$. For this to be the case, it is sufficient to show that our algorithm satisfies

$$2\mu_k t_k^2 \geq \eta k^2 \tag{3.10}$$

while maintaining

$$\mu_k t_k^2 \geq \mu_{k+1} t_{k+1} (t_{k+1} - 1). \tag{3.11}$$

From the update of t_k in Algorithm 1,

$$t_{k+1} = (1 + \sqrt{1 + 4\theta_k t_k^2})/2, \tag{3.12}$$

it follows that

$$\theta_k t_k^2 = t_{k+1} (t_{k+1} - 1). \tag{3.13}$$

Hence, as long as $\theta_k \leq \mu_k / \mu_{k+1}$, property (3.11) holds. We know that if we implement the constraint $\mu_{k+1} \leq \mu_k$ and use $\theta_k = 1$, as is done in the FISTA algorithm

in [1], then both conditions (3.10) and (3.11) are satisfied with $\eta = \beta/L(f)$. However, if we want to allow $\mu_{k+1} > \mu_k$, say $\mu_{k+1} \geq \mu_k/\beta$, then we need to have $\theta_k < 1$, e.g., $\theta_k \leq \beta$. On the other hand, condition (3.10) requires $t_k \geq k\sqrt{\frac{\eta}{2\mu_k}} \geq k\sqrt{\frac{\eta}{2\mu_0}}$, hence t_k needs to grow at the same rate as k . From (3.13) and the fact that $t_k > 1$ for all $k > 1$,

$$(t_{k+1} - 1)^2 < t_{k+1}(t_{k+1} - 1) = \theta_k t_k^2,$$

and hence, $t_{k+1} < \theta_k^{1/2} t_k + 1$. Assume that for all k , we allow $\theta_k \leq \beta < 1$; then if $t_k > 1/(1 - \beta^{1/2})$, we have $t_{k+1} < t_k$. This means that the sequence t_k will not grow at the required rate if we simply allow $\theta_k < 1$ for all iterations.

To maintain FISTA’s rate of convergence while allowing $\theta_k < 1$ on some iterations, we need to ensure that $\theta_k > 1$ on some of the other iterations.

This can be accomplished on iterations on which $\mu_k/\mu_{k+1} > 1$. The immediate difficulty is that we do not know the value of μ_{k+1} on iteration k when θ_k and t_{k+1} are computed. Simply setting $\theta_k \leq \mu_k/\mu_0$, where μ_0 is an upper bound on the step size, will imply that $\theta_k < 1$ for all k . Hence, it is necessary to update θ_k and t_{k+1} along with μ_{k+1} , thereby expanding the backtracking part of the algorithm,

We now present Algorithm 2, which is an extension of Algorithm 1, which allows for a full backtracking and any size μ_k . It satisfies conditions (3.10)–(3.11) while setting the step size μ_k initially to some value μ_k^0 at the beginning of each iteration. As we will show, Algorithm 2 is designed to maintain $\theta_k = \mu_k/\mu_{k+1}$ for all $k \geq 1$.

Recall that the parameter θ_k is used to compute t_{k+1} and y^{k+1} in Algorithm 1 using the updates

$$\begin{aligned} t_{k+1} &:= (1 + \sqrt{1 + 4\theta_k t_k^2})/2, \\ y^{k+1} &:= x^k + \frac{t_k - 1}{t_{k+1}} [x^k - x^{k-1}]. \end{aligned}$$

Let us denote this computation by

$$(t_{k+1}, y^{k+1}) = FistaStep(x^k, x^{k-1}, t_k, \theta_k).$$

At the end of iteration $k - 1$ of Algorithm 2, $\theta_{k-1} = \mu_{k-1}/\mu_k^0$, where μ_k^0 is an initial “guess” for the value of μ_k . Hence, t_k and y^k are computed using this guess. Once the backtracking is called at iteration k , this guess may turn out to be correct, or it may turn out to be an overestimate of μ_k . If the guess is correct, then no correction to θ_{k-1} is needed. If μ_k is reduced during the backtracking, then θ_{k-1} is recomputed to account for the new value of μ_k so that $\theta_{k-1} = \mu_{k-1}/\mu_k$ is satisfied. Another call to $FistaStep(x^{k-1}, x^{k-2}, t_{k-1}, \theta_{k-1})$ is then necessary. After such a call is made, since the iterate y^k has changed, $p_{\mu_k}(y^k)$ must be recomputed and a new backtracking step performed. The backtracking starts with the value of μ_k with which the previous backtracking stopped and, hence, that was used to compute the most recent value of θ_{k-1} . If the value of μ_k is not reduced any further, then θ_{k-1} , y^k and t_k have the correct value and the backtracking part of the iteration is complete. If the value of μ_k is reduced further, then θ_{k-1} is recomputed again and backtracking continues.

Algorithm 2

FISTA-BKTR

0. Set $t_1 = 1, t_0 = 0, 0 < \beta < 1, \theta_0 = 1$ and $y^1 = x^0 = x^{-1}, \mu_1^0 > 0$;

1. for $k = 1, 2, \dots$ do

(1) Set $\mu_k := \mu_k^0$

(2) Compute $\nabla f(y^k), p_{\mu^k}(y^k)$

If $F(p_{\mu^k}(y^k)) > Q_{\mu^k}(p_{\mu^k}(y^k), y^k)$

$\mu_k := \beta \mu_k, \theta_{k-1} := \theta_{k-1}/\beta$

$(t_k, y^k) := FistaStep(x^{k-1}, x^{k-2}, t_{k-1}, \theta_{k-1})$

Return to (2)

(3) $x^k := p_{\mu^k}(y^k)$

Choose $\mu_{k+1}^0 > 0$, and set $\theta_k := \mu_k/\mu_{k+1}^0$

$(t_{k+1}, y^{k+1}) := FistaStep(x^k, x^{k-1}, t_k, \theta_k)$

We have the following lemma, whose proof follows from the preceding algorithm and discussion.

Lemma 3.3 *At each iteration, upon completion of the backtracking, the iterate x_k is computed as $p_{\mu_k}(y^k)$, where*

$$y^k = x^{k-1} + \frac{t_{k-1} - 1}{t_k} [x^{k-1} - x^{k-2}]$$

and

$$t_k = (1 + \sqrt{1 + 4(\mu_{k-1}/\mu_k)t_{k-1}^2})/2, \tag{3.14}$$

which is equivalent to (3.12) since $\theta_{k-1} = \mu_{k-1}/\mu_k$.

From the expression for t_k in this lemma, (3.11) is satisfied for each k . We now need to show that condition (3.14) implies (3.10), in which case the complexity result (3.9) follows.

Lemma 3.4 *Let $\{\mu_k, t_k\}$ be a sequence that satisfies (3.14); then we have $\mu_k t_k^2 \geq (\sum_{i=1}^k \sqrt{\mu_i}/2)^2$.*

Proof We will prove the lemma using induction. Since $t_1 = 1$, the statement trivially holds for $k = 1$. Let us assume that $\mu_k t_k^2 \geq (\sum_{i=1}^k \sqrt{\mu_i}/2)^2$ for a given $k \geq 1$. From the fact that (3.14) holds for each k it follows that

$$t_{k+1} \geq 1/2 + \sqrt{\frac{\mu_k}{\mu_{k+1}}} t_k,$$

and hence,

$$\sqrt{\mu_{k+1}} t_{k+1} \geq \sqrt{\mu_{k+1}}/2 + \sqrt{\mu_k} t_k.$$

We know that $\sqrt{\mu_k}t_k \geq \sum_{i=1}^k \sqrt{\mu_i}/2$; hence, from the induction assumption

$$\sqrt{\mu_{k+1}}t_{k+1} \geq \sqrt{\mu_{k+1}}/2 + \sum_{i=1}^k \sqrt{\mu_i}/2 = \sum_{i=1}^{k+1} \sqrt{\mu_i}/2,$$

which concludes the induction argument.

