An Asynchronous Parallel Stochastic Coordinate Descent Algorithm

Ji Liu† JI-LIU@CS.WISC.EDU
Stephen J. Wright† SWRIGHT@CS.WISC.EDU
Christopher Ré‡ CHRISMRE@STANFORD.EDU
Victor Bittorf† BITTORF@CS.WISC.EDU
Srikrishna Sridhar† SRIKRIS@CS.WISC.EDU
†Department of Computer Sciences, University of Wisconsin-Madison, 1210 W. Dayton St., Madison, WI 53706
‡Department of Computer Science, Stanford University, 353 Serra Mall, Stanford, CA 94305

Abstract

We describe an asynchronous parallel stochastic coordinate descent algorithm for minimizing smooth unconstrained or separably constrained functions. The method achieves a linear convergence rate on functions that satisfy an essential strong convexity property and a sublinear rate ($1/K$) on general convex functions. Near-linear speedup on a multicore system can be expected if the number of processors is $O(n^{1/2})$ in unconstrained optimization and $O(n^{1/4})$ in the separable-constrained case, where n is the number of variables. We describe results from implementation on 40-core processors.

1. Introduction

Consider the convex optimization problem

$$\min_{x\in\Omega} \quad f(x), \tag{1}$$

where $\Omega \subset \mathbb{R}^n$ is a closed convex set and f is a smooth convex mapping from an open neighborhood of Ω to $\mathbb{R}$. We consider two particular cases of Ω in this paper: the unconstrained case $\Omega = \mathbb{R}^n$, and the separable case

$$\Omega = \Omega_1 \times \Omega_2 \times \ldots \times \Omega_n, \tag{2}$$

where each Ω_i, $i = 1, 2, \ldots, n$ is a closed subinterval of the real line.

Formulations of the type (1,2) arise in many data analysis and machine learning problems, for example, support vector machines (linear or nonlinear dual formulation) (Cortes & Vapnik, 1995), LASSO (after decomposing x into positive and negative parts) (Tibshirani, 1996), and logistic regression. Algorithms based on gradient and approximate or partial gradient information have proved effective in these settings. We mention in particular gradient projection and its accelerated variants (Nesterov, 2004), accelerated proximal gradient methods for regularized objectives (Beck & Teboulle, 2009), and stochastic gradient methods (Nemirovski et al., 2009; Shamir & Zhang, 2013). These methods are inherently serial, in that each iteration depends on the result of the previous iteration. Recently, parallel multicore versions of stochastic gradient and stochastic coordinate descent have been described for problems involving large data sets; see for example (Niu et al., 2011; Richtárik & Takáč, 2012; Avron et al., 2014).

This paper proposes an asynchronous stochastic coordinate descent (AsySCD) algorithm for convex optimization. Each step of AsySCD chooses an index $i \in \{1, 2, \ldots, n\}$ and subtracts a short, constant, positive multiple of the partial gradient $\partial f/\partial x_i$ from component i of x. When separable constraints (2) are present, the update is "clipped" to maintain feasibility with respect to Ω_i. Updates take place in parallel across the cores of a multicore system.

We use a simple model of computation that matches well to modern multicore architectures. We assume that each core makes coordinate-descent modifications to a centrally stored vector x in an asynchronous, uncoordinated fashion. We assume that there is a bound τ on the age of the updates, that is, no more than τ updates to x occur between the time at which a processor reads x (and uses it to evaluate one element of the gradient) and the time at which this processor makes its update to a single element of x. (A similar model of parallel asynchronous computation was used in HOGWILD! (Niu et al., 2011).) Our implementation, described in Section 6, is a little more complex than this simple model would suggest, as it is tailored to the architecture of the Intel Xeon machine that we use for experiments.

Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32. Copyright 2014 by the author(s).

We show that linear convergence can be attained if an "essential strong convexity" property (3) holds, while sublinear convergence at a "$1/K$" rate can be proved for general convex functions. Our analysis also defines a sufficient condition for near-linear speedup in the number of cores used. This condition relates the value of delay parameter τ (which relates to the number of cores / threads used in the computation) to the problem dimension n. A parameter that quantifies the cross-coordinate interactions in ∇f also appears in this relationship. When the Hessian of f is nearly diagonal, the minimization problem can almost be separated along the coordinate axes, so higher degrees of parallelism are possible.

We review related work in Section 2. Section 3 specifies the proposed algorithm. Convergence results for unconstrained and constrained cases are described in Sections 4 and 5, respectively, with proofs given in the full version of this paper (Liu et al., 2013). Computational experience is reported in Section 6. Some conclusions are given in Section 7.

Notation and Assumption

We use the following notation.

- $e_i \in \mathbb{R}^n$ denotes the ith natural basis vector.
- $\|\cdot\|$ denotes the Euclidean norm $\|\cdot\|_2$.
- $S \subset \Omega$ denotes the set on which f attains its optimal value, which is denoted by f^*.
- $\mathcal{P}_S(\cdot)$ and $\mathcal{P}_\Omega(\cdot)$ denote Euclidean projection onto S and Ω, respectively.
- We use x_i for the ith element of x, and $\nabla_i f(x)$, $(\nabla f(x))_i$, or $\partial f/\partial x_i$ for the ith element of $\nabla f(x)$.
- We define the following *essential strong convexity* condition for a convex function f with respect to the optimal set S, with parameter $l > 0$:

$$f(x) - f(y) \geq \langle \nabla f(y), x - y\rangle + \frac{l}{2}\|x-y\|^2 \quad \forall\, x, y \in \Omega \text{ with } \mathcal{P}_S(x) = \mathcal{P}_S(y). \tag{3}$$

