FLEXIBLE PARALLEL ALGORITHMS FOR BIG DATA OPTIMIZATION

Francisco Facchinei[1], *Simone Sagratella*[1], *Gesualdo Scutari*[2]

[1]Dpt. of Computer, Control, and Management Eng., University of Rome "La Sapienza", Roma, Italy.

[2]Dpt. of Electrical Eng., State University of New York (SUNY) at Buffalo, Buffalo, NY 14260, USA.

ABSTRACT

We propose a decomposition framework for the parallel optimization of the sum of a differentiable function and a (block) separable nonsmooth, convex one. The latter term is typically used to enforce structure in the solution as, for example, in LASSO problems. Our framework is very flexible and includes both fully parallel Jacobi schemes and Gauss-Seidel (Southwell-type) ones, as well as virtually all possibilities in between (e.g., gradient- or Newton-type methods) with only a subset of variables updated at each iteration. Our theoretical convergence results improve on existing ones, and numerical results show that the new method compares favorably to existing algorithms.

Index Terms— Parallel optimization, Jacobi method, LASSO, Sparse solution.

1. INTRODUCTION

The minimization of the sum of a smooth function, F, and of a nonsmooth (separable) convex one, G,

$$\min_{\mathbf{x}\in X} V(\mathbf{x}) \triangleq F(\mathbf{x}) + G(\mathbf{x}), \tag{1}$$

is an ubiquitous problem that arises in many fields of engineering, so diverse as compressed sensing, basis pursuit denoising, sensor networks, neuroelectromagnetic imaging, machine learning, data mining, sparse logistic regression, genomics, metereology, tensor factorization and completion, geophysics, and radio astronomy. Usually the nonsmooth term is used to promote sparsity of the optimal solution, that often corresponds to a parsimonious representation of some phenomenon at hand. Many of the mentioned applications can give rise to extremely large problems so that standard optimization techniques are hardly applicable. And indeed, recent years have witnessed a flurry of research activity aimed at developing solution methods that are simple (for example based solely on matrix/vector multiplications) but yet capable to converge to a good approximate solution in reasonable time. It is hardly possible here to even summarize the huge amount of work done in this field; we refer the reader to the recent works [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19] as entry points to the literature.

It is clear however that if one wants to solve really large problems, parallel methods exploiting the computational power of multicore processors have to be employed. It is then surprising that while serial solutions methods for Problem (1) have been widely investigated, the analysis of parallel algorithms suitable to large-scale implementations lags behind. Gradient-type methods can of course be easily parallelized. However, beyond that, we are only aware of very few papers, all very recent, that deal with parallel solution methods [2, 6, 13, 17]. These papers analyze both randomized and deterministic block Coordinate Descent Methods (CDMs) that, essentially, are still (regularized) gradient-based methods. One advantage of the analyses in [2, 6, 13, 17] is that they provide an interesting (global) rate of convergence. On the other hand they apply only to convex problems and are not flexible enough to include, among other things, very natural Jacobi-type methods (where at each iteration a partial minimization of the original function is performed with respect to a block variable while all other variables are kept fixed) and the possibility to deal with a nonconvex F.

In this paper, building on the approach proposed in [20, 21], we present a broad, deterministic algorithmic framework for the solution of Problem (1) with the following novel features: i) it is parallel, with a degree of parallelism that can be chosen by the user and that can go from a complete parallelism (each variable is updated in parallel to all the others) to the sequential (one variable only is updated at each iteration); ii) it can tackle a nonconvex F; iii) it is very flexible and includes, among others, updates based on gradient- or Newton-type methods; and iv) it easily allows for inexact solutions. Our framework allows us to define different schemes, *all converging under the same conditions*, that can accommodate different problem features and algorithmic requirements. Even in the most studied case in which F is convex and $G(\mathbf{x}) \equiv 0$ our results compare favourably to existing ones and the numerical results show our approach to be very promising.

2. PROBLEM DEFINITION

We consider Problem (1), where the feasible set $X = X_1 \times \cdots \times X_N$ is a cartesian product of lower dimensional convex sets $X_i \subseteq \mathbb{R}^{n_i}$, and $\mathbf{x} \in \mathbb{R}^n$ is partitioned accordingly to $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$, with each $\mathbf{x}_i \in \mathbb{R}^{n_i}$. F is smooth (and not necessarily convex) and G is convex and possibly nondifferentiable, with $G(\mathbf{x}) = \sum_{i=1}^N g_i(\mathbf{x}_i)$ with $\mathbf{x}_i \in X_i$. This format is very general and includes problems of great interest. Below we list some instances of Problem (1).

