Parallel Coordinate Descent Newton Method for Efficient $L_1$-Regularized Loss Minimization

Yatao An Bian, Xiong Li, Yuncai Liu, and Ming-Hsuan Yang, Fellow, IEEE

Abstract—The recent years have witnessed advances in parallel algorithms for large-scale optimization problems. Notwithstanding the demonstrated success, existing algorithms that parallelize over features are usually limited by divergence issues under high parallelism or require data preprocessing to alleviate these problems. In this paper, we propose a Parallel Coordinate Descent algorithm using approximate Newton steps (PCDN) that is guaranteed to converge globally without data preprocessing. The key component of the PCDN algorithm is the high-dimensional line search, which guarantees the global convergence with high parallelism. The PCDN algorithm randomly partitions the feature set into $b$ subsets/bundles of size $P$, and sequentially processes each bundle by first computing the descent directions for each feature in parallel and then conducting $P$-dimensional line search to compute the step size. We show that: 1) the PCDN algorithm is guaranteed to converge globally despite increasing parallelism and 2) the PCDN algorithm converges to the specified accuracy $\epsilon$ within the limited iteration number of $T_\epsilon$, and $T_\epsilon$ decreases with increasing parallelism. In addition, the data transfer and synchronization cost of the $P$-dimensional line search can be minimized by maintaining intermediate quantities. For concreteness, the proposed PCDN algorithm is applied to $L_1$-regularized logistic regression and $L_1$-regularized $L_2$-loss support vector machine problems. Experimental evaluations on seven benchmark data sets show that the PCDN algorithm exploits parallelism well and outperforms the state-of-the-art methods.

Index Terms—Armijo line search, convergence rate, coordinate descent, $L_1$-regularized loss minimization.

I. INTRODUCTION

HIGH-dimensional $L_1$-minimization problems arise in a wide range of applications, including sparse logistic regression [16], $L_1$-regularized support vector machine (SVM) classification [4], image coding [12], and face recognition [25]. To solve $L_1$-optimization problems efficiently, several algorithms based on coordinate gradient descent (CGD) [24], stochastic gradient [22], interior point [9], and trust region [13] have been developed, among which the coordinate descent Newton (CDN) [26] and improved GLMNET [27] methods have demonstrated promising results for $L_1$-regularized linear optimization problems.

Within the $L_1$-optimization framework, large data sets with high-dimensional features entail scalable and efficient parallel algorithms. Several methods perform parallelization over samples [10], [17], [28], [29] although usually there are more features than samples in $L_1$-regularized problems. Richtárik and Takáč [19] show that randomized coordinate descent methods can be accelerated by parallelization for solving Lasso problems, and the work is further extended to distributed settings [15], [18]. In addition, Bradley et al. [2] proposed the shotgun CDN (SCDN) method for $L_1$-regularized logistic regression by directly parallelizing the updates of features based on the CDN algorithm [26]. However, the SCDN method is not guaranteed to converge when the number of updated features in parallel is greater than a threshold, and thereby limits its ability of exploiting high parallelism. While this problem can be alleviated by preprocessing samples (e.g., feature clustering) to achieve higher parallelism [21], it requires additional computational overhead. The accelerated shotgun method [14] is a first-order algorithm without backtrack line search which has fast convergence. However, it can only deal with the objective functions without regularization terms. Scherrer et al. [20] presented a generic framework for parallel coordinate descent methods, which includes Shotgun, Greedy, Thread-Greedy and Coloring. Their empirical convergence and scalability tests do not favor any of these methods over the others, and no theoretical analysis is presented for the general framework.

Bian et al. [1] presented a high-dimensional line search algorithm to ensure global convergence while performing parallel coordinate updates for $L_1$-regularized logistic regression problem. While this method performs well, no analysis of convergence rate is presented. In this paper, by further exploring the idea, we propose a generalized Parallel Coordinate Descent method using approximate Newton steps (PCDN) for generic $L_1$-optimization problems and present thorough theoretical analysis on the proposed method.

The contributions and novelty of this paper are summarized as follows. We present a theoretical analysis on the upper bound of the expected line search step in each iteration. We analyze the iteration complexity of the proposed PCDN algorithm and show that, for any bundle size $P$ (i.e., parallelism), it is guaranteed to converge to a specified accuracy $\epsilon$ within $T_\epsilon$ iterations. The iteration number $T_\epsilon$ decreases with the
increasing of parallelism (bundle size \( P \)). In addition, we show that in our implementation, the \( P \)-dimensional line search does not need to access all the training data on each thread, and the synchronization cost of the \( P \)-dimensional line search can be minimized. Extensive experiments on \( L_1 \)-regularized classification and regression problems with real-world data sets demonstrate that the proposed PCDN algorithm is a highly parallelized approach with guaranteed global convergence and fast convergence rate.

II. \( L_1 \)-REGULARIZED LOSS MINIMIZATION

For the ease of presentation, we summarize the mathematical notations in Table I.

Consider an unconstrained \( L_1 \)-regularized minimization problem over a training set \( \{ (x_i, y_i) \}_{i=1}^n \) with the following general form:

\[
\min_{w \in \mathbb{R}^n} F(w) := \min_{w \in \mathbb{R}^n} c \sum_{i=1}^n \phi(w; x_i, y_i) + \|w\|_1
\]

where \( L(w) := c \sum_{i=1}^n \phi(w; x_i, y_i) \) is the overall loss function, \( \phi(w; x_i, y_i) \) is a convex and nonnegative loss function, and \( c > 0 \) is the regularization parameter. For \( L_1 \)-regularized logistic regression, the loss function is

\[
\phi_{\log}(w; x_i, y_i) = \log(1 + e^{-y_i w^T x_i})
\]

and for \( L_1 \)-regularized \( L_2 \)-loss SVM, the loss function is

\[
\phi_{\text{svm}}(w; x_i, y_i) = \max(0, 1 - y_i w^T x_i)^2.
\]

A number of algorithms have been proposed to solve these problems. We discuss two related solvers based on CDN [26] and its parallel variant, SCDN [2], in this section.

A. Coordinate Descent Newton

Based on the CGD method [24], Yuan et al. [26] demonstrate that the CDN method is efficient for solving large-scale \( L_1 \)-regularized minimization. The overall procedure is summarized in Algorithm 1. Given the current model \( w \), for the selected feature \( j \in \mathcal{N} \), \( w \) is updated in the direction

\[
d^j = d(w; j)e_j, \ \text{where}^1
\]

\[
d(w; j) := \arg \min_d \left[ \nabla_j L(w)d + \frac{1}{2}\nabla^2_j L(w)d^2 + |w_j + d| \right]
\]

which has the following closed-form solution:

\[
d(w; j) = \begin{cases} 
\frac{-\nabla_j L(w) + 1}{\nabla^2_j L(w)} & \text{if } \nabla_j L(w) + 1 \leq \nabla^2_j L(w)w_j \\
\frac{-\nabla_j L(w) - 1}{\nabla^2_j L(w)} & \text{if } \nabla_j L(w) - 1 \geq \nabla^2_j L(w)w_j \\
-w_j & \text{otherwise.}
\end{cases}
\]

The Armijo rule [3] is used to determine the step size. Let \( q \) be the line search step index, and the step size \( \alpha = \alpha(w, d) \) is determined by

\[
\alpha(w, d) := \max_{q=0, 1, \ldots} \{ \beta^q |F(w + \beta^q d) - F(w) | \leq \beta^q \sigma \Delta \}
\]

where \( \beta \in (0, 1), \sigma \in (0, 1), \) and

\[
\Delta := \nabla L(w)^T d + \gamma d^T H d + \|w + d\|_1 - \|w\|_1
\]

A. Coordinate Descent Newton

Based on the CGD method [24], Yuan et al. [26] demonstrate that the CDN method is efficient for solving large-scale \( L_1 \)-regularized minimization. The overall procedure is summarized in Algorithm 1. Given the current model \( w \), for the selected feature \( j \in \mathcal{N} \), \( w \) is updated in the direction

\[
d^j = d(w; j)e_j, \ \text{where}^1
\]

\[
d(w; j) := \arg \min_d \left[ \nabla_j L(w)d + \frac{1}{2}\nabla^2_j L(w)d^2 + |w_j + d| \right]
\]

which has the following closed-form solution:

\[
d(w; j) = \begin{cases} 
\frac{-\nabla_j L(w) + 1}{\nabla^2_j L(w)} & \text{if } \nabla_j L(w) + 1 \leq \nabla^2_j L(w)w_j \\
\frac{-\nabla_j L(w) - 1}{\nabla^2_j L(w)} & \text{if } \nabla_j L(w) - 1 \geq \nabla^2_j L(w)w_j \\
-w_j & \text{otherwise.}
\end{cases}
\]

