Projective Splitting with Forward Steps: Asynchronous and Block-Iterative Operator Splitting

Patrick R. Johnstone* Jonathan Eckstein*

August 8, 2018

Abstract

This work is concerned with the classical problem of finding a zero of a sum of maximal monotone operators. For the projective splitting framework recently proposed by Combettes and Eckstein, we show how to replace the fundamental subproblem calculation using a backward step with one based on two forward steps. The resulting algorithms have the same kind of coordination procedure and can be implemented in the same block-iterative and potentially distributed and asynchronous manner, but may perform backward steps on some operators and forward steps on others. Prior algorithms in the projective splitting family have used only backward steps. Forward steps can be used for any Lipschitz-continuous operators provided the stepsize is bounded by the inverse of the Lipschitz constant. If the Lipschitz constant is unknown, a simple backtracking linesearch procedure may be used. For affine operators, the stepsize can be chosen adaptively without knowledge of the Lipschitz constant and without any additional forward steps. We close the paper by empirically studying the performance of several kinds of splitting algorithms on large-scale lasso problems.

1 Introduction

For a collection of real Hilbert spaces $\{\mathcal{H}_i\}_{i=0}^n$, consider the problem of finding $z \in \mathcal{H}_0$ such that

$$0 \in \sum_{i=1}^{n} G_i^* T_i(G_i z), \tag{1}$$

where $G_i : \mathcal{H}_0 \to \mathcal{H}_i$ are linear and bounded operators, $T_i : \mathcal{H}_i \to 2^{\mathcal{H}_i}$ are maximal monotone operators and additionally there exists a subset $\mathcal{I}_F \subseteq \{1, \ldots, n\}$ such that for all $i \in \mathcal{I}_F$ the operator T_i is Lipschitz continuous. An important instance of this problem is

$$\min_{x \in \mathcal{H}_0} \sum_{i=1}^{n} f_i(G_i x), \tag{2}$$

*Department of Management Science and Information Systems, Rutgers Business School Newark and New Brunswick, Rutgers University

where every $f_i : \mathcal{H}_i \to \mathbb{R}$ is closed, proper and convex, with some subset of the functions also being differentiable with Lipschitz-continuous gradients. Under appropriate constraint qualifications, (1) and (2) are equivalent. Problem (2) arises in a host of applications such as machine learning, signal and image processing, inverse problems, and computer vision; see [4, 9, 11] for some examples. Operator splitting algorithms are now a common way to solve structured monotone inclusions such as (1). Until recently, there were three underlying classes of operator splitting algorithms: forward-backward [26], Douglas/Peaceman-Rachford [24], and forward-backward-forward [32]. In [13], Davis and Yin introduced a new operator splitting algorithm which does not reduce to any of these methods. Many algorithms for more complicated monotone inclusions and optimization problems involving many terms and constraints are in fact applications of one of these underlying techniques to a reduced monotone inclusion in an appropriately defined product space [6, 21, 12, 5, 10]. These four operator splitting techniques are, in turn, a special case of the *Krasnoselskii-Mann (KM) iteration* for finding a fixed point of a nonexpansive operator [23, 25].

A different, relatively recently proposed class of operator splitting algorithms is *projective splitting*: this class has a different convergence mechanism based on projection onto separating sets and does not in general reduce to the KM iteration. The root ideas underlying projective splitting can be found in [20, 30, 31] which dealt with monotone inclusions with a single operator. The algorithm of [17] significantly built on these ideas to address the case of two operators and was thus the original projective "splitting" method. This algorithm was generalized to more than two operators in [18]. The related algorithm in [1] introduced a technique for handling compositions of linear and monotone operators, and [8] proposed an extension to "block-iterative" and asynchronous operation — block-iterative operation meaning that only a subset of the operators making up the problem need to be considered at each iteration (this approach may be called "incremental" in the optimization literature). A restricted and simplified version of this framework appears in [16]. The asynchronous and block-iterative nature of projective splitting as well as its ability to handle composition with linear operators gives it an unprecedented level of flexibility compared with prior classes of operator splitting methods, none of which can be readily implemented in an asynchronous or block-iterative manner. Further, in the projective splitting methods of [8, 16] the order with which operators can be processed is deterministic, variable, and highly flexible. It is not necessary that each operator be processed the same number of times either exactly or approximately; in fact, one operator may be processed much more often than another. The only constraint is that there is an upper bound on the number of iterations between the consecutive times that each operator is processed.

Projective splitting algorithms work by performing separate calculations on each individual operator to construct a separating hyperplane between the current iterate and the problem's *Kuhn-Tucker set* (essentially the set of primal and dual solutions), and then projecting onto the this hyperplane. In prior projective splitting algorithms, the only operation performed on the individual operators T_i is a proximal (backward) step, which consists of evaluating the operator resolvents $(I + \rho T_i)^{-1}$ for some scalar $\rho > 0$. In this paper, we show how, for the Lipschitz continuous operators, the same kind of framework can also make use of forward steps on the individual operators, equivalent to applying $I - \rho T_i$. Typically, such "explicit" steps are computationally much easier than "implicit", proximal steps. Our procedure requires two forward steps each time it evaluates an operator, and in this sense is reminiscent of Tseng's forward-backward-forward method [32] and Korpelevich's extragradient method [22]. Indeed, for the special case of only one operator, projective splitting with the new procedure reduces to the variant of the extragradient method in [20]. Each stepsize must be bounded by the inverse of the Lipschitz constant of T_i. However, a simple backtracking procedure can eliminate the need to estimate the Lipschitz constant, and other options are available for selecting the stepsize when T_i is affine.

1.1 Intuition and contributions: basic idea

We first provide some intuition into our fundamental idea of incorporating forward steps into projective splitting. For simplicity, consider (1) without the linear operators G_i, that is, we want to find z such that $0 \in \sum_{i=1}^n T_i z$, where $T_1, \ldots, T_n : \mathcal{H}_0 \to 2^{\mathcal{H}_0}$ are maximal monotone operators on a single real Hilbert space $\mathcal{H}_0$. We formulate the Kuhn-Tucker solution set of this problem as

$$\mathcal{S} = \left\{(z, w_1, \ldots, w_{n-1}) \;\middle|\; (\forall\, i \in \{1, \ldots, n-1\}) \;\; w_i \in T_i z, \;\; -\textstyle\sum_{i=1}^{n-1} w_i \in T_n z \right\}. \tag{3}$$

It is clear that z^* solves $0 \in \sum_{i=1}^n T_i z^*$ if and only if there exist $w_1^*, \ldots, w_{n-1}^*$ such that $(z^*, w_1^*, \ldots, w_{n-1}^*) \in \mathcal{S}$. A separator-projector algorithm for finding a point in $\mathcal{S}$ will, at each iteration k, find a closed and convex set H_k which separates $\mathcal{S}$ from the current point, meaning $\mathcal{S}$ is entirely in the set and the current point is not. One can then move closer to the solution set by projecting the current point onto the set H_k.

If we define $\mathcal{S}$ as in (3), then the separator formulation presented in [8] constructs the set H_k through the function

$$\varphi_k(z, w_1, \ldots, w_{n-1}) = \sum_{i=1}^{n-1} \langle z - x_i^k, y_i^k - w_i \rangle + \left\langle z - x_i^n, y_i^n + \sum_{i=1}^{n-1} w_i \right\rangle \tag{4}$$

$$= \left\langle z, \sum_{i=1}^{n} y_i^k \right\rangle + \sum_{i=1}^{n-1} \langle x_i^k - x_n^k, w_i \rangle - \sum_{i=1}^{n} \langle x_i^k, y_i^k \rangle, \tag{5}$$

for some $x_i^k, y_i^k \in \mathcal{H}_0$ such that $y_i^k \in T_i x_i^k$, $i \in 1, \ldots, n$. From its expression in (5) it is clear that φ_k is an affine function on $\mathcal{H}_0^n$. Furthermore, it may easily be verified that for any $p = (z, w_1, \ldots, w_{n-1}) \in \mathcal{S}$, one has $\varphi_k(p) \leq 0$, so that the separator set H_k may be taken to be the halfspace $\{p \mid \varphi_k(p) \leq 0\}$. The key idea of projective splitting is, given a current iterate $p^k = (z^k, w_1^k, \ldots, w_{n-1}^k) \in \mathcal{H}_0^n$, to pick (x_i^k, y_i^k) so that $\varphi_k(p^k)$ is positive if $p^k \notin \mathcal{S}$. Then, since the solution set is entirely on the other side of the hyperplane $\{p \mid \varphi_k(p) = 0\}$, projecting the current point onto this hyperplane makes progress toward the solution. If it can be shown that this progress is sufficiently large, then it is possible to prove (weak) convergence.