Moreover, since $\mu_i \geq \beta/L(f)$,

$$\left(\sum_{i=1}^k \sqrt{\mu_i}/2\right)^2 \geq \frac{k^2\beta}{4L(f)}.$$

□

Since Algorithm 2 maintains $\theta_k = \mu_k/\mu_{k+1}$, (3.11) holds automatically, and we have shown that (3.10) holds with $\eta = \beta L(f)/4$ by Lemma 3.4. Thus the following two complexity results follow immediately from Theorem 3.1.

Theorem 3.2 *At the k th iteration of Algorithm 2 we have*

$$F(x^k) - F(x^*) = v_k \leq \frac{2L(f)\|x^0 - x^*\|^2}{\beta k^2}. \tag{3.15}$$

Theorem 3.3 *Let $\sqrt{L_k} = (\sum_{i=1}^k \sqrt{L(f, g, y^i)})/k$ be the average estimate of local composite Lipschitz constants encountered during the first k iterations of Algorithm 2. Assume that $\mu_i^0 \geq 1/L(f, g, y^i)$ for all $1 \leq i \leq k$. Then*

$$F(x^k) - F(x^*) = v_k \leq \frac{2L_k\|x^0 - x^*\|^2}{\beta k^2}. \tag{3.16}$$

Proof The proof follows trivially by observing that for any $i = 1, \dots, k$ $\mu_i \geq \beta/L(f, g, y^i)$ and applying Lemma 3.4. □

In [9], Nesterov gives an accelerated method for solving problem (1.1) that involves a backtracking process and has an iteration complexity of $O(\sqrt{L(f)}/\epsilon)$. Like Algorithm 2, Nesterov’s method allows repeated reduction of the prox parameter μ_k by a given factor on the k th iteration until a backtracking condition is met.

While Nesterov’s method minimizes the same function Q_{μ_k} on each inner step, as do FISTA and Algorithm 2, it utilizes a different update of the parameters that are used to compute the new extrapolated point y^k and relies on a criterion for terminating each backtracking step other than $F(p_{\mu_k}(y^k)) > Q_{\mu_k}(p_{\mu_k}(y^k), y^k)$. In particular, the criterion in [9] involves $\nabla f(p_{\mu_k}(y_k))$ at each backtracking step, thereby increasing the per-iteration cost compared to that of Algorithm 2 [assuming that the cost of computing $\nabla f(y)$ is higher than that of computing $F(p_{\mu_k}(y^k))$, as in compressed sensing, for instance, where the former requires two matrix vector multiplications, while the latter requires only one]. Whether either of the criteria yields larger step sizes and, hence, possibly fewer iterations remains to be investigated. For the sake

of theoretical comparison, one can easily modify the results in [9] to obtain a lemma similar to Lemma 3.4. However, Theorem 3.3 does not apply directly to the method in [9] because in that case μ_k increases by a constant factor between two consecutive iterations. We discuss this strategy and its advantages next.

Each iteration of the algorithm can start with any value of the prox parameter μ_k^0 , and the condition $\mu_k t_k^2 = \mu_{k+1}(t_{k+1}^2 - t_{k+1})$ is maintained at each iteration. If μ_k^0 is chosen to be a large fixed value μ_0 at each iteration, this implies a potentially large number of calls to the *FistaStep* procedure to update y^k and t^k and the consequent recomputation of $\nabla f(y^k)$, $p_{\mu_k}(y^k)$, and $F(p_{\mu_k}(y^k))$. However, we know that the number of such internal steps does not exceed $\log_{\frac{1}{\beta}}(\mu_0 L(f)) + 1$; hence, the complexity of Algorithm 2 is at most a logarithmic factor $L(f)$ worse than that of FISTA, i.e., Algorithm 1. The advantage of choosing large μ_k^0 is in the potentially rapid increase of the prox parameter if such an increase is possible and desirable due to a rapid change in the local Lipschitz behavior of $\nabla f(x)$.

On the other hand, μ_k^0 can be selected to have a more conservative value (as long as the value of θ_k is chosen accordingly). For instance, one can choose μ_k^0 to equal μ_{k-1}/β at the beginning of the k th iteration. This will significantly reduce the number of backtracking steps. This strategy of increasing μ_k incrementally is what is used in Nesterov’s method in [9], where μ_k^0 is set to μ_{k-1}/α for some fixed $\alpha \geq \beta$. It is then possible to bound the total number of line search steps used by the algorithm. The following lemma is proved in [9] and applies readily to Algorithm 2 if μ_k^0 is chosen to equal μ_{k-1}/α instead of some fixed large value μ_0 .

Lemma 3.5 *For any $k \geq 1$ the number of calls to *FistaStep* by the end of the k th iteration is bounded by*

$$\left[1 + \frac{\ln \alpha}{\ln \beta} \right] (k + 1) + \frac{1}{\ln \beta} \left[\ln \frac{\alpha \mu_0}{\beta(1/L(f))} \right]_+.$$

If we choose $\alpha = \beta$, then we see that the average number of calls of *FistaStep* per iteration is 2. Hence, the logarithmic factor is removed from the overall complexity of Algorithm 2.

It is easy to see that (3.14) still holds for this case, and hence Lemma 3.4 applies. However, the drawback of this approach is that we can no longer guarantee that $\mu_k \geq \beta/L_k$ for each k and Theorem 3.3 may not hold for L_k as the estimate of the local composite Lipschitz constant defined in Sect. 2.1. The bound on μ_k is now $\mu_k \geq \min\{\mu_{k-1}/\beta, \beta L_k\}$. Ideally, one should select the increase of the prox term based on the estimated increase of the local Lipschitz constant. The applicability of such an approach is a subject for future investigation.

Let us consider the extra work required by each of the backtracking steps. The *FistaStep*($x^k, x^{k-1}, t_k, \theta_k$) requires only an additional $O(n)$ operations; hence, the main additional work comes from the necessity of recomputing $F(p_{\mu_k}(y^k))$ and $Q_{\mu_k}(p_{\mu_k}(y^k), y^k)$. In the case of compressed sensing and Lasso problems, which we consider for our computational results, the dominating cost of all these computations can be reduced to one additional matrix vector multiplication per backtracking step.

3.1 A Practical Backtracking FISTA Algorithm for Compressed Sensing and Lasso Problems

The additional cost of each backtracking step of Algorithm 2 compared to that of Algorithm 1 lies in a call to *FistaStep* updates and recomputation of $\nabla f(y^k)$, which is needed to construct $Q_\mu(y^k, x)$. All remaining computational cost is the same for both algorithms. The number of backtracking steps is defined solely by the choice of μ_0^k at each iteration, as discussed in the previous section. The choice of a practical approach is likely to depend on the comparisons of the cost of computation of $\nabla f(y^k)$, $p_{\mu_k}(y^k)$, and $F(p_{\mu_k}(y^k))$. Here we consider specific application of our backtracking algorithm to problems of the form

$$(P) \quad \min\{F(x) \equiv \|Ax - b\|^2 + g(x) : x \in \mathbf{R}^n\}. \tag{3.17}$$

We assume here that $g(x)$ is a simple function, such as $\|x\|_1$, as in the case of CS or Lasso [4, 13] or $\sum_i \|x_i\|_2$ as in the case of group Lasso [15]. In this case, $\nabla f(x) = A^\top(Ax - b)$. Recall the expression for y^k :

$$y^k = x^{k-1} + \frac{t_{k-1} - 1}{t_k} [x^{k-1} - x^{k-2}],$$

which implies that

$$\nabla f(y^k) = \nabla f(x^{k-1}) + \frac{t_{k-1} - 1}{t_k} [\nabla f(x^{k-1}) - \nabla f(x^{k-2})].$$

Hence, if $\nabla f(x^{k-1})$ and $\nabla f(x^{k-2})$ are available, then $\nabla f(y^k)$ can be computed in $O(n)$ operations for any value of t_k . Since the backtracking step changes only the value of t_k but not x^{k-1} or x^{k-2} , this means that the extra cost of each backtracking step of Algorithm 2 compared to that of Algorithm 1 is only $O(n)$, which is negligible.