The Armijo rule [3] is used to determine the step size. Let \( q \) be the line search step index, and the step size \( \alpha = \alpha(w, d) \) is determined by

\[
\alpha(w, d) := \max_{q=0, 1, \ldots} \{ \beta^q |F(w + \beta^q d) - F(w) | \leq \beta^q \sigma \Delta \}
\]

where \( \beta \in (0, 1), \sigma \in (0, 1), \) and

\[
\Delta := \nabla L(w)^T d + \gamma d^T H d + \|w + d\|_1 - \|w\|_1
\]
Based on the spectral radius measure for the parallelization potential of the SCDN method, we can show that

\[ \rho = \frac{1}{1 + \frac{\text{tr} H}{n}} \]

is guaranteed to reach global convergence for high parallelism, whereas SCDN does not. The PCDN algorithm sequentially processes each bundle by computing the approximate Newton descent direction in each iteration of the inner loop. In the \( t \)-th iteration, the \( P \)-dimensional approximate Newton descent direction is computed by

\[ d = \arg \min_{d} \left( \nabla_{d} L(w) \right) d + \frac{1}{2} d^{T} H_{d} d + \| w_{d} + d \| \]

where we only use the diagonal elements of the Hessian, i.e., \( H_{d} \equiv \text{diag}(\nabla_{d}^{2} L(w)) \) to make the computing of 1-D Newton descent direction independent of each other and enable the parallelization. That is

\[ d = \arg \min_{d} \left( \nabla_{d} L(w) \right) d + \frac{1}{2} d^{T} H_{d} d + \| w_{d} + d \| \]

where (9) is from the definition of \( d(w; j) \) in (4). In the \( t \)-th iteration, we first compute the 1-D descent directions \( d_{j}^{t} \) (step 7) for the \( P \) features in \( B' \) in parallel, which constitutes the \( P \)-dimensional descent direction \( d^{t} \) (\( d_{j}^{t} = 0, \forall j \notin B' \)). We then use the \( P \)-dimensional Armijo line search (step 6) to compute the step size \( \alpha^{t} \) of the bundle along \( d^{t} \) and update the model for the features in \( B' \) (step 6).

The PCDN algorithm is different from the SCDN method in three aspects: 1) PCDN randomly partitions the feature set into bundles and performs parallelization for features of each bundle, while SCDN does not; 2) PCDN performs \( P \)-dimensional line search for a bundle of features, while SCDN performs 1-D line search for each feature; and 3) PCDN is guaranteed to reach global convergence for high parallelism, whereas SCDN is not.

The \( P \)-dimensional line search is the key procedure that guarantees the convergence of PCDN. With \( P \)-dimensional line search, the objective function \( F(w) \) in (1) is ensured to be nonincreasing for any bundle \( B' \) [see Lemma 1-3]. In general, the \( P \)-dimensional line search tends to have a large step size when the features in \( B' \) are less correlated, and a small step size otherwise.

The bundle size \( P \) controls the ratio between computation and data communication. From Algorithm 3, in each outer iteration, it updates \( n \) features (computation) while conducts \( \lceil (n/P) \rceil \) times high-dimensional line search (which requires synchronization and communication). The bundle size \( P \) affects convergence rate (see Theorem 2) as well, and the choice of \( P \) is discussed in Section V-A.

\[ N = B^{kb} \cup B^{kb+1} \cup \cdots \cup B^{(k+1)b-1}, \quad k = 0, 1, 2, \ldots \]

where \( B \) denotes a subset, i.e., a bundle, in this paper; \( P = |B| \) is the bundle size; and \( b = \lceil (n/P) \rceil \) is the number of bundles partitioned from \( N \). The PCDN algorithm sequentially processes each bundle by computing the approximate Newton descent direction in each iteration of the inner loop. In the \( t \)-th iteration, the \( P \)-dimensional approximate Newton descent direction is computed by

\[ d = \arg \min_{d} \left( \nabla_{d} L(w) \right) d + \frac{1}{2} d^{T} H_{d} d + \| w_{d} + d \| \]

where (9) is from the definition of \( d(w; j) \) in (4). In the \( t \)-th iteration, we first compute the 1-D descent directions \( d_{j}^{t} \) (step 7) for the \( P \) features in \( B' \) in parallel, which constitutes the \( P \)-dimensional descent direction \( d^{t} \) (\( d_{j}^{t} = 0, \forall j \notin B' \)). We then use the \( P \)-dimensional Armijo line search (step 6) to compute the step size \( \alpha^{t} \) of the bundle along \( d^{t} \) and update the model for the features in \( B' \) (step 6).

The PCDN algorithm is different from the SCDN method in three aspects: 1) PCDN randomly partitions the feature set into bundles and performs parallelization for features of each bundle, while SCDN does not; 2) PCDN performs \( P \)-dimensional line search for a bundle of features, while SCDN performs 1-D line search for each feature; and 3) PCDN is guaranteed to reach global convergence for high parallelism, whereas SCDN is not.

The \( P \)-dimensional line search is the key procedure that guarantees the convergence of PCDN. With \( P \)-dimensional line search, the objective function \( F(w) \) in (1) is ensured to be nonincreasing for any bundle \( B' \) [see Lemma 1-3]. In general, the \( P \)-dimensional line search tends to have a large step size when the features in \( B' \) are less correlated, and a small step size otherwise.

The bundle size \( P \) controls the ratio between computation and data communication. From Algorithm 3, in each outer iteration, it updates \( n \) features (computation) while conducts \( \lceil (n/P) \rceil \) times high-dimensional line search (which requires synchronization and communication). The bundle size \( P \) affects convergence rate (see Theorem 2) as well, and the choice of \( P \) is discussed in Section V-A.
The PCDN algorithm can better exploit parallelism than the SCDN method. In step 7 of Algorithm 3, the descent direction for \( P \) features can be computed in parallel on \( P \) threads. We show in Section IV that the proposed PCDN algorithm is guaranteed to reach global convergence, for any \( P \in [1, n] \). Therefore, the bundle size \( P \) which measures the parallelism can be large when the number of features \( n \) is large. In contrast, for SCDN, the number of parallel updates \( \tilde{P} \) is no more than \( n/\rho + 1 \) [2].

### A. Implementation Techniques of PCDN on Multicore

We use the technique of retaining intermediate quantities, in a way similar to the that in [6], by which two crucial implementation issues are addressed simultaneously. First, due to limited memory bandwidth, we lower data transfer by minimizing the number of intermediate quantities in the following equation

\[
F(w + \beta \theta d) - F(w) = \|w + \beta \theta d\|_1 - \|w\|_1 + \beta \sum_{i,j=1}^{s} d^T x_i,
\]

which is equivalent to the descent condition in (6). More specifically, in Algorithm 3, the core processing the \( j \)th feature only needs to access \( x^j \) twice in the \( t \)th iteration.

For the first time at step 7 of Algorithm 3, \( x^j \) is accessed and the retained \( e^{w^T x_i} \) is used to compute the \( j \)th gradient and Hessian

\[
\nabla_j L(w) = c \sum_{i=1}^{s} \tau(y_i w^T x_i)(1 - \tau(y_i w^T x_i))x_{ij}
\]

\[
\nabla_j^2 L(w) = c \sum_{i=1}^{s} \tau(y_i w^T x_i)(1 - \tau(y_i w^T x_i))x_{ij}^2
\]

where \( \tau(s) = (1/(1 + e^{-s})) \). They are then used to compute \( d(w; j) \) in (4). For the second time at step 6 of Algorithm 3, \( x^j \) is accessed and \( d \) is used to update \( d^T x_i \), which is then used with \( e^{w^T x_i} \) to check the descent condition in (10).

The PCDN algorithm requires much less time for each outer iteration than the CDN method, which is analyzed in Appendix G.

### IV. ON CONVERGENCE OF PCDN

In this section, we analyze the convergence of the PCDN algorithm from three aspects: convergence of \( P \)-dimensional line search, global convergence, and convergence rate. For presentation clarity, we first discuss the main results and present all the proofs in the appendix. Before analyzing the convergence of PCDN, we present Lemma 1.

**Lemma 1**: Let \{\( w \)\}, \{\( d \)\}, \{\( \alpha' \)\} as well as \{\( B' \)\} be sequences generated by Algorithm 3, \( \lambda(B') \) be the maximum element of \( (X^T X)_{jj} \) where \( j \in B' \), and \( \lambda_k \) be the \( k \)th minimum element of \( (X^T X)_{jj} \) where \( j \in N \). The following results hold.

