BLOCK-ACTIVATED ALGORITHMS FOR MULTICOMPONENT FULLY NONSMOOTH MINIMIZATION

Minh N. Bùi, Patrick L. Combettes, and Zev C. Woodstock

North Carolina State University, Department of Mathematics, Raleigh, NC 27695-8205, USA

ABSTRACT

Under consideration are multicomponent minimization problems involving a separable nonsmooth convex function penalizing the components individually, and nonsmooth convex coupling terms penalizing linear mixtures of the components. We investigate the application of block-activated proximal algorithms for solving such problems, i.e., algorithms which, at each iteration, need to use only a block of the underlying functions, as opposed to all of them as in standard methods. For smooth coupling functions, several block-activated algorithms exist and they are well understood. By contrast, in the fully nonsmooth case, few block-activated methods are available and little effort has been devoted to assessing them. Our goal is to shed more light on the implementation, the features, and the behavior of these algorithms, compare their merits, and provide machine learning and image recovery experiments illustrating their performance.

Index Terms— Block-activated algorithm, image recovery, machine learning, nonsmooth convex minimization, proximal splitting.

1. INTRODUCTION

The goal of many signal processing and machine learning tasks is to exploit the observed data and the prior knowledge to produce a solution that represents information of interest. In this process of extracting information from data, structured convex optimization has established itself as an effective modeling and algorithmic framework; see, for instance, [3, 5, 8, 14, 19]. In state-of-the-art applications, the sought solution is often a tuple of vectors which reside in different spaces [1, 2, 4, 6, 12, 16, 13, 17, 20]. The following multicomponent minimization formulation captures such problems. It consists of a separable term penalizing the components individually, and of coupling terms penalizing linear mixtures of the components.

Problem 1 Let $(\mathcal{H}_i)_{1\leqslant i\leqslant m}$ and $(\mathcal{G}_k)_{1\leqslant k\leqslant p}$ be Euclidean spaces. For every $i\in\{1,\ldots,m\}$ and every $k\in\{1,\ldots,p\}$, let $f_i\colon\mathcal{H}_i\to\left]-\infty,+\infty\right]$ and $g_k\colon\mathcal{G}_k\to\left]-\infty,+\infty\right]$ be proper lower semicontinuous convex functions, and let $L_{k,i}\colon\mathcal{H}_i\to\mathcal{G}_k$ be a linear operator. The objective is to

$$\underset{x_1\in\mathcal{H}_1,\ldots,x_m\in\mathcal{H}_m}{\operatorname{minimize}}\ \underbrace{\sum_{i=1}^{m}f_i(x_i)}_{\text{separable term}}+\sum_{k=1}^{p}\underbrace{g_k\Bigg(\sum_{i=1}^{m}L_{k,i}x_i\Bigg)}_{k\text{th coupling term}}. \tag{1}$$

To solve Problem 1 reliably without adding restrictions (for instance, smoothness or strong convexity of some functions involved in the model), we focus on flexible proximal algorithms that have the following features:

① **Nondifferentiability:** None of the functions $f_1,\ldots,f_m,g_1,\ldots,g_p$ needs to be differentiable.

② **Splitting:** The functions $f_1,\ldots,f_m,g_1,\ldots,g_p$ and the linear operators are activated separately.

③ **Block activation:** Only a block of the functions $f_1,\ldots,f_m,g_1,\ldots,g_p$ is activated at each iteration. This is in contrast with most splitting methods which require full activation, i.e., that all the functions be used at every iteration.

④ **Operator norms:** Bounds on the norms of the linear operators involved in Problem 1 are not assumed since they can be hard to compute.

⑤ **Convergence guarantee:** The algorithm produces a sequence which converges (possibly almost surely) to a solution to Problem 1.

In view of features ① and ②, the algorithms of interest should activate the functions $f_1,\ldots,f_m,g_1,\ldots,g_p$ via their proximity operators (even if some functions happened to be smooth, proximal activation is often preferable [6, 10]). The motivation for ② is that proximity operators of composite functions are typically not known explicitly. Feature ③ is geared towards current large-scale problems. In such scenarios, memory and computing power limitations make the execution of standard proximal splitting algorithms, which require activating all the functions at each iteration, inefficient or simply impossible. We must therefore turn our attention to algorithms which employ only blocks of functions $(f_i)_{i\in I_n}$ and $(g_k)_{k\in K_n}$ at iteration n. If the functions $(g_k)_{1\leqslant k\leqslant p}$ were all smooth, one could use block-activated versions of the forward-backward algorithm proposed in [15, 25] and the references therein; in particular, when $m=1$, methods such as those of [11, 18, 23, 26] would be pertinent. As noted in [15, Remark 5.10(iv)], another candidate of interest could be the randomly block-activated algorithm of [15, Section 5.2], which leads to block-activated versions of several primal-dual methods (see [24] for detailed developments and [7] for an inertial version when $m=1$). However, this approach violates ④ as it imposes bounds on the proximal scaling parameters which depend on the norms of the linear operators. Finally, ⑤ rules out methods that guarantee merely minimizing sequences or ergodic convergence.