However, as discussed earlier, using larger values of μ_0^k may result in a higher number of backtracking steps; hence, we should analyze the cost of a backtracking step itself. For simple forms of $g(x)$, computing $p_{\mu_k}(y^k)$ given $\nabla f(y^k)$ takes $O(n)$ operations. Finally, computing $F(p_{\mu_k}(y^k))$ requires one matrix-vector product for computing $Ax - b$. Once x^k is determined via backtracking, an additional matrix-vector product is employed to compute $\nabla f(x^k)$; however, this last computation is not a part of the backtracking procedure. Assuming that matrix-vector products comprise the dominant cost of each iteration, the total cost of an iteration without backtracking equals two matrix-vector products, while the cost of an iteration with backtracking contains an additional matrix-vector product per backtracking step. For instance, if $\mu_k^0 = \mu_{k-1}/\beta$, then, by Lemma 3.5, the average cost of an iteration of Algorithm 2 is $3/2$ that of Algorithm 1. Such a cost increase is beneficial if the number of iterations of Algorithm 2 is proportionately smaller.

In the examples we consider in our computational experiments in Sect. 5, increasing μ_k at each iteration appears to be wasteful. Hence, we choose to allow for the increase

in the value of the prox parameter every l iteration, where l is chosen heuristically. In particular, we try several different values of l during the first 100 iterations and then settle with the value of l , which gives the least number of failed backtracking steps. The key assumption in using such a heuristic is that the behavior of the algorithm at later iterations is similar to that during earlier iterations. While the behavior of the algorithm is problem-dependent, in our experiments this heuristic produced good results. We believe that this is due to the fact that the local Lipschitz constants do not vary dramatically for the problems in our experiments. If significant changes occur in the Lipschitz constants, we believe that any backtracking heuristic will produce significant improvements over the pure FISTA algorithm because it will allow the prox parameter to increase sooner or later.

4 Fast Alternating Linearization Methods

In this section, we propose a backtracking version of the fast alternating linearization method (FALM) with step skipping [7], which is given below as Algorithm 3. This algorithm can be viewed as an extension of FISTA, where the linearization is applied to each of the two parts of $F(x)$ in turn. This similarity to FISTA allows us to extend our backtracking approach and convergence rates results to FALM.

The main difference between FISTA and FALM is that FISTA approximates only the first part of the composite objective function $F(x)$, while FALM applies alternating approximations to each of the two parts of $F(x)$. Hence, we consider the following two approximations of $F(x)$ and their minimizers:

Definition 4.1

$$Q_\mu^f(u, v) := f(v) + \langle u - v, \nabla f(v) \rangle + \frac{1}{2\mu} \|u - v\|^2 + g(u);$$

$$p_\mu^f(v) := \arg \min_u Q_\mu^f(u, v);$$

$$Q_\mu^g(u, v) := f(v) + \langle v - u, \lambda \rangle + \frac{1}{2\mu} \|u - v\|^2 + g(u),$$

where λ is some element of the subdifferential $\partial g(u)$;

$$p_\mu^g(u) := \arg \min_v Q_\mu^g(u, v).$$

Algorithm 3 (FALM-S) given below was proposed in [7] for cases where computing $p_\mu^f(v)$ and $p_\mu^g(u)$, defined earlier, is relatively easy, i.e., comparable to a gradient computation of g and f , respectively. We call the k th iteration of this algorithm a *skipping step* if $x^k = z^k$ and a *regular step* if $x^k \neq z^k$. Skipping steps are designed to accommodate those cases where the condition $F(p_\mu^g(u)) \leq Q_\mu^g(u, p_\mu^g(u))$ fails. Since g is not a smooth function, it may not be possible to satisfy the preceding condition by choosing a sufficiently small value of μ . Hence, when the minimization of $Q_\mu^g(v, u)$ in v does not produce the desired function reduction, the corresponding step is simply skipped.

Algorithm 3

FALM with Skipping Steps (FALM-S)

0. Choose $x^0 = y^0 = y^{-1} = z^1$ and $\lambda^1 \in -\partial g(z^1)$, set $t_1 = 1$ and $t_0 = 0$;

1. for $k = 1, 2, \dots$ do

(1) $x^k := p_{\mu}^g(z^k)$

(2) If $F(x^k) > Q_{\mu}^g(z^k, x^k)$ then

If x -step was not skipped at iteration $k - 1$, $\theta_{k-1} := 2$

else $\theta_{k-1} := 1$

$(t_k, z^k) := FistaStep(y^{k-1}, y^{k-2}, t_{k-1}, \theta_{k-1})$

$x^k := z^k$

(3) $y^k := p_{\mu}^f(x^k)$

If $x^k = z^k$ then $\theta_k := 1/2$

else $\theta_k := 1$

$(t_{k+1}, z^{k+1}) := FistaStep(y^k, y^{k-1}, t_k, \theta_k)$

(4) Choose $\lambda^{k+1} \in -\partial g(z^{k+1})$

For the efficient computation of $\lambda^{k+1} \in -\partial g(z^{k+1})$ we refer the reader to [7].

As one can observe, Algorithm 3 does not include a mechanism for selecting μ at each iteration – the prox parameter is assumed to be fixed. In that case, the θ_k parameter in *FistaStep* computation equals $(\mu + \mu)/(\mu + \mu) = 1$ when on both iterations, k and $k + 1$, there are no skipped steps. Alternatively, $\theta_k = \mu/(\mu + \mu) = 1/2$ when a step is skipped on iteration k but not on iteration $k + 1$, while $\theta_k = (\mu + \mu)/\mu = 2$ when a step is skipped on iteration $k + 1$ but not on iteration k . The convergence rate results in [7] can be easily extended to the case where μ_k is chosen via backtracking, as is done in FISTA, as long as $\mu_k \leq \mu_{k-1}$, with the same values for θ_k . Here we propose a backtracking variant of FALM-S, which is given below as Algorithm 4. We note that we now have two linearization and minimization steps per iteration; hence, we allow the parameter μ to be set separately from μ_k^x and μ_k^y , respectively, for each of the minimization steps. The average $\mu_k = (\mu_k^x + \mu_k^y)/2$ is then used to compute θ_k and, hence, to update the parameter t_{k+1} . The new algorithm is a generalization of Algorithm 3 and includes skipping steps as a special case when *skip* = true and $\mu_k^x = 0$.

Let us first observe that the algorithm generates a sequence $\{\mu_k, t_k\}$ that satisfies a condition analogous to condition (3.11), which we required for FISTA-BKTR. Indeed, it is evident from the computation of θ_k that $\theta_k = \mu_k/\mu_{k+1}$, with $\mu_k = \frac{\mu_k^x + \mu_k^y}{2}$, and hence $\mu_k t_k^2 = \mu_{k+1} t_{k+1} (t_{k+1} - 1)$ for all k .

To obtain a bound on the number of iterations required by Algorithm 4 to reach an ϵ -optimal solution, we need the following lemma.

Lemma 4.1 *The sequence $\{x^k, y^k\}$ generated by Algorithm 4 satisfies*

$$2(\mu_k t_k^2 v_k - \bar{\mu}_{k+1} t_{k+1}^2 v_{k+1}) \geq \|u^{k+1}\|^2 - \|u^k\|^2, \tag{4.1}$$

where $u^k := t_k y^k - (t_k - 1) y^{k-1} - x^*$, $v_k := 2F(y^k) - 2F(x^*)$ and $\mu_k = (\mu_k^x + \mu_k^y)/2$.