1) \( \mathcal{E}_B[\lambda(B')] \) is monotonically increasing with respect to \( P \), \( \mathcal{E}_B[\lambda(B')] \) is constant with respect to \( P \) if \( x_i \) is constant (i.e., \( \lambda_1 = \cdots = \lambda_n \)), and \( \mathcal{E}_B[\lambda(B')] / P \) is monotonically decreasing with respect to \( P \).

2) For \( L_1 \)-regularized logistic regression in (2) and \( L_1 \)-regularized \( L_2 \)-loss SVMs in (3), the diagonal elements of the (generalized) Hessian of the loss function \( L(w) \) have positive lower bound \( \bar{h} \) and upper bound \( h \), and the upper bound only depends on the design matrix \( X \). That is, \( \forall j \in N \)

\[
\nabla_{jj}^2 L(w) \leq \theta c (X^T X)_{jj} = \theta c \sum_{i=1}^{s} x_{ij}^2 \leq \frac{\theta c}{1 - (1/\rho)} \leq \theta c \lambda(N)
\]

where \( \theta = (1/\gamma) \) for logistic regression and \( \theta = 2 \) for \( L_2 \)-loss SVM.

3) The objective \{\( F(w') \)\} is nonincreasing and \( \Delta' \) (7) in the Armijo line search rule satisfies

\[
\Delta' \leq (\gamma - 1)(d')^T H' d' \leq F(w') - F(w) \leq \sigma \alpha' \Delta' \leq 0.
\]

We note that Lemma 1-1 is used to analyze the iteration number \( T_e \) given the expected accuracy \( \epsilon \). Lemma 1-2 is used to prove Theorems 1 and 2, and Lemma 1-3 ensures the descent of the objective theoretically, gives an upper bound for \( \Delta' \) in the Armijo line search, and is used to prove Theorems 1 and 2. Note that the upper bound \( (\gamma - 1)(d')^T H' d' \) is only related to the second-order measurement.

**Theorem 1 (Convergence of \( P \)-Dimensional Line Search)**: Let \{\( B' \)\} be a sequence generated by Algorithm 3, and

```plaintext
Algorithm 4: Efficient High-Dimensional Armijo Line Search (Logistic Regression Here for Example)

1. compute \( d^T x_i, i = 1, \ldots, s \); // parallel
2. for \( q = 0, 1, 2, \ldots \) do
3.   if (10) is satisfied then
4.     \( w \leftarrow w + \beta \theta d; \)
5.     \( e^{w^T x_i} \leftarrow e^{w^T x_i} e^{(\beta \theta d^T x_i)} \); // parallel
6.     break;
7. else
8.     \( \Lambda \leftarrow \beta \Delta; \)
9.     \( d^T x_i \leftarrow \beta d^T x_i, i = 1, \ldots, s \); // parallel
```
δ(B') = \max\{(X^TX)_{jj} \mid j \in B'\}. The P-dimensional line search converges in finite steps, and the expected line search step number in each iteration is bounded by
\[ E[q^t] \leq 1 + \log_{\beta^{-1}} \left( \frac{\theta c}{2h(1 - \sigma + \sigma \gamma)} + \frac{1}{2} \log_{\beta^{-1}} P + \log_{\beta^{-1}} E[\tilde{\lambda}(B')] \right) \tag{16} \]
where the expectation is with respect to the random choice of B', q^t is the line search step number in the rth iteration, \( \beta \in (0, 1) \), \( \sigma \in (0, 1) \) and \( \gamma \in [0, 1) \) are the parameters of the Armijo rule (6), and \( \theta \) and \( h \) is in Lemma 1-2.

As \( E[\tilde{\lambda}(B')] \) is monotonically increasing with respect to \( P \) (Lemma 1-1), Theorem 1 dictates that the upper bound of \( E[q^t] \) (the expected line search step number in each iteration) increases with the bundle size \( P \). Since more line search steps lead to smaller step size (a), Theorem 1 is consistent with the intuition that smaller step size is used when features inside a bundle are more correlated.

A. Global Convergence of PCDN

In Appendix E, we prove the global convergence of PCDN by connecting it to the general framework in [24]. By proving that all assumptions are satisfied, we show that, assuming that \( \{w^t\} \) is the sequence generated by Algorithm 3, any limit point of \( \{w^t\} \) is an optimum. This analysis guarantees that the PCDN algorithm converges globally for any bundle size \( P \in [1, n] \) (i.e., without regard to the level of parallelism).

Theorem 2 (Convergence Rate of PCDN): Assume \( w^* \) minimize (1), \( \{w^t\} \) and \( \{B^t\} \) be sequences generated by Algorithm 3, and \( \tilde{\lambda}(B') := \max\{(X^TX)_{jj} \mid j \in B'\} \) and \( w^\dagger \) be the output of Algorithm 3 after \( T + 1 \) iterations. Then
\[ E[F(w^\dagger)] - F(w^*) \leq \frac{nE[\tilde{\lambda}(B')]}{P(T + 1)} \cdot \frac{\theta c}{2\bar{c}} \left( \frac{\|w^*\|^2 + F(0)}{\sigma(1 - \gamma)\bar{h}} \right) \]
where the expectation is computed with respect to the random choice of \( B' \), and \( \sigma \in (0, 1) \) and \( \gamma \in [0, 1) \) are the parameters in the Armijo rule (6). In addition, \( \theta \) and \( \bar{h} \) are given in Lemma 1-2, and \( E[\tilde{\lambda}(B')] \) is determined by the bundle size \( P \) and design matrix \( X \); \( \bar{c} \) is a positive constant.

Based on Theorem 2, we obtain the upper bound \( (T^\uparrow_e) \) of the iteration number \( T_e \) satisfying a specified accuracy \( \epsilon \)
\[ T_e \leq \frac{nE[\tilde{\lambda}(B')]}{P\epsilon} \cdot \frac{\theta c}{2\bar{c}} \left( \frac{\|w^*\|^2 + F(0)}{\sigma(1 - \gamma)\bar{h}} \right) \]
\[ := T^\uparrow_e \propto E[\tilde{\lambda}(B')] \tag{17} \]
which means that PCDN achieves speedups linear in the bundle size \( P \) if \( E[\tilde{\lambda}(B')] \) remains constant.\(^3\) In general, \( E[\tilde{\lambda}(B')] \) increases with respect to \( P \) (from Lemma 1-1), and thus makes the speedup sublinear. Furthermore, since \( E[\tilde{\lambda}(B')] \) decreases with respect to \( P \) from Lemma 1-1, \( T^\uparrow_e \) decreases with respect to \( P \), and so does \( T_e \). Thus, the PCDN algorithm requires fewer iterations with larger bundle size \( P \).

\(^3\) If we perform featurewise normalization over the training data \( X \) to ensure \( \lambda_1 = \lambda_2 = \cdots = \lambda_n \), then \( E[\tilde{\lambda}(B')] \) remains constant according to Lemma 1-1.

To verify the upper bound \( T^\uparrow_e \) (17) of the iteration number \( T_e \) for a given accuracy \( \epsilon \), we set \( \epsilon = 10^{-3} \) and show the iteration number \( T_e \) as a function of \( P \) in Fig. 1, where two document data sets, \( \text{a9a} \) and \( \text{real-sim} \) (see Section V-A for details about the data sets) are used. Since \( T^\uparrow_e \) is proportional to \( E[\tilde{\lambda}(B')] \) (see Section V-A), we plot \( E[\tilde{\lambda}(B')] \) instead of \( T^\uparrow_e \) in Fig. 1. The results match the upper bound in (17): for a given \( \epsilon \), \( T_e \) (green solid lines) is positively correlated with \( E[\tilde{\lambda}(B')] \) (blue dotted lines). In addition, \( T_e \) decreases with respect to \( P \). These results show that with larger bundle size \( P \), fewer iterations are needed by the PCDN algorithm to converge to \( \epsilon \) accuracy.

V. EXPERIMENTS

In this section, we present the experimental results of the proposed PCDN algorithm with comparison to the state-of-the-art methods on \( L_1 \)-regularized loss minimization problems using several benchmark data sets. The source code of this paper is available at https://github.com/bianan/ParallelCDN.