Let the iterates of such an algorithm be $p^k = (z^k, w_i^k, \ldots, w_{n-1}^k) \in \mathcal{H}_0^n$. To simplify the subsequent analysis, define $w_n^k \triangleq -\sum_{i=1}^{n-1} w_i^k$ at each iteration k, whence it is immediate from (4) that $\varphi_k(p^k) = \varphi_k(z^k, w_1^k, \ldots, w_{n-1}^k) = \sum_{i=1}^n \langle z^k - x_i^k, y_i^k - w_i^k \rangle$. To construct a function φ_k of the form (4) such that $\varphi_k(p^k) = \varphi_k(z^k, w_1^k, \ldots, w_n^k) > 0$ whenever $p^k \notin \mathcal{S}$, it is sufficient to be able to perform the following calculation on each individual operator T_i:

for $(z^k, w_i^k) \in \mathcal{H}_0^2$, find $x_i^k, y_i^k \in \mathcal{H}_0$ such that $y_i^k \in T_i x_i^k$ and $\langle z^k - x_i^k, y_i^k - w_i^k \rangle \geq 0$, with $\langle z^k - x_i^k, y_i^k - w_i^k \rangle > 0$ if $w_i^k \notin T_i z^k$. In earlier work on projective splitting [17, 18, 8, 1], the calculation of such a (x_i^k, y_i^k) is accomplished by a proximal (implicit) step on the operator T_i: given a scalar $\rho > 0$, we find the unique pair $(x_i^k, y_i^k) \in \mathcal{H}_0^2$ such that $y_i^k \in T_i x_i^k$ and

$$x_i^k + \rho y_i^k = z^k + \rho w_i^k \quad \Rightarrow \quad z^k - x_i^k = \rho(y_i^k - w_i^k). \tag{6}$$

We immediately conclude that

$$\langle z^k - x_i^k, y_i^k - w_i^k \rangle = (1/\rho)\|z^k - x_i^k\|^2 \geq 0, \tag{7}$$

and furthermore that $\langle z^k - x_i^k, y_i^k - w_i^k \rangle > 0$ unless $x_i^k = z^k$, which would in turn imply that $y_i^k = w_i^k$ and $w_i^k \in T_i z^k$. If we perform such a calculation for each $i = 1, \ldots, n$, we have constructed a separator of the form (4) which, in view of $\varphi_k(p^k) = \sum_{i=1}^n \langle z^k - x_i^k, y_i^k - w_i^k \rangle$, has $\varphi_k(p^k) > 0$ if $p^k \notin \mathcal{S}$. This basic calculation on T_i is depicted in Figure 1(a) for $\mathcal{H}_0 = \mathbb{R}^1$: since $z^k - x_i^k = \rho(y_i^k - w_i^k)$, the line segment between (z^k, w_i^k) and (x_i^k, y_i^k) must have slope $-1/\rho$, meaning that $\langle z^k - x_i^k, w_i^k - y_i^k \rangle \leq 0$ and thus that $\langle z^k - x_i^k, y_i^k - w_i^k \rangle \geq 0$. It also bears mentioning that the relation (7) plays (in generalized form) a key role in the convergence proof.

Consider now the case that T_i is Lipschitz continuous with modulus $L_i \geq 0$ (and hence single valued) and defined throughout $\mathcal{H}_0$. We now introduce a technique to accomplish something similar to the preceding calculation through two forward steps instead of a single backward step. We begin by evaluating $T_i z^k$ and using this value in place of y_i^k in the right-hand equation in (6), yielding

$$z^k - x_i^k = \rho\big(T_i z^k - w_i^k\big) \quad \Rightarrow \quad x_i^k = z^k - \rho\big(T_i z^k - w_i^k\big), \tag{8}$$

and we use this value for x_i^k. This calculation is depicted by the lower left point in Figure 1(b). We then calculate $y_i^k = T_i x_i^k$, resulting in a pair (x_i^k, y_i^k) on the graph of the operator; see the upper left point in Figure 1(b). For this choice of (x_i^k, y_i^k), we next observe that

$$\begin{aligned}
\langle z^k - x_i^k, y_i^k - w_i^k \rangle &= \big\langle z^k - x_i^k, T_i z^k - w_i^k \big\rangle - \big\langle z^k - x_i^k, T_i z^k - y_i^k \big\rangle \\
&= \Big\langle z^k - x_i^k, \tfrac{1}{\rho}(z^k - x_i^k) \Big\rangle - \big\langle z^k - x_i^k, T_i z^k - T_i x_i^k \big\rangle && (9) \\
&\geq \tfrac{1}{\rho} \big\|z^k - x_i^k\big\|^2 - L_i \big\|z^k - x_i^k\big\|^2 && (10) \\
&= \left(\frac{1}{\rho} - L_i\right) \big\|z^k - x_i^k\big\|^2 . && (11)
\end{aligned}$$

Here, (9) follows because $T_i z^k - w_i^k = (1/\rho)(z^k - x_i^k)$ from (8) and because we let $y_i^k = T_i x_i^k$. The inequality (10) then follows from the Cauchy-Schwarz inequality and the hypothesized Lipschitz continuity of T_i. If we require that $\rho < 1/L_i$, then we have $1/\rho > L_i$ and (11) therefore establishes that $\langle z^k - x_i^k, y_i^k - w_i^k \rangle \geq 0$, with $\langle z^k - x_i^k, y_i^k - w_i^k \rangle > 0$ unless $x_i^k = z^k$, which would imply that $w_i^k = T_i z^k$. We thus obtain a conclusion very similar to (7) and the results immediately following from it, but using the constant $1/\rho - L_i > 0$ in place of the positive constant $1/\rho$.

For $\mathcal{H}_0 = \mathbb{R}^1$, this process is depicted in Figure 1(b). By construction, the line segment between $\big(z^k, T_i z^k\big)$ and (x_i^k, w_i^k) has slope $1/\rho$, which is "steeper" than the graph of the

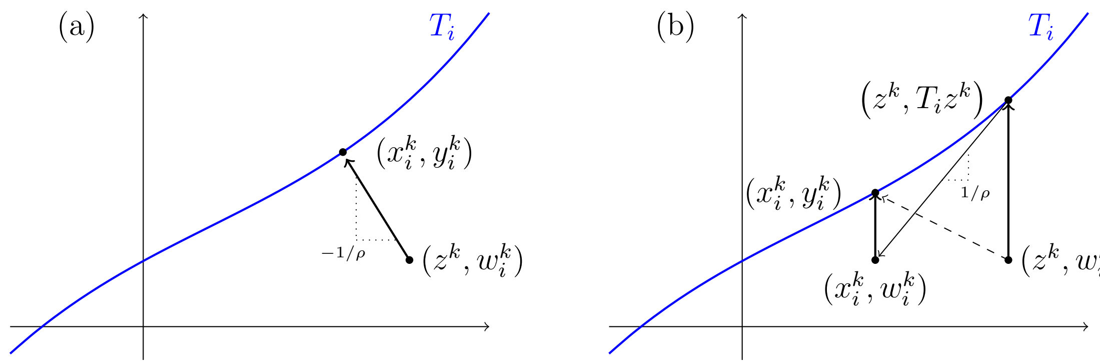

Figure 1: Backward and forward operator calculations in $\mathcal{H}_0 = \mathbb{R}^1$. The goal is to find a point (x_i^k, y_i^k) on the graph of the operator such that line segment connecting (z^k, w_i^k) and (x_i^k, y_i^k) has negative slope. Part (a) depicts a standard backward-step-based construction, while (b) depicts our new construction based on two forward steps.

operator, which can have slope at most L_i by Lipschitz continuity. This guarantees that the line segment between (z^k, w_i^k) and (x_i^k, y_i^k) must have negative slope, which in $\mathbb{R}^1$ is equivalent to the claimed inner product property.

Using a backtracking line search, we will also be able to handle the situation in which the value of L_i is unknown. If we choose any positive constant $\Delta > 0$, then by elementary algebra the inequalities $(1/\rho) - L_i \geq \Delta$ and $\rho \leq 1/(L_i + \Delta)$ are equivalent. Therefore, if we select some positive $\rho \leq 1/(L_i + \Delta)$, we have from (11) that

$$\langle z^k - x_i^k, y_i^k - w_i^k \rangle \geq \Delta \|z^k - x_i^k\|^2, \tag{12}$$

which implies the key properties we need for the convergence proofs. Therefore we may start with any $\rho = \rho^0 > 0$, and repeatedly halve ρ until (12) holds; in Section 4.1 below, we bound the number of halving steps required. In general, each trial value of ρ requires one application of the Lipschitz continuous operator T_i. However, for the case of affine operators T_i, we will show that it is possible to compute a stepsize such that (12) holds with a total of only two applications of the operator. By contrast, most backtracking procedures in optimization algorithms require evaluating the objective function at each new candidate point, which in turn usually requires an additional matrix multiply operation in the quadratic case [3].

1.2 Summary of Contributions

The main thrust of the remainder of this paper is to incorporate the second, forward-step construction of (x_i^k, y_i^k) above into an algorithm resembling those of [8, 16], allowing some operators to use backward steps, and others to use forward steps. Thus, projective splitting may become useful in a broad range of applications in which computing forward steps is preferable to computing or approximating proximal steps. The resulting algorithm inherits the asynchronous and block-iterative features of [8, 16]. It is worth mentioning that the stepsize constraints are unaffected by asynchrony — increasing the delays involved in communicating information between parts of the algorithm does not require smaller stepsizes. This contrasts with other asynchronous optimization and operator splitting algorithms [28, 27]. Another useful feature of the stepsizes is that they are allowed to vary across operators and iterations.

Like previous asynchronous projective splitting methods [16, 8], the asynchronous method developed here does not rely on randomization, nor is the algorithm formulated in terms of some fixed communication graph topology.

We will work with a slight restriction of problem (1), namely

$$0 \in \sum_{i=1}^{n-1} G_i^* T_i(G_i z) + T_n(z). \tag{13}$$

In terms of problem (1), we are simply requiring that G_n be the identity operator and thus that $\mathcal{H}_n = \mathcal{H}_0$. This is not much of a restriction in practice, since one could redefine the last operator as $T_n \leftarrow G_n^* \circ T_n \circ G_n$, or one could simply append a new operator T_n with $T_n(z) = \{0\}$ everywhere.