Algorithm 4

FALM-BKTR
 0. Choose $x^0 = y^0 = y^{-1} = z^1, \mu_0 = \bar{\mu}_1^x = \bar{\mu}_1^y > 0, 0 < \beta < 1$, set $t_1 = 1$ and $t_0 = 0$;
 1. for $k = 1, 2, \dots$ do
 (1) Set $\mu_k^x := \bar{\mu}_k^x, \mu_k^y := \bar{\mu}_k^y, skip := true$
 (2) Compute $p_{\mu_k^x}^g(z^k)$
 If $F(p_{\mu_k^x}^g(z^k)) \leq Q_{\mu_k^x}^g(z^k, p_{\mu_k^x}^g(z^k))$ then $x^k := p_{\mu_k^x}^g(z^k), skip := false$
 else $\mu_k^x = 0, \theta_{k-1} := 2\mu_{k-1}/\mu_k^y, (t_k, z^k) := FistaStep(y^{k-1}, y^{k-2}, t_{k-1}, \theta_{k-1}), x^k := z^k$
 (3) Compute $p_{\mu_k^y}^f(x^k)$
 If $F(p_{\mu_k^y}^f(x^k)) \leq Q_{\mu_k^y}^f(p_{\mu_k^y}^f(x^k), x^k)$, then $y^k := p_{\mu_k^y}^f(x^k)$
 else $\mu_k^y := \beta\mu_k^y, \mu_k := \frac{\mu_k^x + \mu_k^y}{2}, \theta_k := \mu_{k-1}/\mu_k, (t_k, z^k) := FistaStep(y^{k-1}, y^{k-2}, t_{k-1}, \theta_{k-1})$
 If $skip = false$, return to (2)
 else $x_k := z_k$, return to (3)
 (4) Choose $\bar{\mu}_{k+1}^y, \bar{\mu}_{k+1}^x$
 $\theta_k := \frac{2\mu_k}{\bar{\mu}_{k+1}^y + \bar{\mu}_{k+1}^x}$,
 $(t_{k+1}, z^{k+1}) := FistaStep(y^k, y^{k-1}, t_k, \theta_k)$

Proof We will prove that the following inequality holds:

$$2(\mu_k t_k^2 v_k - \mu_{k+1} t_{k+1}^2 v_{k+1}) \geq t_{k+1}(t_{k+1} - 1) \left(\|y^{k+1} - y^k\|^2 - \|z^{k+1} - y^k\|^2 \right) + t_{k+1} \left(\|y^{k+1} - x^*\|^2 - \|z^{k+1} - x^*\|^2 \right).$$

The proof of (4.1), and hence the lemma, then follows from the fact that the right-hand side of inequality (4.2) equals

$$\|t_{k+1}y^{k+1} - (t_{k+1} - 1)y^k - x^*\|^2 - \|t_{k+1}z^{k+1} - (t_{k+1} - 1)y^k - x^*\|^2 = \|u^{k+1}\|^2 - \|u^k\|^2,$$

where we have used the fact that $t_{k+1}z^{k+1} := t_{k+1}y^k + t_k(y^k - y^{k-1}) - (y^k - y^{k-1})$.

First applying Lemma 2.1 to the function $Q_{\mu}^f(\cdot, \cdot)$, with $\mu = \mu_{k+1}^y, x = y^k$, and $y = x^{k+1}$, we have $p_{\mu_{k+1}^y}^f(y) = y^{k+1}$ and

$$2\mu_{k+1}^y(F(y^k) - F(y^{k+1})) \geq \|y^{k+1} - y^k\|^2 - \|x^{k+1} - y^k\|^2. \tag{4.2}$$

In the case of a regular iteration, that is, when $\mu_{k+1}^x > 0$, applying Lemma 2.1 to the function $Q_{\mu}^g(\cdot, \cdot)$, with $\mu = \mu_{k+1}^x, x = y^k$, and $y = z^{k+1}$, we obtain $p_{\mu_{k+1}^x}^g(y) = x^{k+1}$ and

$$2\mu_{k+1}^x(F(y^k) - F(x^{k+1})) \geq \|x^{k+1} - y^k\|^2 - \|z^{k+1} - y^k\|^2. \tag{4.3}$$

In the case where $\mu_{k+1}^x = 0$, since $x^{k+1} := z^{k+1}$, (4.3) still holds.

Summing (4.2) and (4.3), and using the fact that $F(y^{k+1}) \leq F(x^{k+1})$, we obtain

$$2\mu_{k+1}(v_k - v_{k+1}) = 2\mu_{k+1}(2F(y^k) - 2F(y^{k+1})) \geq \|y^{k+1} - y^k\|^2 - \|z^{k+1} - y^k\|^2. \tag{4.4}$$

Again, applying Lemma 2.1 in the case where $\mu_{k+1}^x > 0$ to the function $Q_{\mu_{k+1}^x}^g(\cdot, \cdot)$ with $x = x^*$ and $y = z^{k+1}$, we obtain $p_{\mu_{k+1}^x}^g(y) = x^{k+1}$ and

$$2\mu_{k+1}^x(F(x^*) - F(x^{k+1})) \geq \|x^{k+1} - x^*\|^2 - \|z^{k+1} - x^*\|^2. \tag{4.5}$$

On the other hand, if $\mu_{k+1}^x = 0$, then $x^{k+1} := z^{k+1}$, and (4.5) still holds. Lemma 2.1 applied to the function $Q_{\mu_{k+1}^y}^f(\cdot, \cdot)$, with $x = x^*$, $y = x^{k+1}$, and $p_{\mu_{k+1}^y}^f(x^{k+1}) = y^{k+1}$, gives

$$2\mu_{k+1}^y(F(x^*) - F(y^{k+1})) \geq \|y^{k+1} - x^*\|^2 - \|x^{k+1} - x^*\|^2. \tag{4.6}$$

Summing (4.5) and (4.6), and again using the fact that $F(y^{k+1}) \leq F(x^{k+1})$ and $\mu_k = (\mu_k^x + \mu_k^y)/2$, we obtain

$$-2\mu_{k+1}v_{k+1} = 2\mu_{k+1}(2F(x^*) - 2F(y^{k+1})) \geq \|y^{k+1} - x^*\|^2 - \|z^{k+1} - x^*\|^2. \tag{4.7}$$

If we multiply (4.4) by $t_{k+1}(t_{k+1} - 1)$, and (4.7) by t_{k+1}^2 , and take the sum of the two resulting inequalities, we obtain (4.2) by using the fact that $\mu_k t_k^2 \geq \mu_{k+1} t_{k+1}(t_{k+1} - 1)$. □

Since Algorithm 4 maintains $\theta_k = \mu_k/\mu_{k+1}$ on each iteration, then from Lemma 3.4 we know that for the sequence $\{\mu_k, t_k\}$ generated by Algorithm 4,

$$\mu_k t_k^2 \geq \left(\sum_{i=1}^k \sqrt{\mu_i}/2\right)^2. \tag{4.8}$$

Now we are ready to give the complexity of Algorithm 4. Applying a proof equivalent to that of Theorem 3.1 we obtain

$$F(x^k) - F(x^*) \leq \frac{\|x^0 - x^*\|^2}{4\mu_k t_k^2}. \tag{4.9}$$

In the worst case, we have $\mu_0^x = 0$ and $\mu_k \geq \frac{1}{2L(f)}$ and the algorithm reduces to FISTA. Hence, just as for FISTA, from (4.8) we have the following theorem.

Theorem 4.1 *At each iteration k of Algorithm 2 we have*

$$F(x^k) - F(x^*) = v_k \leq \frac{2L(f)\|x^0 - x^*\|^2}{\beta k^2}. \tag{4.10}$$

More generally, similarly to the case of FISTA-BKTR, we have the following complexity result.

Theorem 4.2 *Let $\sqrt{\mu(k)} = (\sum_{i=1}^k \sqrt{\mu_i})/k$ be the average of the square roots of the average prox parameters μ_i used during the first k iterations of Algorithm 4. Then*

$$F(x^k) - F(x^*) = v_k \leq \frac{2\|x^0 - x^*\|^2}{\mu(k)k^2}. \tag{4.11}$$

Note that the prox parameters μ_i now depend on the local composite Lipschitz constants of both f and g , i.e, $L(f, g, y^k)$ and $L(g, f, z^k)$. For simplicity we write the statement of the theorem in terms of $\mu(k)$ instead of $L(f, g, y^k)$ and $L(g, f, z^k)$ since the latter may not be well defined in some cases and the skipped steps need to be accounted for.

4.1 A Practical Backtracking FALM Algorithm for Compressed Sensing and Lasso Problems

We now discuss the additional cost of each backtracking step of Algorithm 4 compared to that of Algorithm 3 and a general efficient implementation of the algorithm targeted at the problems of the form (3.17).