- $G(\mathbf{x}) = 0$; in this case the problem reduces to the minimization of a smooth, possibly nonconvex problem with convex constraints.
- $F(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2$ and $G(\mathbf{x}) = c\|\mathbf{x}\|_1$, $X = \mathbb{R}^n$, with $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$, and $c \in \mathbb{R}_{++}$ given constants; this is the very famous and much studied LASSO problem [22].
- $F(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2$ and $G(\mathbf{x}) = c\sum_{i=1}^N \|\mathbf{x}_i\|_2$, $X = \mathbb{R}^n$, with $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$, and $c \in \mathbb{R}_{++}$ given constants; this is the group LASSO problem [23].
- $F(\mathbf{x}) = \sum_{j=1}^m \log(1+e^{-a_i \mathbf{y}_i^T \mathbf{x}})$ and $G(\mathbf{x}) = c\|\mathbf{x}\|_1$ (or $G(\mathbf{x}) = c\sum_{i=1}^N \|\mathbf{x}_i\|_2$), with $\mathbf{y}_i \in \mathbb{R}^n$, $a_i \in \mathbb{R}$, and $c \in \mathbb{R}_{++}$ given constants; this is the sparse logistic regression problem [24, 25].
- $F(\mathbf{x}) = \sum_{j=1}^m \max\{0, 1 - a_i \mathbf{y}_i^T \mathbf{x}\}^2$ and $G(\mathbf{x}) = c\|\mathbf{x}\|_1$, with $a_i \in \{-1, 1\}$, $\mathbf{y}_i \in \mathbb{R}^n$, and $c \in \mathbb{R}_{++}$ given; this is the ℓ_1-regularized ℓ_2-loss Support Vector Machine problem, see e.g. [18].
- Other problems that can be cast in the form (1) include the Nuclear Norm Minimization problem, the Robust Principal Component Analysis problem, the Sparse Inverse Covariance Selection problem, the Nonnegative Matrix (or Tensor) Factorization problem, see e.g. [16, 26] and references therein.

Given (1), we make the following standard, blanket assumptions:

(A1) Each X_i is nonempty, closed, and convex;
(A2) F is C^1 on an open set containing X;
(A3) ∇F is Lipschitz continuous on X with constant L_F;
(A4) $G(\mathbf{x}) = \sum_{i=i}^N g_i(\mathbf{x}_i)$, with all g_i continuous and convex on X_i;
(A5) V is coercive.

The order of the authors is alphabetic. The work of Scutari was supported by the USA National Science Foundation under Grants CMS 1218717 and CAREER Award No. 1254739

3. MAIN RESULTS

We want to develop *parallel* solution methods for Problem (1) whereby operations can be carried out on some or (possibly) all (block) variables $\mathbf{x}_i$ at the *same* time. The most natural parallel (Jacobi-type) method one can think of is updating *all* blocks simultaneously: given $\mathbf{x}^k$, each (block) variable $\mathbf{x}_i^{k+1}$ is computed as the solution of $\min_{\mathbf{x}_i}[F(\mathbf{x}_i,\mathbf{x}_{-i}^k)+g_i(\mathbf{x}_i)]$ (where $\mathbf{x}_{-i}$ denotes the vector obtained from $\mathbf{x}$ by deleting the block $\mathbf{x}_i$). Unfortunately this method converges only under very restrictive conditions [27] that are seldom verified in practice. To cope with this issue we introduce some "memory" and set the new point to be a convex combination of $\mathbf{x}^k$ and the solutions of $\min_{\mathbf{x}_i}[F(\mathbf{x}_i,\mathbf{x}_{-i}^k)+g_i(\mathbf{x}_i)]$. However our framework has many additional features, as discussed next.

Approximating F: Solving each $\min_{\mathbf{x}_i}[F(\mathbf{x}_i,\mathbf{x}_{-i}^k)+g_i(\mathbf{x}_i)]$ may be too costly or difficult in some situations. One may then prefer to approximate this problem, in some suitable sense, in order to facilitate the task of computing the new iteration. To this end, we assume that for all $i\in\mathcal{N}\triangleq\{1,\dots,N\}$ we can define a function $P_i(\mathbf{z};\mathbf{w}):X_i\times X\to\mathbb{R}$ having the following properties (we denote by ∇P_i the partial gradient of P_i with respect to $\mathbf{z}$):

(P1) $P_i(\bullet;\mathbf{w})$ is convex and continuously differentiable on X_i for all $\mathbf{w}\in X$;

(P2) $\nabla P_i(\mathbf{x}_i;\mathbf{x})=\nabla_{\mathbf{x}_i}F(\mathbf{x})$ for all $\mathbf{x}\in X$;

(P3) $\nabla P_i(\mathbf{z};\bullet)$ is Lipschitz continuous on X for all $\mathbf{z}\in X_i$.

Such a function P_i should be regarded as a (simple) convex approximation of F at the point $\mathbf{x}$ with respect to the block of variables $\mathbf{x}_i$ that preserves the first order properties of F with respect to $\mathbf{x}_i$. Based on this approximation we can define at any point $\mathbf{x}^k\in X$ a *regularized* approximation $\widetilde{h}_i(\mathbf{x}_i;\mathbf{x}^k)$ of V with respect to $\mathbf{x}_i$ where F is replaced by P_i while the nondifferentiable term is preserved, and a quadratic regularization is added to make the overall approximation strongly convex. More formally, we have

$$\widetilde{h}_i(\mathbf{x}_i;\mathbf{x}^k)\triangleq\underbrace{P_i(\mathbf{x}_i;\mathbf{x}^k)+\frac{\tau_i}{2}\left(\mathbf{x}_i-\mathbf{x}_i^k\right)^T\mathbf{Q}_i(\mathbf{x}^k)\left(\mathbf{x}_i-\mathbf{x}_i^k\right)}_{\triangleq h_i(\mathbf{x}_i;\mathbf{x}^k)}+g_i(\mathbf{x}_i),$$

where $\mathbf{Q}_i(\mathbf{x}^k)$ is an $n_i\times n_i$ positive definite matrix (possibly dependent on $\mathbf{x}^k$). We always assume that the functions $h_i(\bullet,\mathbf{x}_i^k)$ are uniformly strongly convex.