The principle reason for adopting a formulation involving the linear operators G_i is that in many applications of (13) it may be relatively easy to compute the proximal step of T_i but difficult to compute the proximal step of $G_i^* \circ T_i \circ G_i$. Our framework will include algorithms for (13) that may compute the proximal steps on T_i, forward steps when T_i is Lipschitz continuous, and applications ("matrix multiplies") of G_i and G_i^*. An interesting feature of the forward steps in our method is that while the allowable stepsizes depend on the Lipschitz constants of the T_i for $i \in \mathcal{I}_{\mathrm{F}}$, they do not depend on the linear operator norms $\|G_i\|$, in contrast with primal-dual methods [6, 12, 33]. Furthermore as mentioned the stepsizes used for each operator can be chosen independently and may vary by iteration.

We also suggest a greedy heuristic for selecting operators in block-iterative splitting, based on a simple proxy. Augmenting this heuristic with a straightforward safeguard allows one to retain all of the convergence properties of the main algorithm. The heuristic is not specifically tied to the use of forward steps and also applies to the earlier algorithms in [8, 16]. The numerical experiments in Section 5 below attest to its usefulness.

2 Mathematical Preliminaries

2.1 Notation

Summations of the form $\sum_{i=1}^{n-1} a_i$ for some collection $\{a_i\}$ will appear throughout this paper. To deal with the case $n = 1$, we use the standard convention that $\sum_{i=1}^{0} a_i = 0$. To ease the mathematical presentation, we use the following notation throughout the rest of the paper:

$$G_n : \mathcal{H}_n \to \mathcal{H}_n \triangleq I \quad \text{(the identity operator)} \qquad (\forall\, k \in \mathbb{N}) \quad w_n^k \triangleq -\sum_{i=1}^{n-1} G_i^* w_i^k. \tag{14}$$

Note that when $n = 1$, $w_1^k = 0$. We will use a boldface $\mathbf{w} = (w_1, \ldots, w_{n-1})$ for elements of $\mathcal{H}_1 \times \ldots \times \mathcal{H}_{n-1}$.

Throughout, we will simply write $\|\cdot\|_i = \|\cdot\|$ as the norm for $\mathcal{H}_i$ and let the subscript be inferred from the argument. In the same way, we will write $\langle\cdot,\cdot\rangle_i$ as $\langle\cdot,\cdot\rangle$ for the inner product of $\mathcal{H}_i$. For the collective primal-dual space defined in Section 2.3 we will use a special norm and inner product with its own subscript.

For any maximal monotone operator A we will use the notation $\text{prox}_{\rho A} = (I + \rho A)^{-1}$, for any scalar $\rho > 0$, to denote the *proximal operator*, also known as the backward or implicit step with respect to A. This means that

$$x = \text{prox}_{\rho A}(a) \quad \Longrightarrow \quad \exists y \in Ax : x + \rho y = a.$$

The x and y satisfying this relation are unique. Furthermore, $\text{prox}_{\rho A}$ is defined everywhere and $\text{range}(\text{prox}_A) = \text{dom}(A)$ [2, Prop. 23.2].

We use the standard "$\rightharpoonup$" notation to denote weak convergence, which is of course equivalent to ordinary convergence in finite-dimensional settings.

The following basic result will be used several times in our proofs:

Lemma 1. *For any vectors* $v_1, \ldots, v_n$, $\left\|\sum_{i=1}^n v_i\right\|^2 \leq n \sum_{i=1}^n \|v_i\|^2$.

Proof. $\left\|\sum_{i=1}^n v_i\right\|^2 = n^2 \left\|\frac{1}{n}\sum_{i=1}^n v_i\right\|^2 \leq n^2 \cdot \frac{1}{n}\sum_{i=1}^n \|v_i\|^2$, where the inequality follows from the convexity of the function $\|\cdot\|^2$. □

2.2 A Generic Linear Separator-Projection Method

Suppose that $\boldsymbol{\mathcal{H}}$ is a real Hilbert space with inner product $\langle\cdot,\cdot\rangle_{\boldsymbol{\mathcal{H}}}$ and norm $\|\cdot\|_{\boldsymbol{\mathcal{H}}}$. A generic linear separator-projection method for finding a point in some closed and convex set $\mathcal{S} \subseteq \boldsymbol{\mathcal{H}}$ is given in Algorithm 1.

Algorithm 1: Generic linear separator-projection method for finding a point in a closed and convex set $\mathcal{S} \subseteq \boldsymbol{\mathcal{H}}$.

```
   Input: p^1, 0 < β_ ≤ β̄ < 2
1  for k = 1, 2, ..., do
2      Find an affine function φ_k such that ∇φ_k ≠ 0 and φ_k(p) ≤ 0 for all p ∈ S.
3      Choose β_k ∈ [β_, β̄]
4      p^{k+1} = p^k − (β_k max{0, φ_k(p^k)} / ||∇φ_k||²_H) ∇φ_k
```

Input: p^1, $0 < \underline{\beta} \leq \overline{\beta} < 2$

1 **for** $k = 1, 2, \ldots,$ **do**

2 Find an affine function φ_k such that $\nabla\varphi_k \neq 0$ and $\varphi_k(p) \leq 0$ for all $p \in \mathcal{S}$.

3 Choose $\beta_k \in [\underline{\beta}, \overline{\beta}]$

4 $p^{k+1} = p^k - \frac{\beta_k \max\{0, \varphi_k(p^k)\}}{\|\nabla\varphi_k\|^2_{\boldsymbol{\mathcal{H}}}} \nabla\varphi_k$

The update on line 4 is the β_k-relaxed projection of p^k onto the halfspace $\{p : \varphi_k(p) \leq 0\}$ using the norm $\|\cdot\|_{\boldsymbol{\mathcal{H}}}$. In other words, if $\hat{p}^k$ is the projection onto this halfspace, then the update is $p^{k+1} = (1 - \beta_k)p^k + \beta_k \hat{p}^k$. Note that we define the gradient $\nabla\varphi_k$ with respect to the inner product $\langle\cdot,\cdot\rangle_{\boldsymbol{\mathcal{H}}}$, meaning we can write

$$(\forall p, \tilde{p} \in \boldsymbol{\mathcal{H}}) : \quad \varphi_k(p) = \langle\nabla\varphi_k, p - \tilde{p}\rangle_{\boldsymbol{\mathcal{H}}} + \varphi_k(\tilde{p}).$$

We will use the following well-known properties of algorithms fitting the template of Algorithm 1; see for example [7, 17]:

Lemma 2. *Suppose $\mathcal{S}$ is closed and convex. Then for Algorithm 1,*

1. *The sequence $\{p^k\}$ is bounded.*

2. $\|p^k - p^{k+1}\|_{\boldsymbol{\mathcal{H}}} \to 0$;

3. *If all weak limit points of* $\{p^k\}$ *are in* $\mathcal{S}$*, then* p^k *converges weakly to some point in* $\mathcal{S}$*.*

Note that we have not specified how to choose the affine function φ_k. For our specific application of the separator projector framework, we will do so in Section 3.2.

2.3 Main Assumptions Regarding Problem (13)

Let $\boldsymbol{\mathcal{H}} = \mathcal{H}_0 \times \mathcal{H}_1 \times \cdots \times \mathcal{H}_{n-1}$ and $\mathcal{H}_n = \mathcal{H}_0$. Define the *extended solution set* or *Kuhn-Tucker set* of (13) to be

$$\mathcal{S} = \left\{ (z, w_1, \ldots, w_{n-1}) \in \boldsymbol{\mathcal{H}} \;\middle|\; w_i \in T_i(G_i z),\; i = 1, \ldots, n-1,\; -\sum_{i=1}^{n-1} G_i^* w_i \in T_n(z) \right\}. \quad (15)$$

Clearly $z \in \mathcal{H}_0$ solves (13) if and only if there exists $\mathbf{w} \in \mathcal{H}_1 \times \cdots \times \mathcal{H}_{n-1}$ such that $(z, \mathbf{w}) \in \mathcal{S}$. Our main assumptions regarding (13) are as follows:

Assumption 1. *Problem (13) conforms to the following:*

1. $\mathcal{H}_0 = \mathcal{H}_n$ *and* $\mathcal{H}_1, \ldots, \mathcal{H}_{n-1}$ *are real Hilbert spaces.*

2. *For* $i = 1, \ldots, n$*, the operators* $T_i : \mathcal{H}_i \to 2^{\mathcal{H}_i}$ *are monotone.*

3. *For all* i *in some subset* $\mathcal{I}_{\mathrm{F}} \subseteq \{1, \ldots, n\}$*, the operator* T_i *is* L_i*-Lipschitz continuous (and thus single-valued) and* $\mathrm{dom}(T_i) = \mathcal{H}_i$*.*

4. *For* $i \in \mathcal{I}_{\mathrm{B}} \triangleq \{1, \ldots, n\} \backslash \mathcal{I}_{\mathrm{F}}$*, the operator* T_i *is maximal and that the map* $\mathrm{prox}_{\rho T_i} : \mathcal{H}_i \to \mathcal{H}_i$ *can be computed to within the error tolerance specified below in Assumption 3 (however, these operators are not precluded from also being Lipschitz continuous).*

5. *Each* $G_i : \mathcal{H}_0 \to \mathcal{H}_i$ *for* $i = 1, \ldots, n-1$ *is linear and bounded.*

6. *The solution set* $\mathcal{S}$ *defined in (15) is nonempty.*

Lemma 3. *Suppose Assumption 1 holds. The set* $\mathcal{S}$ *defined in* (15) *is closed and convex.*

Proof. We first remark that for $i \in \mathcal{I}_{\mathrm{F}}$ the operators T_i are maximal by [2, Proposition 20.27], so $T_1, \ldots, T_n$ are all maximal monotone. The claimed result is then a special case of [5, Proposition 2.8(i)] with the following change of notatation:

Notation here		**Notation in [5]**
T_n	$\longrightarrow$	A (a maximal monotone operator)
$(x_1, \ldots, x_{n-1}) \mapsto T_1 x_1 \times \cdots \times T_{n-1} x_{n-1}$	$\longrightarrow$	B (a maximal monotone operator)
$z \mapsto (G_1 z, \ldots, G_{n-1} z)$	$\longrightarrow$	L (a bounded linear operator).