We again assume here that $g(x)$ is a simple function, hence computing $p_{\mu^y}^g(y)$ is a relatively cheap operation. Computing $p_{\mu^y}^f(x)$, however, involves solving a system of linear equations with the matrix $A^\top A + \frac{1}{2\mu^y}I$. In some compressed sensing problems, A has a special structure, such that this system can be solved in $O(n \log n)$ operations – the same work as is required to multiply A or A^\top by a vector and, hence, to compute the gradient $\nabla f(x) = A^\top(Ax - b)$ [7]. In cases where such a special structure is not present, work that involves factorization of $A^\top A + \frac{1}{2\mu^y}I$ may be the dominant cost of the iteration because it generally requires $O(m^3)$ operations.

If μ^y is fixed beforehand, then the matrix $A^\top A + \frac{1}{2\mu^y}I$ can be factorized once, at the initialization stage of the algorithm, and hence the per-iteration cost only involves additional matrix-vector products. If μ^y is updated in an arbitrary way on each iteration, then the factorization must be repeated each time. Recall that ideally we want μ^y to have the largest possible value that satisfies the line search conditions in Step 2 of Algorithm 4 and that keeping μ^y constant may result in very slow progress. Hence, again, there exists a tradeoff between choosing a suitable value of the prox parameter and the per-iteration cost. For practical efficiency we strive to achieve a reasonable balance. Assume that we choose some value $\bar{\mu}_1^y$ at the beginning of the algorithm and we choose $\bar{\mu}_{k+1}^y = \beta^i \mu_k^y$ for some $i \in Z$ at each iteration k . Note that in this case, for all k , μ_k^y can only take values $\beta^j \mu_1^y$ for $j \in Z$. Let us consider $\beta = 0.5$. We also impose the following restriction of μ_k^y : if $\mu_k^y < \mu_1^y/1000$, then the expected improvement achieved by Step 2 is too small and the step is automatically skipped; if $\mu_k^y > 1000\mu_1^y$, the prox parameter is large enough and no additional increase of μ_k^y is necessary. Hence the only values allowed for μ_k^y throughout the algorithm are $\{2^{-10}\mu_1^y, 2^{-9}\mu_1^y, \dots, 0, 2\mu_1^y, 4\mu_1^y, \dots, 2^{10}\mu_1^y\}$, overall 21 possible values. As soon as

one of these values occurs in the algorithm, the corresponding matrix factorization can be computed and stored for future use.

If the skipping of Step 2 occurs for a few consecutive iterations, we may choose to automatically skip this step in the further iterations and thus avoid the additional cost of computing $p_{\mu_y}^f(x)$. In this case, the FALM-BKTR algorithm reduces FISTA-BKTR. We found it beneficial to attempt Step 2 from time to time, even after it has been skipped consistently on prior iterations.

The management of the μ_k^x parameter and the additional per-iteration cost of Step 3 can be executed similarly to what is described in Sect. 3.1 for the FISTA-BKTR implementation.

5 Computational Results

We now present numerical results for several sparse optimization problems of the standard compressed sensing or Lasso form:

$$\bar{x} := \arg \min_x \frac{1}{2} \|Ax - b\|_2^2 + \rho \|x\|_1, \quad (5.1)$$

with $f(x) := \frac{1}{2} \|Ax - b\|_2^2$ and $g(x) := \rho \|x\|_1$.

We compare the following algorithms:

- *FISTA*: the original FISTA, [1], as described in Algorithm 1 with $\theta_k = 1$.
- *FISTA-BKTR*: an efficient implementation of Algorithm 2, as discussed in Sect. 3.1.
- *FALM*: an implementation of Algorithm 3, as discussed in Sect. 4.
- *FALM-BKTR*: an efficient implementation of Algorithm 4, as discussed in Sect. 4.1.
- *SpaRSA*: a gradient-based algorithm, with the use of shrinking, described in [5].
- *Yall1*: a solver based on alternating-direction methods, described in [16].

We compared the performance of the algorithms, benchmarking them against FISTA. In particular, we ran FISTA for j iterations and recorded the best objective function value $FISTA(j)$ achieved by FISTA thus far. Then for all other algorithms we recorded the number of iterations it took to reach a function value smaller than $FISTA(j)$. We report the number of iterations as well as the number of matrix-vector multiplications. Throughout our tests, the maximum number of iterations is set to be 100,000, and the tolerance is set to be 10^{-3} , which means that the algorithm terminates when the objective function value is within 10^{-3} from the optimal (precomputed). We report the final objective function value when each algorithm terminates.

The main goal of our experiments is to demonstrate that our full backtracking strategy provides not only theoretical but practical advantage when applied to FISTA and FALM methods. The comparison to Yall1 and SpaRSA methods is only presented here to gauge the difficulty of the problems in our experiments and to demonstrate that the behavior of our methods is reasonable. Our implementations were written in MATLAB and run in MATLAB R2010b on a laptop with an Intel Core Duo 1.8 GHz CPU and 2 GB RAM. We used the default setting for both Yall1 and SpaRSA, which likely accounts for the bad performance of these algorithms on some of the problems.

5.1 Spear Examples

This set of instances is obtained from the Sparse Exact and Approximate Recovery Project and can be downloaded from either of the following links:

- <http://imo.rz.tu-bs.de/mo/spear/>
- <https://coral.ie.lehigh.edu/projects/SPAROPTLIB/wiki/SparseOptimizationLibrary>.

Spear10 (1024 × 512) Dynamic range is 3.02e+4. Sparsity is 18 (i.e., 18 nonzero elements in the true solution). This problem provides a relatively easy instance when $\rho = 1$, but the difficulty increases substantially as ρ decreases.

From Table 1, we see that the algorithms with backtracking (FISTA-BKTR and FALM-BKTR) were generally faster than their basic counterparts (FISTA, FALM) in terms of the number of matrix-vector multiplications. For example, it only takes 627 iterations and 1,343 matrix-vector multiplications for FISTA-BKTR to reduce the objective function value below $FISTA(1000) = 1.0035e + 5$. Comparing FALM-S1 and FALM-BKTR, we observe that the latter is faster for the same initial choice of μ_g ($\mu_g = 1$). The initial performance of FALM-S2 and, especially, FALM-S3 is good due to the larger starting values for μ_g ; however, this performance slows down compared to FALM-BKTR as iterations progress. This indicates that the full backtracking strategy can help accelerate the original algorithms at any stage.

In Fig. 1 we plot how μ changes during iterations taken by FISTA and FISTA-BKTR when solving (5.1) with $\rho = 1$ on Spear10. We see that FISTA-BKTR can achieve larger values of μ by allowing backtracking and thus performs large steps on some of the iterations, which corresponds to the smaller number of iterations required by FISTA-BKTR.

Table 2 shows the results on a Spear10 problem with $\rho = 0.01$. This problem provides a difficult instance where our backtracking method appears to provide a clear advantage. SpaRSA did not converge to the proximity of the solution, while Yall1 achieved an accuracy of only 10^{-1} , not 10^{-3} . Here we observe that FISTA-BKTR

Table 1 Comparison of algorithms for solving (5.1) with $\rho = 1$ on Spear10

solver	Iter	Mult	Iter	Mult	Iter	Mult	Final iter	Mult	Final Obj.
FISTA	100	206	500	1006	1000	2006	1065	2133	9.997e+4
FISTA_BKTR	69	170	283	619	627	1343	647	1404	9.997e+4
FALM-S1	30	94 (2)	117	355 (2)	376	1037 (11)	586	1457 (11)	9.997e+4
FALM-S2	10	34 (2)	39	121 (2)	206	565 (11)	273	699 (11)	9.997e+4
FALM-S3	4	16 (2)	17	57 (4)	355	812 (11)	464	1030 (11)	9.997e+4
FALM_BKTR	8	28 (8)	26	112 (10)	85	396 (21)	136	546 (21)	9.997e+4
SpaRSA	98	196	1487	2974	1689	3378	1729	3458	9.997e+4
YALL1	18	55	30	91	89	268	93	280	9.997e+4