  This condition is significantly weaker than the usual strong convexity condition, which requires the inequality to hold for *all* $x, y \in \Omega$. In particular, it allows for non-singleton solution sets S, provided that f increases at a uniformly quadratic rate with distance from S. (This property is noted for convex quadratic f in which the Hessian is rank deficient.) Other examples of essentially strongly convex funcions that are not strongly convex include:
  - $f(Ax)$ with arbitrary linear transformation A, where $f(\cdot)$ is strongly convex;
  - $f(x) = \max(a^T x - b, 0)^2$, for $a \neq 0$.
- Define L_{res} as the *restricted Lipschitz constant* for ∇f, where the "restriction" is to the coordinate directions: We have for all $i = 1, 2, \ldots, n$ and $t \in \mathbb{R}$, with $x, x + te_i \in \Omega$

$$\|\nabla f(x) - \nabla f(x + te_i)\| \leq L_{\text{res}}|t|.$$

- Define L_i as the *coordinate Lipschitz constant* for ∇f in the ith coordinate direction: We have for $i \in \{1, 2, \ldots, n\}$, and $x, x + te_i \in \Omega$

$$f(x + te_i) - f(x) \leq \langle \nabla_i f(x),\ t\rangle + \frac{L_i}{2}t^2,$$

  or equivalently

$$|\nabla_i f(x) - \nabla_i f(x + te_i)| \leq L_i |t|.$$

- $L_{\max} := \max_{i=1,2,\ldots,n} L_i$.

Note that $L_{\text{res}} \geq L_{\max}$.

We use $\{x_j\}_{j=0,1,2,\ldots}$ to denote the sequence of iterates generated by the algorithm from starting point x_0. Throughout the paper, we make the following assumption.

Assumption 1.

- *The optimal solution set S of (1) is nonempty.*
- *The radius of the iterate set $\{x_j\}_{j=0,1,2,\ldots}$ defined by*

$$R := \sup_{j=0,1,2,\ldots} \|x_j - \mathcal{P}_S(x_j)\|$$

  is bounded, that is, $R < +\infty$.

Lipschitz Constants

The nonstandard Lipschitz constants L_{res}, $L_{\max}$, and L_i, $i = 1, 2, \ldots, n$ defined above are crucial in the analysis of our method. Besides bounding the nonlinearity of f along various directions, these quantities capture the interactions between the various components in the gradient ∇f, as quantified in the off-diagonal terms of the Hessian $\nabla^2 f(x)$ (when this matrix exists).

We have noted already that $L_{\text{res}}/L_{\max} \geq 1$. Let us consider upper bounds on this ratio under certain conditions. When f is twice continuously differentiable, we have

$$L_i = \sup_{x\in\Omega} \max_{i=1,2,\ldots,n} [\nabla^2 f(x)]_{ii}.$$

Since $\nabla^2 f(x) \succeq 0$ for $x \in \Omega$, we have that

$$|[\nabla^2 f(x)]_{ij}| \leq \sqrt{L_i L_j} \leq L_{\max}, \quad \forall\, i, j = 1, 2, \ldots, n.$$

Thus L_{res}, which is a bound on the largest column norm for $\nabla^2 f(x)$ over all $x \in \Omega$, is bounded by $\sqrt{n}L_{\max}$, so that

$$\frac{L_{\text{res}}}{L_{\max}} \leq \sqrt{n}.$$

If the Hessian is structurally sparse, having at most p nonzeros per row/column, the same argument leads to $L_{\text{res}}/L_{\max} \leq \sqrt{p}$.

If $f(x)$ is a convex quadratic with Hessian Q, We have

$$L_{\max} = \max_i Q_{ii}, \quad L_{\text{res}} = \max_i \|Q_{\cdot i}\|_2,$$

where $Q_{\cdot i}$ denotes the ith column of Q. If Q is diagonally dominant, we have for any column i that

$$\|Q_{\cdot i}\|_2 \leq Q_{ii} + \|[Q_{ji}]_{j\neq i}\|_2 \leq Q_{ii} + \sum_{j\neq i} |Q_{ji}| \leq 2Q_{ii},$$

which, by taking the maximum of both sides, implies that $L_{\text{res}}/L_{\max} \leq 2$ in this case.

Finally, consider the objective $f(x) = \frac{1}{2}\|Ax - b\|^2$ and assume that $A \in \mathbb{R}^{m\times n}$ is a random matrix whose entries are i.i.d from $\mathcal{N}(0, 1)$. The diagonals of the Hessian are $A_{\cdot i}^T A_{\cdot i}$ (where $A_{\cdot i}$ is the ith column of A), which have expected value m, so we can expect $L_{\max}$ to be not less than m. Recalling that L_{res} is the maximum column norm of $A^T A$, we have

$$\begin{aligned}
&\mathbb{E}(\|A^T A_{\cdot i}\|) \leq \mathbb{E}(|A_{\cdot i}^T A_{\cdot i}|) + \mathbb{E}(\|[A_{\cdot j}^T A_{\cdot i}]_{j\neq i}\|) \\
=& m + \mathbb{E}\sqrt{\sum_{j\neq i} |A_{\cdot j}^T A_{\cdot i}|^2} \leq m + \sqrt{\sum_{j\neq i} \mathbb{E}|A_{\cdot j}^T A_{\cdot i}|^2} \\
=& m + \sqrt{(n-1)m},
\end{aligned}$$

where the second inequality uses Jensen's inequality and the final equality uses

$$\begin{aligned}
\mathbb{E}(|A_{\cdot j}^T A_{\cdot i}|^2) &= \mathbb{E}(A_{\cdot j}^T \mathbb{E}(A_{\cdot i} A_{\cdot i}^T) A_{\cdot j}) \\
&= \mathbb{E}(A_{\cdot j}^T I A_{\cdot j}) = \mathbb{E}(A_{\cdot j}^T A_{\cdot j}) = m.
\end{aligned}$$

We can thus estimate the upper bound on $L_{\text{res}}/L_{\max}$ roughly by $1 + \sqrt{n/m}$ for this case.