To the best of our knowledge, there are two primary methods that fulfill ①–⑤:

- Algorithm A: The stochastic primal-dual Douglas–Rachford algorithm of [15].
- Algorithm B: The deterministic primal-dual projective splitting algorithm of [9].

In the case of smooth coupling functions $(g_k)_{1\leqslant k\leqslant p}$, in (1), extensive numerical experience has been accumulated to understand

M. N. Bùi and P. L. Combettes were supported by the National Science Foundation under grant CCF-1715671 and Z. C. Woodstock by the National Science Foundation under grant DGE-1746939.

the behavior of block-activated methods, especially in the case of stochastic gradient methods. By contrast, to date, very few numerical experiments with the recent, fully nonsmooth Algorithms A and B have been conducted and no comparison of their merits and performance has been undertaken. Thus far, Algorithm A has been employed only in the context of machine learning (see also the variant of A in [6] for partially smooth problems). On the other hand, Algorithm B has been used in image recovery in [10], but only in full activation mode, and in feature selection in [22], but with $m = 1$.

Contributions and novelty: This paper investigates for the first time the application of block-activated methods in fully nonsmooth multivariate minimization problems. It sheds more light on the implementation, the features, and the numerical behavior of Algorithms A and B, compares their merits, and provides experiments illustrating their performance.

Outline: Algorithms A and B are presented in Section 2. In Section 3, we analyze and compare their features, implementation, and asymptotic properties. This investigation is complemented in Section 4 by numerical experiments in the context of machine learning and image recovery.

2. BLOCK-ACTIVATED ALGORITHMS FOR PROBLEM 1

The subdifferential, the conjugate, and the proximity operator of a proper lower semicontinuous convex function $f\colon \mathcal{H} \to]-\infty,+\infty]$ are denoted by ∂f, f^*, and prox_f, respectively. Let us consider the setting of Problem 1 and let us set $\boldsymbol{\mathcal{H}} = \mathcal{H}_1 \times \cdots \times \mathcal{H}_m$ and $\boldsymbol{\mathcal{G}} = \mathcal{G}_1 \times \cdots \times \mathcal{G}_p$. A generic element in $\boldsymbol{\mathcal{H}}$ is denoted by $\boldsymbol{x} = (x_i)_{1\leqslant i\leqslant m}$ and a generic element in $\boldsymbol{\mathcal{G}}$ by $\boldsymbol{y} = (y_k)_{1\leqslant k\leqslant p}$.

As discussed in Section 1, two primary algorithms fulfill requirements ①–⑤. Both operate in the product space $\boldsymbol{\mathcal{H}} \times \boldsymbol{\mathcal{G}}$. The first one employs random activation of the blocks. To present it, let us introduce

$$\begin{cases} \boldsymbol{L}\colon \boldsymbol{\mathcal{H}} \to \boldsymbol{\mathcal{G}}\colon \boldsymbol{x} \mapsto \Big(\sum_{i=1}^{m} L_{1,i}x_i, \dots, \sum_{i=1}^{m} L_{p,i}x_i\Big) \\ \boldsymbol{V} = \big\{(\boldsymbol{x},\boldsymbol{y}) \in \boldsymbol{\mathcal{H}} \times \boldsymbol{\mathcal{G}} \mid \boldsymbol{y} = \boldsymbol{L}\boldsymbol{x}\big\} \\ \boldsymbol{F}\colon \boldsymbol{\mathcal{H}} \times \boldsymbol{\mathcal{G}} \ni (\boldsymbol{x},\boldsymbol{y}) \mapsto \sum_{i=1}^{m} f_i(x_i) + \sum_{k=1}^{p} g_k(y_k). \end{cases} \tag{2}$$

Then (1) is equivalent to

$$\underset{(\boldsymbol{x},\boldsymbol{y})\in\boldsymbol{V}}{\text{minimize}} \;\; \boldsymbol{F}(\boldsymbol{x},\boldsymbol{y}). \tag{3}$$

The idea is then to apply the Douglas–Rachford algorithm in block form to this problem [15]. To this end, we need $\operatorname{prox}_{\boldsymbol{F}}$ and $\operatorname{prox}_{\iota_{\boldsymbol{V}}} = \operatorname{proj}_{\boldsymbol{V}}$. Note that $\operatorname{prox}_{\boldsymbol{F}}\colon (\boldsymbol{x},\boldsymbol{y}) \mapsto ((\operatorname{prox}_{f_i} x_i)_{1\leqslant i\leqslant m}, (\operatorname{prox}_{g_k} y_k)_{1\leqslant k\leqslant p})$. Now let $\boldsymbol{x} \in \boldsymbol{\mathcal{H}}$ and $\boldsymbol{y} \in \boldsymbol{\mathcal{G}}$, and set $\boldsymbol{t} = (\mathbf{Id} + \boldsymbol{L}^*\boldsymbol{L})^{-1}(\boldsymbol{x} + \boldsymbol{L}^*\boldsymbol{y})$ and $\boldsymbol{s} = (\mathbf{Id} + \boldsymbol{L}\boldsymbol{L}^*)^{-1}(\boldsymbol{L}\boldsymbol{x} - \boldsymbol{y})$. Then