(A6) All $h_i(\bullet;\mathbf{x}^k)$ are uniformly strongly convex on X_i with a common positive definiteness constant $q>0$; furthermore, $\mathbf{Q}_i(\bullet)$ is Lipschitz continuous on X.

Note that an easy and standard way to satisfy A6 is to take, for any i and for any k, $\tau_i=q>0$ and $\mathbf{Q}_i(\mathbf{x}^k)=\mathbf{I}$. However, if $P_i(\bullet;\mathbf{x}^k)$ is already uniformly strongly convex, one can avoid the proximal term and set $\tau_i=0$ while satisfying A6.

Associated with each i and point $\mathbf{x}^k\in X$ we can define the following optimal solution map:

$$\widehat{\mathbf{x}}_i(\mathbf{x}^k,\tau_i)\triangleq\operatorname*{argmin}_{\mathbf{x}_i\in X_i}\tilde{h}_i(\mathbf{x}_i;\mathbf{x}^k). \tag{2}$$

Note that $\widehat{\mathbf{x}}_i(\mathbf{x}^k,\tau_i)$ is always well-defined, since the optimization problem in (2) is strongly convex. Given (2), we can then introduce

$$X\ni\mathbf{y}\mapsto\widehat{\mathbf{x}}(\mathbf{y},\boldsymbol{\tau})\triangleq\left(\widehat{\mathbf{x}}_i(\mathbf{y},\tau_i)\right)_{i=1}^N.$$

The algorithm we are about to described is based on the computation of $\widehat{\mathbf{x}}$. Therefore the approximating functions P_i should lead to as easily computable functions $\widehat{\mathbf{x}}$ as possible. An appropriate choice depends on the problem at hand and on computational requirements. We discuss some possible choices for P_i after introducing the main algorithm (Algorithm 1); see [28] for more details.

Inexact solutions: In many situations (especially in the case of large-scale problems), it can be useful to further reduce the computational effort needed to solve the subproblems in (2) by allowing *inexact* computations $\mathbf{z}_i^k$ of $\widehat{\mathbf{x}}_i(\mathbf{x}^k,\tau_i)$, i.e., $\|\mathbf{z}_i^k-\widehat{\mathbf{x}}_i\left(\mathbf{x}^k,\boldsymbol{\tau}\right)\|\le\varepsilon_i^k$, where ε_i^k measures the accuracy in computing the solution.

Updating only some blocks: Another important feature of our algorithmic framework is the possibility of updating only *some* of the variables at each iteration, a feature that has been observed to be very effective numerically. In fact, our schemes are guaranteed to converge under the update of only a *subset* of the variables at each iteration; the only condition is that such a subset contains at least one (block) component which is within a factor $\rho\in(0,1]$ "far away" from the optimality, in the sense explained next. First of all $\mathbf{x}_i^k$ is optimal for $\tilde{h}_i(\mathbf{x}_i;\mathbf{x}^k)$ if and only if $\widehat{\mathbf{x}}_i(\mathbf{x}^k,\tau_i)=\mathbf{x}_i^k$. Ideally we would like then to select the indices to update according to the optimality measure $d_i^k\triangleq\|\widehat{\mathbf{x}}_i(\mathbf{x}^k,\tau_i)-\mathbf{x}_i^k\|$ (e.g., opting for blocks exhibiting larger d_i^k's); but in some situations this could be computationally too expensive. Building on the same idea, we can introduce alternative less expensive metrics based on a computationally cheaper *error bound*, i.e., a function $E_i(\mathbf{x})$ such that

$$\underline{s}_i\|\widehat{\mathbf{x}}_i(\mathbf{x}^k,\tau_i)-\mathbf{x}_i^k\|\le E_i(\mathbf{x}^k)\le\bar{s}_i\|\widehat{\mathbf{x}}_i(\mathbf{x}^k,\tau_i)-\mathbf{x}_i^k\|, \tag{3}$$

for some $0<\underline{s}_i\le\bar{s}_i$. Of course we can always set $E_i(\mathbf{x}^k)=\|\widehat{\mathbf{x}}_i(\mathbf{x}^k,\tau_i)-\mathbf{x}_i^k\|$, but other choices are also possible; we discuss some of them after introducing the algorithm.

We are now ready to formally introduce our algorithm, Algorithm 1, that enjoys all the features discussed above. Its convergence properties to stationary solutions[1] of (1) are given in Theorem 1, whose proof is omitted because of space limitation, see [28].