$FISTA(100) = 5.3839e + 5$, $FISTA(500) = 1.2799e + 5$, $FISTA(1000) = 1.0035e + 5$. The starting μ for FISTA/FISTA-BKTR, μ_f for FALM/FALM-BKTR and μ_g for FALM-BKTR are all set at 1. For FALM (with skipping), we tried different values for μ_g . The starting μ_g for FALM-S1, FALM-S2, and FALM-S3 is 1, 10, and 100, respectively. Moreover, for FALM/FALM-BKTR the number in parentheses is the number of matrix factorization required overall

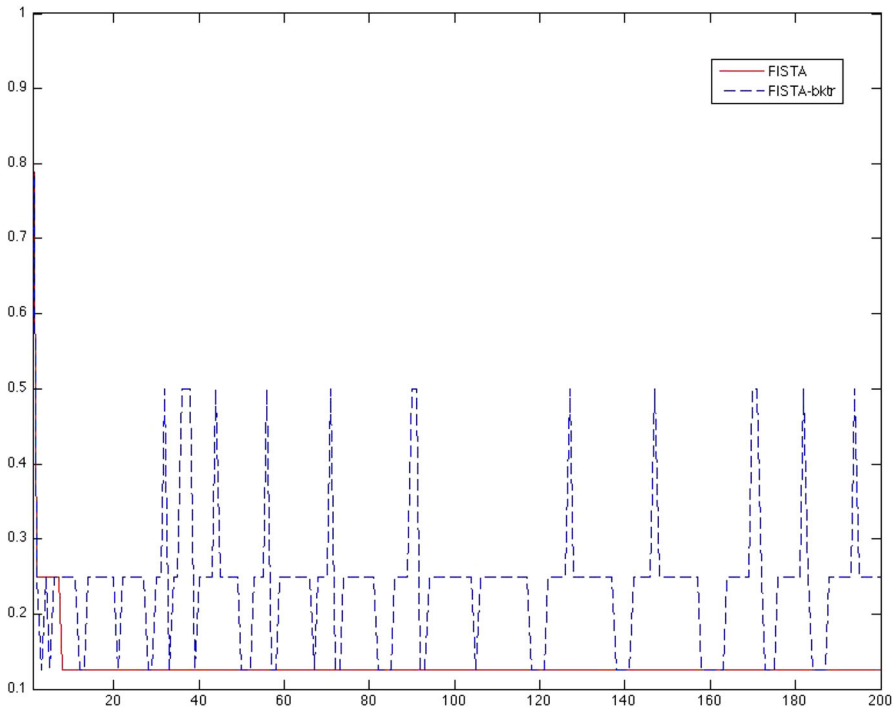


Fig. 1 Comparison on μ values while solving (5.1) with $\rho = 1$ on Spear10

Table 2 Comparison of algorithms for solving (5.1) with $\rho = 0.01$ on Spear10

Solver	Iter	Mult	Iter	Mult	Iter	Mult	Final iter	Mult	Final obj.
FISTA	100	206	500	1006	1000	2006	17846	35695	999.4
FISTA_BKTR	79	190	372	804	746	1607	12547	26527	999.4
FALM-S1	24	76 (2)	156	472 (2)	313	943 (2)	5298	15903 (11)	999.4
FALM-S2	7	25 (2)	53	163 (2)	108	328 (2)	1221	3674 (11)	999.4
FALM-S3	4	16 (2)	16	52 (2)	33	103 (2)	287	869 (11)	999.4
FALM_BKTR	7	25 (7)	12	40 (11)	18	64 (11)	252	1078 (21)	999.4
SpaRSA	30	60	687	1374	5024	10048	100000	200000	2151.1 (Failed)
YALL1	65	196	65	196	66	199	95	286	1015.3

The starting μ is set to be 0.01. FISTA(100) = 6.0980e + 3, FISTA(500) = 5.8943e + 3, FISTA(1000) = 5.4176e + 3

retains its advantage, while FALM-BKTR seems to slow down compared to FALM method when it gets closer to the solutions. The reasons for this behavior will be investigated in the future (Table 3).

Spear3 (1024 × 512) with $\rho = 0.1$. The dynamic range is 2.7535e+4, the sparsity is 6. We observe that the behavior of FALM-BKTR converges to that of FISTA-BKTR in later iterations due to persistent skipping of Step 2.

Table 3 Comparison of algorithms for solving (5.1) with $\rho = 0.1$ on Spear3

Solver	Iter	Mult	Iter	Mult	Iter	Mult	Final iter	Mult	Final obj.
FISTA	100	211	500	1011	1000	2011	28517	57045	7.33e+3
FISTA_BKTR	236	535	241	547	324	722	3149	6767	7.33e+3
FALM-S1	8	28 (2)	166	378 (11)	559	1164 (11)	27287	54620 (11)	7.33e+3
FALM-S2	6	34 (2)	112	266 (11)	547	1136 (11)	28063	56168 (11)	7.33e+3
FALM-S3	105	245 (11)	506	1047 (11)	826	1687 (11)	22430	44895 (11)	7.33e+3
FALM_BKTR	6	34 (6)	145	485 (17)	276	827 (17)	3819	10271 (17)	7.33e+3
SpaRSA	5	10	264	528	1215	2430	100000	200000	1.02e+4 (Failed)
YALL1	418	1255	418	1255	418	1255	809	2428	7.33e+3

The starting μ is set to be 1. FISTA(100) = 1.1825e + 4, FISTA(500) = 1.1793e + 4, FISTA(1000) = 1.1784e + 4

Table 4 Comparison of algorithms for solving (5.1) with $\rho = 0.0001$ on Bdata1

Solver	Iter	Mult	Iter	Mult	Iter	Mult	Final iter	Mult	Final obj.
FISTA	10	22	50	102	100	202	200	402	1.63e-4
FISTA_BKTR	8	18	40	99	81	193	160	368	1.63e-4
FALM-S1	8	22 (2)	37	112 (5)	114	282 (11)	190	434 (11)	1.63e-4
FALM-S2	4	10 (2)	17	51 (5)	95	224 (11)	172	378 (11)	1.63e-4
FALM-S3	2	4 (1)	12	38 (8)	93	212 (11)	169	364 (11)	1.63e-4
FALM_BKTR	5	13 (5)	14	60 (9)	80	331 (19)	142	479 (19)	1.63e-4
SpaRSA	8	16	55	110	166	332	230	460	1.63e-4
YALL1	16	49	27	82	36	109	81	244	1.63e-4

The starting μ is set to be 1. FISTA(10) = 0.0015, FISTA(50) = 4.6868e - 4, FISTA(100) = 1.8933e - 4, FISTA(200) = 1.6275e - 4. The tolerance ϵ_b is set to be 0.001

5.2 Bdata Problems

A Bdata test set was originally created by A. Nemirovski with the aim of imitating examples with worst-case complexity for the first-order methods. This problem, however, provides a relatively easy instance, probably due to the presence of the ℓ_1 term in the objective. Here we present the results for Bdata1 (1036 × 1036), with $\rho = 0.0001$; the dynamic range is 5.9915 and the sparsity is 16 (Table 4).

5.3 Sparco Problems

This category of instances is obtained from [2]. Due to the fact that the Sparco instances use function handles for matrix computation, which our FALM implementation is not equipped to do, we do not include FALM in this comparison. We present the results for Sparco3 (2048 × 1024), with $\rho = 0.01$, dynamic range 2, and sparsity 2. We observe that, for this relatively easy instance, FISTA-BKTR has a minor advantage over FISTA in terms of the number of matrix-vector multiplications. For this example, FISTA

Table 5 Comparison of algorithms for solving (5.1) with $\rho = 0.01$ on Sparco3

Solver	Iter	Mult	Iter	Mult	Iter	Mult	Final iter	Mult	Final obj.
FISTA	10	24	50	104	100	204	207	418	2.22278
FISTA_BKTR	6	18	38	97	78	187	173	387	2.22278
SpaRSA	7	14	11	22	72	144	99	198	2.22278
YALL1	10	20	10	20	19	38	262	534	2.22278

The starting μ is set to be 1. $FISTA(10) = 13.07180$, $FISTA(50) = 8.187212$, $FISTA(100) = 2.710062$. The tolerance ϵ_b is set to be 0.001

outperforms the alternating-direction-based method Yall1, while SpaRSA seems to be the winning method for this instance (Table 5).

