# FAST FIRST-ORDER METHODS FOR COMPOSITE CONVEX OPTIMIZATION WITH BACKTRACKING

KATYA SCHEINBERG*, DONALD GOLDFARB†, AND XI BAI‡

**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 [1] and FALM algorithms [7] 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 has to 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 allows 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 the original FISTA iteration. Our computational results show the benefit of the new algorithm.

**Key words.** Convex programming, first-order method, alternating linearization method, iterative shrinkage, optimal gradient method, line search.

**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:

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

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

†Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, 18015, U.S.A. Email: katyas@lehigh.edu. Research supported in part by NSF Grant DMS 10-16571, AFOSR Grant FA9550-11-1-0239 and DARPA grant FA 9550-12-1-0406 negotiated by AFOSR.

*Department of Industrial Engineering and Operations Research, Columbia University, New York, 10027, U.S.A. Email: goldfarb@columbia.edu. Research supported in part by NSF Grant DMS 10-06571, ONR Grant N00014-08-1-1118 and DOE Grant DE-FG02-08ER58562.

‡Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, 18015, U.S.A. Email: xib210@lehigh.edu

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 it 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 in all subsequent, 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 (ADM or ADMM) [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 that those of the accelerated gradient methods such as FISTA. A recently proposed Fast Alternating Linearization Method (FALM) [7] is an accelerated version of an alternating direction method 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 special structure which allows the proximity operators $p_\mu^f(y)$ and $p_\mu^g(y)$ (defined below) to be easily computable. Here we introduce the backtracking 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 Section 3 we introduce and analyze the FISTA algorithm with full line search. In Section 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 Section 4 we extend our line search approach and our complexity analysis to the FALM method. We conclude in Section 5 with some computational results.

**2. Preliminary results.** The following assumption is made throughout the paper:

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

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

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

We define the following notation.

DEFINITION 2.2.

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

$$p_\mu(v) := \arg\min_u Q_\mu(u,v).$$

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

LEMMA 2.3. *For any $y, x \in \mathbf{R}^n$ and $\mu > 0$, if*

$$F(p_\mu(y)) \le Q_\mu(p_\mu(y), y), \tag{2.1}$$

*where $p_\mu(y)$ is given in Definition 2.2, then*

$$2\mu(F(x) - F(p_\mu(y))) \ge \|p_\mu(y) - y\|^2 + 2\langle y - x, p_\mu(y) - y\rangle = \|p_\mu(y) - x\|^2 - \|y - x\|^2. \tag{2.2}$$

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

The following fact is well-known and will be used in our theoretical analysis: if $\mu \le 1/L(f)$, then (2.1) holds for any $y$, and hence (2.2) holds for any $x$ and $y$.

**2.1. Local composite Lipschitz constants.** The lower bound on the value of the prox parameter $\mu$ 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 $\mu$ at each iteration is determined via a back-tracking line search so that condition (2.1) is satisfied. A lower bound on $\mu$ can be derived from the facts that $\mu$ is reduced by a constant factor at each line search step and as soon as $\mu \le 1/L(f)$, condition (2.1) is satisfied and the line search terminates. On the other hand condition (2.1) may be satisfied with a larger value of $\mu$. As we will see from our results, increasing the lower bound on $\mu$ 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 $u, v \in \mathbb{R}^n$ and let $[u,v]$ denote the set of points on a segment between $u$ and $v$, in other words, $[u,v] = \{x : x = \lambda u + (1-\lambda)v, 0 \le \lambda \le 1\}$. Let $L_{[u,v]}(f)$ be the Lipschitz constant of $\nabla f(x)$ restricted to $[u,v]$; i.e.,

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

From simple calculus it follows that

$$f(u) \le f(v) + \nabla f(v)^\top (u - v) + \frac{L_{[u,v]}(f)}{2}\|u - v\|^2. \tag{2.3}$$

Note that the roles of $u$ and $v$ are interchangeable.

DEFINITION 2.4. *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 $\mu$ 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 2.4 it follows that for any $\mu$ and $y$, such that $\mu \leq 1/L(f,g,y)$ (2.1) holds (by (2.3)), and hence (2.2) 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 $\mu$ than $1/L(f)$.

EXAMPLE 2.5. *Consider a compressed sensing or Lasso setting:*

$$(P) \qquad \min\{F(x) \equiv \|Ax - b\|^2 + \rho\|x\|_1 : x \in \mathbb{R}^n\}. \tag{2.4}$$

*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 $L(f,g,\bar{y}) = \|A_1A_1^\top\|_2$, 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 which is still lower than $L(f)$. For instance, assume that for a given $f$, $g$ and $y$

$$\|\nabla f(x) - \nabla f(y)\| \leq L(f,g,y)\|x - y\|, \forall x :\ \|x - y\| \leq \|p_{1/L(f,g,y)}(y) - y\|, \tag{2.5}$$

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 $\mu < 1/L(f,g,y)$ (2.1) 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 2.6. *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 (2.2). 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,

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

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 2.7. *To illustrate the case when 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*

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

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

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

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

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

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

*and therefore,*

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

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

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

*This implies that (2.5) 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*

$$L(f,g,y) \le \frac{\|A^\top A\|_2}{2\|Ay-b\|}, \ \forall y : \|Ay-b\| \ge 2\nu, \tag{2.6}$$

The above 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, 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 3.1) 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 3.1.

