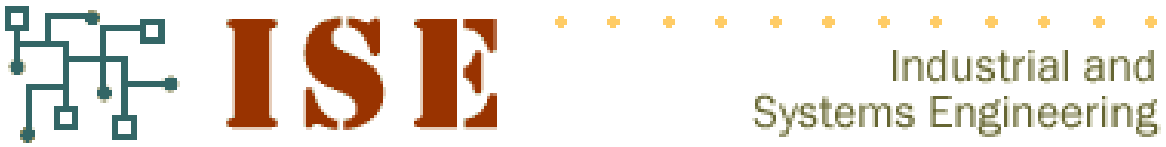



# Practical Inexact Proximal Quasi-Newton Method with Global Complexity Analysis

Katya Scheinberg and Xiaocheng Ta

Department of Industrial and Systems Engineering, Lehigh University, Harold S. Mohler Laboratory, 200

COR@L Technical Report 13T-02-R1


# Practical Inexact Proximal Quasi-Newton Method with Global Complexity Analysis

Katya Scheinberg*[1] and Xiaocheng Tang†[1]

[1]Department of Industrial and Systems Engineering, Lehigh University, Harold S. Mohler Laboratory, 200 West Packer Avenue, Bethlehem, PA 18015-1582, USA.

June 15, 2015

**Abstract**

Recently several methods were proposed for sparse optimization which make careful use of second-order information [11, 31, 17, 3] to improve local convergence rates. These methods construct a composite quadratic approximation using Hessian information, optimize this approximation using a first-order method, such as coordinate descent and employ a line search to ensure sufficient descent. Here we propose a general framework, which includes slightly modified versions of existing algorithms and also a new algorithm, which uses limited memory BFGS Hessian approximations, and provide a global convergence rate analysis in the spirit of proximal gradient methods, which includes analysis of method based on coordinate descent.

## 1 Introduction

In this paper, we are interested in the following convex optimization problem:

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

where $f, g : \mathbb{R}^n \to \mathbb{R}$ are both convex functions such that $\nabla f(x)$ is assumed to be Lipschitz continuous with Lipschitz constant $L(f)$, i.e.,

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq L(f)\|x - y\|_2, \quad \forall x, y \in \mathbb{R}^n,$$

and $g(x)$ is convex and has some structure that can be exploited. In particular, in much of the research on first order methods for problem (1.1) $g(x)$ is considered to be such that the following problem has a closed form solution for any $z \in \mathbb{R}^n$:

$$\min_{x\in\mathbb{R}^n} \left\{ g(x) + \frac{1}{2}\|x - z\|^2 \right\}.$$

*katyas@lehigh.edu. The work of this author is partially supported by NSF Grants DMS 10-16571, DMS 13-19356, AFOSR Grant FA9550-11-1-0239, and DARPA grant FA 9550-12-1-0406 negotiated by AFOSR.

†xct@lehigh.edu. The work of this author is partially supported by DARPA grant FA 9550-12-1-0406 negotiated by AFOSR.

Here our general requirement on $g(x)$ is slightly different - we assume that the following problem is computationally inexpensive to solve approximately, relative to minimizing $F(x)$ for any $z \in \mathbb{R}^n$ and some class of positive definite matrices $H$:

$$\min_{x \in \mathbb{R}^n} \left\{ g(x) + \frac{1}{2}\|x - z\|_H^2 \right\}. \tag{1.2}$$

Here $\|y\|_H^2$ denotes $y^\top H y$. Clearly, the computational cost of approximately solving (1.2) depends on the choice of matrix $H$ and the solution approach.

We are particularly interested in the case of sparse optimization, where $g(x) = \lambda\|x\|_1$. While the theory we present here applies to the general form (1.1), the efficient method for solving (1.2) that we consider in this paper is designed with $g(x) = \lambda\|x\|_1$ example in mind. In this case problem (1.2) takes a form of an unconstrained Lasso problem [26]. We consider matrices $H$ which are a sum of a diagonal matrix and a low-rank matrix and we apply randomized coordinate descent to solve (1.2) approximately. An extension to the group sparsity term $g(x) = \lambda \sum \|x_i\|_2$ [18], is rather straightforward.

Problems of the form (1.1) with $g(x) = \lambda\|x\|_1$ have been the focus of much research lately in the fields of signal processing and machine learning. This form encompasses a variety of machine learning models, in which feature selection is desirable, such as sparse logistic regression [30, 31, 24], sparse inverse covariance selection [11, 17, 21] and unconstrained Lasso [26], etc. These settings often present common difficulties to optimization algorithms due to their large scale. During the past decade most optimization effort aimed at these problems focused on development of efficient first-order methods, such as accelerated proximal gradient methods [15, 2, 28], block coordinate descent methods [31, 10, 9, 21] and alternating directions methods [20]. These methods enjoy low per-iteration complexity, but typically have slow local convergence rates. Their performance is often hampered by small step sizes. This, of course, has been known about first-oder methods for a long time, however, due to the very large size of these problems, second-order methods are often not a practical alternative. In particular, constructing and storing a Hessian matrix, let alone inverting it, is prohibitively expensive for values of $n$ larger than 10000, which often makes the use of the Hessian in large-scale problems prohibitive, regardless of the benefits of fast local convergence rate.

Nevertheless, recently several new methods were proposed for sparse optimization which make careful use of second-order information [11, 31, 17, 3]. These new methods are designed to exploit the special structure of the Hessian of specific functions to improve efficiency of solving (1.2). Several successful methods employ coordinate descent to approximately solve (1.2). While other approaches to solve Lasso subproblem were considered in [3], none generally outperform coordinate descent, which is well suited when special structure of the Hessian approximation, $H$, can be exploited and when low accuracy of the subproblem solutions is sufficient. In particular, [31] proposes a specialized GLMNET [10] implementation for sparse logistic regression, where coordinate descent method is applied to the unconstrained Lasso subproblem constructed using the Hessian of $f(x)$. The special structure of the Hessian is used to reduce the complexity cost of each coordinate step so that it is linear in the number of training instances, and a two-level shrinking scheme proposed to focus the minimization on smaller subproblems. Similar ideas are used in [11] in a specialized algorithm called QUIC for sparse inverse covariance selection, where the Hessian of $f(x)$ also has a favorable structure for solving Lasso subproblems. Another specialized method for graphical Markov random fields was recently proposed in [29]. This method also exploits special Hessian structure to improve coordinate descent efficiency.

There are other common features shared by the methods described above. These methods are often referred to as proximal Newton-type methods. The overall algorithmic framework can be described as follows:

- At each iteration $k$ the smooth function $f(x)$ is approximated near the current iterate $x^k$ by a convex quadratic function $q^k(x)$.
- A working subset of coordinates (elements) of $x$ is selected for subproblem optimization.
- Then $l(k)$ passes of coordinate descent are applied to optimize (approximately) the function $q^k(x) + g(x)$ over the working set, which results in a trial point. Here $l(k)$ is some linear function of $k$.
- The trial point is accepted as the new iterate if it satisfies some sufficient decrease condition (to be specified).
- Otherwise, a line search is applied to compute a new trial point.

In this paper we *do not* include the theoretical analysis of various working set selection strategies. Some of these have been analyzed in the prior literature (e.g., see [13]). Combining such existing analysis with the rate of convergence results in this paper is a subject of a future study.

This paper contains the following three main results.

(1) We discuss the theoretical properties of the above framework in terms of global convergence rates. In particular, we show that if we replace the line search by a prox-parameter update mechanism, we can derive sublinear global convergence results for the above methods under mild assumptions on Hessian approximation matrices, which can include diagonal, quasi-Newton and limited memory quasi-Newton approximations. We also provide the convergence rate for the case of inexact subproblem optimization.

(2) The heuristic of applying $l(k)$ passes of coordinate descent to the subproblem is very useful in practice, but has not yet been theoretically justified, due to the lack of known complexity estimates. Here we use probabilistic complexity bounds of randomized coordinate descent to show that this heuristic is indeed well justified theoretically. In particular, it guarantees the sufficiently rapid decrease of the expectation of the error in the subproblems and hence allows for sublinear global convergence rate to hold for the entire algorithm (again, in expectation). This gives us the first complete global convergence rate result for the algorithmic schemes for practical proximal Newton-type methods.

(3) Finally, we propose an efficient *general purpose* algorithm that uses the same theoretical framework, but which does not rely on the special structure of the Hessian, and yet in our tests compares favorably with the state-of-the-art, specialized methods such as QUIC and GLMNET. We replace the exact Hessian computation by the limited memory BFGS Hessian approximations [16] (LBFGS) and exploit their special structure within a coordinate descent approach to solve the subproblems.

Let us elaborate a bit further on the new approaches and results developed in this paper and discuss related prior work.

In [4] Byrd et al. propose that the methods in the framework described above should be referred to as sequential quadratic approximation (SQA) instead of proximal Newton methods. They reason that there is no proximal operator or proximal term involved in this framework. This is indeed the case, if a line search is used to ensure sufficient decrease. Here we propose to consider a prox term as a part of the quadratic approximation. Instead of a line search procedure, we update the prox term of our quadratic model, which allows us to extend global convergence bounds of proximal gradient methods to the case of proximal (quasi-)Newton methods. The criteria for accepting a

new iteration is based on sufficient decrease condition (much like in trust region methods, and unlike that in proximal gradient methods). We show that our mechanism of updating the prox parameter, based on sufficient decrease condition, leads to an improvement in performance and robustness of the algorithm compared to the line search approach as well as enabling us to develop global convergence rates.

Convergence results for, so-called, proximal Newton method have been shown in [12] and more recently in [4] (with the same sufficient decrease condition as ours, but applied within a line search). These results apply to our framework when exact Hessian of $f(x)$ is used to construct $q(x)$. But they do not apply in the case of LBFGS Hessian approximations, moreover they do not provide global convergence rates.

To provide such rates we use techniques similar to those in [2] and [23] for the proof of convergence rates of the (inexact) proximal gradient method, where we replace the diagonal Hessian approximation with a general positive definite Hessian approximation matrix. We extend the results in [2] and [23] to accept iterates based on sufficient decrease condition instead of full decrease, which allows more flexibility in the algorithm. Finally, we use the complexity analysis of randomized coordinate descent in [19] to provide a simple and efficient stopping criterion for the subproblems and thus derive the total complexity of proximal (quasi-)Newton methods based on randomized coordinate descent to solve Lasso subproblems. Our theory can be easily extended to the case of inexact gradient computation, in addition to inexact subproblem optimization, similarly to the theory developed in [23].