$$\operatorname{proj}_{\boldsymbol{V}}(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{t}, \boldsymbol{L}\boldsymbol{t}) = (\boldsymbol{x} - \boldsymbol{L}^*\boldsymbol{s}, \boldsymbol{y} + \boldsymbol{s}), \tag{4}$$

and we write it coordinate-wise as

$$\operatorname{proj}_{\boldsymbol{V}}(\boldsymbol{x},\boldsymbol{y}) = \big(Q_1(\boldsymbol{x},\boldsymbol{y}), \dots, Q_{m+p}(\boldsymbol{x},\boldsymbol{y})\big). \tag{5}$$

Thus, given $\gamma \in]0,+\infty[$, $\boldsymbol{z}_0 \in \boldsymbol{\mathcal{H}}$, and $\boldsymbol{y}_0 \in \boldsymbol{\mathcal{G}}$, the standard Douglas–Rachford algorithm for (3) is

for $n = 0, 1, \dots$
  $\lambda_n \in]0, 2[$
  for every $i \in \{1, \dots, m\}$
    $x_{i,n+1} = Q_i(\boldsymbol{z}_n, \boldsymbol{y}_n)$
    $z_{i,n+1} = z_{i,n}$
    $+\lambda_n\big(\operatorname{prox}_{\gamma f_i}(2x_{i,n+1} - z_{i,n}) - x_{i,n+1}\big)$
  for every $k \in \{1, \dots, p\}$
    $w_{k,n+1} = Q_{m+k}(\boldsymbol{z}_n, \boldsymbol{y}_n)$
    $y_{k,n+1} = y_{k,n}$
    $+\lambda_n\big(\operatorname{prox}_{\gamma g_k}(2w_{k,n+1} - y_{k,n}) - w_{k,n+1}\big)$.

The block-activated version of this algorithm is as follows.

Algorithm A ([15]) Let $\gamma \in]0,+\infty[$, let $\delta \in]0,1[$, let $\boldsymbol{x}_0$ and $\boldsymbol{z}_0$ be $\boldsymbol{\mathcal{H}}$-valued random variables (r.v.), let $\boldsymbol{y}_0$ and $\boldsymbol{w}_0$ be $\boldsymbol{\mathcal{G}}$-valued r.v. Iterate

for $j = 1, \dots, m+p$
  compute Q_j as in (4)–(5)
for $n = 0, 1, \dots$
  $\lambda_n \in [\delta, 2-\delta]$
  select randomly $\varnothing \neq I_n \subset \{1, \dots, m\}$
    and $\varnothing \neq K_n \subset \{1, \dots, p\}$
  for every $i \in I_n$
    $x_{i,n+1} = Q_i(\boldsymbol{z}_n, \boldsymbol{y}_n)$
    $z_{i,n+1} = z_{i,n}$
    $+\lambda_n\big(\operatorname{prox}_{\gamma f_i}(2x_{i,n+1} - z_{i,n}) - x_{i,n+1}\big)$
  for every $i \in \{1, \dots, m\} \smallsetminus I_n$
    $(x_{i,n+1}, z_{i,n+1}) = (x_{i,n}, z_{i,n})$
  for every $k \in K_n$
    $w_{k,n+1} = Q_{m+k}(\boldsymbol{z}_n, \boldsymbol{y}_n)$
    $y_{k,n+1} = y_{k,n}$
    $+\lambda_n\big(\operatorname{prox}_{\gamma g_k}(2w_{k,n+1} - y_{k,n}) - w_{k,n+1}\big)$
  for every $k \in \{1, \dots, p\} \smallsetminus K_n$
    $(w_{k,n+1}, y_{k,n+1}) = (w_{k,n}, y_{k,n})$.

The second algorithm operates by projecting onto hyperplanes which separate the current iterate from the set $\boldsymbol{\mathsf{Z}}$ of Kuhn–Tucker points of Problem 1, i.e., the points $\widetilde{\boldsymbol{x}} \in \boldsymbol{\mathcal{H}}$ and $\widetilde{\boldsymbol{v}}^* \in \boldsymbol{\mathcal{G}}$ such that

$$\begin{cases} (\forall i \in \{1, \dots, m\}) & -\sum_{k=1}^{p} L_{k,i}^*\widetilde{v}_k^* \in \partial f_i(\widetilde{x}_i) \\ (\forall k \in \{1, \dots, p\}) & \sum_{i=1}^{m} L_{k,i}\widetilde{x}_i \in \partial g_k^*(\widetilde{v}_k^*). \end{cases} \tag{6}$$

This process is explained in Fig. 1.