5.4 Smoothed ℓ_2 Norm Minimization

As an alternative to problem (5.1) one may wish to solve the following problem with an exact ℓ_2 penalty term:

$$\bar{x} := \arg \min_x \|Ax - b\|_2 + \rho \|x\|_1. \tag{5.2}$$

To apply the FISTA and FALM family of methods, we can smooth the ℓ_2 term with the well-known Huber penalty function and obtain the following minimization problem:

$$\bar{x} := \arg \min_x H_\nu(\|Ax - b\|_2) + \rho \|x\|_1, \tag{5.3}$$

where

$$H_\nu(y) = \begin{cases} \frac{y^2}{2\nu}, & 0 \leq |y| \leq \nu \\ |y| - \frac{\nu}{2}, & |y| \geq \nu, \end{cases}$$

for $\nu > 0$. If $\nu < \epsilon$, then the solution of (5.3) is an ϵ -solution to (5.2). We define $f(x) := H_\nu(\|Ax - b\|_2)$ and $g(x) := \rho \|x\|_1$. It is well known that the global Lipschitz constant of $\nabla f(x)$ is $O(1/\nu)$. Recall Example 2.2, where we analyze the local composite Lipschitz constant for the case where $g(x) \equiv 0$ and show that away from the optimal solution the local composite Lipschitz constant of $\nabla f(x)$ is much smaller than $O(1/\nu)$. The analysis of the case where $g(x) := \rho \|x\|_1$ is much more complex, but the essential expectation remains for our first-order schemes: the prox parameter μ will be relatively large away from the solutions, while it will decrease as the algorithm converges. In fact, FISTA and FALM in their original form will observe the same behavior of the prox parameter because they allow the prox parameter to decrease but not increase. Hence, we do not expect a significant saving using backtracking in this setting; however, we present experiments for illustration and to confirm our expectation of the prox parameter behavior.

Table 6 Comparison of algorithms for solving (5.3) with $\rho = 0.1$ on Spear10

Solver	Iter	Mult	Iter	Mult	Iter	Mult	Final iter	Mult	Final obj.
FISTA	100	200	500	1000	1000	2000	2242	4486	9.99e+3
FISTA_BKTR	18	36	27	54	29	62	48	144	9.99e+3
SpaRSA	1558	3116	36014	72028	57198	114396	60313	120626	9.99e+3

The starting μ is set to be 1. FISTA(100) = 4.4552e + 4, FISTA(500) = 2.4248e + 4, FISTA(1000) = 1.1554e + 4

Table 7 Comparison of the algorithms for solving (5.3) with $\rho = 0.1$ on Spear10

Solver	Iter	Mult	Iter	Mult	Iter	Mult	Final iter	Mult	Final obj.
FISTA	10	20	20	40	40	91	42	97	9.99e+3
FISTA_BKTR	7	14	11	28	25	92	28	102	9.99e+3
SpaRSA	18129	36258	57282	114564	59130	118260	59131	118262	9.99e+3

The starting μ is set to be 1000. FISTA(10) = 3.4189e + 4, FISTA(20) = 1.0866e + 4, FISTA(40) = 9.9944e + 3

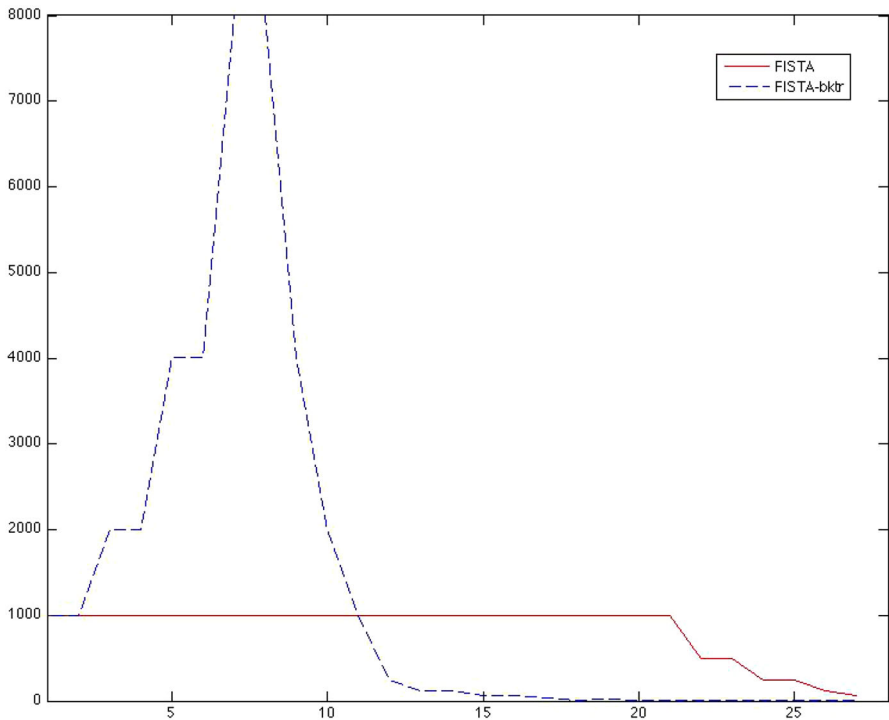


Fig. 2 Comparison on μ values while solving (5.3) with $\rho = 0.1$ on Spear10

In Tables 6 and 7 we present a comparison of the first-order methods on Spear10 data and formulation (5.3). We observe that FISTA-BKTR is much faster than FISTA and SpaRSA in the case where the initial prox parameter value is not very large. As

Table 8 Comparison of algorithms for solving (5.3) with $\rho = 0.01$ on Bdata1

Solver	Iter	Mult	Iter	Mult	Iter	Mult	Final iter	Mult	Final obj.
FISTA	100	210	500	1010	1000	2010	2105	5249	0.0164
FISTA_BKTR	120	342	539	1290	539	1290	1624	3822	0.0164

The starting μ is set to be 1. FISTA(100) = 0.0196, FISTA(500) = 0.0164, FISTA(1000) = 0.0164

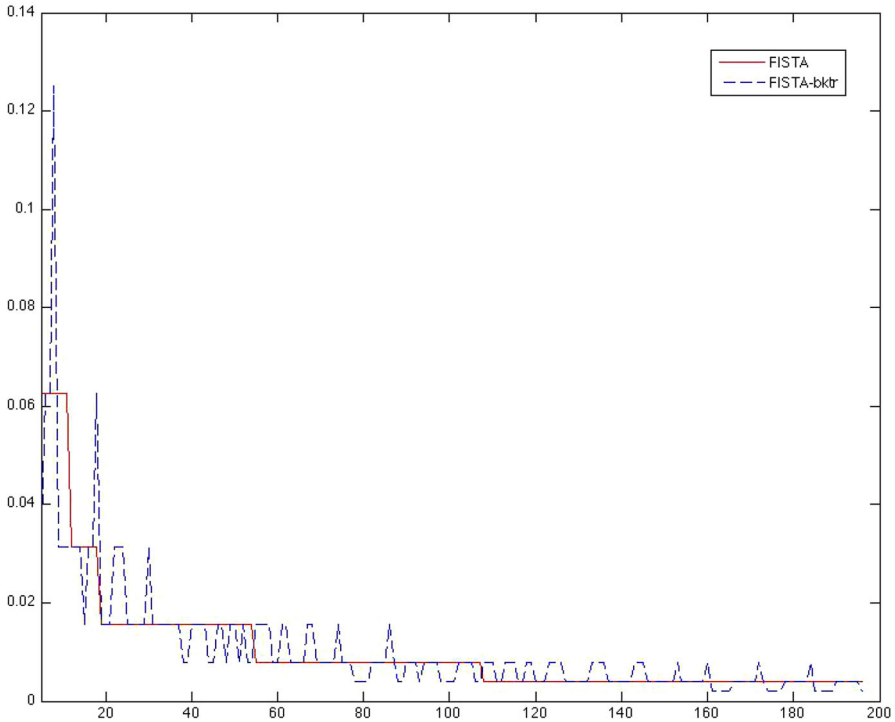


Fig. 3 Comparison on μ values while solving (5.3) with $\rho = 0.001$ on Bdata

compared with FISTA, FISTA-BKTR allows for a huge increase in μ . If the initial μ is set to 1, then FISTA’s performance is very slow, as is shown in Table 6. But if $\mu = 1000$, then FISTA’s performance improves, to the level of FISTA-BKTR, as seen in 7. This is well explained by showing graphically the change of μ in Fig. 2.