A. Experimental Setup

1) Data Sets: Seven benchmark data sets\(^4\) are used in our experiments, and the characteristics are summarized in Table II. The \textit{news20}, \textit{rcv1}, \textit{a9a}, and \textit{real-sim} data sets consist of document data points that are normalized to unit vectors. The \textit{a9a} data set is from UCI data repository, and the \textit{gisette} set consists of handwriting digit data points from the NIPS 2003 feature selection challenge where features are linearly scaled to the \([-1, 1]\) interval. The \textit{kdd99} data set has been used for the KDD Cup 2010 data mining competition. The \textit{webspam} data set is the collection of Web pages that are created to manipulate search engines and deceive Web users.

2) Bundle Size Choice: For each data set, the optimal bundle size \( B^* \) under which PCDN achieves minimum runtime is determined as follows. For Algorithm 3, the expected runtime of the rth inner iteration of the PCDN algorithm time(\( r \)) can be approximated by
\[ E[\text{time}(r)] \approx (P/\#\text{thread}) \cdot t_{dc} + E[q^t] \cdot t_{ls} \tag{18} \]
where the expectation is based on a random choice of \( B' \), \#thread is the number of threads used by PCDN and fixed to be 23 in our experiments, \( t_{dc} \) is the time for computing the

\(^4\) The data sets are available at http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/ data sets.
**TABLE II**

**SUMMARY OF DATA SETS.** The number of nonzero elements in training data is denoted by “NNZ.” The average number of NNZs corresponding to each feature is “NNZ/Feature.” “Sparse” means training data sparsity, which is the ratio of zero elements in X. “c* SVM” and “c* Logistic” denote the best regularization parameter c* for L2-loss SVM and logistic regression, respectively, which are determined according to [26].

| Data Set | P | n_t | NNZ | NNZ/Feature | Sparsity% | c* SVM | c* Logistic |
|----------|---|-----|-----|-------------|----------|-------|-------------|
| a9a      | 26,049 | 123 | 361,276 | 2,937 | 88.72 | 0.5 | 2.0 |
| real-sim | 57,848 | 20,958 | 2,968,110 | 142 | 99.76 | 1.0 | 4.0 |
| news20   | 15,997 | 1,355,191 | 7,281,110 | 5 | 99.97 | 64.0 | 64.0 |
| gisette  | 6,000  | 5,000 | 29,729,997 | 5,946 | 0.9 | 0.25 | 0.25 |
| rcv1     | 541,920 | 47,236 | 39,625,144 | 839 | 99.85 | 1.0 | 4.0 |
| kdda     | 8,407,752 | 20,216,830 | 305,613,310 | 15 | 99.99 | 1.0 | 1.0 |
| webspam  | 280,000 | 16,609,143 | 1,043,724,776 | 63 | 99.97 | 64.0 | 64.0 |

As $E[q^I]$ increases with respect to the bundle size $P$ (based on Theorem 1), $E[\text{time}(t)]$ increases with respect to $P$ based on (18). In addition, as the PCDN algorithm requires fewer iterations for larger $P$ to converge to $\epsilon$ accuracy [from (17)], it is essential to make a tradeoff between the increasing runtime per iteration $E[\text{time}(t)]$ and the decreasing iteration number $T_\epsilon$ to select the optimal bundle size $P^\ast$. In practice, we run PCDN with varying $P$. Fig. 2 shows the training time as a function of bundle size $P$ for the real-sim data set, and the optimal bundle size $P^\ast$ can be determined. In this paper, we empirically select the optimal $P^\ast$ for each data set (see Table III).

We note that it is not necessary to obtain the optimal $P$ to achieve significant speedup, as a wide range of $P$ will suffice to achieve the same goal in practice. As shown in Fig. 2(a), when $P$ is greater than 500, it achieves considerable speedup higher than SCDN (five times faster than SCDN for $P = 500$ from our experiments). For a new data set, one can first select a relatively large $P$ (about 5% of #features) with the most relaxed stopping criteria for a pilot experiment and then adjust $P$ for best performance when necessary.

3) Evaluated Methods: We evaluate the proposed PCDN algorithm against the state-of-the-art $L_1$-regularized optimization approaches, including newGLMNET [27], CDN [26], SCDN$^5$ [2], interior-point method (IPM) [9], and trust region Newton (TRON) [13] methods with C/C++ implementations. For the Armijo line search procedure (6) in the PCDN, CDN, and SCDN methods, we set $\sigma = 0.01$, $\gamma = 0$, and $\beta = 0.5$ for fair comparisons. The OpenMP library is used for parallel programming. The stopping criteria used in the experiments are similar to the outer stopping condition used in [27]. The implementation details are listed in the following.

1) CDN: We implement this method based on the source code in the LIBLINEAR$^6$ toolbox. Since the shrinking procedure cannot be performed inside the parallel loop of the SCDN and PCDN methods, we use an equivalent implementation of the CDN scheme for fair comparisons, where the shrinking procedure is modified such that it is consistent with the other parallel algorithms.

2) SCDN: We set $\bar{P} = 8$ for the SCDN method following Bradley et al. [2].

3) PCDN: We implement this algorithm with conditions consistent with all other methods.

4) TRON: We set $\sigma = 0.01$ and $\beta = 0.1$ in the projected line search according to Yuan et al. [27]. We use it as baseline algorithm for $L_2$-loss SVM.

5) newGLMNET: We use the same setting and implementation provided by Yuan et al. [27]. Since it is outperformed by CDN for $L_2$-loss SVM, we only use it as baseline algorithm for logistic regression experiments.

6) IPM: We use the source code$^7$ and default settings in [9]. We use it as a baseline interior-point algorithm for logistic regression.

4) Platform: All experiments are carried out on a 64-bit machine with Intel Xeon 2.4-GHz CPU and 64-GB main memory. We set #thread = 23 for PCDN on a 24-core machine, which is far less than the optimal bundle size $P^\ast$ given in Table III. We note that the descent direction (step 7 in Algorithm 3) of the PCDN algorithm can be fully parallelized on several hundreds even to thousands of threads.

---

$^5$Since the experimental validation in [2] has shown that SCDN is much faster than the SGD-type algorithms (including SGD, Parallel SGD [17], [28], and SMIDAS [22]) for data sets with more features and SCDN performs well on data sets with more samples than features, we only compare the PCDN algorithm with the SCDN scheme here. Note that SCDN is also a competitive representative of the generic parallel coordinate descent algorithms in [20].

$^6$liblinear version 1.7, http://www.csie.ntu.edu.tw/~cjlin/liblinear/.

$^7$version 0.8.2, http://www.stanford.edu/~boyd/l1_logreg/
C. L1-Regularized Logistic Regression

is a relatively dense data set with fewer features than samples
is sometimes slightly slower than the TRON method since it
For the
number of features (47 236 and 1 355 191). In such cases,
approach. We note that for the
in Table II, is 99
X
ratio of zero elements in design matrix
which are very sparse (training data sparsity, defined by the
(determined based on [26]) and varying stopping criteria
In addition, the computational cost of line search can be further
explained by Amdahl’s law. First, as the number of cores
increases, the parallelized part takes less and less time. How-
the correlation among features and memory bandwidth limit,
another issue that would significantly affect the performance of the PCDN method is the workload of the parallel threads. For the PCDN algorithm, each thread first processes one feature of the data and then switches to the next feature. Thus, the parallel processing time of PCDN, which contributes to the acceleration, is proportional to the workload of the parallel threads. The workload of each thread is approximately proportional to the number of nonzero elements (NNZs) of the data corresponding to the feature being processed. To verify that, we compute the average number of NNZs per feature (NNZ/feature column in Table II) and show that there are only 15 NNZs/feature in the kdda data set, while there are 63 NNZs/feature in the webspam data set. These results explain the performance difference of the PCDN algorithm on these two large data sets.

D. Scalability of PCDN

We evaluate the scalability of the PCDN method in two
aspects: whether this method can maintain the speed-up when the data size is increased and whether it can achieve better speed-up when the available computing resource (e.g., the number of cores) is increased.

To analyze the effect of data size, we maintain all the
other factors, e.g., correlation among features, the same in the experiments. To this end, we duplicate the samples to create data sets from 100% of original size to 1000%. Fig. 5 shows the scalability over different numbers of cores and data size.

1) Effect of Number of Cores: Fig. 5(a) shows that the speedup of the PCDN algorithm is larger at the beginning when the number of cores is increased (i.e., the parallel efficiency decreases with more parallelism), which can be explained by Amdahl’s law. First, as the number of cores increases, the parallelized part takes less and less time. However, the serial part takes approximately the same constant time. Second, with more cores, there is increasing parallelization overhead, e.g., more data transfer, and thereby lowering parallel efficiency.

2) Effect of Data Size: Fig. 5(b) shows that the speedup is approximately constant with larger data size, which shows the weak scaling property of parallel algorithms. It is noteworthy that for very large data set, the size of the data for each feature is also quite large that it may exceed the memory bandwidth.