Algorithm B ([9]) Set $I_0 = \{1, \dots, m\}$ and $K_0 = \{1, \dots, p\}$, and let $\delta \in]0,1[$. For every $i \in I_0$ and every $k \in K_0$, let $\{\gamma_i, \mu_k\} \subset]0,+\infty[$, $x_{i,0} \in \mathcal{H}_i$, and $v_{k,0}^* \in \mathcal{G}_k$. Iterate

for $n = 0, 1, \dots$
  $\lambda_n \in [\delta, 2-\delta]$
  if $n > 0$
    select $\varnothing \neq I_n \subset I_0$ and $\varnothing \neq K_n \subset K_0$
  for every $i \in I_n$
    $x_{i,n}^* = x_{i,n} - \gamma_i \sum_{k=1}^{p} L_{k,i}^* v_{k,n}^*$
    $a_{i,n} = \operatorname{prox}_{\gamma_i f_i} x_{i,n}^*$
    $a_{i,n}^* = \gamma_i^{-1}(x_{i,n}^* - a_{i,n})$
  for every $i \in I_0 \smallsetminus I_n$
    $(a_{i,n}, a_{i,n}^*) = (a_{i,n-1}, a_{i,n-1}^*)$
  for every $k \in K_n$
    $y_{k,n}^* = \mu_k v_{k,n}^* + \sum_{i=1}^{m} L_{k,i} x_{i,n}$
    $b_{k,n} = \operatorname{prox}_{\mu_k g_k} y_{k,n}^*$
    $b_{k,n}^* = \mu_k^{-1}(y_{k,n}^* - b_{k,n})$
    $t_{k,n} = b_{k,n} - \sum_{i=1}^{m} L_{k,i} a_{i,n}$
  for every $k \in K_0 \smallsetminus K_n$
    $(b_{k,n}, b_{k,n}^*) = (b_{k,n-1}, b_{k,n-1}^*)$
    $t_{k,n} = b_{k,n} - \sum_{i=1}^{m} L_{k,i} a_{i,n}$
  for every $i \in I_0$
    $t_{i,n}^* = a_{i,n}^* + \sum_{k=1}^{p} L_{k,i}^* b_{k,n}^*$
  $\tau_n = \sum_{i=1}^{m} \|t_{i,n}^*\|^2 + \sum_{k=1}^{p} \|t_{k,n}\|^2$

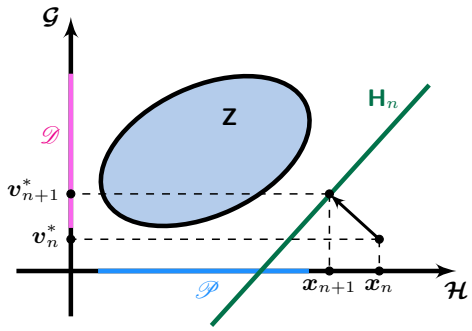

Fig. 1. Let $\mathscr{P}$ be the set of solutions to Problem 1 and let $\mathscr{D}$ be the set of solutions to its dual. Then the Kuhn–Tucker set $\mathbf{Z}$ is a subset of $\mathscr{P} \times \mathscr{D}$. At iteration n, the proximity operators of blocks of functions $(f_i)_{i\in I_n}$ and $(g_k)_{k\in K_n}$ are used to construct a hyperplane $\mathbf{H}_n$ separating the current primal-dual iterate $(\boldsymbol{x}_n, \boldsymbol{v}^*_n)$ from $\mathbf{Z}$, and the update $(\boldsymbol{x}_{n+1}, \boldsymbol{v}^*_{n+1})$ is obtained as its projection onto $\mathbf{H}_n$ [9].

$$
\begin{array}{l}
\text{if } \tau_n > 0 \\
\quad \pi_n = \sum_{i=1}^{m} \big(\langle x_{i,n} \mid t^*_{i,n}\rangle - \langle a_{i,n} \mid a^*_{i,n}\rangle\big) \\
\qquad + \sum_{k=1}^{p} \big(\langle t_{k,n} \mid v^*_{k,n}\rangle - \langle b_{k,n} \mid b^*_{k,n}\rangle\big) \\
\text{if } \tau_n > 0 \text{ and } \pi_n > 0 \\
\quad \theta_n = \lambda_n \pi_n / \tau_n \\
\quad \text{for every } i \in I_0 \\
\quad\quad x_{i,n+1} = x_{i,n} - \theta_n t^*_{i,n} \\
\quad \text{for every } k \in K_0 \\
\quad\quad v^*_{k,n+1} = v^*_{k,n} - \theta_n t_{k,n} \\
\text{else} \\
\quad \text{for every } i \in I_0 \\
\quad\quad x_{i,n+1} = x_{i,n} \\
\quad \text{for every } k \in K_0 \\
\quad\quad v^*_{k,n+1} = v^*_{k,n}.
\end{array}
$$

3. ASYMPTOTIC BEHAVIOR AND COMPARISONS

Let us first state the convergence results available for Algorithms A and B. We make the standing assumption that $\mathbf{Z} \neq \varnothing$ (see (6)), which implies that the solution set $\mathscr{P}$ of Problem 1 is nonempty.