In Table 8 and Fig. 3 we show the outcome of the experiments on the Bdata1 test set. In this case, FISTA and FISTA-BKTR perform as expected, with FISTA-BKTR retaining a small advantage. In fact, after 500 iterations, the μ value for both algorithms becomes small, which indicates a large Lipschitz constant for solving (5.3) on Bdata1.

5.5 ℓ_2 Regularized Logistic Regression

Finally, we illustrate the behavior of FISTA vs. FISTA-BKTR on an example of ℓ_2 regularized logistic regression applied to an optical character recognition data set,

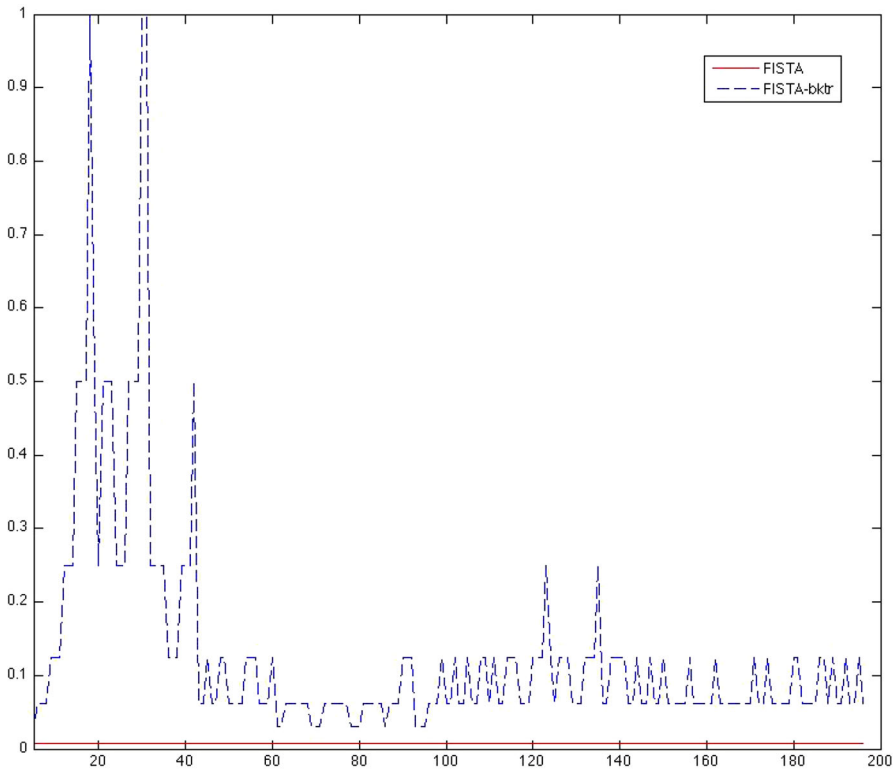


Fig. 4 Comparison on μ values while solving logistic regression problem on *Optdigits* data set

Optdigits, from the UCI repository [6]. We present this example here purely for illustration purposes to show that on settings other than compressed sensing the backtracking strategy can produce significant improvements. As we discussed, to obtain optimal performance from backtracking, a careful implementation is needed that tries to take into account the problem structure. Such an implementation for logistic regression and other problems will be the subject of future research. Here we present the results of the basic approach where μ is increased by a factor of 2 on each iteration. FISTA required 5705 iterations and 11417 matrix-vector multiplications to obtain a solution with a gradient norm less than 10^{-2} , while FISTA-BKTR required 1315 iterations and 3512 matrix-vector multiplications. In Fig. 4 we plot the behavior of the μ parameter for both algorithms, which clearly shows that FISTA-BKTR benefits from much larger steps. The analysis of local Lipschitz constants for logistic regression will be the subject of future research.

6 Conclusion

We present a generalized version of accelerated first-order schemes capable of estimating the prox parameter via backtracking, thereby allowing for the value of this

parameter to increase as well as decrease. We show that the value of the parameter depends on the so-called local composite Lipschitz constant of the gradient, rather than the global Lipschitz constant. We show that the complexity results can then be derived in terms of the average of the local composite Lipschitz constants along the iterates of the algorithm. We show via some examples that the local constants can be much smaller than the global ones; hence, one could potentially obtain better convergence bounds. To produce such bounds, one would need to combine the analysis in this paper with the analysis of the iterates of a first-order algorithm, which will be the subject of a future study. Our computational experiments and the discussion of a practical implementation show that in practice our proposed backtracking schemes offer improvements in terms of accelerated first-order algorithms.

Acknowledgments The authors are grateful to A. d’Aspremont for helpful discussions on the average case behavior and to S. Ma for help with preparing the manuscript. We are also grateful to the anonymous referees for their very helpful comments. The research of K. Scheinberg on this work was supported in part by National Science Foundation (NSF) Grant DMS 10-16571, Air Force Office of Scientific Research (AFOSR) Grant FA9550-11-1-0239, and Defense Advanced Research Projects Agency (DARPA) Grant FA 9550-12-1-0406 negotiated by AFOSR. The research of D. Goldfarb on this work was supported in part by NSF Grant DMS 10-16571, Office of Naval Research (ONR) Grant N00014-08-1-1118, and Department of Energy (DOE) Grant DE-FG02-08ER25856.

References

1. A. Beck and M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, *SIAM J. Imaging Sciences*, 2 (2009), pp. 183–202.
2. E. van den Berg, M. P. Friedlander, G. Hennenfent, F. Herrmann, R. Saab and Ö. Yılmaz, Sparco: a testing framework for sparse reconstruction, Tech. Report TR-2007-20, Dept. Computer Science, University of British Columbia, Vancouver, Oct 2007.
3. S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, *Foundations and Trends in Machine Learning*, 3 (2011), pp. 1–122.
4. E. Candès, Compressive sampling, *Proc. International Congress of Mathematics*, 3 (2006), pp. 1433–1452.
5. 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. Sel. Top. Signal Process*, 1 (2007).
6. A. Frank and A. Asuncion, UCI machine learning repository, 2010.
7. D. Goldfarb, S. Ma, and K. Scheinberg, Fast alternating linearization methods for minimizing the sum of two convex functions, *Math. Prog.* (2013) 141, 349–382.
8. A. Nemirovski and D. Yudin, Informational complexity and efficient methods for solution of convex extremal problems, Wiley, New York, 1983.
9. Y. E. Nesterov, Gradient methods for minimizing composite objective function. <http://rwww.optimization-online.org>.
10. Y. E. Nesterov, A method for unconstrained convex minimization problem with the rate of convergence $\mathcal{O}(1/k^2)$, *Dokl. Akad. Nauk SSSR*, 269 (1983), pp. 543–547.
11. Y. E. Nesterov, *Introductory lectures on convex optimization: a basic course*, vol. 87, Springer, 2004, pp. xviii+236.
12. Y. E. Nesterov, Smooth minimization for non-smooth functions, *Math. Program. Ser. A*, 103 (2005), pp. 127–152.
13. R. Tibshirani, Regression shrinkage and selection via the lasso, *Journal Royal Statistical Society B*, 58 (1996), pp. 267–288.
14. P. Tseng, On accelerated proximal gradient methods for convex–concave optimization. submitted to, *SIAM J. Optim.* (2008).

15. M. Yuan and Y. Lin, Model selection and estimation in regression with grouped variables, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68 (2006), pp. 49–67.
16. Y. Zhang, Yall1: Your algorithms for ℓ_1 . <http://www.caam.rice.edu/optimization/L1/YALL1/> (2009).