Another very relevant work [8] was brought to our attention. In that paper the authors analyze global convergence rates of an accelerated proximal quasi-Newton method, as an extension of FISTA method [2]. The convergence rate they obtain match that of accelerated proximal gradient methods, hence it is a faster rate than that of our method presented here. However, as in the case of many accelerated gradient methods, they have to impose much stricter conditions on the Hessian approximation matrix, in particular they require that the difference between any two consecutive Hessian approximations (i.e., $H_{k+1} - H_k$) is positive semidefinite. This is a natural extensions of the FISTA's requirement that the prox parameter is never increased. Such a condition is fairly restrictive in practice and in particular would not apply to our simple LBFGS approximation strategy. Additionally, the convergence rate dependence on the error in the subproblem minimization is more complex for the method in [8] and it is unclear whether the use of randomized coordinate descent will maintain the convergence rate of this method, as it does for ours. Investigating accelerated version of our approach without restrictive assumptions on the Hessian approximations and with the use of randomized coordinate descent is a subject of future research.

The paper is organized as follows: in Section 2 we describe the algorithmic framework. Then, in Section 3 we present the convergence rate for the exact method using sufficient decrease condition, and address the inexact case in Section 4. We show the convergence analysis for randomized coordinate descent in Section 5. Brief description of the details of our proposed algorithm are in Section 6 and computational results validating the theory are presented in Section 7.

## 2 Basic algorithmic framework and theoretical analysis

The following function is used throughout the paper as an approximation of the objective function $F(x)$.

$$Q(H, u, v) := f(v) + \langle \nabla f(v), u - v \rangle + \frac{1}{2} \langle (u - v), H(u - v) \rangle + g(u). \tag{2.1}$$

For a fixed point $\bar{x}$, the function $Q(H, x, \bar{x})$ serves as an approximation of $F(x)$ around $\bar{x}$. Matrix $H$ controls the quality of this approximation. In particular, if $f(x)$ is smooth and $H = \frac{1}{\mu}I$, then $Q(H, x, \bar{x})$ is a sum of the prox-gradient approximation of $f(x)$ at $\bar{x}$ and $g(x)$. This particular form of $H$ plays a key role in the design and analysis of proximal gradient methods (e.g., see [2]) and alternating direction augmented Lagrangian methods (e.g, see [20]). If $H = \nabla^2 f(\bar{x})$, then $Q(H, x, \bar{x})$ is a second order approximation of $F(x)$ [12, 22]. In this paper we assume that $H$ is a positive definite matrix such that $MI \succeq H \succeq \sigma I$ for some positive constants $M$ and $\sigma$.

Minimizing the function $Q(H, u, v)$ over $u$ reduces to solving problem (1.2). We will use the following notation to denote the accurate and approximate solutions of (1.2).

$$p_H(v) := \arg\min_u Q(H, u, v), \tag{2.2}$$

and

$$p_{H,\phi}(v) \text{ is a vector such that : } \begin{array}{l} Q(H, p_{H,\phi}(v), v) \leq Q(H, v, v) = F(v), \text{ and} \\ Q(H, p_{H,\phi}(v), v) \leq Q(H, p_H(v), v) + \phi. \end{array} \tag{2.3}$$

The method that we consider in this paper computes iterates by (approximately) optimizing $Q(H, u, v)$ with respect to $u$ using some particular $H$ which is chosen at each iteration. The basic algorithm is described in Algorithms 1 and 2.

**Algorithm 1:** Proximal Quasi-Newton method

Choose $0 < \rho \leq 1$ and $x^0$;
**for** $k = 0, 1, 2, \cdots$ **do**
  Choose $0 < \bar{\mu}_k, \phi_k > 0, G_k \succeq 0$;
  Find $H_k = G_k + \frac{1}{2\mu_k}I$ and $x^{k+1} := p_{H_k}(x^k)$
  by applying *Prox Parameter Update* $(\bar{\mu}_k, G_k, x^k, \rho)$;

**Algorithm 2:** Prox Parameter Update $(\bar{\mu}, G, x, \rho)$

Select $0 < \beta < 1$ and set $\mu = \bar{\mu}$;
**for** $i = 1, 2, \cdots$ **do**
  Define $H = G + \frac{1}{2\mu}I$ and compute $p(x) := p_H(x)$;
  If $F(p(x)) - F(x) \leq \rho(Q(H, p(x), x) - F(x))$, then output $H$ and $p(x)$, **Exit** ;
  Else $\mu = \beta^i \bar{\mu}$;

Algorithm 2 chooses Hessian approximations of the form $H_k = \frac{1}{\mu_k}I + G_k$. However, it is possible to consider any procedure of choosing positive definite $H_k$ which ensures $MI \succeq H_k \succeq \sigma I$ and $F(p_{H_k}(x)) - F(x) \leq \rho(Q(H_k, p_{H_k}(x), x) - F(x))$, for a given $0 < \rho \leq 1$, - a step acceptance condition which is a relaxation of conditions used in [2] and [23].

An inexact version of Algorithm 1 is obtained by simply replacing $p_{H_k}$ by $p_{H_k,\phi_k}$ in both Algorithms 1 and 2 for some sequence of $\phi_k$ values.

## 3 Sufficient decrease condition and convergence rate

In the next three sections we present the analysis of convergence rate of Algorithm 1. Recall that we assume that $f(x)$ is convex and smooth, in other words $\|\nabla f(x) - \nabla f(y)\| \leq L(f)\|x - y\|$

for all $x$ and $y$ in the domain of interest, while $g(x)$ is simply convex. In Section 5 we assume that $g(x) = \lambda\|x\|_1$. Note that we do not assume that $f(x)$ is strongly convex or that it is twice continuously differentiable, because we do not rely on any accurate second order information in our framework. We only assume that the Hessian approximations are positive definite and bounded, but their accuracy can be arbitrary, as long as sufficient decrease condition holds. Hence we only achieve sublinear rate of convergence. To achieve higher local rates of convergence stronger assumptions on $f(x)$ and on the Hessian approximations have to be made, see for instance, [4] and [12] for related local convergence analysis.

First we present a helpful lemma which is a simple extension of Lemma 2 in [23] to the case of general positive definite Hessian estimate. This lemma establishes some simple properties of an $\epsilon$-optimal solution to the proximal problem (1.2). It uses the concept of the $\epsilon$-subdifferential of a convex function $a$ at $x$, $\partial_\epsilon a(x)$, which is defined as the set of vectors $y$ such that $a(x) - y^T x \leq a(t) - y^T t + \epsilon$ for all $t$.

**Lemma 1** *Given $\epsilon > 0$, a p.d. matrix $H$ and $v \in \mathbb{R}^n$, let $p_\epsilon(v)$ denote the $\epsilon$-optimal solution to the proximal problem (1.2) in the sense that*

$$g(p_\epsilon(v)) + \frac{1}{2}\|p_\epsilon(v) - z\|_H^2 \leq \epsilon + \min_{x\in\mathbb{R}^n}\left\{g(x) + \frac{1}{2}\|x - z\|_H^2\right\}, \tag{3.1}$$

*where $z = v - H^{-1}\nabla f(v)$. Then there exists $\eta$ such that $\frac{1}{2}\|\eta\|_{H^{-1}}^2 \leq \epsilon$ and*

$$H(v - p_\epsilon(v)) - \eta \in \partial_\epsilon g(p_\epsilon(v)). \tag{3.2}$$

*Proof.* (3.1) indicates that $p_\epsilon(v)$ is an $\epsilon$-minimizer of the convex function $a(x) := \frac{1}{2}\|x - z\|_H^2 + g(x)$. If we let $a_1(x) = \frac{1}{2}\|x - z\|_H^2$ and $a_2(x) = g(x)$, then this is equivalent to

$$0 \subset \partial_\epsilon a(p_\epsilon(v)) \subset \partial_\epsilon a_1(p_\epsilon(v)) + \partial_\epsilon a_2(p_\epsilon(v)). \tag{3.3}$$

Hence,

$$\begin{aligned}\partial_\epsilon a_1(p_\epsilon(v)) &= \left\{y \in \mathbb{R}^n \mid \frac{1}{2}\|y + H(z - p_\epsilon(v))\|_{H^{-1}}^2 \leq \epsilon\right\} \\ &= \left\{y \in \mathbb{R}^n,\ y = \eta - H(z - p_\epsilon(v)) \mid \frac{1}{2}\|\eta\|_{H^{-1}}^2 \leq \epsilon\right\}.\end{aligned}$$

From (3.3) we have

$$H(z - p_\epsilon(v)) - \eta \in \partial_\epsilon g(p_\epsilon(v)) \text{ with } \frac{1}{2}\|\eta\|_{H^{-1}}^2 \leq \epsilon. \tag{3.4}$$

Then (3.2) follows using $z = v - H^{-1}\nabla f(v)$. □

The following lemma, is a generalization of Lemma 2.3 in [2] and of a similar lemma in [23]. This lemma serves to provide a bound on the change in the objective function $F(x)$.

**Lemma 2** *Given $\epsilon$, $\phi$ and $H$ such that*

$$\begin{aligned}F(p_\phi(v)) &\leq Q(H, p_\phi(v), v) + \epsilon \\ Q(H, p_\phi(v), v) &\leq \min_{x\in\mathbb{R}^n} Q(H, x, v) + \phi,\end{aligned} \tag{3.5}$$

*where $p_\phi(v)$ is the $\phi$-approximate minimizer of $Q(H, x, v)$, then for any $u$ and $\eta$ such that $\frac{1}{2}\|\eta\|_{H^{-1}}^2 \leq \phi$*

$$2(F(u) - F(p_\phi(v))) \geq \|p_\phi(v) - u\|_H^2 - \|v - u\|_H^2 - 2\epsilon - 2\phi - 2\langle\eta, u - p_\phi(v)\rangle.$$

*Proof.* The proof closely follows that in [2]. Recall that $p_\phi(v)$ denotes $p_{H,\phi}(v)$. From (3.5) and (2.1), we have

$$\begin{aligned} F(u) - F(p_\phi(v)) &\geq F(u) - Q(H, p_\phi(v), v) - \epsilon \\ &= F(u) - (f(v) + g(p_\phi(v)) + \langle \nabla f(v), p_\phi(v) - v \rangle \\ &+ \tfrac{1}{2}\|p_\phi(v) - v\|_H^2) - \epsilon. \end{aligned} \tag{3.6}$$

Also

$$g(u) \geq g(p_\phi(v)) + \langle u - p_\phi(v), \gamma_g(p_\phi(v)) \rangle - \phi \tag{3.7}$$

by the definition of $\phi$-subgradient, and

$$f(u) \geq f(v) + \langle u - v, \nabla f(v) \rangle, \tag{3.8}$$

due to the convexity of $f$. Here $\gamma_g(\cdot)$ is any subgradient of $g(\cdot)$ and $\gamma_g(p_\phi(v))$ is an $\phi$-subgradient, which satisfies the first-order optimality conditions for $\phi$-approximate minimizer from Lemma 1, i.e.,

$$\gamma_g(p_\phi(v)) = H(v - p_\phi(v)) - \nabla f(v) - \eta, \text{ with } \frac{1}{2}\|\eta\|_{H^{-1}}^2 \leq \phi. \tag{3.9}$$

Summing (3.7) and (3.8) yields

$$F(u) \geq g(p_\phi(v)) + \langle u - p_\phi(v), \gamma_g(p_\phi(v)) \rangle - \phi + f(v) + \langle u - v, \nabla f(v) \rangle. \tag{3.10}$$

Therefore, from (3.6), (3.9) and (3.10) it follows that

$$\begin{aligned} F(u) - F(p_\phi(v)) &\geq \langle \nabla f(v) + \gamma_g(p_\phi(v)), u - p_\phi(v) \rangle - \frac{1}{2}\|p_\phi(v) - v\|_H^2 - \epsilon - \phi \\ &= \langle -H(p_\phi(v) - v) - \eta, u - p_\phi(v) \rangle - \frac{1}{2}\|p_\phi(v) - v\|_H^2 - \epsilon - \phi \\ &= \frac{1}{2}\|p_\phi(v) - u\|_H^2 - \frac{1}{2}\|v - u\|_H^2 - \epsilon - \phi - \langle \eta, u - p_\phi(v) \rangle. \end{aligned}$$

□

Note that if $\phi = 0$, that is the subproblems are solved accurately, then we have $2(F(u) - F(p(v))) \geq \|p(v) - u\|_H^2 - \|v - u\|_H^2 - 2\epsilon$.