Theorem 2 ([15]) *In Algorithm A, define, for every $n \in \mathbb{N}$ and every $j \in \{1, \ldots, m+p\}$, $\varepsilon_{j,n} = 1$, if $j \in I_n$ or $j - m \in K_n$; $\varepsilon_{j,n} = 0$, otherwise. Suppose that the following hold:*

(i) *The r.v. $(\boldsymbol{\varepsilon}_n)_{n\in\mathbb{N}}$ are identically distributed.*

(ii) *For every $n \in \mathbb{N}$, the r.v. $\boldsymbol{\varepsilon}_n$ and $(\boldsymbol{z}_j, \boldsymbol{y}_j)_{0\leqslant j\leqslant n}$ are mutually independent.*

(iii) $(\forall j \in \{1, \ldots, m+p\})\ \mathsf{Prob}[\varepsilon_{j,0} = 1] > 0$.

Then $(\boldsymbol{x}_n)_{n\in\mathbb{N}}$ converges almost surely to a $\mathscr{P}$-valued r.v.

Theorem 3 ([9]) *In Algorithm B, suppose that there exists $T \in \mathbb{N}$ such that $(\forall n \in \mathbb{N}) \bigcup_{j=n}^{n+T} I_j = \{1, \ldots, m\}$ and $\bigcup_{j=n}^{n+T} K_j = \{1, \ldots, p\}$. Then $(\boldsymbol{x}_n)_{n\in\mathbb{N}}$ converges to a point in $\mathscr{P}$.*

Let us compare Algorithms A and B.

a/ **Auxiliary tasks:** A requires the construction and storage of the operators $(Q_j)_{1\leqslant j\leqslant m+p}$ of (4)–(5), which can be quite demanding as they involve inversion of a linear operator acting on the product space $\boldsymbol{\mathcal{H}}$ or $\boldsymbol{\mathcal{G}}$ (see (5)). By contrast, B does not require such tasks.

b/ **Proximity operators:** Both algorithms are block-activated: only the blocks of functions $(f_i)_{i\in I_n}$ and $(g_k)_{k\in K_n}$ need to be activated at iteration n. Each selected function is activated via its proximity operator.

c/ **Linear operators:** In A, the operators $(Q_i)_{i\in I_n}$ and $(Q_{m+k})_{k\in K_n}$ selected at iteration n are evaluated at $(z_{1,n}, \ldots, z_{m,n}, y_{1,n}, \ldots, y_{p,n}) \in \boldsymbol{\mathcal{H}} \times \boldsymbol{\mathcal{G}}$. On the other hand, B activates the local operators $L_{k,i}\colon \mathcal{H}_i \to \mathcal{G}_k$ and $L^*_{k,i}\colon \mathcal{G}_k \to \mathcal{H}_i$ once or twice, depending on whether they are selected. Thus, if we set $N = \dim \boldsymbol{\mathcal{H}}$ and $M = \dim \boldsymbol{\mathcal{G}}$ and if the linear operators are implemented in matrix form, then the corresponding load per iteration in full activation mode of A is $\mathcal{O}((M+N)^2)$ versus $\mathcal{O}(MN)$ in B.

d/ **Activation scheme:** Both algorithms have the ability to activate several functions simultaneously, and they are therefore naturally implementable on parallel architectures. As A selects the blocks randomly, the user does not have complete control of the computational load of an iteration, whereas the load of B is more predictable because of its deterministic activation scheme.

e/ **Parameters:** A single scale parameter γ is used in A, while B allows the proximity operators to have their own scale parameters $(\gamma_1, \ldots, \gamma_m, \mu_1, \ldots, \mu_p)$. This gives B more flexibility, but more effort may be needed *a priori* to find efficient parameters. Further, in both algorithms, there is no restriction on the parameter values.

f/ **Convergence:** B, which activates the blocks deterministically, guarantees sure convergence under the mild repetitive activation condition in Theorem 3, while A, which activates the blocks randomly, guarantees only almost sure convergence.

g/ **Other features:** Although this point is omitted for brevity, unlike A, B can be executed asynchronously with iteration-dependent scale parameters [9].

4. NUMERICAL EXPERIMENTS

We present two experiments which are reflective of our numerical investigations in solving various problems using Algorithms A and B. The main objective is to illustrate the block processing ability of the algorithms (when implemented with full activation, i.e., $I_n = I_0$ and $K_n = K_0$, Algorithm B was already shown in [10] to be quite competitive compared to existing methods).