2. Related Work

This section reviews some related work on coordinate relaxation and stochastic gradient algorithms.

Among *cyclic coordinate descent* algorithms, Tseng (2001) proved the convergence of a block coordinate descent method for nondifferentiable functions with certain conditions. Local and global linear convergence were established under additional assumptions, by Luo & Tseng (1992) and Wang & Lin (2013), respectively. Global linear (sublinear) convergence rate for strongly (weakly) convex optimization was proved in (Beck & Tetruashvili, 2013). Block-coordinate approaches based on proximal-linear subproblems are described in (Tseng & Yun, 2009; 2010). Wright (2012) uses acceleration on reduced spaces (corresponding to the optimal manifold) to improve the local convergence properties of this approach.

Stochastic coordinate descent is almost identical to cyclic coordinate descent except selecting coordinates in a random manner. Nesterov (2012) studied the convergence rate for a stochastic block coordinate descent method for unconstrained and separably constrained convex smooth optimization, proving linear convergence for the strongly convex case and a sublinear $1/K$ rate for the convex case. Extensions to minimization of composite functions are described by Richtárik & Takáč (2011) and Lu & Xiao (2013).

Synchronous parallel methods distribute the workload and data among multiple processors, and coordunate the computation among processors. Ferris & Mangasarian (1994) proposed to distribute variables among multiple processors and optimize concurrently over each subset. The synchronization step searches the affine hull formed by the current iterate and the points found by each processor. Similar ideas appeared in (Mangasarian, 1995), with a different synchronization step. Goldfarb & Ma (2012) considered a multiple splitting algorithm for functions of the form $f(x) = \sum_{k=1}^N f_k(x)$ in which N models are optimized separately and concurrently, then combined in an synchronization step. The alternating direction method-of-multiplier (ADMM) framework (Boyd et al., 2011) can also be implemented in paralle. It dissects the problem into multiple subproblems (possibly after replication of primal variables) and optimizes concurrently, then synchronizes to update multiplier estimates. Duchi et al. (2012) described a subgradient dual-averaging algorithm for partially separable objectives, with subgradient evaluations distributed between cores and combined in ways that reflect the structure of the objective. Parallel stochastic gradient approaches have received broad attention; see Agarwal & Duchi (2012) for an approach that allows delays between evaluation and update, and (Cotter et al., 2011) for a minibatch stochastic gradient approach with Nesterov acceleration. Shalev-Shwartz & Zhang (2013) proposed an accelerated stochastic dual coordinate ascent method.

Among *synchronous parallel methods for (block) coordinate descent*, Richtárik & Takáč (2012) described a method of this type for convex composite optimization problems. All processors update randomly selected coordinates or blocks, concurrently and synchronously, at each iteration. Speedup depends on the sparsity of the data matrix that defines the loss functions. Several variants that select blocks greedily are considered by Scherrer et al. (2012) and Peng et al. (2013). Yang (2013) studied the parallel stochastic dual coordinate ascent method and emphasized the balance between computation and communication.

We turn now to *asynchronous parallel methods*. Bertsekas & Tsitsiklis (1989) introduced an asynchronous parallel implementation for general fixed point problems $x = q(x)$ over a separable convex closed feasible region. (The optimization problem (1) can be formulated in this way by defining $q(x) := \mathcal{P}_\Omega[(I - \alpha\nabla f)(x)]$ for some fixed $\alpha > 0$.)

Their analysis allows inconsistent reads for x, that is, the coordinates of the read x have different "ages." Linear convergence is established if all ages are bounded and $\nabla^2 f(x)$ satisfies a diagonal dominance condition guaranteeing that the iteration $x = q(x)$ is a maximum-norm contraction mapping for sufficient small α. However, this condition is strong — stronger, in fact, than the strong convexity condition. For convex quadratic optimization $f(x) = \frac{1}{2}x^T Ax + bx$, the contraction condition requires diagonal dominance of the Hessian: $A_{ii} > \sum_{i \neq j} |A_{ij}|$ for all $i = 1, 2, \ldots, n$. By comparison, AsySCD guarantees linear convergence rate under the essential strong convexity condition (3), though we do not allow inconsistent read. (We require the vector x used for each evaluation of $\nabla_i f(x)$ to have existed at a certain point in time.)

Hogwild! (Niu et al., 2011) is a lock-free, asynchronous parallel implementation of a stochastic-gradient method, targeted to a multicore computational model similar to the one considered here. Its analysis assumes consistent reading of x, and it is implemented without locking or coordination between processors. Under certain conditions, convergence of Hogwild! approximately matches the sublinear $1/K$ rate of its serial counterpart, which is the constant-steplength stochastic gradient method analyzed in (Nemirovski et al., 2009).

We also note recent work by Avron et al. (2014), who proposed an asynchronous linear solver to solve $Ax = b$ where A is a symmetric positive definite matrix, proving a linear convergence rate. Both inconsistent- and consistent-read cases are analyzed in this paper, with the convergence result for inconsistent read being slightly weaker.