If the sufficient decrease condition

$$(F(x^{k+1}) - F(x^k)) \leq \rho(Q(H_k, x^{k+1}, x^k) - F(x^k)) \tag{3.11}$$

is satisfied, then

$$\begin{aligned} F(x^{k+1}) &\leq Q(H_k, x^{k+1}, x^k) - (1 - \rho)\left(Q(H_k, x^{k+1}, x^k) - F(x^k)\right) \\ &\leq Q(H_k, x^{k+1}, x^k) - \frac{1 - \rho}{\rho}(F(x^{k+1}) - F(x^k)) \end{aligned}$$

and Lemma 2 holds at each iteration $k$ of Algorithm 1 with $\epsilon_k = -\frac{1-\rho}{\rho}(F(x^{k+1}) - F(x^k))$.

ISTA [2] is a particular case of Algorithm 1 with $G_k = 0$, for all $k$, and $\rho = 1$. In this case, the value of $\mu_k$ is chosen so that the conditions of Lemma 2 hold with $\epsilon = 0$. In other words, the

reduction achieved in the objective function $F(x)$ is at least the amount of reduction achieved in the model $Q(\mu_k, p(x^k), x^k)$. It is well known that as long as $\mu_k \leq 1/L(f)$ (recall that $L(f)$ is the Lipschitz constant of the gradient) then condition (3.11) holds with $\rho = 1$. Relaxing condition (3.11) by using $\rho < 1$ allows us to accept larger values of $\mu_k$, which in turn implies larger steps taken by the algorithm. This basic idea is the cornerstone of step size selection in most nonlinear optimization algorithms. Instead of insisting on achieving "full" predicted reduction of the objective function (even when possible), a fraction of this reduction is usually sufficient. In our experiments small values of $\rho$ provided much better performance than values close to 1.

We now derive the complexity bound for the exact version of Algorithm 1. Here are the assumptions made in our analysis.

**Assumptions 1**

*(i) Denote $X^*$ as the set of optimal solutions of (1.1) and $x^*$ as any element of that set.*

*(ii) There exists positive constants $M_k$ and $\sigma$ such that at the k-th iteration of Algorithm 1:*

$$\sigma I \preceq H_k \preceq M_k I \preceq MI$$

**Theorem 3** *Let Assumptions 1 hold. In Algorithm 1, suppose that at iteration k, $H_k$, is chosen so that the sufficient decrease condition (3.11) holds. Then the iterates $\{x^k\}$ in Algorithm 1 satisfy*

(3.12)
$$F(x^k) - F(x^*) \leq \frac{1}{2k_m}\left(\|x^0 - x^*\|^2_{H_0/M_0} + \sum_{i=0}^{k-1} \|x^{i+1} - x^*\|^2_{H_{i+1}/M_{i+1} - H_i/M_i} + \frac{2(1-\rho)(F(x^0) - F(x^*))}{\rho\sigma}\right).$$

*where $k_m = \sum_{i=0}^{k-1} \frac{1}{M_i}$. Thus, the sequence $\{F(x^k)\}$ produced by Algorithm 1 converges to $F(x^*)$ if*

$$\frac{\sum_{i=0}^{k-1} \|x^{i+1} - x^*\|^2_{H_{i+1}/M_{i+1} - H_i/M_i}}{k_m} \to 0 \text{ as } k \to \infty.$$

*Proof.* As is done in [2] and [23], we apply Lemma 2, sequentially, with $\phi = 0$, $u = x^*$ and $p_\phi(v) = x^i$ for $i = 0, \ldots, k-1$.

$$F(x^{i+1}) - F(x^*) \leq \frac{1}{2}\left(\|x^i - x^*\|^2_{H_i} - \|x^{i+1} - x^*\|^2_{H_i}\right) + \epsilon_i \tag{3.13}$$

Dividing both sides of (3.13) by $M_i$ and adding up resulting inequalities, we obtain

$$\begin{aligned}
&\sum_{i=0}^{k-1} \frac{F(x^{i+1}) - F(x^*)}{M_i} \leq \frac{1}{2}\sum_{i=0}^{k-1}\left(\|x^i - x^*\|^2_{H_i/M_i} - \|x^{i+1} - x^*\|^2_{H_i/M_i}\right) + \sum_{i=0}^{k-1}\frac{\epsilon_i}{M_i} \\
&\leq \frac{1}{2}\sum_{i=0}^{k-1}\left(\|x^i - x^*\|^2_{H_i/M_i} - \|x^{i+1} - x^*\|^2_{H_i/M_i}\right) + \sum_{i=0}^{k-1}\frac{\epsilon_i}{\sigma} \\
&= \frac{1}{2}\left(\|x^0 - x^*\|^2_{H_0/M_0} + \sum_{i=0}^{k-1}\|x^{i+1} - x^*\|^2_{H_{i+1}/M_{i+1} - H_i/M_i} - \|x^k - x^*\|^2_{H_k/M_k}\right) + \frac{1}{\sigma}\sum_{i=0}^{k-1}\epsilon_i \\
&\text{use the already established bound } \textstyle\sum_{i=0}^{k-1}\epsilon_i \leq \frac{(1-\rho)(F(x^0) - F(x^*))}{\rho} \\
&\leq \frac{1}{2}\left(\|x^0 - x^*\|^2_{H_0/M_0} + \sum_{i=0}^{k-1}\|x^{i+1} - x^*\|^2_{H_{i+1}/M_{i+1} - H_i/M_i}\right) + \frac{(1-\rho)(F(x^0) - F(x^*))}{\rho\sigma}.
\end{aligned}$$

Letting $k_m = \sum_{i=0}^{k-1} \frac{1}{M_i}$, we have

$$F(x^k) - F(x^*) \leq \frac{1}{k_m}(\sum_{i=0}^{k-1} \frac{F(x^{i+1}) - F(x^*)}{M_i})$$
$$\leq \frac{1}{2k_m}\left(\|x^0 - x^*\|^2_{H_0/M_0} + \sum_{i=0}^{k-1} \|x^{i+1} - x^*\|^2_{H_{i+1}/M_{i+1} - H_i/M_i} + \frac{2(1-\rho)(F(x^0) - F(x^*))}{\rho\sigma}\right).$$

□

# 4 Analysis of inexact proximal Quasi-Newton method

We now follow the theory proposed in [23] to analyze Algorithm 1 in the case when the computation of $p_H(v)$ is performed inexactly. In other words, we consider the version of Algorithm 1 (and 2) where we compute $x^{k+1} := p_{H_k,\phi_k}(x^k)$.

**Assumptions 2**

*(i) There exists a positive constant $\delta$ such that for all iterates $\{x^k\}$ of Algorithm 1:*

$$\max_{x^* \in X^*} \|x^k - x^*\| \leq \delta$$

We note that while it is possible to remove Assumption 2($i$) by directly establishing a uniform bound on $\|x^k - x^*\|$ (rf. [25, 23]), here we instead make it an explicit assumption, mainly for simplicity of the presentation. This allows us to provide a much shorter and more concise proof than the one given in [25]. Note also that all iterates $\{x^k\}$ fall into the level set $\{x \in \mathbb{R}^n : F(x) \leq F(x^0)\}$, due to the sufficient decrease condition (3.11) that demands a monotonic decrease on the objective values. Assumption 2($i$) thus follows straightforwardly from the fact that the level set is bounded, which often holds true in real-world problems.

We can now state the general convergence rate result for the inexact version of Algorithm 1.

**Theorem 4** *Suppose that Assumptions 1 and 2 hold. Assume that all iterates $\{x^k\}$ of inexact Algorithm 1 are generated with some $\phi_k > 0$ (cf. (3.5)), then*

$$F(x^k) - F(x^*) \leq \frac{1}{k_m}\left(\frac{1}{2}\|x^0 - x^*\|^2_{\frac{H_0}{M_0}} + \frac{(1-\rho)(F(x^0) - F(x^*))}{\rho\sigma} + \sum_{i=0}^{k-1} \frac{\phi_i}{M_i} + \delta \sum_{i=0}^{k-1} \sqrt{\frac{2\phi_i}{M_i}}\right), \tag{4.1}$$

*Proof.* As is done in Theorem 3, we apply Lemma 2, sequentially, with $u = x^*$, $p_\phi(v) = x^i$ and nonzero subproblem minimization residual $\phi_i$ for $i = 0, \ldots, k-1$. Adding up resulting inequalities,

we obtain

(4.2)
$$\sum_{i=0}^{k-1}\frac{F(x^{i+1})-F(x^*)}{M_i}\leq\frac{1}{2}\sum_{i=0}^{k-1}\left(\|x^i-x^*\|^2_{H_i/M_i}-\|x^{i+1}-x^*\|^2_{H_i/M_i}\right)+\sum_{i=0}^{k-1}\frac{\epsilon_i}{M_i}+\sum_{i=0}^{k-1}\frac{\phi_i}{M_i}+\sum_{i=0}^{k-1}\frac{\langle\eta_i,x^*-x^{i+1}\rangle}{M_i}$$