*FISTA*
*0. Set* $t_1 = 1, 0 < \beta < 1$ *and* $y^1 = x^0, \mu_0 > 0$*;*
*1. for* $k = 1, 2, \ldots$ *do*
(1) *Choose* $0 < \bar{\mu}_k < \mu_0, \theta > 0$
(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) $x^k := p_{\mu^k}(y^k)$
$t_{k+1} := \left(1 + \sqrt{1 + 4\theta_k t_k^2}\right)/2$
$y^{k+1} := x^k + \frac{t_k - 1}{t_{k+1}}\left[x^k - x^{k-1}\right]$

According to Step 2 of this algorithm, if $\mu_0$ 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 3.2. *Let* $\mu_0 > 1/L(f)$*, then at each iteration of Algorithm 3.1 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 Section 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 3.1 for $\theta_k \equiv 1$ is analysed 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 $\mu_k$ is monotonically nonincreasing. Here we will use an analysis, similar to that in [1], but which allows for increases in the step size in Algorithm 3.1, while preserving the algorithm's $O(\sqrt{L(f)/\epsilon})$ complexity. As we will show, we can accomplish this by choosing appropriate values of $\theta_k$.

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

LEMMA 3.3. *Assume that on each step of Algorithm 3.1,* $\mu_k t_k^2 \geq \mu_{k+1} t_{k+1}(t_{k+1} - 1)$*. 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 2.3. 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 3.1 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 3.4. *Assume that on each step of Algorithm 3.1, $\mu_k t_k^2 \geq \mu_{k+1}t_{k+1}(t_{k+1}-1)$. The sequence $\{x^k\}$ generated by Algorithm 3.1 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 3.3 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 2.3 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 3.1

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

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] 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 $\mu_{k+1}$ on iteration $k$, when $\theta_k$ and $t_{k+1}$ are computed. Simply setting $\theta_k \leq \mu_k/\mu_0$, where $\mu_0$ is an upper bound on the step size will imply that $\theta_k < 1$ for all $k$. Hence it is necessary to update $\theta_k$ and $t_{k+1}$ along with $\mu_{k+1}$, thus expanding the backtracking part of the algorithm,

We now present Algorithm 3.5 which is an extension of Algorithm 3.1 which allows for a full backtracking and any size $\mu_k$. It satisfies conditions (3.10)-(3.11), while setting the step size $\mu_k$ initially to some value $\mu_k^0$ at the beginning of each iteration. As we will show, Algorithm 3.5 is designed to maintain $\theta_k = \mu_k/\mu_{k+1}$ for all $k \geq 1$.

Recall that the parameter $\theta_k$ is used to compute $t_{k+1}$ and $y^{k+1}$ in Algorithm 3.1 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 3.5, $\theta_{k-1} = \mu_{k-1}/\mu_k^0$, where $\mu_k^0$ is an initial "guess" for the value of $\mu_k$. 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 $\mu_k$. If the "guess" is correct, then no correction to $\theta_{k-1}$ is needed. If $\mu_k$ is reduced during the backtracking, then $\theta_{k-1}$ is recomputed to account for the new value of $\mu_k$ 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, $p_{\mu_k}(y^k)$ has to be recomputed and a new backtracking step has to be performed. The backtracking starts with the value of $\mu_k$ with which the previous backtracking stopped, and hence, which was used to compute the most recent value of $\theta_{k-1}$. If the value of $\mu_k$ is not reduced any further then $\theta_{k-1}$, $y^k$ and $t_k$ have the correct value and the backtracking part of the iteration is complete. If the value of $\mu_k$ is reduced further then $\theta_{k-1}$ is recomputed again and backtracking continues.

ALGORITHM 3.5.

*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* $k = 1, 2, \ldots$ *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 algorithm and the discussion above.

LEMMA 3.6. *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 3.7. *Let* $\{\mu_k, t_k\}$ *be a sequence which 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 $\mu_i \geq \beta/L(f)$,

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

□

Since Algorithm 3.5 maintains $\theta_k = \mu_k/\mu_{k+1}$, then (3.11) holds automatically and we have shown that (3.10) holds with $\eta = \beta L(f)/4$ by Lemma 3.7. Thus the following two complexity results follow immediately from Theorem 3.4.