Algorithm 1 Asynchronous Stochastic Coordinate Descent Algorithm $x_{K+1} = \text{AsySCD}(x_0, \gamma, K)$

```
Require: $x_0 \in \Omega$, $\gamma$, and $K$
Ensure: $x_{K+1}$
1: Initialize $j \leftarrow 0$;
2: while $j \leq K$ do
3:     Choose $i(j)$ from $\{1, \ldots, n\}$ with equal probability;
4:     $x_{j+1} \leftarrow \mathcal{P}_\Omega \left( x_j - \frac{\gamma}{L_{\max}} e_{i(j)} \nabla_{i(j)} f(x_{k(j)}) \right)$;
5:     $j \leftarrow j + 1$;
6: end while
```

3. Algorithm

In AsySCD, multiple processors have access to a shared data structure for the vector x, and each processor is able to compute a randomly chosen element of the gradient vector $\nabla f(x)$. Each processor repeatedly runs the following coordinate descent process (the steplength parameter γ is discussed further in the next section):

R: Choose an index $i \in \{1, 2, \ldots, n\}$ at random, read x, and evaluate $\nabla_i f(x)$;

U: Update component i of the shared x by taking a step of length $\gamma / L_{\max}$ in the direction $-\nabla_i f(x)$.

Since these processors are being run concurrently and without synchronization, x may change between the time at which it is read (in step R) and the time at which it is upatted (step U). We capture the system-wide behavior of AsySCD in Algorithm 1. There is a global counter j for the total number of updates; x_j denotes the state of x after j updates. The index $i(j) \in \{1, 2, \ldots, n\}$ denotes the component updated at step j. $k(j)$ denotes the x-iterate at which the update applied at iteration j was calculated. Obviously, we have $k(j) \leq j$, but we assume that the delay between the time of evaluation and updating is bounded uniformly by a positive integer τ, that is, $j - k(j) \leq \tau$ for all j. The value of τ captures the essential parallelism in the method, as it indicates the number of processors that are involved in the computation.

The projection operation P_Ω onto the feasible set is not needed in the case of unconstrained optimization. For separable constraints (2), it requires a simple clipping operation on the $i(j)$ component of x.

We note several differences with earlier asynchronous approaches. Unlike the asynchronous scheme in Bertsekas & Tsitsiklis (1989, Section 6.1), the *latest* value of x is updated at each step, not an earlier iterate. Although our model of computation is similar to Hogwild! (Niu et al., 2011), the algorithm differs in that each iteration of AsySCD evaluates a single component of the gradient exactly, while Hogwild! computes only a (usually crude) estimate of the full gradient. Our analysis of AsySCD below is comprehensively different from that of (Niu et al., 2011), and we obtain stronger convergence results.

4. Unconstrained Smooth Convex Case

This section presents results about convergence of AsySCD in the unconstrained case $\Omega = \mathbb{R}^n$. The theorem encompasses both the linear rate for essentially strongly convex f and the sublinear rate for general convex f. The result depends strongly on the delay parameter τ. (Proofs of results in this section appear in the full version of this paper (Liu et al., 2013).)

A crucial issue in AsySCD is the choice of steplength parameter γ. This choice involves a tradeoff: We would like γ to be long enough that significant progress is made at each step, but not so long that the gradient information computed at step $k(j)$ is stale and irrelevant by the time the update is applied at step j. We enforce this tradeoff by means of a bound on the ratio of expected squared norms on ∇f at successive iterates; specifically,

$$\rho^{-1} \le \frac{\mathbb{E}\|\nabla f(x_{j+1})\|^2}{\mathbb{E}\|\nabla f(x_j)\|^2} \le \rho, \tag{4}$$

where $\rho > 1$ is a user defined parameter. The analysis becomes a delicate balancing act in the choice of ρ and steplength γ between aggression and excessive conservatism. We find, however, that these values can be chosen to ensure steady convergence for the asynchronous method at a *linear* rate, with rate constants that are almost consistent with vanilla short-step full-gradient descent.

Theorem 1. *Suppose that $\Omega = \mathbb{R}^n$ in (1) and that Assumption 1 is satisfied. For any $\rho > 1$, define the quantity ψ as follows:*

$$\psi := 1 + \frac{2\tau\rho^\tau L_{\text{res}}}{\sqrt{n}L_{\max}}. \tag{5}$$

Suppose that the steplength parameter $\gamma > 0$ satisfies the following three upper bounds:

$$\gamma \le \frac{1}{\psi},\ \gamma \le \frac{(\rho-1)\sqrt{n}L_{\max}}{2\rho^{\tau+1}L_{\text{res}}},\ \gamma \le \frac{(\rho-1)\sqrt{n}L_{\max}}{L_{\text{res}}\rho^\tau(2+\frac{L_{\text{res}}}{\sqrt{n}L_{\max}})}.$$

Then we have that for any $j \ge 0$ that

$$\rho^{-1}\mathbb{E}(\|\nabla f(x_j)\|^2) \le \mathbb{E}(\|\nabla f(x_{j+1})\|^2) \le \rho\mathbb{E}(\|\nabla f(x_j)\|^2).$$

Moreover, if the essentially strong convexity property (3) holds with $l > 0$, we have

$$\mathbb{E}(f(x_j)-f^*) \le \left(1 - \frac{2l\gamma}{nL_{\max}}\left(1-\frac{\psi}{2}\gamma\right)\right)^j (f(x_0)-f^*), \tag{6}$$

while for general smooth convex functions f, we have

$$\mathbb{E}(f(x_j)-f^*) \le \frac{1}{(f(x_0)-f^*)^{-1} + \frac{\gamma}{nL_{\max}R^2}(1-\frac{\psi}{2}\gamma)j}. \tag{7}$$

This theorem demonstrates linear convergence (6) for AsySCD in the unconstrained essentially strongly convex case. This result is better than that obtained for Hogwild! (Niu et al., 2011), which guarantees only sublinear convergence under the stronger assumption of strict convexity.

The following corollary proposes an interesting particular choice of the parameters for which the convergence expressions become more comprehensible. The result requires a condition on the delay bound τ in terms of n and the ratio $L_{\max}/L_{\text{res}}$.