(4.3)
$$=\frac{1}{2}\left(\|x^0-x^*\|^2_{\frac{H_0}{M_0}}+\sum_{i=0}^{k-1}\|x^{i+1}-x^*\|^2_{\frac{H_{i+1}}{M_{i+1}}-\frac{H_i}{M_i}}-\|x^k-x^*\|^2_{\frac{H_k}{M_k}}\right)+\sum_{i=0}^{k-1}\frac{\epsilon_i}{M_i}+\sum_{i=0}^{k-1}\frac{\phi_i}{M_i}+\sum_{i=0}^{k-1}\frac{\langle\eta_i,x^*-x^{i+1}\rangle}{M_i}$$

from the assumptions that $H_{i+1}\preceq H_i$ and that $\|x^i-x^*\|\leq\delta,\forall i$

$$\leq\frac{1}{2}\|x^0-x^*\|^2_{\frac{H_0}{M_0}}+\sum_{i=0}^{k-1}\frac{\epsilon_i}{M_i}+\sum_{i=0}^{k-1}\frac{\phi_i}{M_i}+\delta\sum_{i=0}^{k-1}\frac{\|\eta_i\|}{M_i}$$

note that $\|\eta_i\|\leq\sqrt{2M_i\phi_i}$, and use the already established bound $\sum_{i=0}^{k-1}\epsilon_i\leq\frac{(1-\rho)(F(x^0)-F(x^*))}{\rho}$

$$\leq\frac{1}{2}\|x^0-x^*\|^2_{\frac{H_0}{M_0}}+\frac{(1-\rho)(F(x^0)-F(x^*))}{\rho\sigma}+\sum_{i=0}^{k-1}\frac{\phi_i}{M_i}+\delta\sum_{i=0}^{k-1}\sqrt{\frac{2\phi_i}{M_i}}.$$

Hence,

$$\begin{aligned}F(x^k)-F(x^*)&\leq\frac{1}{k_m}\sum_{i=0}^{k-1}\frac{F(x^{i+1})-F(x^*)}{M_i}\\&\leq\frac{1}{k_m}\left(\frac{1}{2}\|x^0-x^*\|^2_{\frac{H_0}{M_0}}+\frac{(1-\rho)(F(x^0)-F(x^*))}{\rho\sigma}+\sum_{i=0}^{k-1}\frac{\phi_i}{M_i}+\delta\sum_{i=0}^{k-1}\sqrt{\frac{2\phi_i}{M_i}}\right).\end{aligned}$$

□

As in [23] it follows that the inexact version of Algorithm 1 converges to the optimal value if $(\sum_{i=0}^{k-1}\sqrt{\phi_i})/k\to 0$ and it has sublinear convergence rate if $\sum_{i=0}^{\infty}\sqrt{\phi_i}$ is bounded. To ensure such a bound, it is sufficient to require that $\phi_i\leq\alpha^i$, for some $\alpha\in(0,1)$ for all $i$. We are hence interested in practical approaches to subproblem optimization which can be terminated upon reducing the gap in the function value of the subproblem to $\alpha^i$, where $0<\alpha<1$ is a fixed number and $i=1,2,\ldots$ is the index of the corresponding iteration of Algorithm 1. The question now is: what method and what stopping criterion should we use for subproblem optimization, so that sufficient accuracy is achieved and no excessive computations are performed, in other word, how can we guarantee the bound on $\phi_i$, while maintaining efficiency of the subproblem optimization? It is possible to consider terminating the optimization of the $i$-th subproblem once the duality gap is smaller than $\alpha^i$. However, checking duality gap can be computationally very expensive. Alternatively one can use an algorithm with a known convergence rate. This way it can be determined apriori how many iterations of such an algorithm should be applied to the $i$-th subproblem to achieve $\alpha^i$ accuracy. In particular, we note that the objective functions in our subproblems are all $\sigma$-strongly convex, so a simple proximal gradient method, or some of its accelerated versions, will enjoy linear convergence rates when applied to these subproblems. Hence, after $i$ iterations of optimizing $Q_i$, such a method will achieve accuracy $\alpha^i$, for some fixed $\alpha\in(0,1)$. Under this trivial termination criterion, the subproblem is solved more and more accurately as the outer iteration approaches optimality, which is a common condition required in classic inexact Newton method analysis [6]. Note that the same property holds for any linearly convergent method, such as the proximal gradient or a semi-smooth Newton method, as discussed above.

As we pointed out in the introduction, the most efficient practical approach to subproblem optimization, in the cases when $g(x) = \lambda\|x\|_1$, seems to be the coordinate descent method. One iteration of a coordinate descent step can be a lot less expensive than that of a proximal gradient or a Newton method. In particular, if matrix $H$ is constructed via the LBFGS approach, then one step of a coordinate decent takes a constant number of operations, $m$ (the memory size of LBFGS, which is typically 10-20). On the other hand, one step of proximal gradient takes $O(mn)$ operations and Newton method takes $O(nm^2)$.

Unfortunately, cyclic (Gauss-Seidel) coordinate descent, does not have deterministic complexity bounds, hence it is not possible to know when the work on a particular subproblem can be terminated to guarantee the desired (i.e., $\alpha^i$) level of accuracy. However, a randomized coordinate descent has probabilistic complexity bounds, which can be used to demonstrate the linear rate of convergence in expectation. Hence we will terminate the randomized coordinate descent after it performs a number of iterations which is a linear function of $i$ (it is possible to simply use $i$ for the number of iteration, but other linear functions of $i$ are more practical, as we will discuss below).

In the next section we bring practice and theory together by showing that randomized coordinate descent can guarantee sufficient accuracy for subproblem solutions and hence maintain the sub linear convergence rate (in expectation) of Algorithm 1. Moreover, we show in Section 7, that the randomized coordinate descent is as efficient in practice as the cyclic one.

# 5 Analysis of Subproblem Optimization via Randomized Coordinate Descent

Here we propose a simple termination rule for the subproblem optimization phase, when using randomized coordinate descent which ensures overall convergence rates shown in the previous section. In randomized coordinate descent the model function $Q(\cdot)$ is iteratively minimized over one randomly chosen coordinate, while the others remain fixed. The method is presented in Algorithm 3 and is applied for $l$ steps, with $l$ being an input parameter.

**Algorithm 3:** Randomized Coordinate Descent for optimizing Model Function $Q(H, v, x)$ over $v$: $RCD\ (Q(H, v, x), x, l)$

Set $p(x) \leftarrow x$;
**for** $i = 1, 2, \cdots, l$ **do**
  Choose $j \in \{1, 2, ..., n\}$ with probability $\frac{1}{n}$;
  $z^* = \arg\min_z Q(H, p(x) + ze_j, x)$;
  $p(x) \leftarrow p(x) + z^* e_j$;
Return $p(x)$.

Our analysis is based on Richtarik and Takac's results on iteration complexity of randomized coordinate descent [19]. In particular, we make use of Theorem 7 in [19], which we restate below without proof, while adapting it to our context.

**Lemma 5** *Let $v$ be the initial point and $Q^* := \min_{u\in\mathbb{R}^n} Q(H, u, v)$. If $v_i$ is the random point generated by applying $i$ randomized coordinate descent steps to a strongly convex function $Q$, then for some constant $0 < \alpha < 1$ (dependent only on $n$ and $\sigma$ - the bound on the smallest eigenvalue of $H$) we have*

$$E[Q(H, v_i, v) - Q^*] \leq \alpha^i (Q(H, v, v) - Q^*). \tag{5.1}$$

Next, we establish a bound on the maximal possible reduction of the model function $Q(\cdot)$ objective function value, for any positive definite matrix $H \succ 0$, and fixed point $v \in \mathbb{R}^n$.

**Lemma 6** *Assume that for any $v \in \mathbb{R}^n$ such that $F(v) \leq F(x^0)$, all of the the subgradients of $F$ at $v$ are bounded in norm by a constant $\kappa$, i.e. $\|\nabla f(v)+\gamma\| \leq \kappa$ for all $\gamma \in \partial g(v)$. Then the maximum function reduction for $Q(\cdot)$ is uniformly bounded from above by*

$$Q(H, v, v) - Q^* \leq R, \text{ with } R = \tfrac{M\kappa^2}{2\sigma^2}, \tag{5.2}$$

*where $M$ and $\sigma$ are respectively the bounds on the largest and smallest eigenvalues of $H$ and $Q^* := \min_{u\in\mathbb{R}^n} Q(H, u, v)$.*

*Proof. Let $v^* = \arg\min_{u\in\mathbb{R}^n} Q(H, u, v)$ and let $\gamma_g(v^*)$ be any subgradient of $g(\cdot)$ at $v^*$. From the first-order optimality conditions*

$$H(v^* - v) + \nabla f(v) + \gamma_g = 0, \tag{5.3}$$

*we can obtain an upper bound on $\|v^* - v\|$*

$$\|v^* - v\| = \|H^{-1}\| \cdot \|\nabla f(v) + \gamma_g\| \leq \kappa/\sigma. \tag{5.4}$$

*Now, we bound the reduction in the objective function in terms of $||v^* - v||$. From the convexity of $g$,*

$$g(v) - g(v^*) \leq \langle \gamma_g, v - v^* \rangle. \tag{5.5}$$

*Multiplying (5.3) with $v^* - v$, we obtain*

$$-\langle \nabla f(v), v^* - v\rangle = \|v^* - v\|_H^2 + \langle \gamma_g, v^* - v\rangle. \tag{5.6}$$

*From (5.6), (5.3), (5.5) and the definition of $Q$, we have*

$$\begin{aligned} Q(H, v, v) - Q^* &= g(v) - g(v^*) - \langle \nabla f(v), v^* - v\rangle - \frac{1}{2}\|v^* - v\|_H^2 \\ &\leq \langle \gamma_g, v - v^*\rangle + \|v^* - v\|_H^2 + \langle \gamma_g, v^* - v\rangle - \frac{1}{2}\|v^* - v\|_H^2 \\ &= \frac{1}{2}\|v^* - v\|_H^2 \leq \frac{M\kappa^2}{2\sigma^2}, \end{aligned}$$

*which concludes the proof of the lemma.* □

It follows immediately from Lemma 6 that the subproblem optimization error $\phi_i$ is also bounded for all $i$, say by $\Phi$, and that $\Phi \leq R$. Using this bound we now present the auxiliary result that derives the bounds on separate terms that appear on the right hand side of (4.1) and involve $\phi_i$.