4.1. Experiment 1: Group-Sparse Binary Classification

We revisit the classification problem of [12], which is based on the latent group lasso formulation in machine learning [21]. Let $\{G_1, \ldots, G_m\}$ be a covering of $\{1, \ldots, d\}$ and define $X = \big\{(x_1, \ldots, x_m) \mid x_i \in \mathbb{R}^d,\ \mathrm{support}(x_i) \subset G_i\big\}$. The sought vector is $\widetilde{y} = \sum_{i=1}^{m} \widetilde{x}_i$, where $(\widetilde{x}_1, \ldots, \widetilde{x}_m)$ solves

$$
\underset{(x_1,\ldots,x_m)\in X}{\text{minimize}} \ \sum_{i=1}^{m} \|x_i\|_2 + \sum_{k=1}^{p} g_k\Bigg(\sum_{i=1}^{m} \langle x_i \mid u_k\rangle\Bigg), \qquad (7)
$$

with $u_k \in \mathbb{R}^d$ and $g_k\colon \mathbb{R} \to \mathbb{R}\colon \xi \mapsto 10\max\{0, 1 - \beta_k\xi\}$, where $\beta_k = \omega_k \operatorname{sign}(\langle \overline{y} \mid u_k\rangle)$ is the kth measurement of the true vector

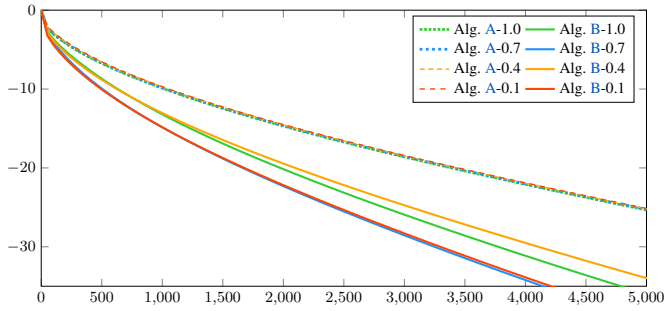

Fig. 2. Normalized error $20\log_{10}(\|\boldsymbol{x}_n - \boldsymbol{x}_\infty\|/\|\boldsymbol{x}_0 - \boldsymbol{x}_\infty\|)$ (dB), averaged over 20 runs, versus epoch count in Experiment 1. The variations around the averages were not significant. The computational load per epoch for both algorithms is comparable.

$\overline{y} \in \mathbb{R}^d$ ($d = 10000$) and $\omega_k \in \{-1, 1\}$ induces 25% classification error. There are $p = 1000$ measurements and the goal is to reconstruct the group-sparse vector $\overline{y}$. There are $m = 1429$ groups. For every $i \in \{1, \dots, m-1\}$, each G_i has 10 consecutive integers and an overlap with G_{i+1} of length 3. We obtain an instance of (1), where $\mathcal{H}_i = \mathbb{R}^{10}$, $f_i = \|\cdot\|_2$, and $L_{k,i} = \langle \cdot \mid u_k|_{G_i} \rangle$. The auxiliary tasks for Algorithm A (see a/) are negligible [12]. For each $\alpha \in \{0.1, 0.4, 0.7, 1.0\}$, at iteration $n \in \mathbb{N}$, I_n has $\lceil \alpha m \rceil$ elements and the proximity operators of the scalar functions $(g_k)_{1 \leqslant k \leqslant p}$ are all used, i.e., $K_n = \{1, \dots, p\}$. We display in Fig. 2 the normalized error versus the epoch, that is, the cumulative number of activated blocks in $\{1, \dots, m\}$ divided by m.

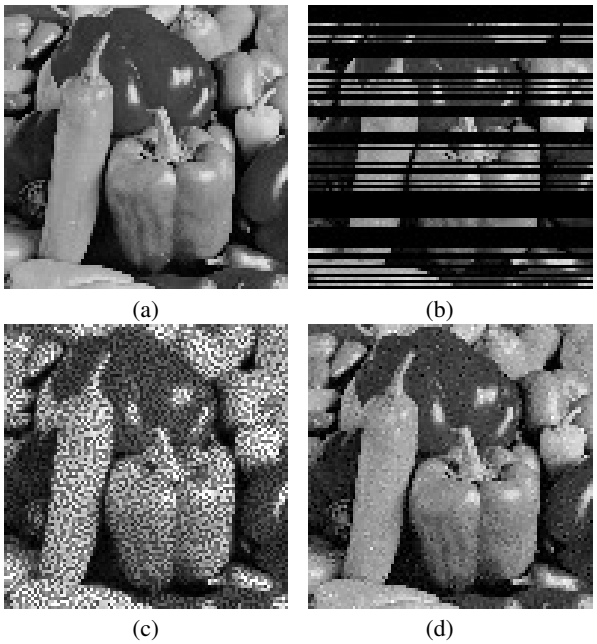

Fig. 3. Experiment 2: (a) Original $\overline{x}$. (b) Observation b. (c) Observation c. (d) Recovery (all recoveries were visually indistinguishable).