Algorithm 1: Inexact Parallel Algorithm (FLEXA)

Data : $\{\varepsilon_i^k\}$ for $i\in\mathcal{N}$, $\boldsymbol{\tau}\ge\mathbf{0}$, $\{\gamma^k\}>0$, $\mathbf{x}^0\in X$, $\rho\in(0,1]$.
Set $k=0$.

(S.1) : If $\mathbf{x}^k$ satisfies a termination criterion: STOP;

(S.2) : For all $i\in\mathcal{N}$, solve (2) with accuracy ε_i^k :
Find $\mathbf{z}_i^k\in X_i$ s.t. $\|\mathbf{z}_i^k-\widehat{\mathbf{x}}_i\left(\mathbf{x}^k,\boldsymbol{\tau}\right)\|\le\varepsilon_i^k$;

(S.3) : Set $M^k\triangleq\max_i\{E_i(\mathbf{x}^k)\}$.
Choose a set S^k that contains at least one index i for which $E_i(\mathbf{x}^k)\ge\rho M^k$.
Set $\widehat{\mathbf{z}}_i^k=\mathbf{z}_i^k$ for $i\in S^k$ and $\widehat{\mathbf{z}}_i^k=\mathbf{x}_i^k$ for $i\notin S^k$

(S.4) : Set $\mathbf{x}^{k+1}\triangleq\mathbf{x}^k+\gamma^k\left(\widehat{\mathbf{z}}^k-\mathbf{x}^k\right)$;

(S.5) : $k\leftarrow k+1$, and go to (S.1).

Theorem 1 *Let $\{\mathbf{x}^k\}$ be the sequence generated by Algorithm 1, under A1-A6. Suppose that $\{\gamma^k\}$ and $\{\varepsilon_i^k\}$ satisfy the following conditions: i) $\gamma^k\in(0,1]$; ii) $\gamma^k\to0$; iii) $\sum_k\gamma^k=+\infty$; iv) $\sum_k\left(\gamma^k\right)^2<+\infty$; and v) $\varepsilon_i^k\le\gamma^k\alpha_1\min\{\alpha_2,1/\|\nabla_{\mathbf{x}_i}F(\mathbf{x}^k)\|\}$ for all $i\in\mathcal{N}$ and some nonnegative constants α_1 and α_2. Additionally, if inexact solutions are used in Step 2, i.e., $\varepsilon_i^k>0$ for some i and infinite k, then assume also that G is globally Lipschitz on X.*

Then, either Algorithm 1 converges in a finite number of iterations to a stationary solution of (1) *or every limit point of $\{\mathbf{x}^k\}$ (at least one such points exists) is a stationary solution of* (1).

On Algorithm 1. The proposed algorithm is extremely flexible. We can always choose $S^k=\mathcal{N}$ resulting in the simultaneous update

[1] We recall that a stationary points $\mathbf{x}^*$ of (1) is a point for which a subgradient $\xi\in\partial G(\mathbf{x}^*)$ exists such that $(\nabla F(\mathbf{x}^*)+\xi)^T(\mathbf{y}-\mathbf{x}^*)\ge0$ for all $\mathbf{y}\in X$. If F is convex, stationary points coincide with global minimizers.

of all the (block) variables (full Jacobi scheme); or, at the other extreme, one can update a single (block) variable per time, thus obtaining a Gauss-Southwell kind of method. One can also compute inexact solutions (Step 2) while preserving convergence, provided that the error term ε_i^k and the step-size γ^k's are chosen according to Theorem 1. We emphasize that the Lipschitzianity of G is required only if $\widehat{\mathbf{x}}(\mathbf{x}^k,\tau)$ is not computed exactly for infinite iterations. At any rate this Lipschitz conditions is automatically satisfied if G is a norm (and therefore in LASSO and group LASSO problems for example) or if X is bounded.

On the choice of the stepsize γ^k. An example of step-size rule satisfying i-iv in Theorem 1 is: given $\gamma^0=1$, let

$$\gamma^k=\gamma^{k-1}\left(1-\theta\,\gamma^{k-1}\right),\quad k=1,\ldots, \tag{4}$$

where $\theta\in(0,1)$ is a given constant; see [21] for others rules. This is actually the rule we used in our practical experiments (cf. Sec. 4). Note that while this rule may still require some tuning for optimal behaviour, it is quite reliable, since in general we are not using a (sub)gradient direction, so that many of the well-known practical drawbacks associated with a (sub)gradient method with diminishing step-size are mitigated in our setting. Furthermore, this choice of step-size does not require any form of centralized coordination, which is a favourable feature in a parallel environment.

We remark that it is possible to prove convergence of Algorithm 1 also using other step-size rules, such as a standard Armijo-like linesearch procedure or a (suitably small) constant step-size; see [28] for more details. We omit the discussion of these options because of lack of space, but also because the former is not in line with our parallel approach while the latter is numerically less efficient.

On the choice of $E_i(\mathbf{x})$.

• As we mentioned, the most obvious choice is to take $E_i(\mathbf{x})=\|\widehat{\mathbf{x}}_i(\mathbf{x}^k,\tau_i)-\mathbf{x}_i^k\|$. This is a valuable choice if the computation of $\widehat{\mathbf{x}}_i(\mathbf{x}^k,\tau_i)$ can be easily accomplished. For instance, in the LASSO problem with $\mathcal{N}=\{1,\ldots,n\}$ (i.e., when each block reduces to a scalar variable), it is well-known that $\widehat{\mathbf{x}}_i(\mathbf{x}^k,\tau_i)$ can be computed in closed form using the soft-thresholding operator.