**Lemma 7** *Assume that $\phi_i$ are nonnegative bounded independent random variables whose value lies in an interval $[0, \Phi]$. Assume also that $E[\phi_i] \leq \bar{R}\bar{\alpha}^i$ for all $i$, and some constants $0 < \bar{\alpha} < 1$ and $\bar{R} > 0$. Then the following inequalities hold*

$$E[\sum_{i=1}^{k} \sqrt{\phi_i}] \leq \frac{\sqrt{\bar{R}\bar{\alpha}}}{1 - \sqrt{\bar{\alpha}}}, \quad E[\sum_{i=1}^{k} \phi_i] \leq \frac{\bar{R}}{1 - \bar{\alpha}} \tag{5.7}$$

*Proof.* First we note that

$$E[\sqrt{\phi_i \phi_j}] = E[\sqrt{\phi_i}]E[\sqrt{\phi_j}], \tag{5.8}$$

due to independence of $\phi_i$'s, and

$$E[\sqrt{\phi_i}] \leq \sqrt{E[\phi_i]}, \tag{5.9}$$

due to Jensen inequality and the fact that the square root function is concave and $\phi_i \geq 0$. Then, given (5.8) and (5.9), we derive (5.7) using a bound on $E[\phi_i]$.

$$E[\sum_{i=1}^{k} \sqrt{\phi_i}] = \sum_{i=1}^{k} E[\sqrt{\phi_i}] \leq \sum_{i=1}^{k} \sqrt{E[\phi_i]} \leq \sum_{i=1}^{k} \sqrt{\bar{R}}\bar{\alpha}^{i/2} \leq \frac{\sqrt{\bar{R}\bar{\alpha}}}{1-\sqrt{\bar{\alpha}}}.$$

And the second bound on $E[\sum_{i=1}^{k} \phi_i]$ can be established in a similar fashion. □

## 5.1 Complexity of inexact proximal Quasi-Newton method, based on randomized coordinate descent

. The key result in this section is showing that, to provide convergence rates developed in the previous sections, the number of the coordinate descent steps should increase as a linear function of the index of the outer iteration. Here, we assume that some linear function of $k$, $l(k) = ak + b$ is chosen to indicate the number of coordinate steps for the $k$-th subproblem. In our computational experiments we use a step function which adds $n_k$ coordinate decent steps to subproblem optimization after each $m$ iterations, where $m$ is the size of L-BFGS memory and $n_k$ is the size of the working set at the $k$-th iteration - in other words it is the number of variables in the $k$-th subproblem. While this function is more complicated than a simple linear function of $k$ it can always be bounded from below by such a linear function, hence for simplicity of analysis we consider only simple linear functions $l(k)$.

Let us now state the version of Algorithms 1 and 2 which combines the ideas from the previous sections and for which we will be able to derive complexity in expectation.

**Algorithm 4:** Proximal Quasi-Newton method using randomized coordinate descent

1 Choose $0 < \rho \leq 1$, $a, b > 0$ and $x^0$;
2 **for** $k = 0, 1, 2, \cdots$ **do**
3 Choose $0 < \bar{\mu}_k, \theta_k > 0, G_k \succeq 0$.;
4 Find $H_k = G_k + \frac{1}{2\mu_k} I$ and $x^{k+1} := p_{H_k,\phi_k}(x^k)$
5 by applying *Prox Parameter Update with RCD* $(\bar{\mu}_k, G_k, x^k, \rho, a, b)$.;

Note that $\phi_k$ and $\phi$ used in in the $p_{H,\phi_k}(x)$ and $p_{H,\phi}(x)$ in Algorithms 4 and 5 are used to indicate that the inexact subproblem optimization is performed. The algorithm does not select the values of $\phi_k$, the value of each $\phi_k$ is a consequence of applying $ak + b$ RCD iterations to the subproblem. By Lemma 5 we know that there exists a constant $\alpha \in (0, 1)$ such that, if $ak + b$ iterations of RCD are applied for the $k$-th subproblem, then $E(\phi_k) \leq R\alpha^{ak+b} = \bar{R}\bar{\alpha}^k$, for $\bar{R} = R\alpha^b$ and $\bar{\alpha} = \alpha^a$. Applying now Lemma 7 we can bound each element in (4.1) and derive the following theorem.

**Algorithm 5:** Prox Parameter Update with RCD $(\bar{\mu}, G, x, \rho, a, b)$

Select $0 < \beta < 1$ and set $\mu = \bar{\mu}$;
**for** $i = 1, 2, \cdots$ **do**
  Define $H = G + \frac{1}{2\mu} I$, and compute $p(x) := p_{H,\phi}(x)$
  by applying $RCD$ $(Q(H, v, x), x, \lceil ak + b \rceil)$;
  If $F(p(x)) - F(x) \leq \rho(Q(H, p(x), x) - F(x))$, then output $H$ and $p(x)$, **Exit** ;
  Else $\mu = \beta^i \bar{\mu}$;

**Theorem 8** *Assume that for each iterate $\{x^k\}$ of Algorithm 4 the following iterate $\{x^{k+1}\}$ is generated by applying $ak + b$ steps of Algorithm 3, then*

$$E[F(x^k)] - F(x^*) \leq \frac{\zeta}{k},$$

*where $x^*$ is an optimal solution of (1.1) and $\zeta$ is a constant.*

*Proof.* Taking expectation on both sides of (4.1) in Theorem 4, we have

$$E[F(x^k)] - F(x^*) \leq \frac{1}{k}\left(\frac{\sigma}{2}E[B_k] + \frac{\sigma}{2}C + 2E[A_k^2] + \sqrt{C}E[A_k] + E[\sqrt{B_k}A_k]\right). \tag{5.10}$$

Next, we show how to bound $E[B_k], E[A_k], E[A_k^2]$ and $E[\sqrt{B_k}A_k]$. We first note a key observation, that, as a result of Lemma 5, Lemma 6 and Algorithm 3, the expectation of subproblem optimization error $\phi_i$ can be upper bounded by a geometric progression such that

$$E[\phi_i] \leq R\alpha^{ai+b} = \bar{R}\bar{\alpha}^i,$$

where $0 < \alpha < 1$ and $R$ are specified respectively in Lemma 5 and Lemma 6 and $\bar{\alpha} = \alpha^a$ and $\bar{R} = R\alpha^b$. It immediately follows that

$$E[B_k] = \frac{2(M_\sigma + 1)}{\sigma}\sum_{i=0}^{k-1} E[\phi_i] \leq \frac{2(M_\sigma + 1)}{\sigma}\sum_{i=0}^{k-1}\bar{R}\bar{\alpha} \leq \frac{2(M_\sigma + 1)\bar{R}}{\sigma}\frac{1}{1-\bar{\alpha}}.$$

Recalling Lemma 7, we then obtain

$$\begin{aligned}
E[A_k] &= \sqrt{2M}E[\sum_{i=0}^{k-1}\sqrt{\phi_i}] \leq \frac{\sqrt{2M\bar{R}}}{1-\sqrt{\bar{\alpha}}} \\
E[A_k^2] &= E[(\sum_{i=1}^{k}\sqrt{2M\phi_i})^2] = 2ME[(\sum_{i=0}^{k-1}\sqrt{\phi_i})^2] \leq \frac{4M\bar{R}}{(1-\sqrt{\bar{\alpha}})^2} \\
E[\sqrt{B_k}A_k] &= E[\sqrt{\frac{2(M_\sigma+1)}{\sigma}\sum_{i=1}^{k}\phi_i}\sum_{i=1}^{k}\sqrt{2M\phi_i}] = \sqrt{\frac{4M(M_\sigma+1)}{\sigma}}E[\sqrt{\sum_{i=1}^{k}\phi_i}\sum_{i=1}^{k}\sqrt{\phi_i}] \\
&\leq \sqrt{\frac{4M(M_\sigma+1)}{\sigma}}\frac{\bar{R}}{\sqrt{1-\bar{\alpha}}(1-\sqrt{\bar{\alpha}})},
\end{aligned}$$

and $\zeta$ is equal to

$$\zeta = \frac{\bar{R}(M_\sigma+1)}{1-\bar{\alpha}} + \frac{\sigma}{2}C + \frac{8M\bar{R}}{(1-\sqrt{\bar{\alpha}})^2} + \sqrt{C}\frac{\sqrt{2M\bar{R}}}{1-\sqrt{\bar{\alpha}}} + \sqrt{\frac{4M(M_\sigma+1)}{\sigma}}\frac{\bar{R}}{\sqrt{1-\bar{\alpha}}(1-\sqrt{\bar{\alpha}})}.$$

□

**Remark 9** *Note that the choice of the number of coordinates decent steps in each applications of RCD, in other words, the choice of constants $a$ and $b$ has a direct affect on the constants in the complexity bound. Clearly taking more RCD steps leads to fewer iterations of the main algorithm, as larger values of $a$ and $b$ lead to smaller value of $\zeta$. On the other hand, each iterations becomes more expensive. We believe that our practical approach described in Section 7 strikes a good balance in this trade-off.*

## 6 Optimization Algorithm

In this section we briefly describe the specifics of the general purpose algorithm that we propose within the framework of Algorithms 4, 5 and 3 and that takes advantage of approximate second order information while maintaining low complexity of subproblem optimization steps. The algorithm is designed to solve problems of the form (1.1) with $g(x) = \lambda\|x\|_1$, but it does not use any special structure of the smooth part of the objective, $f(x)$.

At iteration $k$ a step $d_k$ is obtained, approximately, as follows

$$d_k = \arg\min_d \{\nabla f(x^k)^T d + d^T H_k d + \lambda\|x^k + d\|_1;\ \text{s.t. } d_i = 0, \forall i \in \mathcal{A}_k\},$$

with $H_k = G_k + \frac{1}{2\mu_k} I$ - a positive definite matrix and $\mathcal{A}_k$ - a set of coordinates fixed at the current iteration.

The positive definite matrix $G_k$ is computed by a limited memory BFGS approach. In particular, we use a specific form of Hessian estimate, (see e.g. [5, 16]),

$$G_k = \gamma_k I - QRQ^T = \gamma_k I - Q\hat{Q} \quad \text{with } \hat{Q} = RQ^T, \tag{6.1}$$

where $Q$, $\gamma_k$ and $R$ are defined below,

$$Q = \begin{bmatrix}\gamma_k S_k & T_k\end{bmatrix},\ R = \begin{bmatrix}\gamma_k S_k^T S_k & M_k \\ M_k^T & -D_k\end{bmatrix}^{-1},\ \gamma_k = \frac{t_{k-1}^T t_{k-1}}{t_{k-1}^T s_{k-1}}.. \tag{6.2}$$

Note that there is low-rank structure present in $G_k$, the matrix given by $Q\hat{Q}$, which we can exploit, but $G_k$ itself by definition is always positive definite. Let $m$ be a small integer which defines the number of latest BFGS updates that are "remembered" at any given iteration (we used $10 - 20$). Then $S_k$ and $T_k$ are the $p \times m$ matrices with columns defined by vector pairs $\{s_i, t_i\}_{i=k-m}^{k-1}$ that satisfy $s_i^T t_i > 0, s_i = x^{i+1} - x_i$ and $t_i = \nabla f(x^{i+1}) - \nabla f(x^i)$, $M_k$ and $D_k$ are the $k \times k$ matrices

$$(M_k)_{i,j} = \begin{cases} s_{i-1}^T t_{j-1} & \text{if } i > j \\ 0 & \text{otherwise,}\end{cases} \qquad D_k = \text{diag}[s_{k-m}^T t_{k-m}, ..., s_{k-1}^T t_{k-1}].$$

The particular choice of $\gamma_k$ is meant to promote well-scaled quasi-Newton steps, so that less time is spent on line search or updating of prox parameter $\mu_k$ [16]. In fact instead of updating and maintaining $\mu_k$, exactly as described in Algorithm 2 we simply double $\gamma_k$ in (6.1) at each backtracking step. This can be viewed as choosing $\mu_k = \infty$ the first step of backtracking, and $\mu_k = 1/(2^{i-1} - 1)\gamma_k$ for the $i$-th backtracking step, when $i > 1$. As long as $G_K$ in (6.1), is positive definite, with smallest eigenvalue bounded by $\sigma > 0$, our theory applies to this particular backtracking procedure.