Corollary 2. *Suppose that Assumption 1 holds, and that*

$$\tau + 1 \le \frac{\sqrt{n}L_{\max}}{2eL_{\text{res}}}. \tag{8}$$

Then if we choose

$$\rho = 1 + \frac{2eL_{\text{res}}}{\sqrt{n}L_{\max}}, \tag{9}$$

define ψ by (5), and set $\gamma = 1/\psi$, we have for the essentially strongly convex case (3) with $l > 0$ that

$$\mathbb{E}(f(x_j)-f^*) \le \left(1 - \frac{l}{2nL_{\max}}\right)^j (f(x_0)-f^*), \tag{10}$$

while for the case of general convex f, we have

$$\mathbb{E}(f(x_j)-f^*) \le \frac{1}{(f(x_0)-f^*)^{-1} + \frac{j}{4nL_{\max}R^2}}. \tag{11}$$

We note that the linear rate (10) is broadly consistent with the linear rate for the classical steepest descent method applied to strongly convex functions, which has a rate constant of $(1-2l/L)$, where L is the standard Lipschitz constant for ∇f. If we assume (not unreasonably) that n steps of stochastic coordinate descent cost roughly the same as one step of steepest descent, and note from (10) that n steps of stochastic coordinate descent would achieve a reduction factor of about $(1-l/(2L_{\max}))$, a standard argument would suggest that stochastic coordinate descent would require about $4L_{\max}/L$ times more computation. (Note that $L_{\max}/L \in [1/n, 1]$.) The stochastic approach may gain an advantage from the parallel implementation, however. Steepest descent would require synchronization and careful division of evaluation work, whereas the stochastic approach can be implemented in an asynchronous fashion.

For the general convex case, (11) defines a sublinear rate, whose relationship with the rate of the steepest descent for general convex optimization is similar to the previous paragraph.

As noted in Section 1, the parameter τ is closely related to the number of cores that can be involved in the computation, without degrading the convergence performance of the algorithm. In other words, if the number of cores is small enough such that (8) holds, the convergence expressions (10), (11) do not depend on the number of cores, implying that linear speedup can be expected. A small value for the ratio $L_{\text{res}}/L_{\max}$ (not much greater than 1) implies a greater degree of potential parallelism. As we note at the end of Section 1, this ratio tends to be small in some important applications. In these situations, the maximal number of cores could be as high as $O(\sqrt{n})$.

5. Constrained Smooth Convex Case

This section considers the case of separable constraints (2). We show results about convergence rates and high-probability complexity estimates, analogous to those of the previous section. Proofs appear in the full version (Liu et al., 2013).

As in the unconstrained case, the steplength γ should be chosen to ensure steady progress while ensuring that update information does not become too stale. Because constraints are present, the ratio (4) is no longer appropriate. We use instead a ratio of squares of expected differences in successive primal iterates:

$$\mathbb{E}\|x_{j-1} - \bar{x}_j\|^2 / \mathbb{E}\|x_j - \bar{x}_{j+1}\|^2, \tag{12}$$

where $\bar{x}_{j+1}$ is the hypothesized full update obtained by applying the single-component update to *every* component of x_j, that is,

$$\bar{x}_{j+1} := \arg\min_{x\in\Omega} \langle \nabla f(x_{k(j)}), x - x_j\rangle + \frac{L_{\max}}{2\gamma}\|x - x_j\|^2.$$

We have the following result concerning convergence of the expected error to zero.

Theorem 3. *Suppose that Ω has the form (2), that Assumption 1 is satisfied, and that $n \geq 5$. Let ρ be a constant with $\rho > (1 - 2/\sqrt{n})^{-1}$, and define the quantity ψ as follows:*

$$\psi := 1 + \frac{L_{\text{res}}\tau\rho^\tau}{\sqrt{n}L_{\max}}\left(2 + \frac{L_{\max}}{\sqrt{n}L_{\text{res}}} + \frac{2\tau}{n}\right). \tag{13}$$

Suppose that the steplength parameter $\gamma > 0$ satisfies the following two upper bounds:

$$\gamma \leq \frac{1}{\psi}, \quad \gamma \leq \left(1 - \frac{1}{\rho} - \frac{2}{\sqrt{n}}\right)\frac{\sqrt{n}L_{\max}}{4L_{\text{res}}\tau\rho^\tau}. \tag{14}$$

Then we have

$$\mathbb{E}\|x_{j-1} - \bar{x}_j\|^2 \leq \rho\mathbb{E}\|x_j - \bar{x}_{j+1}\|^2, \quad j = 1, 2, \dots. \tag{15}$$

If the essential strong convexity property (3) holds with $l > 0$, we have for $j = 1, 2, \dots$ that

$$\mathbb{E}\|x_j - \mathcal{P}_S(x_j)\|^2 + \frac{2\gamma}{L_{\max}}(\mathbb{E}f(x_j) - f^*) \tag{16}$$
$$\leq \left(1 - \frac{l}{n(l + \gamma^{-1}L_{\max})}\right)^j \left(R^2 + \frac{2\gamma}{L_{\max}}(f(x_0) - f^*)\right).$$

For general smooth convex function f, we have

$$\mathbb{E}f(x_j) - f^* \leq \frac{n(R^2 L_{\max} + 2\gamma(f(x_0) - f^*))}{2\gamma(n + j)}. \tag{17}$$

Similarly to the unconstrained case, the following corollary proposes an interesting particular choice for the parameters for which the convergence expressions become more comprehensible. The result requires a condition on the delay bound τ in terms of n and the ratio $L_{\max}/L_{\text{res}}$.