• In situations where the computation of $\|\widehat{\mathbf{x}}_i(\mathbf{x}^k,\tau_i)-\mathbf{x}_i^k\|$ is not possible or advisable, we can resort to estimates. Assume momentarily that $G\equiv 0$. Then it is known [29, Proposition 6.3.1] under our assumptions that $\|\Pi_{X_i}(\mathbf{x}_i^k-\nabla_{\mathbf{x}_i}F(\mathbf{x}^k))-\mathbf{x}_i^k\|$ is an error bound for the minimization problem in (2) and therefore satisfies (3), where $\Pi_{X_i}(\mathbf{y})$ denotes the Euclidean projection of $\mathbf{y}$ onto the closed and convex set X_i. In this situation we can choose $E_i(\mathbf{x}^k)=\|\Pi_{X_i}(\mathbf{x}_i^k-\nabla_{\mathbf{x}_i}F(\mathbf{x}^k))-\mathbf{x}_i^k\|$. If $G(\mathbf{x})\not\equiv 0$ things become more complex. In most cases of practical interest, adequate error bounds can be derived from [15, Lemma 7].

• It is interesting to note that the computation of E_i is only needed if a partial update of the (block) variables is performed. However, an option that is always feasible is to take $S^k=\mathcal{N}$ at each iteration, i.e., update all (block) variables at each iteration. With this choice we can dispense with the computation of E_i altogether.

On the choice of $P_i(\mathbf{x}_i;\mathbf{x})$.

• The most obvious choice for P_i is the linearization of F at $\mathbf{x}^k$ with respect to $\mathbf{x}_i$: $P_i(\mathbf{x}_i;\mathbf{x}^k)=F(\mathbf{x}^k)+\nabla_{\mathbf{x}_i}F(\mathbf{x}^k)^T(\mathbf{x}_i-\mathbf{x}_i^k)$. With this choice, and taking for simplicity $\mathbf{Q}_i(\mathbf{x}^k)=\mathbf{I}$, $\widehat{\mathbf{x}}_i(\mathbf{x}^k,\tau_i)$ is given by

$$\operatorname*{argmin}_{\mathbf{x}_i\in X_i}\left\{F(\mathbf{x}^k)+\nabla_{\mathbf{x}_i}F(\mathbf{x}^k)^T(\mathbf{x}_i-\mathbf{x}_i^k)+\frac{\tau_i}{2}\|\mathbf{x}_i-\mathbf{x}_i^k\|^2+g_i(\mathbf{x}_i)\right\}. \tag{5}$$

This is essentially the way a new iteration is computed in most *sequential* (block-)CDMs for the solution of (group) LASSO problems and its generalizations. Note that contrary to most existing schemes, our algorithm is *parallel*.

• Assuming $F(\mathbf{x}_i,\mathbf{x}_{-i}^k)$ convex, at another extreme we could just take $P_i(\mathbf{x}_i;\mathbf{x}^k)=F(\mathbf{x}_i,\mathbf{x}_{-i}^k)$, which setting for simplicity $\mathbf{Q}_i(\mathbf{x}^k)=\mathbf{I}$ leads to

$$\widehat{\mathbf{x}}_i(\mathbf{x}^k,\tau_i)=\operatorname*{argmin}_{\mathbf{x}_i\in X_i}\left\{F(\mathbf{x}_i,\mathbf{x}_{-i}^k)+\frac{\tau_i}{2}\|\mathbf{x}_i-\mathbf{x}_i^k\|^2+g_i(\mathbf{x}_i)\right\}, \tag{6}$$

thus giving rise to a parallel nonlinear Jacobi type method for the constrained minimization of $V(\mathbf{x})$.

• Between the two "extreme" solutions proposed above one can consider "intermediate" choices. For example, If $F(\mathbf{x}_i,\mathbf{x}_{-i}^k)$ is convex, we can take $P_i(\mathbf{x}_i;\mathbf{x}^k)$ as a second order approximation of $F(\mathbf{x}_i,\mathbf{x}_{-i}^k)$, i.e., $P_i(\mathbf{x}_i;\mathbf{x}^k)=F(\mathbf{x}^k)+\nabla_{\mathbf{x}_i}F(\mathbf{x}^k)^T(\mathbf{x}_i-\mathbf{x}_i^k)+\frac{1}{2}(\mathbf{x}_i-\mathbf{x}_i^k)^T\nabla^2_{\mathbf{x}_i\mathbf{x}_i}F(\mathbf{x}^k)(\mathbf{x}_i-\mathbf{x}_i^k)$. When $g_i(\mathbf{x}_i)\equiv 0$, this essentially corresponds to taking a Newton step in minimizing the "reduced" problem $\min_{\mathbf{x}_i\in X_i}F(\mathbf{x}_i,\mathbf{x}_{-i}^k)$, resulting in

$$\widehat{\mathbf{x}}_i(\mathbf{x}^k,\tau_i)=\operatorname*{argmin}_{\mathbf{x}_i\in X_i}\left\{F(\mathbf{x}^k)+\nabla_{\mathbf{x}_i}F(\mathbf{x}^k)^T(\mathbf{x}_i-\mathbf{x}_i^k)+\right.$$
$$\left.+\frac{1}{2}(\mathbf{x}_i-\mathbf{x}_i^k)^T\nabla^2_{\mathbf{x}_i\mathbf{x}_i}F(\mathbf{x}^k)(\mathbf{x}_i-\mathbf{x}_i^k)+\frac{\tau_i}{2}\|\mathbf{x}_i-\mathbf{x}_i^k\|^2+g_i(\mathbf{x}_i)\right\}.$$