### 6.1 Greedy Active-set Selection $\mathcal{A}_k(\mathcal{I}_k)$

An active-set selection strategy maintains a sequence of sets of indices $\mathcal{A}_k$ that iteratively estimates the optimal active set $\mathcal{A}^*$ which contains indices of zero entries in the optimal solution $x^*$ of (1.1). We introduce this strategy as a heuristic aiming to improve the efficiency of the implementation and to make it comparable with state-of-the-art methods, which also use active set strategies. A theoretical analysis of the effects of these strategies is a subject of future study. The complement set of $\mathcal{A}_k$ is $\mathcal{I}_k = \{i \in \mathcal{P} \mid i \notin \mathcal{A}_k\}$. Let $(\partial F(x^k))_i$ be the $i$-th component of a subgradient of $F(x)$ at $x^k$. We define two sets,

$$\mathcal{I}_k^{(1)} = \{i \in \mathcal{P} \mid (\partial F(x^k))_i \neq 0\},\ \mathcal{I}_k^{(2)} = \{i \in \mathcal{P} \mid (x^k)_i \neq 0\}. \tag{6.3}$$

As is done in [31] and [11] we select $\mathcal{I}_k$ to include the entire set $\mathcal{I}_k^{(2)}$ and the entire set $\mathcal{I}_k^{(1)}$. We also tested a strategy which includes only a small subset of indices from $\mathcal{I}_k^{(1)}$ for which the corresponding elements $|(\partial F(x^k))_i|$ are the largest. This strategy resulted in a smaller size of subproblems (6.1) at the early stages of the algorithm, but did not appear to improve the overall performance of the algorithm.

### 6.2 Solving the inner problem via coordinate descent

We apply coordinate descent method to the piecewise quadratic subproblem (6.1) to obtain the direction $d_k$ and exploit the special structure of $H_k$. Suppose $j$-th coordinate in $d$ is updated, hence $d' = d + ze_j$ ($e_j$ is the $j$-th vector of the identity). Then $z$ is obtained by solving the following one-dimensional problem

$$\min_z (H_k)_{jj} z^2 + ((\nabla f(x^k))_j + (2H_k d)_j)z + \lambda |(x^k)_j + d_j + z|,$$

which has a simple closed-form solution [7, 11].

The most costly step of an iteration of the coordinate descent method is computing or maintaining vector $H_k d$. Naively, or in the case of general $H_k$, this step takes $O(n)$ flops, since the vector needs to be updated at the end of each iteration, when one of the coordinates of vector $d$ changes. The special form of $G_k$ in $H_k = G_k + \frac{1}{\mu} I = \gamma_k I - Q\hat{Q}$ provides us an opportunity to accelerate this step, reducing the complexity from problem-dependent $O(n)$ to $O(m)$ with $m$ chosen as a small constant. In particular we only store the diagonal elements of $G_k$, $(G_k)_{ii} = \gamma_k - q_i^T \hat{q}_i$, where $q_i$ is the $i$th row of the matrix $Q$ and $\hat{q}_i$ is the $i$th column vector of the matrix $\hat{Q}$. We compute $(G_k d)_i$, whenever it is needed, by maintaining a $2m$ dimensional vector $v := \hat{Q}d$, which takes $O(2m)$ flops, and using $(G_k d)_i = \gamma_k d_i - q_i^T v$. After each coordinate step $v$ is updated by $v \leftarrow v + z_i \hat{q}_i$, which costs $O(m)$. We also need to use extra memory for caching $\hat{Q}$ and $\hat{d}$ which takes $O(2mp + 2m)$ space. With the other $O(2p + 2mn)$ space for storing the diagonal of $G_k$, $Q$ and $d$, altogether we need $O(4mp + 2n + 2m)$ space, which is essentially $O(4mn)$ when $n \gg m$.

## 7 Computational experiments

The aim of this section is to provide validation for our general purpose algorithm, but not to conduct extensive comparison of various inexact proximal Newton approaches. In particular, we aim to demonstrate a) that using the exact Hessian is not necessary in these methods, b) that backtracking using prox parameter, based on sufficient decrease condition, which our theory uses, does in fact work well in practice and c) that randomized coordinate descent is at least as effective as the cyclic one, which is standardly used by other methods.

LHAC, for **L**ow rank **H**essian **A**pproximation in **A**ctive-set **C**oordinate descent, is a C/C++ package that implements Algorithms 3- 4 for solving general $\ell_1$ regularization problems. We conduct experiments on two of the most well-known $\ell_1$ regularized models – Sparse Inverse Covariance Selection (SICS) and Sparse Logistic Regression (SLR). The following two specialized C/C++ solvers are included in our comparisons:

- QUIC: the quadratic inverse covariance algorithm for solving SICS described in [11].
- LIBLINEAR: an improved version of GLMNET for solving SLR described in [10, 31].

Note that both of these packages have been shown to be the state-of-the-art solvers in their respective categories (see e.g. [31, 30, 11, 17]).

Both QUIC and LIBLINEAR adopt line search to ensure function reduction. We have implemented line search in LHAC as well to see how it compares to the updating of prox parameter proposed in Algorithm 2. In all the experiments presented below use the following notation.

- LHAC: Algorithm 4 with backtracking on prox parameter.
- LHAC-L: Algorithm 4 with Armijo line search procedure described below in (7.4).

### 7.1 Experimental Settings

For all of the experiments we choose the initial point $x_0 = \mathbf{0}$, and we report running time results in seconds, plotted against log-scale relative objective function decrease given by

$$\log(\frac{F(x) - F^*}{F^*}), \tag{7.1}$$

where $F^*$ is the optimal function value. Since $F^*$ is not available, we compute an approximation by setting a small optimality tolerance, specifically $10^{-7}$, in QUIC and LIBLINEAR. All the experiments are executed through the MATLAB mex interface. We also modify the source code of LIBLINEAR in both its optimization routine and mex gateway function to obtain the records of function values and the running time. We note that we simply store, in a double array, and pass the function values which the algorithm already computes, so this adds little to nothing to LIBLINEAR's computational costs. We also adds a function call of *clock()* at every iteration to all the tested algorithms, except QUIC, which includes a "trace" mode that returns automatically the track of function values and running time, by calling *clock()* iteratively. For both QUIC and LIBLINEAR we downloaded the latest versions of the publicly available source code from their official websites, compiled and built the software on the machine on which all experiments were executed, and which uses 2.4GHz quad-core Intel Core i7 processor, 16G RAM and Mac OS.

The optimal objective values $F^*$ obtained approximately by QUIC and LIBLINEAR are later plugged in LHAC and LHAC-L to terminate the algorithm when the following condition is satisfied

$$\frac{F(x) - F^*}{F^*} \leq 10^{-8}. \tag{7.2}$$

In LHAC we chose $\bar{\mu} = 1, \beta = 1/2$ and $\rho = 0.01$ for sufficient decrease (see Algorithm 5), and for LBFGS we use $m = 10$.

When solving the subproblems, we terminate the RCD procedure whenever the number of coordinate steps exceeds

$$(1 + \lfloor \frac{k}{m} \rfloor)|\mathcal{I}_k|, \tag{7.3}$$

where $|\mathcal{I}_k|$ denotes the number of coordinates in the current working set. Condition (7.3) indicates that we expect to update each coordinate in $\mathcal{I}_k$ only once when $k < m$, and that when $k > m$ we increase the number of expected passes through $\mathcal{I}_l$ by 1 every $m$ iterations, i.e., after LBFGS receives a full update. The idea is not only to avoid spending too much time on the subproblem especially at the beginning of the algorithm when the Hessian approximations computed by LBFGS are often fairly coarse, but also to solve the subproblem more accurately as the iterate moves closer to the optimality. Note that in practice when $|\mathcal{I}_k|$ is large, the value of (7.3) almost always dominates $k$, hence it can be lower bounded by $l(k) = ak + b$ with some reasonably large values of $a$ and $b$, which, as we analyzed in Section 5, guarantees the sub linear convergence rate. We also find that (7.3) works quite well in practice in preventing from "over-solving" the subproblems, particularly for LBFGS type algorithms. In 2 and 4 we plot the data with respect to the number of RCD iterations. In particular Figures 2(a) and 4(a) show the number of RCD steps taken at the $k$-th iteration, as a function of $k$. Figures 2(b) and 4(b) show convergence of the objective function to its optimal value as a function of the total number of RCD steps taken so far (both values are plotted in logarithmic scale). Note that RCD steps are not the only component of the CPU time of the algorithms, since gradient computation has to be performed at least once per iteration.

In LHAC-L, a line search procedure is employed, as is done in QUIC and LIBLINEAR, for the convergence to follow from the framework by [27]. In particular, the Armijo rule chooses the step size $\alpha_k$ to be the largest element from $\{\beta^0, \beta^1, \beta^2, ...\}$ satisfying

$$F(x_k + \alpha_k d_k) \leq F(x_k) + \alpha_k \sigma \Delta_k, \tag{7.4}$$

where $0 < \beta < 1, 0 < \sigma < 1$, and $\Delta_k := \nabla f_k^T d_k + \lambda \|x_k + d_k\|_1 - \lambda \|x_k\|_1$. In all the experiments we chose $\beta = 0.5, \sigma = 0.001$ for LHAC-L.