E. Discussions

The high-dimensional line search plays the key role in ensuring global convergence of the PCDN algorithm. In this paper, we use the Armijo line search as a specific realization, and it is worth exploring other ways to perform the line search. In addition, the computational cost of line search can be further
develop a principled approach to determine the optimal value for $P$.

Although we present an empirical method to choose a good $P$, it is of great interest to develop a principled approach to determine the optimal value for $P$. The bundle size $P$ controls the ratio between computation and communication, thus affecting the running time of the PCDN method.

Another direction to pursue is to extend the PCDN algorithm within a distributed framework in a way similar to the Parallel SGD [29] and Downpour SGD [5] methods, to deal with very large data sets with lots of samples, that do not fit into one single machine. This can be achieved by first randomly distributing training data of different samples to different machines (i.e., parallelizing over samples), applying the PCDN algorithm over a subset (i.e., parallelizing over features) on each machine, and aggregating all the models in the end. As a shared memory parallel algorithm, the PCDN algorithm can also be implemented with the stale synchronous parallel model [8] to achieve a better performance. As a fast local $L_1$ solver, PCDN can be plugged into distributed solvers, such as CoCoA [23] and its decentralized variant CoLa [7], for huge-scale $L_1$ regularized problems.

VI. Conclusion

We propose an algorithm termed PCDN with strong convergence guarantee, fast convergence rate, and high parallelism for $L_1$-regularized minimization problems. We show that the seemingly expensive high-dimensional line search can be calculated efficiently with the implementation technique of maintaining intermediate quantities, which also minimizes the data transfer and synchronization cost of the PCDN algorithm.

The PCDN method can be generalized to solve the problems of minimizing the sum of a convex twice differentiable loss term and a separable regularization term. Thus, it allows $L_1$ (lasso), $L_2$ (ridge regression), and the mixtures of the two penalties (elastic net). Experimental results on several benchmark data sets show that the proposed PCDN algorithm performs favorably against several state-of-the-art methods for $L_1$-regularized optimization problems.

APPENDIX

FULL PROOFS OF THEOREMS 1 AND 2

A. Proof of Lemma 1-1

Proof: We first prove that $E_{B^*} [\hat{\lambda} (B^*)]$ is monotonically increasing with respect to $P$ and $E_{B^*} [\hat{\lambda} (B^*)]$ is constant with respect to $P$, if $\hat{\lambda}_1 = \hat{\lambda}_2 = \cdots = \hat{\lambda}_n$.

Let $\hat{\lambda}_k$ be the $k$th minimum of $(X^\top X)_{jj}, j = 1, \ldots, n$, for $1 \leq P \leq n$. We define

$$f (P) := E_{B^*} [\hat{\lambda}(B^*)]$$

$$= (\hat{\lambda}_n C_{n-1}^{P-1} + \cdots + \hat{\lambda}_k C_{k-1}^{P-1} + \cdots + \hat{\lambda}_1 C_{1}^{P-1}) / C_n^P \quad (19)$$

where $C_n^P$ is a binomial distribution. For $1 \leq P \leq n - 1$

$$f (P + 1) - f (P)$$

$$= -\hat{\lambda}_P C_{P}^{P-1} C_n^P + \sum_{k=1}^{P+1} \hat{\lambda}_k (C_{k-1}^{P} C_n^P - C_{k}^{P-1} C_n^P)$$

$$= -\hat{\lambda}_P C_{P}^{P-1} C_n^P + \sum_{k=1}^{P+1} \hat{\lambda}_k (P + 1) k - P (n + 1) C_{k-1}^{P-1} C_n^P$$

$$= -\hat{\lambda}_P C_{P}^{P-1} C_n^P + \sum_{k=1}^{P+1} \hat{\lambda}_k (P + 1) k - P (n + 1) C_{k-1}^{P-1} C_n^P$$
When \( \bar{k} = \lfloor (P+1)k/P(n+1) \rfloor, (P+1)k - P(n+1) \geq \mathbf{0}, \forall k \geq \bar{k}, \) and \( (P+1)k - P(n+1) \leq \mathbf{0}, \forall k < \bar{k}. \) The above equation is equivalent to

\[
f(P + 1) - f(P) = \sum_{k = \bar{k}}^{P} \lambda_k \frac{(P+1)k - P(n+1) C_{k-1}^{P-1}}{P(n-P)C_n^P} - \lambda_P \frac{C_{P-1}}{C_n^P}.
\]

According to the observations that \( \lambda_k \geq \lambda_{\bar{k}}, \forall k \geq \bar{k} \) and \( \lambda_k \leq \lambda_{\bar{k}}, \forall k < \bar{k}, \) we can decrease the above equation by substituting \( \lambda_k \) with \( \lambda_{\bar{k}}. \) That is

\[
f(P + 1) - f(P) \geq \sum_{k = \bar{k}}^{P} \lambda_{\bar{k}} \frac{(P+1)k - P(n+1) C_{k-1}^{P-1}}{P(n-P)C_n^P} - \lambda_{\bar{k}} \frac{C_{P-1}}{C_n^P} \]

\[
= \lambda_{\bar{k}} \left[ \frac{1}{P} \sum_{k = \bar{k}}^{P} \frac{C_{k-1}^{P-1}}{C_n^P} - \frac{P}{P} \right] - \lambda_{\bar{k}} \frac{C_{P-1}}{C_n^P} \]

\[
= \lambda_{\bar{k}} \left[ \frac{1}{P} \sum_{k = \bar{k}}^{P} \frac{C_{k-1}^{P-1}}{C_n^P} - \frac{P}{P} \right] = \lambda_{\bar{k}}[1 - 1/\bar{k}] = 0.
\]

Thus, \( f(P + 1) - f(P) \geq 0, \forall 1 \leq P \leq n - 1. \) Namely, \( E_{\bar{B}}[\hat{L}(B')] \) is monotonically increasing with respect to \( P. \) Clearly, from (19), if \( \lambda_1 = \lambda_2 = \cdots = \lambda_n, \) then \( E_{\bar{B}}[\hat{L}(B')] = \lambda_1, \) which is constant with respect to \( P. \)

Next, we prove that \( E_{\bar{B}}[\hat{L}(B')] / P \) is monotonically decreasing with respect to \( P. \) Let \( \lambda_{\bar{k}} \) be the \( k \)th minimum of \( (X^T X)_{jj}, j = 1, \ldots, n. \) For \( 1 \leq P \leq n, \) define

\[
g(P) := E_{\bar{B}}[\hat{L}(B')] / P
\]

\[
= \frac{1}{P C_n^P} \left( \lambda_1 C_{n-1}^{P-1} + \cdots + \lambda_{\bar{k}} C_{k-1}^{P-1} + \cdots + \lambda_P C_{P-1}^{P-1} \right).
\]

For \( 1 \leq P \leq n - 1, \) we have

\[
g(P + 1) - g(P) = -\lambda_P \frac{C_{P-1}}{P C_n^P} + \sum_{k = \bar{k}}^{P+1} \lambda_k \frac{C_{k-1}^{P-1}}{(P+1)C_n^P} - \lambda_{\bar{k}} \frac{C_{P-1}^{P-1}}{P C_n^P} \]

\[
= -\lambda_P \frac{C_{P-1}}{P C_n^P} + \sum_{k = \bar{k}}^{P+1} \frac{\lambda_k}{P - n} \frac{C_{k-1}^{P-1}}{P C_n^P}.
\]

According to the observations that \( ((k - n)/(n - P)) \leq 0 \) and \( \lambda_k \geq \lambda_P, \forall k = n, \ldots, P + 1, \) we can increase the above equation by substituting \( \lambda_k \) with \( \lambda_P. \) That is

\[
g(P + 1) - g(P) \leq -\lambda_P \frac{C_{P-1}}{P C_n^P} + \sum_{k = \bar{k}}^{P+1} \frac{\lambda_k}{P - n} \frac{C_{k-1}^{P-1}}{P C_n^P} \]

\[
= \lambda_P \left[ \frac{1}{P} \sum_{k = \bar{k}}^{P+1} \frac{C_{k-1}^{P-1}}{P C_n^P} - \frac{P}{P} \right] - \lambda_P \frac{C_{P-1}}{P C_n^P} \]

\[
= \lambda_P \left[ \frac{1}{P} \sum_{k = \bar{k}}^{P+1} \frac{C_{k-1}^{P-1}}{P C_n^P} - \frac{P}{P} \right] \leq 0.
\]

where (20) comes from \( \lambda_P \geq 0 \) and \( (1/(P + 1)) - (1/P) < 0. \) Thus, \( g(P + 1) - g(P) \leq 0, \forall 1 \leq P \leq n - 1. \) Namely, \( (E_{\bar{B}}[\hat{L}(B')] / P) \) is monotonically decreasing with respect to \( P. \)

\[\tag{20}\]