The framework described in Algorithm 1 can give rise to very different instances, according to the choices one makes for the many variable features it contains, some of which have been detailed above. For lack of space, we cannot fully discuss here all possibilities. We provide next just a few instances of possible algorithms that fall in our framework; more examples can be found in [28].

Example #1−(Proximal) Jacobi algorithms for convex functions: Consider the simplest problem falling in our setting: the unconstrained minimization of a continuously differentiable convex function, i.e., assume in (1) that F is convex, $G(\mathbf{x})\equiv 0$, and $X=\mathbb{R}^n$. Although this is possibly the best studied problem in nonlinear optimization, classical parallel methods for this problem [27, Sec. 3.2.4] require very strong contraction conditions. In our framework we can take $S^k=\mathcal{N}$, $P_i(\mathbf{x}_i;\mathbf{x}^k)=F(\mathbf{x}_i,\mathbf{x}_{-i}^k)$, resulting in a fully parallel Jacobi-type method which does not need any additional assumptions. Furthermore our theory shows that we can even dispense with the convexity assumption and still get convergence of a Jacobi-type method to a stationary point.

Example # 2−Parallel coordinate descent method for LASSO Consider the LASSO problem, i.e., problem (1) with $F(\mathbf{x})=\|\mathbf{A}\mathbf{x}-\mathbf{b}\|^2$, $G(\mathbf{x})=c\|\mathbf{x}\|_1$, and $X=\mathbb{R}^n$. Probably, to date, the most succesful class of methods for this problem is that of CDMs, whereby at each iteration a single variable is updated using (5). We can easily obtain a parallel version for this method by taking $n_i=1$, $S^k=\mathcal{N}$ and still using (5). Alternatively, instead of linearizing $F(\mathbf{x})$, we can better exploit the convexity of $F(\mathbf{x})$ and use (6). Furthermore, we can easily consider similar methods for the group LASSO problem (just take $n_i>1$). As a final remark, we observe that convergence conditions of existing (deterministic) fully distributed parallel versions of CDMs such as [2, 17] impose a constraint on the maximum number of variables that can be simultaneously updated (linked to the spectral radius of some matrices), a constraint that in many large scale problems is likely not satisfied. A key feature of the proposed scheme is that we can parallelize over (possibly) all variables while guaranteeing convergence.

4. NUMERICAL RESULTS

In this section we provide some numerical results providing a solid evidence of the viability of our approach; they clearly show that our algorithmic framework leads to practical methods that exploit well

parallelism and compare favourably to existing schemes, both parallel and sequential. The tests were carried out on LASSO problems, one of the most studied instance of (1); other classes of problems are considered in [28]. We generate four instances of problems using the random generation technique proposed by Nesterov in [7]; ; this method permits to control the sparsity of the solution. For the first three groups, we considered problems with 10,000 variables with the matrix $\mathbf{A}$ having 2,000 rows. The three groups differ in the number of non zeros of the solution, which is 20% (low sparsity), 10% (medium sparsity), and 5% (high sparsity) respectively. The last group is an instance with 100,000 variables and 5000 rows, and solutions having 5% of non zero variables (high sparsity).

We implemented the instance of Algorithm 1 that we described in Example # 2 in the previous section, with the only difference that we used (6) instead of the proximal-linear choice (5). Note that in the case of LASSO problems, the unique solution (6) can be computed in closed form using the soft-thresholding operator, see e.g. [30]. The free parameters of the algorithm are chosen as follows. The proximal parameters are initially set to $\tau_i = \text{tr}(\mathbf{A}^T\mathbf{A})/2n$ for all i, where n is the total number of variables. Furthermore, we allowed a finite number of possible changes to τ_i according to the following rules: (i) all τ_i are doubled if at a certain iteration the objective function does not decrease; and (ii) they are all halved if the objective function decreases for ten consecutive iterations. We updated γ^k according to (4) with $\gamma^0 = 0.9$ and $\theta = 1e-5$. Note that since τ_i are changed only a finite number of times, conditions of Theorem 1 are satisfied, and thus this instance of Algorithm 1 is guaranteed to converge. Finally we choose not to update all variables but set $E_i(\mathbf{x}^k) = \|\widehat{\mathbf{x}}_i(\mathbf{x}^k, \tau_i) - \mathbf{x}_i^k\|$ and $\rho = 0.5$ in Algorithm 1.