### 7.2 Sparse Inverse Covariance Selection

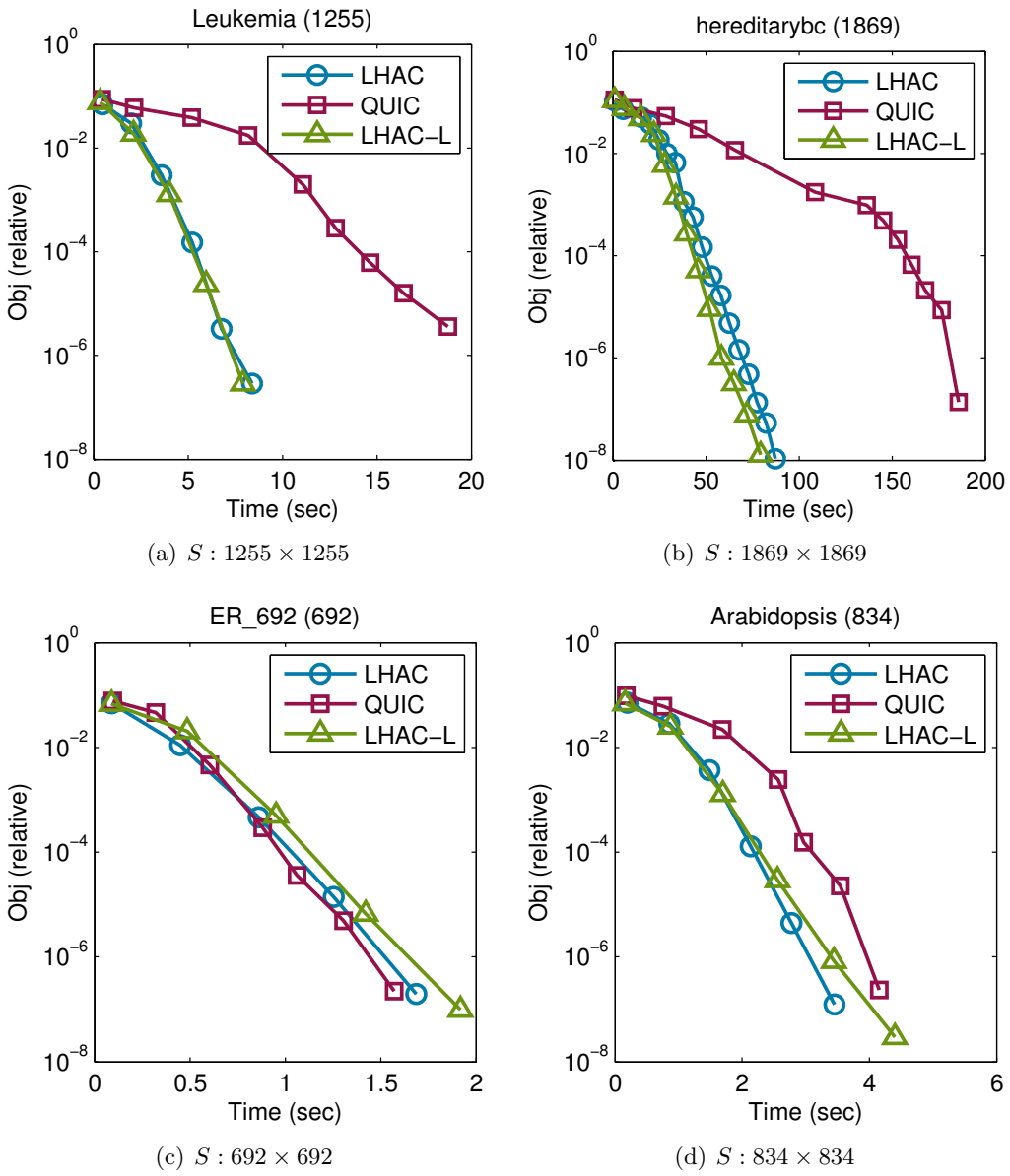
The sparse inverse covariance selection problem is defined by

$$\min_{X \succ 0} \quad F(X) = -\log\det X + \mathrm{tr}(SX) + \lambda ||X||_1, \tag{7.5}$$

where the input $S \in \mathbb{R}^{p \times p}$ is the sample covariance matrix and the optimization is over a symmetric matrix $X \in \mathbb{R}^{p \times p}$ that is required to be positive definite.

For SICS we report results on four real world data sets, denoted as *ER_692*, *Arabidopsis*, *Leukemia* and *hereditarybc*, which are preprocessed from breast cancer data and gene expression networks. We refer to [14] for detailed information about those data sets.

We set the regularization parameter $\lambda = 0.5$ for all experiments as suggested in [14]. The plots presented in Figure 1 show that LHAC and LHAC-L is almost twice as fast as QUIC, in the two largest data sets Leukemia and hereditarybc (see Figure 1(c) and 1(d)). In the other two smaller data sets the results are less clear-cut, but all of the methods solve the problems very fast and the performance of LHAC is comparable to that of QUIC. The performances of LHAC and LHAC-L are fairly similar in all experiments. Again we should note that with the sufficient decrease condition proposed in Algorithm 2 we are able to establish the global convergence rate, which has not been shown in the case of Armijo line search.




(a) $S : 1255 \times 1255$ (b) $S : 1869 \times 1869$

(c) $S : 692 \times 692$ (d) $S : 834 \times 834$

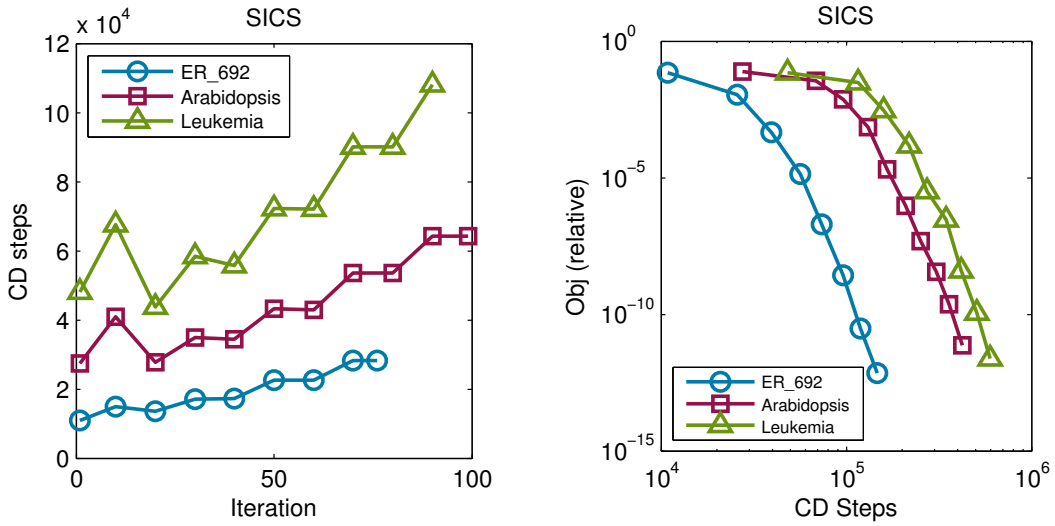
Figure 1: Convergence plots on SICS (the y-axes on log scale).

### 7.3 Sparse Logistic Regression

The objective function of sparse logistic regression is given by

$$F(w) = \lambda \|w\|_1 + \frac{1}{N} \sum_{n=1}^{N} \log(1 + \exp(-y_n \cdot w^T x_n)),$$

where $L(w) = \frac{1}{N} \sum_{n=1}^{N} \log(1+\exp(-y_n \cdot w^T x_n))$ is the average logistic loss function and $\{(x_n, y_n)\}_{n=1}^{N} \in (\mathbb{R}^p \times \{-1, 1\})$ is the training set. The number of instances in the training set and the number of features are denoted by $N$ and $p$ respectively. Note that the evaluation of $F$ requires $O(pN)$ flops and to compute the Hessian requires $O(Np^2)$ flops. Hence, we chose such training sets for our experiment with $N$ and $p$ large enough to test the scalability of the algorithms and yet small




(a) The number of coordinate descent steps on subproblems increases as iterates move towards optimality

(b) Both axes are in log scale. Change of objective w.r.t. the number of coordinate descent steps.

Figure 2: RCD step count of LHAC on different SICS data sets.

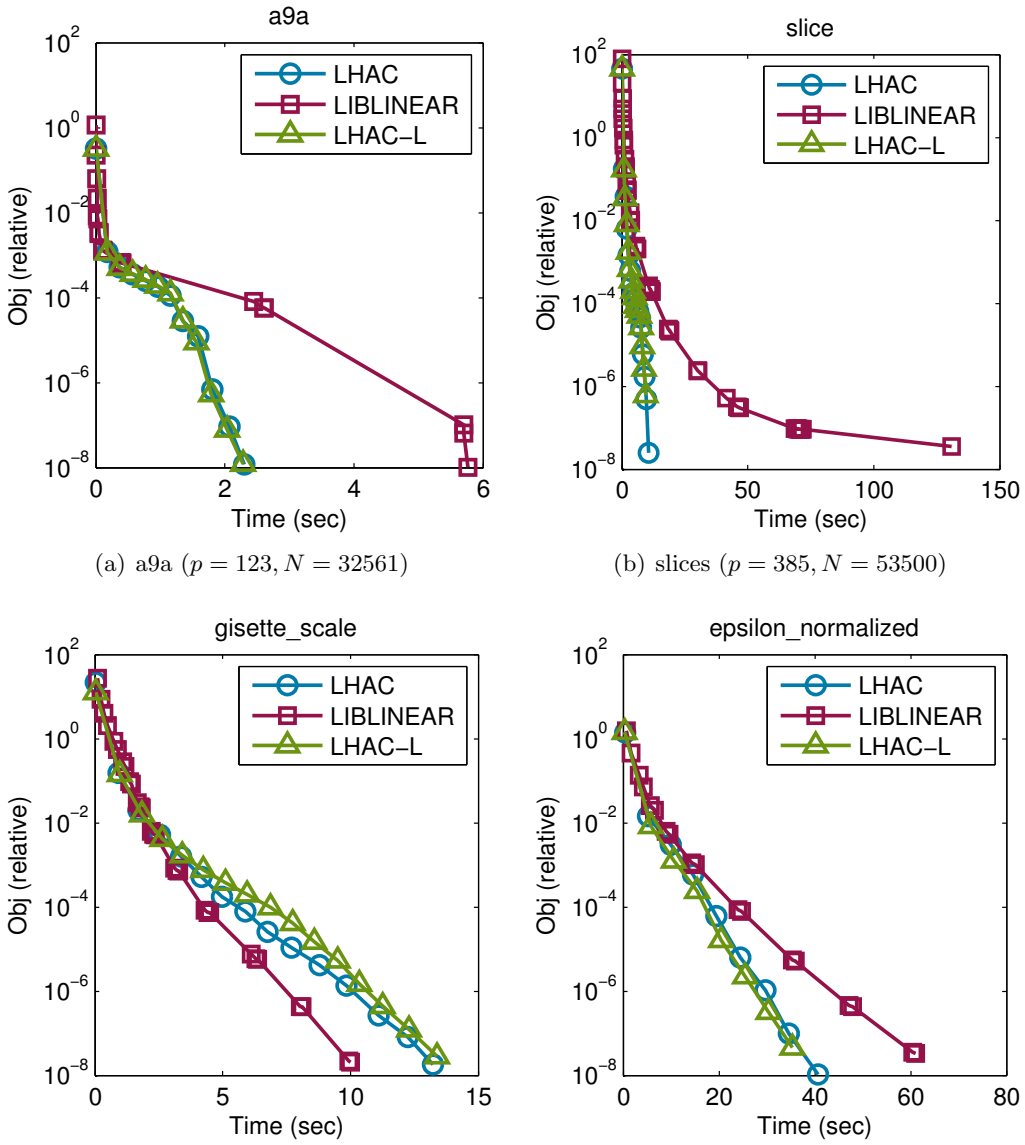
enough to be completed on a workstation.