B. Proof of Lemma 1-2

Proof: First, for logistic regression

\[
\nabla_{ij}^2 L(w) = c \sum_{i=1}^{n} \tau(y_i w^T x_i)(1 - \tau(y_i w^T x_i)) x_{ij}^2
\]

where \( \tau(s) \equiv (1/(1 + e^{-s})) \) is the derivative of the logistic loss function \( \log(1 + e^s). \) Because \( 0 < \tau(s) < 1, \) we have \( 0 < \nabla_{ij}^2 L(w) \leq (1/4)c \sum_{i=1}^{n} x_{ij}^2 \) (the equal sign holds when \( \tau(s) = (1/2). \) and thus (12) holds when \( \theta = (1/4) \) for logistic regression. As \( \hat{L}(N) \) is the maximum element of \( (X^T X)_{jj} \) where \( j \in N, \) \( \nabla_{jj}^2 L(w) \leq \bar{h} = \theta c \hat{L}(N) \) in (13) also holds. In addition, because in practice \( |y_i w^T x_i| < \infty, \) there exist \( \bar{t} \) and \( \bar{h} \) such that \( 0 < \bar{t} \leq \tau(y_i w^T x_i) \leq \tau \leq \bar{h}. \) Thus, there exists \( \bar{h} > 0 \) such that \( 0 < \bar{h} \leq \nabla_{jj}^2 L(w). \)

Second, for \( L_2 \)-loss SVM, use generalized second derivative

\[
2c \sum_{i \in I(w)} x_{ij}^2 \leq 2c \sum_{i=1}^{s} x_{ij}^2 = \|X^T X_{jj}\|
\]

where \( I(w) = \{i \mid y_i w^T x_i < 1\}. \) So, (12) holds for \( \theta = 2 \) for \( L_2 \)-loss SVM. Because \( \hat{L}(N) \) is the maximum element of \( (X^T X)_{jj} \) where \( j \in N, \) \( \nabla_{jj}^2 L(w) \leq \bar{h} = \theta c \hat{L}(N) \) in (13) also holds. To ensure that \( \nabla_{jj}^2 L(w) > 0, \) a very small positive number \( v (v = 10^{-12}) \) is added when \( \nabla_{jj}^2 L(w) \leq 0 \) according to [4]. Thus, \( \bar{h} = v > 0. \)

C. Proof of Lemma 1-3

Proof: We follow the proof in [24], from (4) and the convexity of \( L_1 \)-norm, for any \( a \in (0, 1), \)

\[
\nabla L(w)^T d + \frac{1}{2} d^T H d + \|w + d\|
\]

\[
\leq \nabla L(w)^T (\alpha d) + \frac{1}{2} (\alpha d)^T H (\alpha d) + \|w + (\alpha d)\|
\]
\[ = a \nabla L(w)^T d + \frac{1}{2} a^2 d^T H d + \|a(w + d) + (1 - a)w\|_1 \]
\[ \leq a \nabla L(w)^T d + \frac{1}{2} a^2 d^T H d + a\|w + d\|_1 + (1 - a)\|w\|_1. \]
After rearranging these terms, we have
\[ (1 - a)\nabla L(w)^T d + (1 - a)(\|w + d\|_1 - \|w\|_1) \]
\[ \leq -\frac{1}{2} (1 - a)(1 + a)d^T H d. \]
Dividing both sides by \(1 - a > 0\) and taking \(a\) infinitely approaching 0 yields
\[ \nabla L(w)^T d + \|w + d\|_1 - \|w\|_1 \leq -d^T H d \]
and thus
\[ \Delta = \nabla L(w)^T d + \gamma d^T H d + \|w + d\|_1 - \|w\|_1 \leq \gamma (\gamma - 1)d^T H d \]
which proves (14). From the Armijo rule in (6), we have
\[ F(w + a\delta) - F(w) \leq \sigma a \Delta. \]
By substituting (23) into the above equation and considering that \(\gamma \in [0, 1]\), we obtain
\[ F(w + a\delta) - F(w) \leq \sigma a (\gamma - 1)d^T H d \leq 0. \]
Hence, \(\{F(w')\}\) is nonincreasing. \(\Box\)

D. Proof of Theorem 1: Convergence of P-Dimensional Line Search

Proof: First, we prove that the descent condition in (6)
\[ F(w + a\delta) - F(w) \leq \sigma a \Delta \]
is satisfied for any \(\sigma \in (0, 1)\) whenever \(0 \leq a \leq \min\{1, (2h(1 - \sigma + \gamma)/\theta \sqrt{\bar{P}_j(B')}\}.\)

For any \(a \in [0, 1]\)
\[ F(w + a\delta) - F(w) \]
\[ = L(w + a\delta) - L(w) + \|w + a\delta\|_1 - \|w\|_1 \]
\[ = \int_0^1 \nabla L(w + a\delta)^T (a\delta) du \]
\[ + \|w + a\delta\|_1 - \|w\|_1 \]
\[ = a\nabla L(w)^T d + \|w + a\delta\|_1 - \|w\|_1 \]
\[ + \int_0^1 (\nabla L(w + a\delta) - \nabla L(w))^T (a\delta) du \]
where (24) is based on the definition of definite integration. Because in the \(r\)th iteration of PCDN, \(d_j = 0, \forall j \not\in B'\), we define auxiliary matrix \(G \in \mathbb{R}^{n \times p}\) such that \(g_{jj} = 1, \forall j \in B'\), otherwise \(g_{jj} = 0\). Then, we have
\[ (\nabla L(w + a\delta) - \nabla L(w))^T (a\delta) \]
\[ = (G \cdot (\nabla L(w + a\delta) - \nabla L(w)))^T (a\delta). \]
Substituting (26) into (25), we obtain
\[ F(w + a\delta) - F(w) \]
\[ = a\nabla L(w)^T d + \|w + a\delta\|_1 - \|w\|_1 \]
\[ + \int_0^1 (G \cdot (\nabla L(w + a\delta) - \nabla L(w)))^T (a\delta) du \]
\[ \leq a\nabla L(w)^T d + a(\|w + d\|_1 - \|w\|_1) \]
\[ + a \int_0^1 \|G \cdot (\nabla L(w + u\delta) - \nabla L(w))\| du \]
where (27) is from the convexity of \(L_1\)-norm and the Cauchy–Schwarz inequality. It follows that
\[ \|G \cdot (\nabla L(w + u\delta) - \nabla L(w))\| \]
\[ = \sqrt{\sum_{j \in B'} (\nabla_j L(w + u\delta) - \nabla_j L(w))^2} \]
\[ \leq u a \sqrt{\sum_{j \in B'} (\nabla_j L(w))^2} \|d\| \]
\[ \leq u a \theta c \sqrt{\bar{P}_j(B')} \|d\| \]
\[ = u a \theta c \sqrt{\bar{P}_j(B')} \|d\| \]
which satisfies (28).

If we set \(a \leq (2h(1 - \sigma + \gamma)/\theta \sqrt{\bar{P}_j(B')}\), then
\[ a \theta c \sqrt{\bar{P}_j(B')} \|d\|^2 - a \gamma d^T H d \]
\[ \leq a(\|w + a\delta\|_1 - d^T H d) \]
\[ \leq a(1 - \sigma)(1 - \gamma)d^T H d \]
\[ \leq a(1 - a) \Delta \]
where (30) comes from (13) in Lemma 1-2 and (31) is based on Lemma 1-3. The above equation together with (29) proves that \(F(w + a\delta) - F(w) \leq \sigma a \Delta\) if \(a \leq (2h(1 - \sigma + \gamma)/\theta \sqrt{\bar{P}_j(B')}\).

Second, we prove the upper bound of \(E[q']\). In the Armijo line search procedure, it tests different values of \(a\) from larger to smaller and stops right after finding one value that satisfies \(F(w' + a'\delta') - F(w') \leq a' \Delta'\). Thus, in the \(r\)th iteration, the chosen step size \(a'\) satisfies
\[ a' \geq \frac{2h(1 - \sigma + \gamma)}{\theta \sqrt{\bar{P}_j(B')}}. \]
(32)

From (32), we have \(a' = \beta q'\), and thus the line search step number of the \(r\)th iteration \(q'\)
\[ q' = 1 + \log_{\beta} a' \leq 1 + \log_{\beta - 1} \frac{\theta c \sqrt{\bar{P}_j(B')}}{2h(1 - \sigma + \gamma)} \]
(33)
Taking expectation on both sides with respect to the random choices of $B'$, we obtain
\[
E[q'] \leq 1 + \log_{\beta^{-1}} \left( \frac{\theta c}{2h(1 - \sigma + \sigma \gamma)} + \frac{1}{2} \log_{\beta^{-1}} P \right) + \left( \log_{\beta^{-1}} \lambda(B') \right)
\]
where (34) is based on Jensen’s inequality for concave function $\log_{\beta^{-1}}()$.