THEOREM 3.8. *At the k-th iteration of Algorithm 3.5 we have*

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

THEOREM 3.9. *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 3.5. Assume that* $\mu_i^0 \geq 1/L(f, g, y^i)$ *for all* $1 \geq i \geq k$. *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 $i = 1, \ldots, k$ $\mu_i \geq \beta/L(f, g, y^i)$ and applying Lemma 3.7. □

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 3.5, Nesterov's method allows repeated reduction of the prox parameter $\mu_k$ by a given factor on the $k$-th iteration until a backtracking condition is met. While Nesterov's method minimizes the same function $Q_{\mu_k}$ on each inner step as does FISTA and Algorithms 3.5, it utilizes a different update of the parameters that are used to compute the new extrapolated point $y^k$ and also 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, hence increasing the per-iteration cost compared to that of Algorithm 3.5 (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 steps size 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

3.7. However, Theorem 3.9 does not apply directly to the method in [9] because in that case $\mu_k$ 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 $\mu_k^0$, and the condition $\mu_k t_k^2 = \mu_{k+1}(t_{k+1}^2 - t_{k+1})$ is maintained at each iteration. If $\mu_k^0$ is chosen to be a large fixed value $\mu_0$ 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 re-computation 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 3.5 at most a logarithmic factor $L(f)$ worse than that of FISTA, i.e. Algorithm 3.1. The advantage of choosing large $\mu_k^0$ is in the potentially rapid increase of the prox parameter if such increase if possible and desirable due to a rapid change in local Lipschitz behavior of $\nabla f(x)$.

On the other hand $\mu_k^0$ can be selected to have a more conservative value (as long as the value of $\theta_k$ is chosen accordingly). For instance one can choose $\mu_k^0$ to equal $\mu_{k-1}/\beta$ at the beginning of the $k$-th iteration. This will significantly reduce the number of backtracking steps. This strategy of increasing $\mu_k$ incrementally is what is used in Nesterov's method in [9] where $\mu_k^0$ is set to $\mu_{k-1}/\alpha$, 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 3.5 if $\mu_k^0$ is chosen to equal $\mu_{k-1}/\alpha$ instead of some fixed large value $\mu_0$.

LEMMA 3.10. *For any $k \geq 1$ the number of calls to FistaStep by the end of the $k$-th iteration is bounded by*

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

If we choose $\alpha = \beta$ then we see that the average number calls of *FistaStep* per iteration is 2. Hence the logarithmic factor is removed from the overall complexity of Algorithm 3.5.

It is easy to see that (3.14) still holds for this case, and hence Lemma 3.7 applies. However, the drawback of this approach is that we no longer can guarantee that $\mu_k \geq \beta/L_k$ for each $k$ and Theorem 3.9 may not hold for $L_k$ being the estimate of the local composite Lipschitz constant defined in Section 2.1. The bound on $\mu_k$ 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. Applicability of such an approach is the 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 additional $O(n)$ operations, hence the main additional work comes from the necessity to recompute $F(p_{\mu_k}(y^k))$ 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 3.5 compared to that of Algorithms 3.1 lies in a call to *FistaStep* updates and re-computation 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 solely defined by the choice of $\mu_0^k$ 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 the problem of the form,

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

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 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 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 3.5 compared to that of Algorithms 3.1 is only $O(n)$, which is negligible.

However, as discussed earlier, using larger values of $\mu_0^k$ 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 product comprise the dominant cost of each iteration, then the total cost of an iteration without backtracking equals two matrix vector products, while the cost of an iteration with backtracking contains additional matrix-vector product per each backtracking step. For instance, if $\mu_k^0 = \mu_{k-1}/\beta$, then by Lemma 3.10, the average cost of an iteration of Algorithm 3.5 is 3/2 that of Algorithms 3.1. Such cost increase is beneficial if the number of iterations Algorithm 3.5 is proportionately smaller.

In the examples we consider in our computational experiments in Section 5 increasing $\mu_k$ at each iterations appears to be wasteful. Hence we choose to allow for the increase of 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 least number of failed backtracking steps. The key assumption for using such a heuristics 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. In case of significant changes in the Lipschitz constants we believe any backtracking heuristic will produce significant improvement over pure FISTA algorithm as 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 4.2. 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 $\lambda$ is some element of the subdifferential $\partial g(u)$.*

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

Algorithm 4.2 (FALM-S) given below was proposed in [7] for the cases when computing $p_\mu^f(v)$ and $p_\mu^g(u)$, defined above, is relatively easy; i.e., comparable to a gradient computation of $g$ and $f$ respectively. We call $k$-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 when 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 above condition by choosing a sufficiently small value of $\mu$. 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 4.2.

*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 $k = 1, 2, \ldots$ 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 efficient computation of $\lambda^{k+1} \in -\partial g(z^{k+1})$ we refer the reader to [7].