We report results of SLR on four data sets downloaded from UCI Machine Learning repository [1], whose statistics are summarized in Table 1. In particular, the first data set is the well-known UCI Adult benchmark set *a9a* used for income classification, determining whether a person makes over \$50K/yr or not, based on census data; the second one we use in the experiments is called *epsilon*, an artificial data set for PASCAL large scale learning challenge in 2008; the third one, *slices*, contains features extracted from CT images and is often used for predicting the relative location of CT slices on the human body; and finally we consider *gisette*, a handwritten digit recognition problem from NIPS 2003 feature selection challenge, with the feature set of size 5000 constructed in order to discriminate between two confusable handwritten digits: the four and the nine.

| **Data set** | #features $p$ | #instances $N$ | #non-zeros | Description |
|---|---|---|---|---|
| **a9a** | 123 | 32561 | 451592 | 'census Income' dataset. |
| **epsilon** | 2000 | 100000 | 200000000 | PASCAL challenge 2008. |
| **gisette** | 5000 | 6000 | 29729997 | handwritten digit recognition. |
| **slices** | 385 | 53500 | 20597500 | CT slices location prediction. |

Table 1: Data statistics in sparse logistic regression experiments.

The results are shown in Figure 3. In most cases LHAC and LHAC-L outperform LIBLINEAR. On data set *slice*, LIBLINEAR experiences difficulty in convergence which results in LHAC being faster by an order of magnitude. On the largest data set *epsilon*, LHAC and LHAC-L is faster than LIBLINEAR by about one third and reaches the same precision. Finally we note that the memory usage of LIBLINEAR is more than doubled compared to that of LHAC and LHAC-L, as we observed in all the experiments and is particularly notable on the largest data set *epsilon*.




(a) a9a ($p = 123, N = 32561$)

(b) slices ($p = 385, N = 53500$)

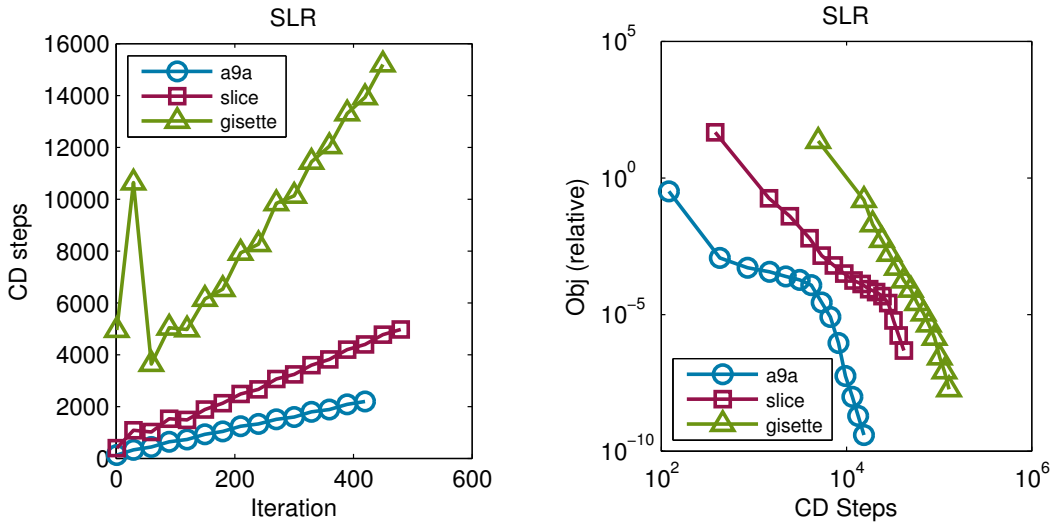
(c) gisette ($p = 5000, N = 6000$)

(d) epsilon ($p = 2000, N = 100000$)

Figure 3: Convergence plots on SLR (the y-axes on log scale).

## 8 Conclusion

In this paper we presented analysis of global convergence rate of inexact proximal quasi-Newton framework, and showed that randomized coordinate descent can be used effectively to find inexact quasi-Newton directions, which guarantee sublinear convergence rate of the algorithm, in expectation. This is the first global convergence rate result for an algorithm that uses coordinate descent to inexactly optimize subproblems at each iteration. Our framework does not rely or exploit the accuracy of second order information, and hence we do not obtain fast local convergence rates. We also do not assume strong convexity of our objective function, hence a sublinear conference rate is the best global rate we can hope to obtain. In [8] an accelerated scheme related to our framework is studied and an optimal sublinear convergence rate is shown, but the assumptions on the Hessian




(a) The number of coordinate descent steps on subproblems increases as iterates move towards optimality

(b) Both axes are in log scale. Change of objective w.r.t. the number of coordinate descent steps.

Figure 4: RCD step count of LHAC on different SLR data sets.

approximations are a lot stronger in [8] than in our paper, hence the accelerated method is not as widely applicable. The framework studied by us in this paper covers several existing efficient algorithms for large scale sparse optimization. However, to provide convergence rates we had to depart from some standard techniques, such as line-search, replacing it instead by a prox-parameter updating mechanism with a trust-region-like sufficient decrease condition for acceptance of iterates. We also use randomized coordinate descent instead of a cyclic one. We demonstrated that this modified framework is, nevertheless, very effective in practice and is competitive with state-of-the-art specialized methods.

## References


[1] K. Bache and M. Lichman, *UCI machine learning repository*, 2013.

[2] A. Beck and M. Teboulle, *A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems*, SIAM Journal on Imaging Sciences, 2 (2009), pp. 183–202.

[3] R. Byrd, G. Chin, J. Nocedal, and F. Oztoprak, *A family of second-order methods for convex l1-regularized optimization*, tech. rep., (2012).

[4] R. Byrd, J. Nocedal, and F. Oztoprak, *An inexact successive quadratic approximation method for convex l-1 regularized optimization*, tech. rep., 2013.

[5] R. H. Byrd, J. Nocedal, and R. B. Schnabel, *Representations of quasi-newton matrices and their use in limited memory methods*, Mathematical Programming, 63 (1994), pp. 129–156.

[6] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982), pp. 400–408.

[7] D. Donoho, *De-noising by soft-thresholding*, Information Theory, IEEE Transactions on, 41 (1995), pp. 613 –627.

[8] J. K. F., S. D. F., and T. K. C., *An inexact accelerated proximal gradient method for large scale linearly constrained convex SDP*, SIAM Journal on Optimization, 3 (2012), p. 10421064.

[9] J. Friedman, T. Hastie, and R. Tibshirani, *Sparse inverse covariance estimation with the graphical lasso.*, Biostatistics Oxford England, 9 (2008), pp. 432–41.

[10] ——, *Regularization paths for generalized linear models via coordinate descent*, Journal of Statistical Software, 33 (2010), pp. 1–22.

[11] C.-J. Hsieh, M. Sustik, I. Dhilon, and P. Ravikumar, *Sparse inverse covariance matrix estimation using quadratic approximation*, NIPS, (2011).

[12] J. D. Lee, Y. Sun, and M. A. Saunders, *Proximal newton-type methods for convex optimization*, in NIPS, 2012.

[13] A. S. Lewis and S. J. Wright, *Identifying activity*, SIAM Journal on Optimization, 21 (2011), pp. 597–614.

[14] L. Li and K.-C. Toh, *An inexact interior point method for L1-regularized sparse covariance selection*, Mathematical Programming, 2 (2010), pp. 291–315.

[15] Y. Nesterov, *Gradient methods for minimizing composite objective function*, CORE report, (2007).

[16] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer Series in Operations Research, Springer, New York, NY, USA, 2nd ed., 2006.

[17] P. A. Olsen, F. Oztoprak, J. Nocedal, and S. J. Rennie, *Newton-Like Methods for Sparse Inverse Covariance Estimation*, 2012.

[18] Z. Qin, K. Scheinberg, and D. Goldfarb, *Efficient block-coordinate descent algorithms for the group lasso*, Mathematical Programming ..., 5 (2010), pp. 143–169.

[19] P. Richtárik and M. Takáč, *Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function*, Mathematical Programming, (2012).

[20] K. Scheinberg, S. Ma, and D. Goldfarb, *Sparse inverse covariance selection via alternating linearization methods*, NIPS, (2010).

[21] K. Scheinberg and I. Rish, *SINCO - a greedy coordinate ascent method for sparse inverse covariance selection problem*, tech. rep., (2009).

[22] M. Schmidt, D. Kim, and S. Sra, *Projected newton-type methods in machine learning*, Optimization for Machine Learning, (2012), p. 305.

[23] M. Schmidt, N. L. Roux, and F. Bach, *Supplementary material for the paper convergence rates of inexact proximal-gradient methods for convex optimization*, in NIPS, 2011.

[24] S. Shalev-Shwartz and A. Tewari, *Stochastic methods for l1 regularized loss minimization*, ICML, (2009), pp. 929–936.

[25] X. Tang, *Optimization in Machine Learning*, PhD thesis, Lehigh University, 2015.

[26] R. Tibshirani, *Regression shrinkage and selection via the lasso*, Journal of the Royal Statistical Society Series B Methodological, 58 (1996), pp. 267–288.

[27] P. Tseng and S. Yun, *A coordinate gradient descent method for nonsmooth separable minimization*, Mathematical Programming, 117 (2009), pp. 387–423.

[28] S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo, *Sparse reconstruction by separable approximation*, Trans. Sig. Proc., 57 (2009), pp. 2479–2493.

[29] M. Wytock and Z. Kolter, *Sparse gaussian conditional random fields: Algorithms, theory, and application to energy forecasting*, in Proceedings of the 30th International Conference on Machine Learning (ICML-13), S. Dasgupta and D. McAllester, eds., vol. 28, JMLR Workshop and Conference Proceedings, May 2013, pp. 1265–1273.

[30] G.-X. Yuan, K.-W. Chang, C.-J. Hsieh, and C.-J. Lin, *A comparison of optimization methods and software for large-scale l1-regularized linear classification*, JMLR, 11 (2010), pp. 3183–3234.

[31] G.-X. Yuan, C.-H. Ho, and C.-J. Lin, *An improved GLMNET for l1-regularized logistic regression and support vector machines*, National Taiwan University, (2011).