**E. Proof of Global Convergence**

**Proof:** We first relate PCDN to the framework in [24]. Note that the selection of bundle $B'$ in (8) is consistent with that used in CGD [24, eq. (12)]. For the descent direction computed in a bundle in Algorithm 3, we have
\[
d' = \sum_{j \in B'} d(w'; j)e_j
\]
\[
= \sum_{j \in B'} \arg \min_{d} \left\{ \nabla_j L(w')d + \frac{1}{2} \nabla_j^2 L(w')d^2 + |w'_j + d_j| \right\} e_j
\]
\[
= \arg \min_{d} \left\{ \sum_{j \in B'} \left( \nabla_j L(w')d_j + \frac{1}{2} \nabla_j^2 L(w')d_j^2 + |w'_j + d_j| \right) \right\}
\]
\[
\times |d_j = 0, \forall j \not\in B'\}
\]
\[
= \arg \min_{d} \left\{ \nabla L(w')^T d + \frac{1}{2} d^T H d + \|w + d\|_1 \right\}
\]
\[
\times |d_j = 0, \forall j \not\in B'\}
\]
\[
= d_H(w'; B')
\]
where (35) is derived by considering the definition of $d(w'; j)$ in (4), (36) is obtained by applying the setting of $H \equiv \text{diag}(\nabla^2 L(w))$, and (37) is defined by following the descent direction definition in [24, eq. (6)]. Therefore, the definition of direction computed is in a manner similar to CGD. Furthermore, since PCDN uses the Armijo line search for $d'$, by taking $H \equiv \text{diag}(\nabla^2 L(w))$, it is clear that we can use the framework in [24] to analyze the global convergence of PCDN.

Second, we use [24, Th. 1(e)] to prove the global convergence, which requires that $\{B'\}$ is chosen under the Gauss–Seidel rule and $\sup_t a' < \infty$. In (6), $a' \leq 1, t = 1, 2, \ldots$, which satisfies $\sup_t a' < \infty$. To ensure global convergence, Tseng and Yun make the following assumption:
\[
0 < h \leq \nabla_j^2 L(w) \leq \bar{h} \quad \forall j = 1, \ldots, n, \ t = 0, 1, \ldots
\]
which is fulfilled by Lemma 1-2. According to [24, Th. 1(e)], any cluster point of $\{w_t\}$ is a stationary point of $F(w)$.

**F. Proof of Theorem 2: Convergence Rate of PCDN**

To analyze the convergence rate, we transform (1) into an equivalent problem with a twice differentiable regularizer following [22]. Let $\hat{w} \in \mathbb{R}^{2n}$ with duplicated features$^8$ $\hat{x}_i \equiv \{x_i; -x_i\} \in \mathbb{R}^{2n}$, the problem becomes
\[
\min_{\hat{w} \in \mathbb{R}^{2n}} F(\hat{w}) \equiv c \sum_{j=1}^{s} \phi(\hat{w}; \hat{x}_i, y_i) + \frac{2n}{j=1} \hat{w}_j.
\]
The descent direction is computed by
\[
\hat{d}_j = \hat{d}(\hat{w}; j) \equiv \arg \min_{d} \left\{ \nabla_j L(\hat{w})d + \frac{1}{2} \nabla_j^2 L(\hat{w})d^2 + \hat{w}_j + \hat{d} \right\}
\]
\[
= -\left( \nabla_j L(\hat{w}) + 1 \right)/\nabla_j^2 L(\hat{w}).
\]
In the following proof, we omit the “∧” above each variables for the ease of presentation.

**Proof:** Assume that $w^*$ minimizes the objective in (38). Define the potential function as
\[
\Psi(w) \equiv \frac{\theta c\lambda(B')}{2} ||w - w^*||^2 + \frac{\theta c\lambda(B')}{2} \sup_t a' F(w)
\]
\[
= a ||w - w^*||^2 + b F(w)
\]
where we define
\[
a = \frac{\theta c\lambda(B')}{2}, \quad b = \frac{\theta c\lambda(B')}{2} \sup_t a'.
\]
Thus, we have
\[
\Psi(w) - \Psi(w + a d)
\]
\[
= a(||w - w^*||^2 - ||w + a d - w^*||^2) + b(F(w) - F(w + a d))
\]
\[
= a a(\bar{w}^T d + 2w^T (\bar{w} - a d) + b(F(w) - F(w + a d))
\]
\[
= a a(\bar{w}^T d + 2w^T (\bar{w} - a d) + b\sup_t a(\bar{w} - a d)^T H d)
\]
where (41) uses (14) and (15) in Lemma 1-3. Using the fact that $d_j = 0, \forall j \not\in B'$, we derive from (41) that
\[
\Psi(w) - \Psi(w + a d)
\]
\[
\geq \sum_{j \in B'} a a(\bar{w}^T d_j + 2w^T d_j - a d_j^2) + b\sup_t a(\bar{w} - a d)^T H d_j
\]
\[
= \sum_{j \in B'} a a(\bar{w}^T d_j + 2w^T d_j) + a[b\sup_t a(\bar{w} - a d)^T H d_j]
\]
\[
\geq \sum_{j \in B'} a a(\bar{w}^T d_j + 2w^T d_j)
\]
and (42) uses the fact that
\[
b\sup_t a(\bar{w} - a d)^T H d_j
\]
\[
= \frac{\theta c\lambda(B')}{2} \left[ \nabla^2_{jj} L(w) \sup_t a' - a \right]
\]
\[
\geq \frac{\theta c\lambda(B')}{2} \left( \sup_t a' - a \right) \geq 0.
\]

$^8$Although our analysis uses duplicate features, they are not required for an implementation.
By substituting \( a = (\theta c \lambda(B'))/2 \) and \( d_j = -(\nabla_j L(w) + 1)/\nabla_j^2 L(w) \) [see (39)] into (42), we have the following equations:

\[
\Psi(w) - \Psi(w + a d) \\
\geq \sum_{j \in B'} \frac{\theta c \lambda(B')}{\nabla_j^2 L(w)} (w_j - w_j^*) (\nabla_j L(w) + 1) \tag{43}
\]

\[
\geq \sum_{j \in B'} \left( \frac{\lambda(B')}{\langle X_j, X_j \rangle} (w_j - w_j^*) (\nabla_j L(w) + 1) \right) \tag{44}
\]

\[
\geq a \sum_{j \in B'} (w_j - w_j^*) (\nabla_j L(w) + 1). \tag{45}
\]

We note that (44) is based on Lemma 1-2 and (45) results from the definition of \( \lambda(B') \).

Taking the expectation with respect to the random choices of \( B' \) on both sides of (45), we have

\[
E_{B'}[\Psi(w) - \Psi(w + a d)] \\
\geq \inf_t a^t E_{B'} \left[ \sum_{j \in B'} (w_j - w_j^*) (\nabla_j L(w) + 1) \right] \tag{46}
\]

where (46) comes from the convexity of \( L(w) \).

By summing over \( T + 1 \) iterations on both sides of (46), with an expectation over the random choices of \( B' \), we obtain

\[
E \left[ \sum_{t=0}^{T} \Psi(w^t) - \Psi(w^{t+1}) \right] \\
\geq \inf_t a^t \frac{P}{2n} \left[ \sum_{t=0}^{T} F(w^t) - F(w^*) \right] \tag{47}
\]

where (47) comes from Lemma 1-3 that \( \{F(w^t)\} \) is non-increasing. From (32), we can bound \( a^t \) by some positive constant \( \zeta = (2\lambda/(1 - \sigma + \sigma \gamma)/\theta c \sqrt{P \lambda(B')}) \)

\[
0 < \zeta \leq a^t \leq 1. \tag{48}
\]

Substituting (48) into (47), we have

\[
E \left[ \sum_{t=0}^{T} \Psi(w^t) - \Psi(w^{t+1}) \right] \geq \frac{P(T + 1)}{2n} \left[ E[F(w^T)] - F(w^*) \right].
\]

By rearranging the above inequality, we have

\[
E[F(w^T)] - F(w^*) \leq \frac{2n}{\zeta P(T + 1)} \left[ E[\Psi(w^0) - \Psi(w^{T+1})] \right] \tag{49}
\]

\[= \frac{2n}{\zeta P(T + 1)} \left[ E[\Psi(w^0) - \Psi(w^{T+1})] \right] \tag{50}
\]

\[
= \frac{2n}{\zeta P(T + 1)} \left[ \frac{\theta c \lambda(B')}{2} + \frac{\theta c \sup_{i} a^t(F(0))}{2\sigma(1 - \gamma)h} \right] \tag{51}
\]

where (49) comes from that \( \Psi(w^{T+1}) \geq 0 \), and (50) is because \( w^0 \) is set to be 0. \( \psi \), (51) holds since \( a^t \leq 1 \).