As one can observe, Algorithm 4.2 does not include a mechanism for selecting $\mu$ at each iteration - the prox parameter is assumed to be fixed. In that case $\theta_k$ parameter in *FistaStep* computation equals $(\mu+\mu)/(\mu+\mu) = 1$ when on both iterations, $k$ and $k+1$, there was no skipped steps. Alternatively, $\theta_k =$

$\mu/(\mu+\mu) = 1/2$ when a step was skipped on iteration $k$ but not on iteration $k+1$, while $\theta_k = (\mu+\mu)/\mu = 2$ when a step was skipped on iteration $k+1$ but not on iteration $k$. The convergence rate results in [7] can be easily extended to the case when $\mu_k$ is chosen via backtracking as it is done in FISTA, as long at $\mu_k \leq \mu_{k-1}$, with the same values for $\theta_k$. Here we propose a backtracking variant of FALM-S, which is given below as Algorithm 4.3. We note that we now have two linearization and minimization steps per iteration; hence, we allow the parameter $\mu$ to be set separately to $\mu_k^x$ and $\mu_k^y$, respectively, for each of the minimization steps. The average $\mu_k = (\mu_k^x + \mu_k^y)/2$ is then used to compute $\theta_k$ and, hence, to update the parameter $t_{k+1}$. The new algorithm is a generalization of Algorithm 4.2 and includes skipping steps as a special case when *skip* = true and $\mu_k^x = 0$.

ALGORITHM 4.3.

*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* $k = 1, 2, \ldots$ *do*

(1) *Set* $\mu_k^x := \bar{\mu}_k^x, \mu_x^y := \bar{\mu}_k^y, skip := true$

(2) *Compute* $p^g_{\mu_k^x}(z^k)$

*If* $F(p^g_{\mu_k^x}(z^k)) \leq Q^g_{\mu_k^x}(z^k, p^g_{\mu_k^x}(z^k))$ *then* $x^k := p^g_{\mu_k^x}(z^k), skip := false$

*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* $p^f_{\mu_k^y}(x^k)$

*If* $F(p^f_{\mu_k^y}(x^k)) \leq Q^f_{\mu_k^y}(p^f_{\mu_k^f}(x^k), x^k)$ *then* $y^k := p^y_{\mu_k^y}(x^k)$

*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* $\bar{\mu}_{k+1}^y, \bar{\mu}_{k+1}^x$

$\theta_k := \frac{2\mu_k}{\bar{\mu}_{k+1}^y + \bar{\mu}_{k+1}^x}, (t_{k+1}, z^{k+1}) := FistaStep(y^k, y^{k-1}, t_k, \theta_k)$

Let us, first, observe that the algorithm generates a sequence $\{\mu_k, t_k\}$ which satisfy condition analogous to condition (3.11), which we required for FISTA-BKTR. Indeed, it is evident from the computation of $\theta_k$ 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 4.3 to reach an $\epsilon$-optimal solution, we need the following lemma.

LEMMA 4.4. *The sequence* $\{x^k, y^k\}$ *generated by Algorithm 4.3 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* $\mu_k = (\mu_k^x + \mu_k^y)/2$.

*Proof.* We will prove that the following inequality holds:

$$\begin{aligned} &2(\mu_k t_k^2 v_k - \mu_{k+1} t_{k+1}^2 v_{k+1}) \\ \geq &t_{k+1}(t_{k+1} - 1)\left(\|y^{k+1} - y^k\|^2 - \|z^{k+1} - y^k\|^2\right) + t_{k+1}\left(\|y^{k+1} - x^*\|^2 - \|z^{k+1} - x^*\|^2\right). \end{aligned} \tag{4.2}$$

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 2.3 to the function $Q^f_\mu(\cdot,\cdot)$ with $\mu = \mu^y_{k+1}$, $x = y^k$ and $y = x^{k+1}$, we have $p^f_{\mu^y_{k+1}}(y) = y^{k+1}$ and

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

In the case of a regular iteration, that is when $\mu^x_{k+1} > 0$, applying Lemma 2.3 to the function $Q^g_\mu(\cdot,\cdot)$ with $\mu = \mu^x_{k+1}$, $x = y^k$ and $y = z^{k+1}$, we obtain $p^g_{\mu^x_{k+1}}(y) = x^{k+1}$ and

$$2\mu^x_{k+1}(F(y^k) - F(x^{k+1})) \geq \|x^{k+1} - y^k\|^2 - \|z^{k+1} - y^k\|^2. \tag{4.4}$$

In the case when $\mu^x_{k+1} = 0$, since $x^{k+1} := z^{k+1}$, (4.4) still holds.