□

Algorithm 2: Asynchronous algorithm for solving (13).

Input: $(z^1, \mathbf{w}^1) \in \boldsymbol{\mathcal{H}}$, $(x_i^0, y_i^0) \in \mathcal{H}_i^2$ for $i = 1, \ldots, n$, $0 < \underline{\beta} \leq \overline{\beta} < 2$, $\gamma > 0$.

for $k = 1, 2, \ldots$ **do**
- **for** $i = 1, 2, \ldots, n$ **do**
  - **if** $i \in I_k$ **then**
    - **if** $i \in \mathcal{I}_{\mathrm{B}}$ **then**
      - $a = G_i z^{d(i,k)} + \rho_i^{d(i,k)} w_i^{d(i,k)} + e_i^k$
      - $x_i^k = \operatorname{prox}_{\rho_i^{d(i,k)} T_i}(a)$
      - $y_i^k = (\rho_i^{d(i,k)})^{-1}\left(a - x_i^k\right)$
    - **else**
      - $x_i^k = G_i z^{d(i,k)} - \rho_i^{d(i,k)}(T_i G_i z^{d(i,k)} - w_i^{d(i,k)})$,
      - $y_i^k = T_i x_i^k$.
  - **else**
    - $(x_i^k, y_i^k) = (x_i^{k-1}, y_i^{k-1})$
- $u_i^k = x_i^k - G_i x_n^k, \quad i = 1, \ldots, n-1,$
- $v^k = \sum_{i=1}^{n-1} G_i^* y_i^k + y_n^k$
- $\pi_k = \|u^k\|^2 + \gamma^{-1}\|v^k\|^2$
- **if** $\pi_k > 0$ **then**
  - Choose some $\beta_k \in [\underline{\beta}, \overline{\beta}]$
  - $\varphi_k(p_k) = \langle z^k, v^k\rangle + \sum_{i=1}^{n-1}\langle w_i^k, u_i^k\rangle - \sum_{i=1}^{n}\langle x_i^k, y_i^k\rangle$
  - $\alpha_k = \frac{\beta_k}{\pi_k}\max\{0, \varphi_k(p_k)\}$
- **else**
  - **if** $\cup_{j=1}^k I_j = \{1, \ldots, n\}$ **then**
    - **return** $z^{k+1} \leftarrow x_n^k, w_1^{k+1} \leftarrow y_1^k, \ldots, w_{n-1}^{k+1} \leftarrow y_{n-1}^k$
  - **else**
    - $\alpha_k = 0$
- $z^{k+1} = z^k - \gamma^{-1}\alpha_k v^k$
- $w_i^{k+1} = w_i^k - \alpha_k u_i^k, \quad i = 1, \ldots, n-1,$
- $w_n^{k+1} = -\sum_{i=1}^{n-1} G_i^* w_i^{k+1}$

3 Our Algorithm and Convergence

3.1 Algorithm Definition

Algorithm 2 is our asynchronous block-iterative projective splitting algorithm with forward steps for solving (13). It is essentially a special case of the weakly convergent Algorithm of [8], except that we use the new forward step procedure to deal with the Lipschitz continuous operators T_i for $i \in \mathcal{I}_{\mathrm{F}}$, instead of exclusively using proximal steps. For our separating hyperplane we use a special case of the formulation of [8], which is slightly different from the one used in [16]. Our method can be reformulated to use the same hyperplane as [16]; however, this requires that it be computationally feasible to project on the subspace given by the equation $\sum_{i=1}^n G_i^* w_i = 0$.

The algorithm has the following parameters:

- For each iteration $k \geq 1$, a subset $I_k \subseteq \{1, \ldots, n\}$.
- For each $k \geq 1$ and $i = 1, \ldots, n$, a positive scalar stepsize ρ_i^k.
- For each iteration $k \geq 1$ and $i = 1, \ldots, n$, a delayed iteration index $d(i,k) \in \{1, \ldots, k\}$ which allows the subproblem calculations to use outdated information.
- For each iteration $k \geq 1$, an overrelaxation parameter $\beta_k \in [\underline{\beta}, \overline{\beta}]$ for some constants $0 < \underline{\beta} \leq \overline{\beta} < 2$.
- A scalar $\gamma > 0$ which controls the relative emphasis on the primal and dual variables in the projection update in lines 25-26; see (16) in Section 3.2 for more details.
- Sequences of errors $\{e_i^k\}_{k \geq 1}$ for $i \in \mathcal{I}_B$, allowing us to model inexact computation of the proximal steps.

There are many ways in which Algorithm 2 could be implemented in various parallel computing environments; a specific suggestion for asynchronous implementation of a closely related class of algorithms is developed in [16, Section 3]. One simple option is a centralized or "master-slave" implementation in which lines 5-7 and 9-10 are implemented on a collection of "worker" processors, while the remainder of the algorithm, most notably the coordination process embodied by lines 13-27, is executed by a single coordinating processor. However, such a simple implementation risks the coordinating processor becoming a serial bottleneck as the number of worker processors grows or the memory required to store the vectors (x_i^k, y_i^k, w_i^k) for $i = 1, \ldots, n$, becomes large, since the amount of work required to execute lines 13-27 is proportional to the total number of elements in (x_i^k, y_i^k, w_i^k). Fortunately, all but a constant amount of the work in the coordination calculations in lines 13-27 involves only sums, inner products, and matrix multiplies by G_i and G_i^*. Summation and hence inner product operations can be efficiently distributed over multiple processors. Therefore, with some care exercised as to where one performs the matrix multiplications in cases in which the G_i are nontrivial, the coordination calculations may be distributed over multiple processors so that the coordination process need not constitute a serial bottleneck.

In the form directly presented in Algorithm 2, the delay indices $d(i,k)$ may seem unmotivated; it might seem best to always select $d(i,k) = k$. However, these indices can play a critical role in modeling asynchronous parallel implementation. In the simple "master-slave" scheme described above, for example, the "master" might dispatch subproblems to worker processors, but not receive the results back immediately. In the meantime, other workers may report back results, which the master could incorporate into its projection calculations. In this context, k counts the number of projection operations performed at the master, and I_k is the set of subproblems whose solutions reached the master between iterations $k-1$ and k. For each $i \in I_k$, $d(i,k)$ is the index of the iteration completed just before subproblem was last dispatched for solution. In more sophisticated parallel implementation, I_k and $d(i,k)$ would have similar interpretations.

We now start our analysis of the weak convergence of the iterates of Algorithm 2 to a solution of problem (13). While the overall proof strategy is similar to [16], considerable innovation is required to incorporate the forward steps.

3.2 The Hyperplane

In this section, we define the affine function our algorithm uses to construct a separating hyperplane. Let $p = (z, \mathbf{w}) = (z, w_1, \ldots, w_{n-1})$ be a generic point in $\boldsymbol{\mathcal{H}}$, the collective primal-dual space. For $\boldsymbol{\mathcal{H}}$, we adopt the following norm and inner product for some $\gamma > 0$:

$$\|(z,\mathbf{w})\|_\gamma^2 = \gamma\|z\|^2 + \sum_{i=1}^{n-1}\|w_i\|^2 \qquad \big\langle (z^1,\mathbf{w}^1),(z^2,\mathbf{w}^2)\big\rangle_\gamma = \gamma\langle z^1, z^2\rangle + \sum_{i=1}^{n-1}\langle w_i^1, w_i^2\rangle. \tag{16}$$

Define the following function generalizing (4) at each iteration $k \geq 1$:

$$\varphi_k(p) \;=\; \sum_{i=1}^{n-1}\big\langle G_i z - x_i^k, y_i^k - w_i\big\rangle + \left\langle z - x_n^k, y_n^k + \sum_{i=1}^{n-1} G_i^* w_i\right\rangle, \tag{17}$$

where the (x_i^k, y_i^k) are chosen so that $y_i^k \in T_i x_i^k$ for $i = 1, \ldots, n$ (recall that each inner product is for the corresponding Hilbert space $\mathcal{H}_i$). This function is a special case of the separator function used in [8]. The following lemma proves some basic properties of φ_k; similar results are in [1, 8, 16] in the case $\gamma = 1$.