We compared our algorithm above, termed FLEXible parallel Algorithm (FLEXA), with a parallel implementation of FISTA [30], that can be regarded as the benchmark algorithm for LASSO problems, and Grock, a parallel algorithm proposed in [17] that seems to perform extremely well on sparse problems. We actually tested two instances of Grock, namely: i) one where only one variable is updated at each iteration; and ii) a second instance where the number of variables simultaneously updated is equal to the number of the parallel processors (16 for the first three set of test problems, 32 for the last). Note that the theoretical convergence of Grock is in jeopardy as the number of updated variables increases; convergence conditions for this method are likely to hold only if the columns of $\mathbf{A}$ are "almost" orthogonal, a feature enjoyed by most our test problems but not satisfied in most applications. As benchmark, we also implemented two classical sequential schemes: (i) a Gauss-Seidel (GS) method computing $\hat{\mathbf{x}}_i$, and then updating $\mathbf{x}_i$ using unitary step-size, in a sequential fashion, and (ii) a classical Alternating Method of Multipliers (ADMM) [31] in the form of [32]. Note that ADMM can be parallelized, but it is known not to scale well and therefore we did not consider this possibility here.

All codes have been written in C++ and use the Message Passing Interface for parallel operations. All algebra is performed by using the GNU Scientific Library (GSL). The algorithms were tested on a cluster computer at the State University of New York at Buffalo. All computations were done on one 32 core node composed of four 8 core CPUs with 96GB of RAM and Infiniband card. The 10,000 variables problems were solved using 16 parallel processes while for the 100,000 variables problems 32 parallel processes were used. GS and ADMM were always run on a single process. Results of our experiments are reported in Fig. 1, where we plot the relative error $(V(\mathbf{x}^k) - V^\star)/V^\star$ versus the CPU time, where $V^\star$ is the optimal value of V. The CPU time includes communication times (for distributed algorithms) and the initial time needed by the methods to

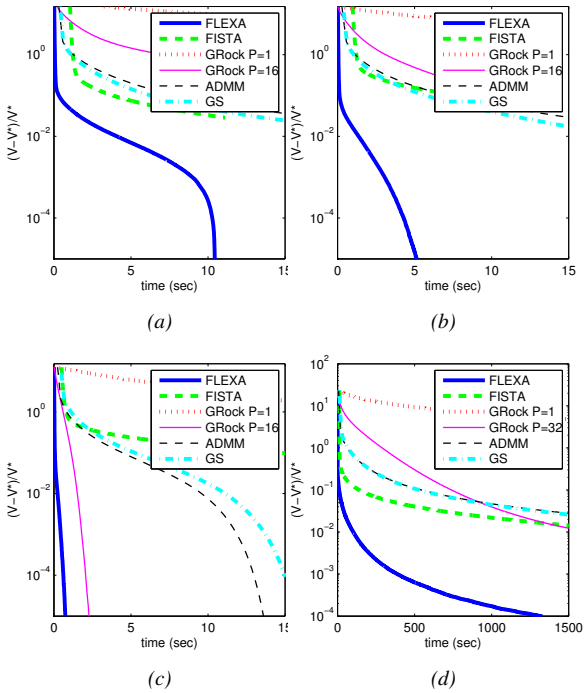

Fig. 1. Relative error vs. time (in seconds, logarithmic scale): (a) medium size and low sparsity - (b) medium size and sparsity - (c) medium size and high sparsity - (d) large size and high sparsity

perform all pre-iterations computations (this explains why the plot of FISTA starts after the others; in fact FISTA requires some nontrivial initializations based on the computation of $\|\mathbf{A}\|_2^2$). The curves are averaged over ten random realizations for each of the 10,000 variables groups, while for large 100,000 variables problems the average is over 3 realizations.

Fig 1 shows that on the tested problems FPA outperforms in a consistent manner all other implemented algorithms. Sequential methods behave strikingly well on the 10,000 variables problems, if one keeps in mind that they only use one process; however, as expected, they cannot compete with parallel methods when the dimensions increase. FISTA is capable to approach relatively fast low accuracy solutions, but has difficulties in reaching high accuracies. The parallel version of Grock is the closest match to FLEXA, but only when the problems are very sparse and the dimensions not too large. This is consistent with the fact that at each iteration Grock updates only a very limited number of variables, and also with the fact that its convergence properties are at stake when the problems are quite dense. Our experiments also suggest that, differently from what one could think (and often claimed in similar situations when using gradient-like methods), updating only a (suitably chosen) subset of blocks rather than all variables may lead to faster algorithms; see [28] for a detailed discussion on this issue. In conclusion, we believe the results overall indicate that our approach can lead to very efficient practical methods for the solution of large problems, with the flexibility to adapt to many different problem characteristics.

5. CONCLUSIONS

We proposed a highly parallelizable algorithmic scheme for the minimization of the sum of a differentiable function and a block-separable nonsmooth one. Our framework easily allows us to analyze parallel versions of well-known sequential methods and leads to entirely new algorithms. When applied to large-scale LASSO problems, our algorithm was shown to outperform existing methods.

6. REFERENCES

[1] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski, "Optimization with sparsity-inducing penalties," *arXiv preprint arXiv:1108.0775*, 2011.