Summing (4.3) and (4.4), and using the fact that $F(y^{k+1}) \leq F(x^{k+1})$, we obtain,

$$2\mu_{k+1}(v_k - v_{k+1}) = 2\mu_{k+1}(2F(y^k) - 2F(y^{k+1})) \geq \|y^{k+1} - y^k\|^2 - \|z^{k+1} - y^k\|^2 \tag{4.5}$$

Again, applying Lemma 2.3 in the case when $\mu^x_{k+1} > 0$ to the function $Q^g_{\mu^x_{k+1}}(\cdot,\cdot)$ with $x = x^*$ and $y = z^{k+1}$, we get $p^g_{\mu^x_{k+1}}(y) = x^{k+1}$ and

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

On the other hand, if $\mu^x_{k+1} = 0$ then $x^{k+1} := z^{k+1}$ and (4.6) still holds. Lemma 2.3 applied to the function $Q^f_{m^y_{k+1}}(\cdot,\cdot)$ with $x = x^*$, $y = x^{k+1}$ and $p^f_{\mu^y_{k+1}}(x^{k+1}) = y^{k+1}$ gives

$$2\mu^y_{k+1}(F(x^*) - F(y^{k+1}) \geq \|y^{k+1} - x^*\|^2 - \|x^{k+1} - x^*\|^2. \tag{4.7}$$

Summing (4.6) and (4.7), and again using the fact that $F(y^{k+1}) \leq F(x^{k+1})$ and $\mu_k = (\mu^x_k + \mu^y_k)/2$, we obtain,

$$-2\mu_{k+1}v_{k+1} = 2\mu_{k+1}(2F(x^*) - 2F(y^{k+1})) \geq \|y^{k+1} - x^*\|^2 - \|z^{k+1} - x^*\|^2. \tag{4.8}$$

If we multiply (4.5) by $t_{k+1}(t_{k+1}-1)$, and (4.8) by $t^2_{k+1}$, and take the sum of the resulting two inequalities, we get (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 4.3 maintains $\theta_k = \mu_k/\mu_{k+1}$ on each iteration, then from Lemma 3.7 we know that for the sequence $\{\mu_k, t_k\}$ generated by Algorithm 4.3,

$$\mu_k t_k^2 \geq (\sum_{i=1}^{k} \sqrt{\mu_i}/2)^2. \tag{4.9}$$

Now we are ready to give the complexity of Algorithm 4.3. Applying a proof equivalent to that of Theorem 3.4 we get

$$F(x^k) - F(x^*) \leq \frac{\|x^0 - x^*\|^2}{4\mu_k t_k^2}. \tag{4.10}$$

In the worst case we have $\mu^x_0 = 0$ and $\mu_k \geq \frac{1}{2L(f)}$ and the algorithm reduces to FISTA. Hence, just as for FISTA, from (4.9) we have

THEOREM 4.5. *At each iteration $k$ of Algorithm 3.5 we have*

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

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

THEOREM 4.6. *Let $\sqrt{\mu(k)} = (\sum_{i=1}^{k} \sqrt{\mu_i})/k$ be the "average" of the square roots of the average prox parameters $\mu_i$ used during the first $k$ iterations of Algorithm 4.3. Then*

$$F(x^k) - F(x^*) = v_k \leq \frac{2\|x^0 - x^*\|^2}{\mu(k)k^2}. \tag{4.12}$$

Note that the prox parameters $\mu_i$ 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 4.3 compared to that of Algorithms 4.2 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 $p^g_{\mu^x}(y)$ is a relatively cheap operation. Computing $p^f_{\mu^y}(x)$, 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 when such special structure is not present, the work which involves factorization of $A^\top A + \frac{1}{2\mu^y} I$ may be the dominant cost of the iteration, as it generally requires $O(m^3)$ operations.

If $\mu^y$ 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 $\mu^y$ is updated in an arbitrary way on each iteration, then the factorization has to be repeated each time. Recall, that ideally we want $\mu^y$ to have the largest possible value which satisfies the line search conditions in Step 2 of Algorithm 4.3 and that keeping $\mu^y$ constant may result in very slow progress. Hence, again, there exists a tradeoff between choosing a suitable value of prox parameter and the per iteration cost. For practical efficiency we strive to achieve a reasonable balance. Assume that we choose some value $\bar{\mu}^y_1$ at the beginning of the algorithm and we choose $\bar{\mu}^y_{k+1} = \beta^i \mu^y_k$ for some $i \in Z$ at each iteration $k$. Note that in this case, for all $k$ $\mu^y_k$ can only take values $\beta^j \mu^y_1$ for $j \in Z$. Let us consider $\beta = 0.5$. We also impose the following restriction of $\mu^y_k$ - if $\mu^y_k < \mu^y_1/1000$, then the expected improvement achieved by Step 2 is too small and the step is automatically skipped; if $\mu^y_k > 1000\mu^y_1$, the prox parameter the step size is large enough and no additional increase of $\mu^y_k$ is necessary. Hence the only values allowed for $\mu^y_k$ throughout the algorithm are $\{2^{-10}\mu^y_1, 2^{-9}\mu^y_1, ..., 0, 2\mu^y_1, 4\mu^y_1, ..., 2^{10}\mu^y_1\}$, 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 $p^f_{\mu^y}(x)$. 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 $\mu_k^x$ parameter and the additional per iteration cost of step 3 can be executed similarly to what is described in Section 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:

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

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 3.1 with $\theta_k = 1$.
- FISTA-BKTR: an efficient implementation of Algorithm 3.5 as discussed in Section 3.1.
- FALM: an implementation of Algorithm 4.2 as discussed in Section 4.
- FALM-BKTR: an efficient implementation of Algorithm 4.3 as discussed in Section 4.1.
- SpaRSA: a gradient based algorithm, with the use of shrinking, described in [5].
- Yall1: a solver based on alternating directions 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 the iterations it took to reach a function value which is 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 100000, 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 behavior of our methods is reasonable. Our implementations were written in MATLAB and run in MATLAB R2010b on a laptop with Intel Core Duo 1.8 GHz CPU and 2GB 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. The Spear Examples.** This set of instances are 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* ($1024 \times 512$). 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 $\rho$ decreases.