Lemma 4. *Let φ_k be defined as in (17). Then:*

1. *φ_k is affine on $\boldsymbol{\mathcal{H}}$.*
2. *With respect to inner product $\langle\cdot,\cdot\rangle_\gamma$ on $\boldsymbol{\mathcal{H}}$, the gradient of φ_k is*

$$\nabla\varphi_k = \left(\frac{1}{\gamma}\left(\sum_{i=1}^{n-1} G_i^* y_i^k + y_n^k\right),\; x_1^k - G_1 x_n^k, x_2^k - G_2 x_n^k, \ldots, x_{n-1}^k - G_{n-1}x_n^k\right).$$

3. *Suppose Assumption 1 holds and that $y_i^k \in T_i x_i^k$ for $i = 1, \ldots, n$. Then $\varphi_k(p) \leq 0$ for all $p \in \mathcal{S}$ defined in (15).*
4. *If Assumption 1 holds, $y_i^k \in T_i x_i^k$ for $i = 1, \ldots, n$, and $\nabla\varphi_k = 0$, then $(x_n^k, y_1^k, \ldots, y_{n-1}^k) \in \mathcal{S}$.*

Proof. To see that φ_k is affine, rewrite (17) as

$$\varphi_k(z,\mathbf{w}) = \sum_{i=1}^{n-1}\langle G_i z, y_i^k - w_i\rangle - \sum_{i=1}^{n-1}\langle x_i^k, y_i^k - w_i\rangle + \left\langle z, y_n^k + \sum_{i=1}^{n-1} G_i^* w_i\right\rangle$$
$$- \left\langle x_n^k, y_n^k + \sum_{i=1}^{n-1} G_i^* w_i\right\rangle$$

$$
\begin{aligned}
&= \sum_{i=1}^{n-1}\langle z, G_i^*(y_i^k - w_i)\rangle + \sum_{i=1}^{n-1}\langle w_i, x_i^k\rangle - \sum_{i=1}^{n}\langle x_i^k, y_i^k\rangle + \left\langle z, y_n^k + \sum_{i=1}^{n-1} G_i^* w_i \right\rangle \\
&\qquad - \sum_{i=1}^{n-1}\langle w_i, G_i x_n^k\rangle \\
&= \left\langle z, \sum_{i=1}^{n-1} G_i^* y_i^k + y_n^k \right\rangle + \sum_{i=1}^{n-1}\langle w_i, x_i^k - G_i x_n^k\rangle - \sum_{i=1}^{n}\langle x_i^k, y_i^k\rangle. \qquad (18)
\end{aligned}
$$

It is now clear that φ_k is an affine function of $p = (z, \mathbf{w})$. Next, fix an arbitrary $\tilde{p} \in \boldsymbol{\mathcal{H}}$. Using the fact that φ_k is affine, we may write

$$
\begin{aligned}
\varphi_k(p) &= \langle p - \tilde{p}, \nabla\varphi_k\rangle_\gamma + \varphi_k(\tilde{p}) = \langle p, \nabla\varphi_k\rangle_\gamma + \varphi_k(\tilde{p}) - \langle \tilde{p}, \nabla\varphi_k\rangle_\gamma \\
&= \gamma\langle z, \nabla_z\varphi_k\rangle + \sum_{i=1}^{n-1}\langle w_i, \nabla_{w_i}\varphi_k\rangle + \varphi_k(\tilde{p}) - \langle \tilde{p}, \nabla\varphi_k\rangle_\gamma
\end{aligned}
$$

Equating terms between this expression and (18) yields the claimed expression for the gradient.

Next, suppose Assumption 1 holds and $y_i^k \in T_i x_i^k$ for $i = 1, \ldots, n$. To prove the third claim, we need to consider $(z, \mathbf{w}) \in \mathcal{S}$ and establish that $\varphi_i(z, \mathbf{w}) \leq 0$. We do so by showing that all n terms in (17) are nonpositive: first, for each $i = 1, \ldots, n-1$, we have $\langle G_i z - x_i^k, y_i^k - w_i\rangle \leq 0$ since T_i is monotone, $w_i \in T_i(G_i z)$, and $y_i^k \in T_i x_i^k$. The nonpositivity of the final term is established similarly by noting that $y_n^k \in T_n x_n^k$, $-\sum_{i=1}^{n-1} G_i^* w_i \in T_n z$, and that T_n is monotone.

Finally, suppose $\nabla\varphi_k = 0$ for some $k \geq 1$. Then

$$
y_n^k = -\sum_{i=1}^{n-1} G_i^* y_i^k, \qquad (19)
$$

$$
x_i^k - G_i x_n^k = 0, \qquad i = 1, \ldots, n-1. \qquad (20)
$$

Now (20) implies

$$
y_i^k \in T_i(G_i x_n^k), \quad i = 1, \ldots, n-1. \qquad (21)
$$

Since we also have $y_n^k \in T_n(x_n^k)$, (19) and (21) imply that $(x_n^k, y_1^k, \ldots, y_{n-1}^k) \in \mathcal{S}$. □

3.3 Asynchrony and Block-Iterative Properties

We now state our assumptions about the block-iterative and asynchronous nature of Algorithm 2. These same assumptions were used in [8, 16].

Assumption 2. *For Algorithm 2, assume:*

1. *For some fixed integer $M \geq 1$, each index i in $1, \ldots, n$ is in I_k at least once every M iterations, that is,*

$$
(\forall\, j \geq 1) \qquad \bigcup_{k=j}^{j+M-1} I_k = \{1, \ldots, n\}.
$$

2. *For some fixed integer $D \geq 0$, we have $k - d(i,k) \leq D$ for all i, k with $i \in I_k$. That is, there is a constant bound on the extent to which the information $z^{d(i,k)}$ and $w_i^{d(i,k)}$ used in lines 5 and 9 is out of date.*

We now deal with the situation where Algorithm 2 terminates at line 22.

Lemma 5. *For Algorithm 2:*

1. *Suppose Assumption 1 holds. If the algorithm terminates via line 22, then $(z^{k+1}, \mathbf{w}^{k+1}) \in \mathcal{S}$. Furthermore $x_i^k = G_i z^{k+1}$ and $y_i^k = w_i^{k+1}$ for $i = 1, \ldots, n-1$, and $x_n^k = z^{k+1}$ and $y_n^k = -\sum_{i=1}^{n-1} G_i^* w_i^{k+1}$.*

2. *Additionally, suppose Assumption 2(1) holds. Then if $\pi_k = 0$ at some iteration $k \geq M$, the algorithm terminates via line 22.*

Proof. The condition $\cup_{j=1}^k I_j = \{1, \ldots, n\}$ implies that $y_i^k \in T_i x_i^k$ for $i = 1, \ldots, n$. Let φ_k be the affine function defined in (17). Simple algebra verifies that for u^k and v^k defined on lines 13 and 14, $u_i^k = \nabla_{w_i} \varphi_k$ for $i = 1, \ldots, n-1$, $v^k = \gamma \nabla_z \varphi_k$, and $\pi_k = \|\nabla \varphi_k\|_\gamma^2$. If for any such k, π_k equals 0, then this implies $\nabla \varphi_k = 0$. Then we can invoke Lemma 4(4) to conclude that $(x_n^k, y_1^k, \ldots, y_{n-1}^k) \in \mathcal{S}$. Thus, the algorithm terminates with $(z^{k+1}, w_1^{k+1}, \ldots, w_{n-1}^{k+1}) = (x_n^k, y_1^k, \ldots, y_{n-1}^k) \in \mathcal{S}$. Furthermore, when $\nabla \varphi_k = 0$, Lemma 4(2) leads to

$$\sum_{i=1}^{n-1} G_i^* y_i^k + y_n^k = 0 \qquad\qquad x_i^k - G_i x_n^k = 0 \quad i = 1, \ldots, n-1.$$

We immediately conclude that $y_n^k = -\sum_{i=1}^{n-1} G_i^* y_i^k = -\sum_{i=1}^{n-1} G_i^* w_i^{k+1}$ and, for $i = 1, \ldots, n-1$, that $x_i^k = G_i x_n^k = G_i z^k$.

Finally, note that for any $k \geq M$, $\cup_{j=1}^k I_j = \{1, \ldots, n\}$ by Assumption 2(1). Therefore whenever $\pi_k = 0$ for $k \geq M$, the algorithm terminates via line 22. □

Lemma 5 asserts that if the algorithm terminates finitely, then the final iterate is a solution. For the rest of the analysis, we therefore assume that $\pi_k \neq 0$ for all $k \geq M$. Under Assumption 2, Algorithm 2 is a projection algorithm:

Lemma 6. *Suppose that Assumption 1 holds for problem* (13) *and Assumption 2(1) holds for Algorithm 2. Then, for all $k \geq M$ such that π_k defined on Line 15 is nonzero, Algorithm 2 is an instance of Algorithm 1 with $\boldsymbol{\mathcal{H}} = \mathcal{H}_0 \times \cdots \times \mathcal{H}_{n-1}$ and the inner product in* (16)*, $\mathcal{S}$ as defined in* (15)*, and φ_k as defined in* (17)*. All the statements of Lemma 2 hold for the sequence $\{p^k\} = \{(z^k, w_1^k, \ldots, w_{n-1}^k)\}$ generated by Algorithm 2.*

Proof. For $k \geq M$ in Algorithm 2, by Assumption 2(1) all (x_i^k, y_i^k) have been updated at least once using either lines 6–7 or lines 9–10, and thus $y_i^k \in T_i x_i^k$ for $i = 1, \ldots, n$. Therefore, Lemma 4 implies that φ_k defined in (17) forms a separating hyperplane for the set $\mathcal{S}$, that is, for any $(z, \mathbf{w}) \in \mathcal{S}$, we must have $\varphi_k(z, \mathbf{w}) \leq 0$.

Next we verify that lines 13-27 of Algorithm 2 are an instantiation of line 4 of Algorithm 1 using φ_k as defined in (17) and the norm defined in (16). As already shown, $\pi_k = \|\nabla \varphi_k\|_\gamma^2$. Considering the decomposition of φ_k in (18), it can then be seen that lines 15-26 of Algorithm 2 implement the projection on line 4 of Algorithm 1.

To conclude the proof, we note that Lemma 3 asserts that $\mathcal{S}$ is closed and convex, so all the results of Lemma 2 apply. □

We need a few more definitions to describe the asynchronous and block-iterative properties of Algorithm 2. These use the same notation as [16]. For all i and k, define

$$S(i,k) = \{j \in \mathbb{N} : j \leq k, i \in I_j\} \qquad s(i,k) = \begin{cases} \max S(i,k), & \text{when } S(i,k) \neq \emptyset \\ 0, & \text{otherwise.} \end{cases}$$

In words, $s(i,k)$ is the most recent iteration up to and including k in which the index-i information in the separator was updated, or 0 if index-i information has never been processed. Assumption 2 ensures that $0 \leq k - s(i,k) < M$.