Corollary 4. *Suppose that Assumption 1 holds, that $\tau \geq 1$ and $n \geq 5$, and that*

$$\tau(\tau + 1) \leq \frac{\sqrt{n}L_{\max}}{4eL_{\text{res}}}. \tag{18}$$

If we choose

$$\rho = 1 + \frac{4e\tau L_{\text{res}}}{\sqrt{n}L_{\max}}, \tag{19}$$

then the steplength $\gamma = 1/2$ will satisfy the bounds (14). In addition, for the essentially strongly convex case (3) with $l > 0$, we have for $j = 1, 2, \dots$ that

$$\mathbb{E}(f(x_j) - f^*) \leq \left(1 - \frac{n^{-1}l}{l + 2L_{\max}}\right)^j (L_{\max}R^2 + f(x_0) - f^*), \tag{20}$$

while for the case of general convex f, we have

$$\mathbb{E}(f(x_j) - f^*) \leq \frac{n(L_{\max}R^2 + f(x_0) - f^*)}{j + n}. \tag{21}$$

Similarly to Section 4, and provided τ satisfies (18), the convergence rate is not affected appreciably by the delay bound τ, and close-to-linear speedup can be expected for multicore implementations when (18) holds. This condition is more restrictive than (8) in the unconstrained case, but still holds in many problems for interesting values of τ. When $L_{\text{res}}/L_{\max}$ is bounded independently of dimension, the maximal number of cores allowed is of the the order of $n^{1/4}$, which is slightly smaller than the $O(n^{1/2})$ value obtained for the unconstrained case.

6. Experiments

We illustrate the behavior of two variants of the stochastic coordinate descent approach on test problems constructed from several data sets. Our interests are in the efficiency of multicore implementations (by comparison with a single-threaded implementation) and in performance relative to alternative solvers for the same problems.

All our test problems have the form (1), with either $\Omega = \mathbb{R}^n$ or Ω separable as in (2). The objective f is quadratic, that is,

$$f(x) = \frac{1}{2}x^T Qx + c^T x,$$

with Q symmetric positive definite.

Our implementation of AsySCD is called DIMMWITTED (or DW for short). It runs on various numbers of threads, from 1 to 40, each thread assigned to a single core in our 40-core Intel Xeon architecture. Cores on the Xeon architecture are arranged into four sockets — ten cores per socket, with each socket having its own memory. Non-uniform memory access (NUMA) means that memory accesses to local memory (on the same socket as the core) are less expensive than accesses to memory on another socket. In our DW implementation, we assign each socket an equal-sized "slice" of Q, a row submatrix. The components of x are partitioned between cores, each core being responsible for updating its own partition of x (though it can read the components of x from other cores). The components of x assigned to the cores correspond to the rows of Q assigned to that core's socket. Computation is grouped into "epochs," where an epoch is defined to be the period of computation during which each component of x is updated exactly once. We use the parameter p to denote the number of epochs that are executed between reordering (shuffling) of the coordinates of x. We investigate both shuffling after every epoch ($p = 1$) and after every tenth epoch ($p = 10$). Access to x is lock-free, and updates are performed asynchronously. This update scheme does not implement exactly the "sampling with replacement" scheme analyzed in previous sections, but can be viewed as a high performance, practical adaptation of the AsySCD method. Please refer to Zhang & Ré (2014) for more details about DW.

To do each coordinate descent update, a thread must read the latest value of x. Most components are already in the cache for that core, so that it only needs to fetch those components recently changed. When a thread writes to x_i, the hardware ensures that this x_i is simultaneously removed from other cores, signaling that they must fetch the updated version before proceeding with their respective computations.

The first test problem `QP` is an unconstrained, regularized least squares problem constructed with synthetic data. It has the form

$$\min_{x \in \mathbb{R}^n} \; f(x) := \frac{1}{2}\|Ax - b\|^2 + \frac{\alpha}{2}\|x\|^2. \qquad (22)$$

All elements of $A \in \mathbb{R}^{m\times n}$, the true model $\tilde{x} \in \mathbb{R}^n$, and the observation noise vector $\delta \in \mathbb{R}^m$ are generated in i.i.d. fashion from the Gaussian distribution $\mathcal{N}(0, 1)$, following which each column in A is scaled to have a Euclidean norm of 1. The observation $b \in \mathbb{R}^m$ is constructed from $A\tilde{x} + \delta\|A\tilde{x}\|/(5m)$. We choose $m = 6000$, $n = 20000$, and $\alpha = 0.5$. We therefore have $L_{\max} = 1 + \alpha = 1.5$ and

$$\frac{L_{\text{res}}}{L_{\max}} \approx \frac{1 + \sqrt{n/m} + \alpha}{1 + \alpha} \approx 2.2.$$

This problem is diagonally dominant, and the condition (8) is satisfied when delay parameter τ is less than about 95. In Algorithm 1, we set the steplength parameter γ to 1, and we choose initial iterate to be $x_0 = \mathbf{0}$. We measure convergence of the residual norm $\|\nabla f(x)\|$.

Our second problem `QPc` is a bound-constrained version of (22):

$$\min_{x \in \mathbb{R}^n_+} \quad f(x) := \frac{1}{2}(x - \tilde{x})^T(A^TA + \alpha I)(x - \tilde{x}). \quad (23)$$

The methodology for generating A and $\tilde{x}$ and for choosing the values of m, n, γ, and x_0 is the same as for (22). We measure convergence via the residual $\|x - \mathcal{P}_\Omega(x - \nabla f(x))\|$ where Ω is the nonnegative orthant $\mathbb{R}^n_+$. At the solution of (23), about half the components of x are at their lower bound of 0.