From Table 5.1, 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 1343 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 $\mu_g$ ($\mu_g = 1$). Initial

TABLE 5.1

*Comparison of the algorithms for solving* (5.1) *with* $\rho = 1$ *on Spear10.* $FISTA(100) = 5.3839e+5$, $FISTA(500) = 1.2799e+5$, $FISTA(1000) = 1.0035e+5$. *The starting* $\mu$ *for FISTA/FISTA-BKTR,* $\mu_f$ *for FALM/FALM-BKTR and* $\mu_g$ *for FALM-BKTR are all set to be 1. For FALM (with skipping), we tried different values for* $\mu_g$. *The starting* $\mu_g$*'s for FALM-S1, FALM-S2, FALM-S3 are 1, 10 and 100, respectively. Moreover, for FALM/FALM-BKTR the number in parentheses is the number of matrix factorization required overall.*

| 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 |

performance of FALM-S2 and, especially, FALM-S3 is good due to larger starting values for $\mu_g$, 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.

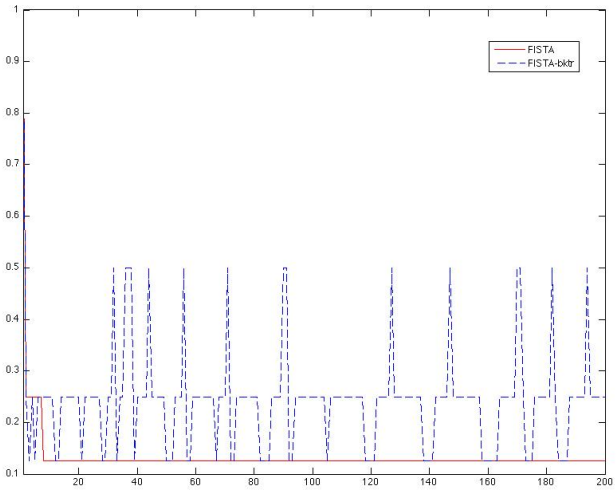


FIG. 5.1. *Comparison on the* $\mu$ *values while solving* (5.1) *with* $\rho = 1$ *on Spear10*

In Figure 5.1 we plot how $\mu$ 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 $\mu$ by allowing backtracking, and thus performs large steps on some of the iterations, which corresponds with the smaller number of iterations required by FISTA-BKTR.

Table 5.2 shows the results on Spear10 problem with $\rho = 0.01$. This problem provides a difficult instance where our backtracking methods appears to provide clear advantage. SpaRSA did not converge to the proximity of the solution, while Yall1 only achieved accuracy of $10^{-1}$, but not $10^{-3}$. Here we observe that FISTA-BKTR 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 5.2

*Comparison of the algorithms for solving* (5.1) *with* $\rho = 0.01$ *on Spear10. The starting* $\mu$ *is set to be 0.01.* $FISTA(100) = 6.0980e+3$, $FISTA(500) = 5.8943e+3$, $FISTA(1000) = 5.4176e+3$.

| 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 |

*Spear3* ($1024 \times 512$). with $\rho = 0.1$. Dynamic range is 2.7535e+4. Sparsity is 6. We observe that behavior of FALM-BKTR converges to that of FISTA-BKTR in later iterations due to persistent skipping of Step 2.

TABLE 5.3

*Comparison of the algorithms for solving* (5.1) *with* $\rho = 0.1$ *on Spear3. The starting* $\mu$ *is set to be 1.* $FISTA(100) = 1.1825e+4$, $FISTA(500) = 1.1793e+4$, $FISTA(1000) = 1.1784e+4$.

| 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 |

**5.2. Bdata problems.** Bdata test set was originally created by A. Nemirovski with the aim to imitate examples with worst-case complexity for the first-order methods. This problem, however, provides a relatively easy instance probably due to presence of the $\ell_1$ term in the objective. Here we present results for Bdata1 ($1036 \times 1036$), with $\rho = 0.0001$; dynamic range is 5.9915 and sparsity is 16.