Next, for all $i = 1,\ldots,n$ and iterations k, define $l(i,k) = d(i, s(i,k))$. Thus, $l(i,k)$ is the iteration in which the algorithm generated the information $z^{l(i,k)}$ and $w_i^{l(i,k)}$ used to compute the current point (x_i^k, y_i^k). Regarding initialization, we set $d(i,0) = 0$; note that the initial points (x_i^0, y_i^0) are arbitrary and were not computed from any instance of (z^k, w_i^k).

We formalize the use of $l(i,k)$ in the following Lemma.

Lemma 7. *Suppose Assumption 2(1) holds. For all iterations $k \geq M$, if Algorithm 2 has not already terminated, then the updates can be written as*

$$(\forall i \in \mathcal{I}_{\rm B}) \qquad x_i^k + \rho_i^{l(i,k)} y_i^k = G_i z^{l(i,k)} + \rho_i^{l(i,k)} w_i^{l(i,k)} + e_i^{s(i,k)}, \qquad y_i^k \in T_i x_i^k, \tag{22}$$

$$(\forall i \in \mathcal{I}_{\rm F}) \qquad x_i^k = G_i z^{l(i,k)} - \rho_i^{l(i,k)}(T_i G_i z^{l(i,k)} - w_i^{l(i,k)}), \qquad y_i^k = T_i x_i^k. \tag{23}$$

Proof. The proof follows from the definition of $l(i,k)$ and $s(i,k)$. After M iterations, all operators must have been in I_k at least once. Thus, after M iterations, every operator has been updated at least once using either the proximal step on lines 5-7 or the forward steps on lines 9-10 of Algorithm 2. Recall the variables defined to ease mathematical presentation, namely $G_n = I$ and w_n^k defined in (14) and line 27. □

We now derive some important properties of $l(i,k)$. The following result was proved in Lemma 6 of [16] but since it is short we include the proof here.

Lemma 8. *Under Assumption 2, $k - l(i,k) < M + D$ for all $i = 1,\ldots,n$ and iterations k.*

Proof. From the definition, we know that $0 \leq k - s(i,k) < M$. Part 2 of Assumption 2 ensures that $s(i,k) - l(i,k) = s(i,k) - d(i, s(i,k)) \leq D$. Adding these two inequalities yields the desired result. □

Lemma 9. *Suppose Assumptions 1 and 2 hold and $\pi_k > 0$ for all $k \geq M$. Then $w_i^{l(i,k)} - w_i^k \to 0$ for all $i = 1,\ldots,n$ and $z^{l(i,k)} - z^k \to 0$.*

Proof. For z^k and w_i^k for $i = 1,\ldots,n-1$, the proof is identical to the proof of [16, Lemma 9]. For $\{w_n^k\}$, we have from line 27 of the algorithm that

$$\begin{aligned}
\|w_n^{l(n,k)} - w_n^k\| &= \left\| \sum_{i=1}^{n-1} G_i^* \left(w_i^k - w_i^{l(n,k)} \right) \right\| \\
&\leq \sum_{i=1}^{n-1} \|G_i^*\| \left\| w_i^k - w_i^{l(n,k)} \right\|.
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=1}^{n-1} \|G_i^*\| \left\| \sum_{j=1}^{k-l(n,k)} \left(w_i^{k-j+1} - w_i^{k-j} \right) \right\| \\
&\le \sum_{i=1}^{n-1} \|G_i^*\| \sum_{j=1}^{k-l(n,k)} \left\| w_i^{k-j+1} - w_i^{k-j} \right\| \\
&\le \sum_{i=1}^{n-1} \|G_i^*\| \sum_{j=1}^{M+D} \left\| w_i^{k-j+1} - w_i^{k-j} \right\|,
\end{aligned}
$$

where in the final line we used Lemma 8. Since the operators G_i are bounded and Lemma 2(2) implies that $w_i^{k+1} - w_i^k \to 0$ for all $i = 1, \ldots, n-1$, we conclude that $w_n^{l(n,k)} - w_n^k \to 0$. □

Next, we define

$$
(\forall i = 1, \ldots, n) \quad \phi_{ik} \triangleq \langle G_i z^k - x_i^k, y_i^k - w_i^k \rangle \qquad \phi_k \triangleq \textstyle\sum_{i=1}^n \phi_{ik} \tag{24}
$$

$$
(\forall i = 1, \ldots, n) \quad \psi_{ik} \triangleq \langle G_i z^{l(i,k)} - x_i^k, y_i^k - w_{l(i,k)}^k \rangle \qquad \psi_k \triangleq \textstyle\sum_{i=1}^n \psi_{ik}. \tag{25}
$$

Note that (24) simply expands the definition of the affine function in (17) and we may write $\varphi_k(p^k) = \phi_k$.

Lemma 10. *Suppose assumptions 1 and 2 hold and $\pi_k > 0$ for all $k \ge M$. Then $\phi_{ik} - \psi_{ik} \to 0$ for all $i = 1, \ldots, n$.*

Proof. In view of Lemma 9, we may follow the same argument as [16, Lemma 12]. □

3.4 Conditions on the Errors and the Stepsizes

We now state our assumptions regarding the errors and stepsizes in Algorithm 2. The assumptions for the errors in evaluating the proximal operators are taken exactly from [16] and we include them here for completeness.

Assumption 3. *The error sequences $\{\|e_i^k\|\}$ are bounded for all $i \in \mathcal{I}_{\mathrm{B}}$. For some σ with $0 \le \sigma < 1$ the following hold for all $k \ge 1$:*

$$
(\forall i \in \mathcal{I}_{\mathrm{B}}) \qquad \langle G_i z^{l(i,k)} - x_i^k, e_i^{s(i,k)} \rangle \ge -\sigma \|G_i z^{l(i,k)} - x_i^k\|^2 \tag{26}
$$

$$
(\forall i \in \mathcal{I}_{\mathrm{B}}) \qquad \langle e_i^{s(i,k)}, y_i^k - w_i^{l(i,k)} \rangle \le \rho_i^{l(i,k)} \sigma \|y_i^k - w_i^{l(i,k)}\|^2. \tag{27}
$$

Assumption 4. *The stepsize conditions for weak convergence of Algorithm 2 are:*

$$
\underline{\rho} \triangleq \min_{i=1,\ldots,n} \left\{ \inf_{k\ge1} \rho_i^k \right\} > 0 \qquad \overline{\rho} \triangleq \max_{i \in \mathcal{I}_{\mathrm{B}}} \left\{ \sup_{k\ge1} \rho_i^k \right\} < \infty
$$

$$
(\forall\, i \in \mathcal{I}_{\mathrm{F}}) \quad \overline{\rho}_i \triangleq \limsup_{k\to\infty} \rho_i^k < \frac{1}{L_i}. \tag{28}
$$

Note that (28) allows the stepsize to be larger than the right hand side for a finite number of iterations.

3.5 Three Technical Lemmas

We now prove three technical lemmas which pave the way to establishing weak convergence of Algorithm 2 to a solution of (13). The first lemma upper bounds the norm of the gradient of φ_k at each iteration.

Lemma 11. *Suppose assumptions 1-4 hold. Suppose that $\pi_k > 0$ for all $k \geq M$. Recall the affine function φ_k defined in (17). There exists $\xi_1 \geq 0$ such that $\|\nabla\varphi_k\|_\gamma^2 \leq \xi_1$ for all $k \geq 1$.*

Proof. For $k < M$ the gradient can be trivially bounded by $\max_{1\leq k<M}\|\nabla\varphi_k\|_\gamma^2$. Now fix any $k \geq M$. Using Lemma 4,

$$\|\nabla\varphi_k\|_\gamma^2 = \gamma^{-1}\left\|\sum_{i=1}^{n-1} G_i^* y_i^k + y_n^k\right\|^2 + \sum_{i=1}^{n-1}\|x_i^k - G_i x_n^k\|^2. \tag{29}$$

Using Lemma 1, we begin by writing the second term on the right of (29) as

$$\sum_{i=1}^{n-1}\|x_i^k - G_i x_n^k\|^2 \leq 2\sum_{i=1}^{n-1}\left(\|x_i^k\|^2 + \|G_i\|^2\|x_n^k\|^2\right) \leq 2\sum_{i=1}^{n-1}\|x_i^k\|^2 + 2(n-1)\max_i\left\{\|G_i\|^2\right\}\|x_n^k\|^2.$$

The linear operators G_i are bounded by Assumption 1. We now check the boundedness of sequences $\{x_i^k\}$, $i = 1, \ldots, n$. For $i \in \mathcal{I}_\text{B}$, the boundedness of $\{x_i^k\}$ follows from exactly the same argument as in [16, Lemma 10]. Now taking any $i \in \mathcal{I}_\text{F}$, we use the triangle inequality and Lemma 7 to obtain

$$\begin{aligned}\|x_i^k\| &\leq \|G_i z^{l(i,k)} - \rho_i^{l(i,k)} T_i G_i z^{l(i,k)}\| + \rho_i^{l(i,k)}\|w_i^{l(i,k)}\| \\ &\leq \|G_i\|\|z^{l(i,k)}\| + \rho_i^{l(i,k)}\|T_i G_i z^{l(i,k)}\| + \rho_i^{l(i,k)}\|w_i^{l(i,k)}\|.\end{aligned}$$

Now the sequences $\{\|z^k\|\}$ and $\{\|w_i^k\|\}$ are bounded by Lemma 2, implying the boundedness of $\{\|z^{l(i,k)}\|\}$ and $\{\|w_i^{l(i,k)}\|\}$. Since $\{z^{l(i,k)}\}$ is bounded, G_i is bounded, and T_i is Lipschitz continuous, $\{T_i G_i z^{l(i,k)}\}$ is bounded. Finally, the stepsizes ρ_i^k are bounded by Assumption 4. Therefore, $\{x_i^k\}$ is bounded for $i \in \mathcal{I}_\text{F}$, and we may conclude that the second term in (29) is bounded.