**G. Computational Complexity of PCDN and CDN**

The proposed PCDN algorithm takes much less time for each outer iteration than the CDN method. We analyze the computational complexity of PCDN for the \( k \)th outer iteration, time(\( k \)), to demonstrate this point (note that CDN is a special case of PCDN with bundle size \( P = 1 \)). Let \( t_{ls} \) denote the time complexity for computing the descent direction (step 7 in Algorithm 3), and \( t_{ls} \) denote the time complexity for a step of \( P \)-dimensional line search, which is approximately constant with varying \( P \) (see the discussions below). When the computation of descent directions (step 6 in Algorithm 3) is fully parallelized, time(\( k \)) can be estimated by

\[
E[\text{time}(k)] \approx \lceil n/P \rceil \cdot t_{ls} + \lceil n/P \rceil \cdot E[q^t] \cdot t_{ls} \tag{52}
\]

where the expectation is with respect to the random choice of \( B' \), and \( q^t \) is the number of line search steps in the \( t \)th iteration. As indicated in (52), the computational complexity of descent directions \( \lceil n/P \rceil \cdot t_{ls} \) decreases linearly with the increase of bundle size \( P \). For the cost of Armijo line search, when approximately estimating \( E[q^t] \) by its upper bound in Theorem 1, \( E[q^t]/P \) decreases with respect to \( P \), and thus \( \lceil n/P \rceil \cdot E[q^t] \cdot t_{ls} \) decreases with the increase of bundle size \( P \).

The overall computational complexity of PCDN’s each outer iteration is lower than that of the CDN method.

We show that the time complexity of one step of \( P \)-dimensional line search \( t_{ls} \) remains approximately constant with varying bundle size \( P \). The reason being that in each line search step of Algorithm 4, the time complexity remains constant with respect to \( P \). The difference of the whole line search procedure results from computing \( \sum_{i=1}^{P} x_i = \sum_{j=1}^{P} d_j x_{ij} \).

However, \( d^T x_i \) in the PCDN algorithm can be computed in parallel with \( P \) threads as well as a reduction-sum operation, and thus the computational complexity remains approximately constant.

**Acknowledgment**

The authors would like to thank H. Zha, X. Wang, M. Takác, and M. Jaggi for their valuable comments and suggestions.

**References**

[1] Y. Bian, X. Li, M. Cao, and Y. Liu, “Bundle CDN: A highly parallelized approach for large-scale \( l_1 \)-regularized logistic regression,” in *Proc. ECML/PKDD*, 2013, pp. 81–95.

[2] J. K. Bradley, A. Kyrola, D. Bickson, and C. Guestrin, “Parallel coordinate descent for \( l_1 \)-regularized loss minimization,” in *Proc. ICML*, 2011, pp. 321–328.

\(^9\)Using the upper bound in Theorem 1, and the fact that \( E_{B'}[\lambda(B')]/P \) is monotonically decreasing with respect to \( P \) in Lemma 1-1, we can easily obtain this.
3. J. Burke, “Descent methods for composite nondifferentiable optimization problems,” *Math. Program.*, vol. 33, nos. 3, pp. 260–279, 1985.

4. K.-W. Chang, C.-J. Hsieh, and C.-J. Lin, “Coordinate descent method for large-scale $l_1$-loss linear support vector machines,” *J. Mach. Learn. Res.*, vol. 9, pp. 1369–1398, Jun. 2008.

5. J. Dean et al., “Large scale distributed deep networks,” in *Proc. NIPS*, 2012, pp. 1223–1231.

6. R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, “LIBLINEAR: A library for large linear classification,” *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, Jun. 2008.

7. L. He, A. Bian, and M. Jaggi, “COLA: Communication-efficient decentralized linear learning,” in *Adv. Neural Inf. Process. Syst. (NIPS)*, 2018, pp. 4537–4547.

8. Q. Ho et al., “More effective distributed ML via a stale synchronous parallel parameter server,” in *Proc. NIPS*, 2013, pp. 1223–1231.

9. K. Koh, S.-J. Kim, and S. P. Boyd, “An interior-point method for large-scale $l_1$-regularized logistic regression,” *J. Mach. Learn. Res.*, vol. 8, no. 8, pp. 1519–1555, 2007.

10. J. Langford, L. Li, and T. Zhang, “Sparse online learning via truncated gradient,” *J. Mach. Learn. Res.*, vol. 10, pp. 777–801, Mar. 2009.

11. C. Lee and D. Roth, “Distributed box-constrained quadratic optimization for dual linear SVM,” in *Proc. ICML*, 2015, pp. 987–996.

12. H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations,” in *Proc. ICML*, 2009, pp. 609–616.

13. J. Lin and J. J. More, “Newton’s method for large bound-constrained optimization problems,” *SIAM J. Optim.*, vol. 9, no. 4, pp. 1100–1127, 1999.

14. H. Luo, P. Haffner, and J.-F. Paiement, (2014). “Accelerated parallel optimization methods for large scale machine learning.” [Online]. Available: https://arxiv.org/abs/1411.6725

15. J. Mareček and P. Richtárik, and M. Takáč, “Distributed block coordinate descent for minimizing partially separable functions,” in *Numerical Analysis and Optimization*, Cham, Switzerland: Springer, 2015, pp. 261–288. doi: 10.1007/978-3-319-17689-5_11.

16. A. Y. Ng, “Feature selection, L1 vs. L2 regularization, and rotational invariance,” in *Proc. ICML*, 2004, pp. 78–85.

17. F. Niu, B. Recht, C. Re, and S. J. Wright, “HOGWILD: A lock-free approach to parallelizing stochastic gradient descent,” in *Proc. NIPS*, 2011, pp. 693–701.

18. P. Richtárik and M. Takáč, “Distributed coordinate descent method for learning with big data,” *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 2657–2681, 2016.

19. P. Richtárik and M. Takáč, “Parallel coordinate descent methods for big data optimization,” *Math. Program.*, vol. 156, nos. 1–2, pp. 433–484, 2016.

20. C. Scherrer, M. Halappanavar, A. Tewari, and D. Hlagin, “Scaling up coordinate descent algorithms for large $l_1$-regularization problems,” in *Proc. ICML*, 2012, pp. 1–8.

21. C. Scherrer, A. Tewari, M. Halappanavar, and D. Hlagin, “Feature clustering for accelerating parallel coordinate descent,” in *Proc. NIPS*, 2012, pp. 28–36.

22. S. Shalev-Shwartz and A. Tewari, “Stochastic methods for $l_1$ regularized loss minimization,” in *Proc. ICML*, 2009, pp. 117–124.

23. V. Smith, S. Forte, M. Chenxin, M. Takač, M. I. Jordan, and M. Jaggi, “CoCoA: A general framework for communication-efficient distributed optimization,” *J. Mach. Learn. Res.*, vol. 18, no. 230, Jul. 2018.

24. P. Tseng and S. Yun, “A coordinate gradient descent method for nonsmooth separable minimization,” *Math. Program.*, vol. 117, nos. 1–2, pp. 387–423, 2009.

25. J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, “Robust face recognition via sparse representation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 210–227, Feb. 2009.

26. 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,” *J. Mach. Learn. Res.*, vol. 11, pp. 3183–3234, Mar. 2010.

27. G.-X. Yuan, C.-H. Ho, and C.-J. Lin, “An improved GLMNIT for $l_1$-regularized logistic regression,” in *Proc. KDD*, 2011, pp. 33–41.

28. M. Zinkevich, A. Smola, and A. Langford, “Slow learners are fast,” in *Proc. NIPS*, 2009, pp. 2331–2339.

29. M. Zinkevich, M. Weimer, A. J. Smola, and L. Li, “Parallelized stochastic gradient descent,” in *Proc. NIPS*, 2010, pp. 2595–2603.