TABLE 5.4

*Comparison of the algorithms for solving* (5.1) *with* $\rho = 0.0001$ *on Bdata1. The starting* $\mu$ *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* $\epsilon_b$ *is set to be 0.001.*

| 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 |

**5.3. Sparco problems.** This category of instances are 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 utilize, we do not include FALM in this comparison. We present results for Sparco3 ($2048 \times 1024$), with $\rho = 0.01$, dynamic range of 2 and sparsity of 2. We observe that, for this relatively easy instance, FISTA-BKTR has minor advantage over FISTA in terms of the number of matrix-vector multiplications. For this example FISTA outperforms the alternating direction based method Yall1, while Sparsa seems to be the winning method for this instance.

TABLE 5.5

*Comparison of the algorithms for solving* (5.1) *with* $\rho = 0.01$ *on Sparco3. The starting* $\mu$ *is set to be 1.* $FISTA(10) = 13.07180$, $FISTA(50) = 8.187212$, $FISTA(100) = 2.710062$. *The tolerance*$\epsilon_b$ *is set to be 0.001.*

| 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 |

**5.4. Smoothed $\ell_2$ norm minimization.** As an alternative to problem (5.1) one may wish to solve the following problem with exact $\ell_2$ penalty term:

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

In order to apply FISTA and FALM family of methods, we can smooth the $\ell_2$ term with the well-known Huber penalty function, and obtain the following minimization problem:

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

where

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

for $\nu > 0$. If $\nu < \epsilon$, then the solution of (5.3) is an $\epsilon$-solution to 5.2. converges to that of (5.2). We define $f(x) := H_\nu(||Ax-b||_2)$ and $g(x) := \rho\|x\|_1$. It is well known that global Lipschitz constant of $\nabla f(x)$ is $O(1/\nu)$. Recall Example 2.7, where we analyze the local composite Lipschitz constant for the case when $g(x) \equiv 0$ and show that away from the optimal solution the local composite Lipschitz constant is of $\nabla f(x)$ is much smaller than $O(1/\nu)$. The analysis of the case when $g(x) := \rho\|x\|_1$ is a lot more complex, but the essential expectation remains for our first-order schemes: the prox parameter $\mu$ will be relatively large away from the solutions, while it will decrease as the algorithms converges. In fact FISTA and FALM in their original form will observe the same behavior of the prox parameter, as they allow for the prox parameter to decrease, but not to 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.

In Tables 5.6 and 5.7 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 when the initial prox parameter value is not very large. As compared with FISTA, FISTA-BKTR allows for a huge increase

in $\mu$. If initial $\mu$ is set to 1, then FISTA performance is very slow, as is shown in Table 5.6. But if $\mu = 1000$, then FISTA performance improves, to the level of FISTA-BKTR, as is seen in 5.7. This is well explained by showing graphically the change of $\mu$ in Fig 5.2.

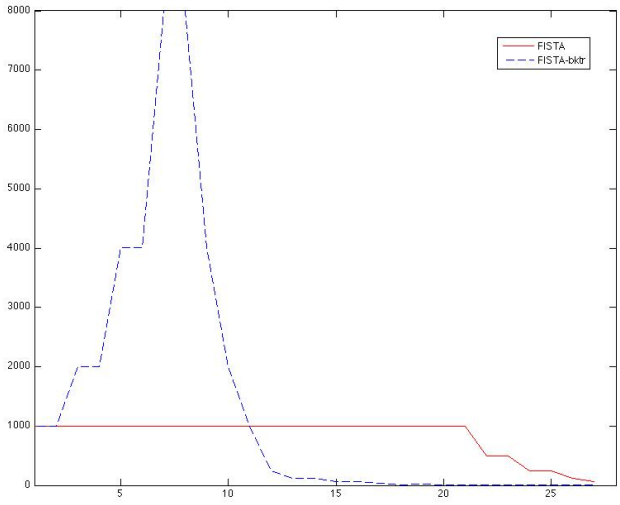


FIG. 5.2. *Comparison on the $\mu$ values while solving* (5.3) *with $\rho = 0.1$ on Spear10*

TABLE 5.6
*Comparison of the algorithms for solving* (5.3) *with $\rho = 0.1$ on Spear10. The starting $\mu$ is set to be 1. $FISTA(100) = 4.4552e+4$, $FISTA(500) = 2.4248e+4$, $FISTA(1000) = 1.1554e+4$*

| 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 |

TABLE 5.7
*Comparison of the algorithms for solving* (5.3) *with $\rho = 0.1$ on Spear10. The starting $\mu$ is set to be 1000. $FISTA(10) = 3.4189e+4$, $FISTA(20) = 1.0866e+4$, $FISTA(40) = 9.9944e+3$*

| 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 |

In Table 5.8 and Figure 5.3 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 $\mu$ value for both algorithms becomes small which indicates large Lipschitz constant for solving (5.3) on Bdata1.