4.2. Experiment 2: Image Recovery

We revisit the image interpolation problem of [10, Section 4.3]. The objective is to recover the image $\overline{x} \in C = [0, 255]^N$ ($N = 96^2$) of Fig. 3(a), given a noisy masked observation $b = M\overline{x} + w_1 \in \mathbb{R}^N$ and a noisy blurred observation $c = H\overline{x} + w_2 \in \mathbb{R}^N$. Here, M masks all but $q = 39$ rows $(x^{(r_k)})_{1 \leqslant k \leqslant q}$ of an image x, and H is a nonstationary blurring operator, while w_1 and w_2 yield signal-to-noise ratios of 28.5 dB and 27.8 dB, respectively. We split H into $s = 384$ subblocks: for every $k \in \{1, \dots, s\}$, $H_k \in \mathbb{R}^{24 \times N}$ and the corresponding block of c is denoted c_k. The goal is to

$$\underset{x \in C}{\text{minimize}} \ \|Dx\|_{1,2} + 10 \sum_{k=1}^{q} \left\| x^{(r_k)} - b^{(r_k)} \right\|_2 + 5 \sum_{k=1}^{s} \|H_k x - c_k\|_2^2, \tag{8}$$

where $D \colon \mathbb{R}^N \to \mathbb{R}^N \times \mathbb{R}^N$ models finite differences and, given vectors $y_1 = (\eta_{1,j})_{1 \leqslant j \leqslant N}$ and $y_2 = (\eta_{2,j})_{1 \leqslant j \leqslant N}$, $\|(y_1, y_2)\|_{1,2} = \sum_{j=1}^{N} \|(\eta_{1,j}, \eta_{2,j})\|_2$. Thus, (8) is an instance of Problem 1, where $m = 1$; $p = q + s + 1$; for every $k \in \{1, \dots, q\}$, $L_{k,1} \colon \mathbb{R}^N \to \mathbb{R}^{\sqrt{N}} \colon x \mapsto x^{(r_k)}$ and $g_k \colon y_k \mapsto 10\|y_k - b^{(r_k)}\|_2$; for every $k \in \{q+1, \dots, q+s\}$, $L_{k,1} = H_{k-q}$, $g_k = 5\|\cdot - c_k\|_2^2$, and $g_p = \|\cdot\|_{1,2}$; $L_{p,1} = D$; $f_1 \colon x \mapsto 0$ if $x \in C$; $+\infty$ if $x \notin C$. At iteration n, K_n has $\lceil \alpha p \rceil$ elements, where $\alpha \in \{0.1, 0.4, 0.7, 1.0\}$. The results are shown in Figs. 3–4, where the epoch is the cumulative number of activated blocks in $\{1, \dots, p\}$ divided by p.

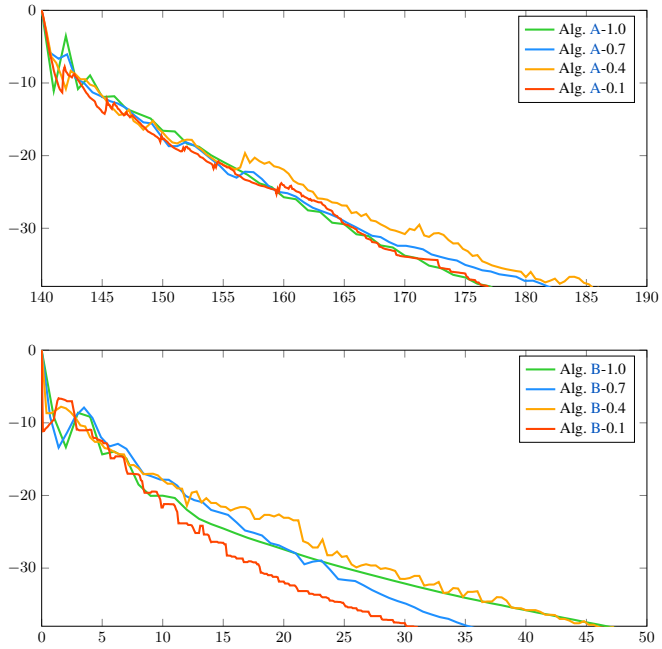

Fig. 4. Normalized error $20\log_{10}(\|\boldsymbol{x}_n - \boldsymbol{x}_\infty\|/\|\boldsymbol{x}_0 - \boldsymbol{x}_\infty\|)$ (dB) versus epoch count in Experiment 2. Top: Algorithm A. The horizontal axis starts at 140 epochs to account for the auxiliary tasks (see a/). Bottom: Algorithm B. The computational load per epoch for Algorithm B was about twice that of Algorithm A.

4.3. Discussion

Our first finding is that, for both Algorithms A and B, even when full activation is computationally possible, it may not be the best strategy (see Figs. 2 and 4). Second, a/–g/ and our experiments suggest that B is preferable to A. Let us add that, in general, A does not scale as well as B. For instance, in Experiment 2, if the image size scales up, B can still operate since it involves only individual applications of the local $L_{k,i}$ operators, while A becomes unmanageable because of the size of the Q_j operators (see a/ and [6]).

5. REFERENCES

[1] A. Argyriou, R. Foygel, and N. Srebro, Sparse prediction with the k-support norm, *Proc. Adv. Neural Inform. Process. Syst. Conf.*, vol. 25, pp. 1457–1465, 2012.