We next consider the first term in (29). Rearranging the update equations for Algorithm 2 as given in Lemma 7, we may write

$$y_i^k = \left(\rho_i^{l(i,k)}\right)^{-1}\left(G_i z^{l(i,k)} - x_i^k + \rho_i^{l(i,k)} w_i^{l(i,k)} + e_i^{s(i,k)}\right), \qquad i \in \mathcal{I}_\text{B} \tag{30}$$

$$T_i G_i z^{l(i,k)} = \left(\rho_i^{l(i,k)}\right)^{-1}\left(G_i z^{l(i,k)} - x_i^k + \rho_i^{l(i,k)} w_i^{l(i,k)}\right), \qquad i \in \mathcal{I}_\text{F}. \tag{31}$$

Using $G_n = I$, the squared norm in the first term of (29) may be written as

$$\begin{aligned}\left\|\sum_{i=1}^{n} G_i^* y_i^k\right\|^2 &= \left\|\sum_{i\in\mathcal{I}_\text{B}} G_i^* y_i^k + \sum_{i\in\mathcal{I}_\text{F}} G_i^*\left(T_i G_i z^{l(i,k)} + y_i^k - T_i G_i z^{l(i,k)}\right)\right\|^2 \\ &\overset{(a)}{\leq} 2\left\|\sum_{i\in\mathcal{I}_\text{B}} G_i^* y_i^k + \sum_{i\in\mathcal{I}_\text{F}} G_i^* T_i G_i z^{l(i,k)}\right\|^2 + 2\left\|\sum_{i\in\mathcal{I}_\text{F}} G_i^*\left(y_i^k - T_i G_i z^{l(i,k)}\right)\right\|^2\end{aligned}$$

$$\overset{(b)}{\leq} 4\left\|\sum_{i=1}^{n}\left(\rho_i^{l(i,k)}\right)^{-1}G_i^*\left(G_iz^{l(i,k)}-x_i^k+\rho_i^{l(i,k)}w_i^{l(i,k)}\right)\right\|^2$$

$$+\ 2|\mathcal{I}_{\mathrm{F}}|\sum_{i\in\mathcal{I}_{\mathrm{F}}}\|G_i\|^2\left\|T_ix_i^k-T_iG_iz^{l(i,k)}\right\|^2+4\left\|\sum_{i\in\mathcal{I}_{\mathrm{B}}}\left(\rho_i^{l(i,k)}\right)^{-1}G_i^*e_i^{s(i,k)}\right\|^2 \quad (32)$$

$$\overset{(c)}{\leq} 4n\underline{\rho}^{-2}\max_i\left\{\|G_i\|\right\}^2\left(\sum_{i=1}^{n}\left\|G_iz^{l(i,k)}-x_i^k+\rho_i^{l(i,k)}w_i^{l(i,k)}\right\|^2+\sum_{i\in\mathcal{I}_{\mathrm{B}}}\|e_i^{s(i,k)}\|^2\right)$$

$$+\ 4|\mathcal{I}_{\mathrm{F}}|\sum_{i\in\mathcal{I}_{\mathrm{F}}}\|G_i\|^2\left(L_i^2\|x_i^k-G_iz^{l(i,k)}\|^2\right) \quad (33)$$

In the above, (a) uses Lemma 1, while (b) is obtained by substituting (30)-(31) into the first squared norm and using $y_i^k = T_ix_i^k$ for $i \in \mathcal{I}_{\mathrm{F}}$ in the second, and then using Lemma 1 on both terms. Finally, (c) uses Lemma 1, the Lipschitz continuity of T_i, and Assumption 4. For each $i = 1, \ldots, n$, we have that G_i is a bounded operator, the sequences $\{z^{l(i,k)}\}$, $\{x_i^k\}$, and $\{w_i^{l(i,k)}\}$ are already known to be bounded, $\{\rho_i^{l(i,k)}\}$ is bounded by Assumption 4, and for $i \in \mathcal{I}_{\mathrm{B}}$, $\{e_i^{s(i,k)}\}$ is bounded by Asssumption 3. We conclude that the right hand side of (33) is bounded. Therefore, the first term in (29) is bounded and the sequence $\{\nabla\varphi_k\}$ must be bounded. □

The second technical lemma establishes a lower bound for the affine function φ_k evaluated at the current point, proving that it is nonegative and "large enough" to guarantee weak convergence of the method. The lower bound applies to the quantity ψ_k defined in (25): this quantity is easier to analyze than ϕ_k and Lemma 10 asserts that the difference between the two converges to zero.

Lemma 12. *Suppose that assumptions 1-4 hold. Suppose $\pi_k > 0$ for all $k \geq M$. Then there exists $\xi_2 > 0$ such that*

$$\limsup_{k\to\infty}\psi_k \geq \xi_2 \limsup_{k\to\infty}\sum_{i=1}^{n}\|G_iz^{l(i,k)}-x_i^k\|^2.$$

Proof. For $k \geq M$, we have

$$\psi_k = \sum_{i=1}^{n}\left\langle G_iz^{l(i,k)}-x_i^k, y_i^k-w_i^{l(i,k)}\right\rangle$$

$$\overset{(a)}{=} \sum_{i\in\mathcal{I}_{\mathrm{B}}}\left\langle G_iz^{l(i,k)}-x_i^k, \left(\rho_i^{l(i,k)}\right)^{-1}\left(G_iz^{l(i,k)}-x_i^k+e_i^{s(i,k)}\right)\right\rangle$$

$$+\sum_{i\in\mathcal{I}_{\mathrm{F}}}\left\langle G_iz^{l(i,k)}-x_i^k, T_iG_iz^{l(i,k)}-w_i^{l(i,k)}\right\rangle+\sum_{i\in\mathcal{I}_{\mathrm{F}}}\left\langle G_iz^{l(i,k)}-x_i^k, y_i^k-T_iG_iz^{l(i,k)}\right\rangle$$

$$\overset{(b)}{=} \sum_{i\in\mathcal{I}_{\mathrm{B}}}\left[\left(\rho_i^{l(i,k)}\right)^{-1}\|G_iz^{l(i,k)}-x_i^k\|^2+\left(\rho_i^{l(i,k)}\right)^{-1}\left\langle G_iz^{l(i,k)}-x_i^k, e_i^{s(i,k)}\right\rangle\right]$$

$$+\sum_{i\in\mathcal{I}_{\mathrm{F}}}\left\langle G_iz^{l(i,k)}-x_i^k, \left(\rho_i^{l(i,k)}\right)^{-1}\left(G_iz^{l(i,k)}-x_i^k\right)\right\rangle$$

$$- \sum_{i\in\mathcal{I}_\mathrm{F}} \left\langle G_i z^{l(i,k)} - x_i^k, T_i G_i z^{l(i,k)} - T_i x_i^k \right\rangle$$

$$\overset{\text{(c)}}{\geq} (1-\sigma)\sum_{i\in\mathcal{I}_\mathrm{B}} \left(\rho_i^{l(i,k)}\right)^{-1}\|G_i z^{l(i,k)} - x_i^k\|^2 + \sum_{i\in\mathcal{I}_\mathrm{F}} \left(\left(\rho_i^{l(i,k)}\right)^{-1} - L_i\right)\|G_i z^{l(i,k)} - x_i^k\|^2. \quad (34)$$

In the above derivation, (a) follows by substitution of (22) into the $\mathcal{I}_\mathrm{B}$ terms and algebraic manipulation of the $\mathcal{I}_\mathrm{F}$ terms. Next, (b) follows by algebraic manipulation of the $\mathcal{I}_\mathrm{B}$ terms and substitution of (23) into the $\mathcal{I}_\mathrm{F}$ terms. Finally, (c) is justified by using (26) in Assumption 3 and the Lipschitz continuity of T_i for $i \in \mathcal{I}_\mathrm{F}$.