[2] J. K. Bradley, A. Kyrola, D. Bickson, and C. Guestrin, "Parallel coordinate descent for l1-regularized loss minimization," *arXiv preprint arXiv:1105.5379*, 2011.

[3] P. L. Bühlmann, S. A. van de Geer, and S. Van de Geer, *Statistics for high-dimensional data*. Springer, 2011.

[4] R. H. Byrd, J. Nocedal, and F. Oztoprak, "An Inexact Successive Quadratic Approximation Method for Convex L-1 Regularized Optimization," *arXiv preprint arXiv:1309.3529*, 2013.

[5] K. Fountoulakis and J. Gondzio, "A Second-Order Method for Strongly Convex L1-Regularization Problems," *arXiv preprint arXiv:1306.5386*, 2013.

[6] I. Necoara and D. Clipici, "Efficient parallel coordinate descent algorithm for convex optimization problems with separable constraints: application to distributed MPC," *Journal of Process Control*, vol. 23, no. 3, pp. 243–253, 2013.

[7] Y. Nesterov, "Gradient methods for minimizing composite functions," *Mathematical Programming*, pp. 1–37, 2012.

[8] ——, "Efficiency of coordinate descent methods on huge-scale optimization problems," *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 341–362, 2012.

[9] Z. Qin, K. Scheinberg, and D. Goldfarb, "Efficient block-coordinate descent algorithms for the group lasso," *Mathematical Programming Computation*, pp. 1–27, 2010.

[10] A. Rakotomamonjy, "Surveying and comparing simultaneous sparse approximation (or group-lasso) algorithms," *Signal processing*, vol. 91, no. 7, pp. 1505–1526, 2011.

[11] M. Razaviyayn, M. Hong, and Z.-Q. Luo, "A unified convergence analysis of block successive minimization methods for nonsmooth optimization," *SIAM Journal on Optimization*, vol. 23, no. 2, pp. 1126–1153, 2013.

[12] P. Richtárik and M. Takáč, "Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function," *Mathematical Programming*, pp. 1–38, 2012.

[13] ——, "Parallel coordinate descent methods for big data optimization," *arXiv preprint arXiv:1212.0873*, 2012.

[14] S. Sra, S. Nowozin, and S. J. Wright, Eds., *Optimization for Machine Learning*, ser. Neural Information Processing. Cambridge, Massachusetts: The MIT Press, Sept. 2011.

[15] P. Tseng and S. Yun, "A coordinate gradient descent method for nonsmooth separable minimization," *Mathematical Programming*, vol. 117, no. 1-2, pp. 387–423, 2009.

[16] Y. Xu and W. Yin, "A block coordinate descent method for multi-convex optimization with applications to nonnegative tensor factorization and completion," DTIC Document, Tech. Rep., 2012. [Online]. Available: http://www.caam.rice.edu/∼optimization/bcu/multiconvex.html

[17] Z. Yin, P. Ming, and Y. Wotao, "Parallel and Distributed Sparse Optimization," 2013. [Online]. Available: http://www.caam.rice.edu/∼optimization/disparse/

[18] 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," *The Journal of Machine Learning Research*, vol. 9999, pp. 3183–3234, 2010.

[19] S. J. Wright, "Accelerated block-coordinate relaxation for regularized optimization," *SIAM Journal on Optimization*, vol. 22, no. 1, pp. 159–186, 2012.

[20] G. Scutari, F. Facchinei, P. Song, D. P. Palomar, and J.-S. Pang, "Decomposition by partial linearization in multiuser systems," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2013)*, May 4-9 2013, pp. 4424–4428.

[21] G. Scutari, F. Facchinei, P. Song, D. Palomar, and J.-S. Pang, "Decomposition by Partial linearization: Parallel optimization of multi-agent systems," *IEEE Trans. Signal Process.*, vol. 62, pp. 641–656, Feb. 2014.

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

[23] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 1, pp. 49–67, 2006.

[24] S. K. Shevade and S. S. Keerthi, "A simple and efficient algorithm for gene selection using sparse logistic regression," *Bioinformatics*, vol. 19, no. 17, pp. 2246–2253, 2003.

[25] L. Meier, S. Van De Geer, and P. Bühlmann, "The group lasso for logistic regression," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 70, no. 1, pp. 53–71, 2008.

[26] D. Goldfarb, S. Ma, and K. Scheinberg, "Fast alternating linearization methods for minimizing the sum of two convex functions," *Mathematical Programming*, pp. 1–34, 2012.

[27] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, 2nd ed. Athena Scientific Press, 1989.

[28] F. Facchinei, S. Sagratella, and G. Scutari, "Parallel Algorithms for Big Data Optimization," *IEEE Trans. Signal Process.*, (under review). [Online]. Available: http://arxiv.org/pdf/1402.5521v2.pdf.

[29] F. Facchinei and J.-S. Pang, *Finite-Dimensional Variational Inequalities and Complementarity Problem*. Springer-Verlag, New York, 2003.

[30] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[31] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[32] Z.-Q. Luo and M. Hong, "On the linear convergence of the alternating direction method of multipliers," *arXiv preprint arXiv:1208.3922*, 2012.