Our third and fourth problems are quadratic penalty functions for linear programming relaxations of vertex cover problems on large graphs. The variable vector x in these problems has dimension $n = |V| + |E|$, where V and E are the vertex and edge sets for the graph, respectively. The feasible set $\Omega = [0, 1]^n$ has the separable form (2), and the problem has the following form, for some penalty parameter β:

$$\min_{x \in \mathbb{R}^n_+} \quad c^Tx + \frac{\beta}{2}\|Ax - b\|^2 + \frac{1}{2\beta}\|x\|^2, \qquad (24)$$

with $\beta = 5$. Details appear in the full version (Liu et al., 2013). `Amazon` has $n = 561050$ and `DBLP` has $n = 520891$.

We tracked the behavior of the gradient as a function of the number of epochs, when executed on different numbers of cores. We found that in this respect, the performance of the algorithm does not change appreciably as the number of cores is increased. Thus, any deviation from linear speedup is due not to degradation of convergence speed in the algorithm but rather to systems issues in the implementation. Graphs of convergence behavior for different numbers of cores are shown in the full version (Liu et al., 2013).

We define speedup of DW on multicore implementations as follows:

$$\frac{\text{runtime a single core using DW}}{\text{runtime on } P \text{ cores}}.$$

Results are shown in Figures 1 for a variant of DW in which the indexes are reshuffled only every tenth epoch ($p = 10$). Near-linear speedup can be observed for the two QP problems with synthetic data. For Problems 3 and 4, speedup is at most 12-14, and there are few gains when the number of cores exceeds about 12. We believe that the degradation is due mostly to memory contention. Although these problems have high dimension, the matrix Q is very sparse (in contrast to the dense Q for the synthetic data set). Thus, the ratio of computation to data movement / memory access is much lower for these problems, making memory contention effects more significant. Figures 1 also shows results of a global-locking strategy for the parallel stochastic coordinate descent method, in which the vector x is locked by a core whenever it performs a read or update. The performance curve for this strategy hugs the horizontal axis; it is not competitive.

The time required for the four test problems on 1 and 40 cores, to reduce residuals below 10^{-5}, are shown in Table 1. (Similar speedups are noted when we use a convergence tolerance looser than 10^{-5}.)

All problems reported on above are essentially strongly convex. Similar speedup properties can be obtained in the weakly convex case as well. We illustrate this claim by showing speedups for the `QPc` problem with $\alpha = 0$, see the full version (Liu et al., 2013).

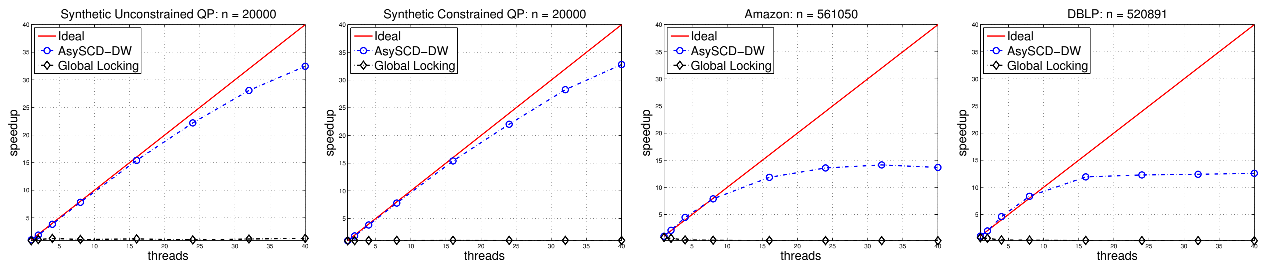

Figure 1. Test problems 1, 2, 3, and 4: Speedup of multicore implementations of DW on up to 40 cores of an Intel Xeon architecture. Ideal (linear) speedup curve is shown for reference, along with poor speedups obtained for a global-locking strategy.

Problem	1 core	40 cores
`QP`	98.4	3.03
`QPc`	59.7	1.82
`Amazon`	17.1	1.25
`DBLP`	11.5	.91

Table 1. Runtimes (s) for the four test problems on 1 and 40 cores.

#cores	Time(sec) SynGD / AsySCD	Speedup SynGD / AsySCD
1	121. / 27.1	0.22 / 1.00
10	11.4 / 2.57	2.38 / 10.5
20	6.00 / 1.36	4.51 / 19.9
30	4.44 / 1.01	6.10 / 26.8
40	3.91 / 0.88	6.93 / 30.8

Table 2. Efficiency comparison between SynGD and AsySCD for the `QP` problem. The running time and speedup are based on the residual achieving a tolerance of 10^{-5}.

Dataset	# of Samples	# of Features	Train time(sec) LIBSVM	Train time(sec) AsySCD
adult	32561	123	16.15	1.39
news	19996	1355191	214.48	7.22
rcv	20242	47236	40.33	16.06
reuters	8293	18930	1.63	0.81
w8a	49749	300	33.62	5.86

Table 3. Efficiency comparison between LIBSVM and AsySCD for kernel SVM using 40 cores using homogeneous kernels ($K(x_i, x_j) = (x_i^T x_j)^2$). The running time and speedup are calculated based on the "residual" 10^{-3}. Here, to make both algorithms comparable, the "residual" is defined by $\|x - \mathcal{P}_\Omega(x - \nabla f(x))\|_\infty$.

Turning now to comparisons between AsySCD and alternative algorithms, we start by considering the basic gradient descent method. We implement gradient descent in a parallel, synchronous fashion, distributing the gradient computation load on multiple cores and updates the variable x in parallel at each step. The resulting implementation is called SynGD. Table 2 reports running time and speedup of both AsySCD over SynGD, showing a clear advantage for AsySCD.