[2] J.-F. Aujol and A. Chambolle, Dual norms and image decomposition models, *Int. J. Comput. Vision*, vol. 63, pp. 85–104, 2005.

[3] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski, Optimization with sparsity-inducing penalties, *Found. Trends Machine Learn.*, vol. 4, pp. 1–106, 2012.

[4] J. M. Bioucas-Dias, A. Plaza, N. Dobigeon, M. Parente, Q. Du, P. Gader, and J. Chanussot, Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches, *IEEE J. Select. Topics Appl. Earth Observ. Remote Sensing*, vol. 5, pp. 354–379, 2012.

[5] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, *Found. Trends Machine Learn.*, vol. 3, pp. 1–122, 2010.

[6] L. M. Briceño-Arias, G. Chierchia, E. Chouzenoux, and J.-C. Pesquet, A random block-coordinate Douglas–Rachford splitting method with low computational complexity for binary logistic regression, *Comput. Optim. Appl.*, vol. 72, pp. 707–726, 2019.

[7] A. Chambolle, M. J. Ehrhardt, P. Richtárik, and C.-B. Schönlieb, Stochastic primal-dual hybrid gradient algorithm with arbitrary sampling and imaging applications, *SIAM J. Optim.*, vol. 28, pp. 2783–2808, 2018.

[8] A. Chambolle and T. Pock, An introduction to continuous optimization for imaging, *Acta Numer.*, vol. 25, pp. 161–319, 2016.

[9] P. L. Combettes and J. Eckstein, Asynchronous block-iterative primal-dual decomposition methods for monotone inclusions, *Math. Program. Ser. B*, vol. 168, pp. 645–672, 2018.

[10] P. L. Combettes and L. E. Glaudin, Proximal activation of smooth functions in splitting algorithms for convex image recovery, *SIAM J. Imaging Sci.*, vol. 12, pp. 1905–1935, 2019.

[11] P. L. Combettes and L. E. Glaudin, Solving composite fixed point problems with block updates, *Adv. Nonlinear Anal.*, vol. 10, 2021.

[12] P. L. Combettes, A. M. McDonald, C. A. Micchelli, and M. Pontil, Learning with optimal interpolation norms, *Numer. Algorithms*, vol. 81, pp. 695–717, 2019.

[13] P. L. Combettes and C. L. Müller, Perspective maximum likelihood-type estimation via proximal decomposition, *Electron. J. Stat.*, vol. 14, pp. 207–238, 2020.

[14] P. L. Combettes and J.-C. Pesquet, Proximal splitting methods in signal processing, *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pp. 185–212. Springer, 2011.

[15] P. L. Combettes and J.-C. Pesquet, Stochastic quasi-Fejér block-coordinate fixed point iterations with random sweeping, *SIAM J. Optim.*, vol. 25, pp. 1221–1248, 2015.

[16] P. L. Combettes and J.-C. Pesquet, Fixed point strategies in data science, *IEEE Trans. Signal Process.*, vol. 69, pp. 3878–3905, 2021.

[17] J. Darbon and T. Meng, On decomposition models in imaging sciences and multi-time Hamilton–Jacobi partial differential equations, *SIAM J. Imaging Sci.*, vol. 13, pp. 971–1014, 2020.

[18] A. J. Defazio, T. S. Caetano, and J. Domke, Finito: A faster, permutable incremental gradient method for big data problems, *Proc. Intl. Conf. Machine Learn.*, pp. 1125–1133, 2014.

[19] R. Glowinski, S. J. Osher, and W. Yin (Eds.), *Splitting Methods in Communication, Imaging, Science, and Engineering*. Springer, 2016.

[20] M. Hintermüller and G. Stadler, An infeasible primal-dual algorithm for total bounded variation-based inf-convolution-type image restoration, *SIAM J. Sci. Comput.*, vol. 28, pp. 1–23, 2006.

[21] L. Jacob, G. Obozinski, and J.-Ph. Vert, Group lasso with overlap and graph lasso, *Proc. Int. Conf. Machine Learn.*, pp. 433–440, 2009.

[22] P. R. Johnstone and J. Eckstein, Projective splitting with forward steps, *Math. Program. Ser. A*, published online 2020-09-30.

[23] K. Mishchenko, F. Iutzeler, and J. Malick, A distributed flexible delay-tolerant proximal gradient algorithm, *SIAM J. Optim.*, vol. 30, pp. 933–959, 2020.

[24] J.-C. Pesquet and A. Repetti, A class of randomized primal-dual algorithms for distributed optimization, *J. Nonlinear Convex Anal.*, vol. 16, pp. 2453–2490, 2015.

[25] S. Salzo and S. Villa, Parallel random block-coordinate forward-backward algorithm: A unified convergence analysis, *Math. Program. Ser. A*, published online 2021-04-11.

[26] M. Schmidt, N. Le Roux, and F. Bach, Minimizing finite sums with the stochastic average gradient, *Math. Program. Ser. A*, vol. 162, pp. 83–112, 2017.