**5.5. $\ell_2$ regularized logistic regression.** Finally, we illustrate the behavior of FISTA vs. FISTA-BKTR on an example of $\ell_2$ regularized logistic regression applied to an optical character recognition data set "Optdigits" from UCI repository [6]. We present this example here purely for illustration purpose to show that on settings other than compressed sensing the backtracking strategy can produce significant

TABLE 5.8

*Comparison of the algorithms for solving* (5.3) *with* $\rho = 0.01$ *on Bdata1. The starting* $\mu$ *is set to be 1.* $FISTA(100) = 0.0196$, $FISTA(500) = 0.0164$, $FISTA(1000) = 0.0164$

| 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 |

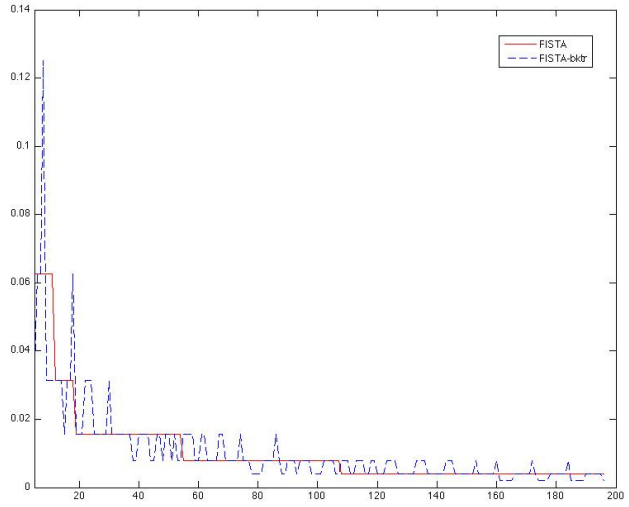


FIG. 5.3. *Comparison on the* $\mu$ *values while solving* (5.3) *with* $\rho = 0.01$ *on Bdata*

improvement. As we discussed, to obtain optimal performance from backtracking a careful implementation is needed that tries to take into account the problem structure. Such implementation for logistic regression and other problems is a subject of future research. Here we present results of basic approach where $\mu$ is increased by a factor of 2 on each iteration. FISTA required 5705 iterations and 11417 matrix-vector multiplications to obtain a solution with gradient norm less that $10^{-2}$, while FISTA-BKTR required 1315 iterations and 3512 matrix-vector multiplications. In Figure 5.4 we plot the behavior of the $\mu$ parameter for both algorithms, which clearly shows that FISTA-BKTR benefits from much larger steps. The analysis of local Lipschitz constants for logistic regression is a subject of future research.

**6. Conclusion.** We present a generalized version of accelerated first-order schemes which are able to estimate 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 then can 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 can 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 is a subject of a future study. Our computational experiments and the discussion of a practical implementation show that in practice our proposed backtracking schemed offer improvement of accelerate first-order algorithms.

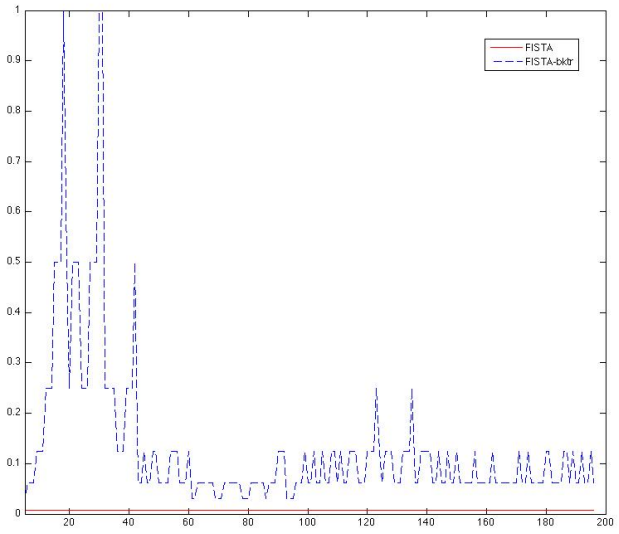


FIG. 5.4. *Comparison on the $\mu$ values while solving logistic regression problem on "Optdigits" dataset.*

**7. Acknowledgements.** The authors are grateful to A. d'Aspremont for helpful discussion on the average case behavior and to S. Ma for the help with the preparation of the manuscript. We are also grateful to the two anonymous referees for their very helpful comments.

**References.**

[1] AMIR 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. YILMAZ, *Sparco: A testing framework for sparse reconstruction*, Tech. Report TR-2007-20, Dept. Computer Science, University of British Columbia, Vancouver, October 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 Journal on Selected Topics in Signal Processing, 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*, tech. report, Department of IEOR, Columbia University, 2010.

[8] A. NEMIROVSKI AND D. YUDIN, *Informational complexity and efficient methods for solution of convex extremal problems*, J. Wiley & Sons, New York, 1983.

[9] Y. E. NESTEROV, *Gradient methods for minimizing composite objective function*, URL:http://rwww.optimization-online.org.

[10] ——, *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] ——, *Introductory lectures on convex optimization*, 87 (2004), pp. xviii+236. A basic course.

[12] ——, *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 l1*, http://www.caam.rice.edu/ optimization/L1/YALL1/, (2009).