Next we compare AsySCD to LIBSVM (Chang & Lin, 2011) a popular parallel solver for kernel support vector machines (SVM). Both algorithms are run on 40 cores to solve the dual formulation of kernel SVM, without an intercept term. All datasets used in Table 3 except reuters were obtained from the LIBSVM dataset repository[1]. The dataset reuters is a sparse binary text classification dataset constructed as a one-versus-all version of Reuters-2159[2]. Our comparisons, shown in Table 3, indicate that AsySCD outperforms LIBSVM on these test sets.

7. Conclusions

This paper proposed an asynchronous parallel stochastic coordinate descent algorithm for problems with no constraints and separable constraints, which is proven to achieve sublinear convergence (at rate $1/K$) on general convex functions and linear convergence on functions that satisfy an essential strong convexity property. Our analysis also indicates the extent to which parallel implementations can be expected to yield near-linear speedup, in terms of a parameter that quantifies the cross-coordinate interactions in the gradient ∇f. Our computational experience confirms the theory, as deviations from linear speedup are due to implementation issues rather than algorithmic issues.

The analysis extends almost immediately to a *block*-coordinate update scheme, provided that the constraints are block-separable.

8. Acknowledgements

This project is supported by NSF Grants DMS-0914524, DMS-1216318, and CCF-1356918; NSF CAREER Award IIS-1353606; ONR Awards N00014-13-1-0129 and N00014-12-1-0041; AFOSR Award FA9550-13-1-0138; a Sloan Research Fellowship; and grants from Oracle, Google, and ExxonMobil.

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

[2] `http://www.daviddlewis.com/resources/testcollections/reuters21578/`

References

Agarwal, A. and Duchi, J. C. Distributed delayed stochastic optimization. *CDC*, pp. 5451–5452, 2012.

Avron, H., Druinsky, A., and Gupta, A. Revisiting asynchronous linear solvers: Provable convergence rate through randomization. *IPDPS*, 2014.

Beck, A. and Teboulle, M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sciences*, 2(1):183–202, 2009.

Beck, A. and Tetruashvili, L. On the convergence of block coordinate descent type methods, 2013. To appear in *SIAM Journal on Optimization*.

Bertsekas, D. P. and Tsitsiklis, J. N. *Parallel and Distributed Computation: Numerical Methods*. Pentice Hall, 1989.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

Chang, C.-C. and Lin, C.-J. LIBSVM: A library for support vector machines, 2011. URL `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`.

Cortes, C. and Vapnik, V. Support vector networks. *Machine Learning*, pp. 273–297, 1995.

Cotter, A., Shamir, O., Srebro, N., and Sridharan, K. Better mini-batch algorithms via accelerated gradient methods. *arXiv: 1106.4574*, 2011.

Duchi, J. C., Agarwal, A., and Wainwright, M. J. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Transactions on Automatic Control*, 57 (3):592–606, 2012.

Ferris, M. C. and Mangasarian, O. L. Parallel variable distribution. *SIAM Journal on Optimization*, 4(4):815–832, 1994.

Goldfarb, D. and Ma, S. Fast multiple-splitting algorithms for convex optimization. *SIAM Journal on Optimization*, 22(2): 533–556, 2012.

Liu, J., Wright, S. J., Ré, C., Bittorf, V., and Sridhar, S. An asynchronous parallel stochastic coordinate descent algorithm. *Arxiv: 1311.1873*, 2013.

Lu, Z. and Xiao, L. On the complexity analysis of randomized block-coordinate descent methods. *TechReport*, 2013.

Luo, Z. Q. and Tseng, P. On the convergence of the coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72:7–35, 1992.

Mangasarian, O. L. Parallel gradient distribution in unconstrained optimization. *SIAM Journal on Optimization*, 33(1):916–1925, 1995.

Nemirovski, A., Juditsky, A., Lan, G., and Shapiro, A. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19:1574–1609, 2009.

Nesterov, Y. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer Academic Publishers, 2004.

Nesterov, Y. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

Niu, F., Recht, B., Ré, C., and Wright, S. J. HOGWILD!: A lock-free approach to parallelizing stochastic gradient descent. *Advances in Neural Information Processing Systems 24*, pp. 693–701, 2011.

Peng, Z., Yan, M., and Yin, W. Parallel and distributed sparse optimization. *preprint*, 2013.

Richtárik, P. and Takáč, M. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *arXiv:1107.2848*, 2011.

Richtárik, P. and Takáč, M. Parallel coordinate descent methods for big data optimization. *ArXiv: 1212.0873*, 2012.

Scherrer, C., Tewari, A., Halappanavar, M., and Haglin, D. Feature clustering for accelerating parallel coordinate descent. *NIPS*, pp. 28–36, 2012.

Shalev-Shwartz, S. and Zhang, T. Accelerated mini-batch stochastic dual coordinate ascent. *Arxiv: 1305.2581*, 2013.

Shamir, O. and Zhang, T. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. *ICML*, 2013.

Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1996.

Tseng, P. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109:475–494, 2001.

Tseng, P. and Yun, S. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming, Series B*, 117:387–423, June 2009.

Tseng, P. and Yun, S. A coordinate gradient descent method for linearly constrained smooth optimization and support vector machines training. *Computational Optimization and Applications*, 47(2):179–206, 2010.

Wang, P.-W. and Lin, C.-J. Iteration complexity of feasible descent methods for convex optimization. *Technical report*, 2013.

Wright, S. J. Accelerated block-coordinate relaxation for regularized optimization. *SIAM Journal on Optimization*, 22(1): 159–186, 2012.

Yang, T. Trading computation for communication: Distributed stochastic dual coordinate ascent. *NIPS*, pp. 629–637, 2013.

Zhang, C. and Ré, C. Dimmwitted: A study of main-memory statistical analytics. *ArXiv: 1403.7550*, 2014.