Now consider any two sequences $\{a_k\} \subset \mathbb{R}, \{b_k\} \subset \mathbb{R}_+$. We note that

$$\limsup_{k\to\infty} a_k b_k \geq \limsup_{k\to\infty} \left\{ \left(\liminf_{k\to\infty} a_k\right) b_k \right\} = \left(\liminf_{k\to\infty} a_k\right)\left(\limsup_{k\to\infty} b_k\right).$$

Applying this fact to the expression in (34) yields the desired result with

$$\xi_2 = \min\left\{ (1-\sigma)\overline{\rho}^{-1},\ \min_{j\in\mathcal{I}_\mathrm{F}} \left\{\overline{\rho}_j^{-1} - L_j\right\} \right\},$$

and Assumption 4 guarantees that $\xi_2 > 0$. □

In the third technical lemma, we provide what is essentially a complementary lower bound for ψ_k:

Lemma 13. *Suppose assumptions 1-4 hold. Suppose $\pi_k > 0$ for all $k \geq M$. Then there exists $\xi_3 > 0$ such that*

$$\limsup_{k\to\infty} \left(\psi_k + \sum_{i\in\mathcal{I}_\mathrm{F}} L_i \|G_i z^{l(i,k)} - x_i^k\|^2 \right)$$
$$\geq \xi_3 \limsup_{k\to\infty} \left(\sum_{i\in\mathcal{I}_\mathrm{B}} \|y_i^k - w_i^{l(i,k)}\|^2 + \sum_{i\in\mathcal{I}_\mathrm{F}} \|T_i G_i z^{l(i,k)} - w_i^{l(i,k)}\|^2 \right). \quad (35)$$

Proof. For all $k \geq M$, we have

$$\psi_k = \sum_{i=1}^n \langle G_i z^{l(i,k)} - x_i^k, y_i^k - w_i^{l(i,k)} \rangle$$
$$\overset{\text{(a)}}{=} \sum_{i\in\mathcal{I}_\mathrm{B}} \langle \rho_i^{l(i,k)}(y_i^k - w_i^{l(i,k)}) - e_i^{s(i,k)}, y_i^k - w_i^{l(i,k)} \rangle$$
$$+ \sum_{i\in\mathcal{I}_\mathrm{F}} \langle G_i z^{l(i,k)} - x_i^k, T_i G_i z^{l(i,k)} - w_i^{l(i,k)} \rangle + \sum_{i\in\mathcal{I}_\mathrm{F}} \langle G_i z^{l(i,k)} - x_i^k, y_i^k - T_i G_i z^{l(i,k)} \rangle$$
$$\overset{\text{(b)}}{=} \sum_{i\in\mathcal{I}_\mathrm{B}} \left(\rho_i^{l(i,k)} \|y_i^k - w_i^{l(i,k)}\|^2 - \langle e_i^{s(i,k)}, y_i^k - w_i^{l(i,k)} \rangle \right)$$
$$+ \sum_{i\in\mathcal{I}_\mathrm{F}} \langle \rho_i^{l(i,k)}(T_i G_i z^{l(i,k)} - w_i^{l(i,k)}), T_i G_i z^{l(i,k)} - w_i^{l(i,k)} \rangle$$

$$
\begin{aligned}
&\quad - \sum_{i \in \mathcal{I}_\mathrm{F}} \langle x_i^k - G_i z^{l(i,k)}, T_i x_i^k - T_i G_i z^{l(i,k)} \rangle && (36)\\
&\overset{\text{(c)}}{\geq} (1-\sigma) \sum_{i \in \mathcal{I}_\mathrm{B}} \rho_i^{l(i,k)} \| y_i^k - w_i^{l(i,k)} \|^2 + \sum_{i \in \mathcal{I}_\mathrm{F}} \rho_i^{l(i,k)} \| T_i G_i z^{l(i,k)} - w_i^{l(i,k)} \|^2 \\
&\quad - \sum_{i \in \mathcal{I}_\mathrm{F}} L_i \| G_i z^{l(i,k)} - x_i^k \|^2. && (37)
\end{aligned}
$$

In the above derivation, (a) follows by substition of (22) into the $\mathcal{I}_\mathrm{B}$ terms and algebraic manipulation of the $\mathcal{I}_\mathrm{F}$ terms. Next (b) is obtained by algebraic simplification of the $\mathcal{I}_\mathrm{B}$ terms and substitution of (23) into the two groups of $\mathcal{I}_\mathrm{F}$ terms. Finally, (c) is obtained by substituting the error criterion (27) from Assumption 3 for the $\mathcal{I}_\mathrm{B}$ terms and using the Lipschitz continuity of T_i for the $\mathcal{I}_\mathrm{F}$ terms. Adding the last term in (37) to both sides yields

$$
\begin{aligned}
&\psi_k + \sum_{i \in \mathcal{I}_\mathrm{F}} L_i \| G_i z^{l(i,k)} - x_i^k \|^2 \\
&\qquad\qquad \geq (1-\sigma) \sum_{i \in \mathcal{I}_\mathrm{B}} \rho_i^{l(i,k)} \| y_i^k - w_i^{l(i,k)} \|^2 + \sum_{i \in \mathcal{I}_\mathrm{F}} \rho_i^{l(i,k)} \| T_i G_i z^{l(i,k)} - w_i^{l(i,k)} \|^2.
\end{aligned}
$$

Assumption 3 requires that $\sigma < 1$ and Assumption 4 requires that $\rho_i^k \geq \underline{\rho} > 0$ for all i, so taking limits in the above inequality implies that (35) holds with $\xi_3 = (1-\sigma)\underline{\rho}$. □

3.6 Weak Convergence of the Asynchronous Algorithm

We now state the main technical result of the paper, showing weak convergence of Algorithm 2 to a solution of (13).

Theorem 1. *Suppose Assumptions 1-4 hold. If Algorithm 2 terminates at line 22, then its final iterate is a member of the extended solution set $\mathcal{S}$. Otherwise, the sequence $\{(z^k, \mathbf{w}^k)\}$ generated by Algorithm 2 converges weakly to some point $(\bar{z}, \overline{\mathbf{w}})$ in the extended solution set $\mathcal{S}$ of* (13) *defined in* (15). *Furthermore, $x_i^k \rightharpoonup G_i \bar{z}$ and $y_i^k \rightharpoonup \overline{w}_i$ for all $i = 1, \ldots, n-1$, $x_n^k \rightharpoonup \bar{z}$, and $y_n^k \rightharpoonup -\sum_{i=1}^{n-1} G_i^* \overline{w}_i$.*

Proof. The assertion regarding termination at line 22 follows immediately from Lemma 5. For the remainder of the proof, we therefore consider only the case that the algorithm runs indefinitely and thus that $\pi_k > 0$ for all $k \geq M$.

The proof consists of three parts. The first part establishes that $G_i z^k - x_i^k \to 0$ for all i and the second part proves that $y_i^k - w_i^k \to 0$ for all i. Finally, the third part uses these results in conjunction with a result in [1] to show that any convergent subsequence of $\{p^k\} = \{(z^k, \mathbf{w}^k)\}$ generated by the algorithm must converge to a point in $\mathcal{S}$, after which we may simply invoke Lemma 2.

Part 1. Convergence of $G_i z^k - x_i^k \to 0$.

Lemma 6 and (24) imply that

$$
p^{k+1} = p^k - \frac{\beta_k \max\{\varphi_k(p^k), 0\}}{\|\nabla \varphi_k\|_\gamma^2} \nabla \varphi_k = p^k - \frac{\beta_k \max\{\phi_k, 0\}}{\|\nabla \varphi_k\|_\gamma^2} \nabla \varphi_k.
$$

Lemma 2(2) guarantees that $p^k - p^{k+1} \to 0$, so it follows that

$$0 = \lim_{k\to\infty} \|p^{k+1} - p^k\|_\gamma = \lim_{k\to\infty} \frac{\beta_k \max\{\phi_k, 0\}}{\|\nabla\varphi_k\|_\gamma} \geq \frac{\underline{\beta} \limsup_{k\to\infty} \max\{\phi_k, 0\}}{\sqrt{\xi_1}},$$

since $\|\nabla\varphi_k\|_\gamma \leq \sqrt{\xi_1} < \infty$ for all k by Lemma 11. Therefore, $\limsup_{k\to\infty} \phi_k \leq 0$. Since Lemma 10 implies that $\phi_k - \psi_k \to 0$, it follows that $\limsup_{k\to\infty} \psi_k \leq 0$. With (a) following from Lemma 12, we next obtain

$$0 \geq \limsup_{k\to\infty} \psi_k \overset{\text{(a)}}{\geq} \xi_2 \limsup_k \sum_{i=1}^n \|G_i z^{l(i,k)} - x_i^k\|^2 \geq \xi_2 \liminf_k \sum_{i=1}^n \|G_i z^{l(i,k)} - x_i^k\|^2 \geq 0.$$

Thus, $G_i z^{l(i,k)} - x_i^k \to 0$ for $i = 1, \ldots, n$. Since $z^k - z^{l(i,k)} \to 0$ and G_i is bounded, we obtain that $G_i z^k - x_i^k \to 0$ for $i = 1, \ldots, n$.

<u>Part 2. Convergence of $y_i^k - w_i^k \to 0$</u>

From $\limsup_{k\to\infty} \psi_k \leq 0$ and $G_i z^{l(i,k)} - x_i^k \to 0$, we obtain

$$\limsup_{k\to\infty} \left\{ \psi_k + \sum_{i\in\mathcal{I}_\mathrm{F}} L_i \|G_i z^{l(i,k)} - x_i^k\|^2 \right\} \leq 0. \tag{38}$$

Combining (38) with (35) in Lemma 13, we infer that

$$\begin{array}{llcl} (\forall\, i \in \mathcal{I}_\mathrm{B}) & y_i^k - w_i^{l(i,k)} \to 0 & \implies & y_i^k - w_i^k \to 0 \\ (\forall\, i \in \mathcal{I}_\mathrm{F}) & T_i G_i z^{l(i,k)} - w_i^{l(i,k)} \to 0 & \implies & T_i G_i z^k - w_i^k \to 0. \end{array} \tag{39}$$

where the implications follow from Lemma 9, the Lipschitz continuity of T_i for $i \in \mathcal{I}_\mathrm{F}$, and the continuity of the linear operators G_i. Finally, for each $i \in \mathcal{I}_\mathrm{F}$ and $k \geq M$, we further reason that

$$\begin{aligned} \|y_i^k - w_i^k\| &= \|T_i G_i z^k - w_i^k + y_i^k - T_i G_i z^k\|, \\ &\leq \|T_i G_i z^k - w_i^k\| + \|y_i^k - T_i G_i z^k\| \\ &\overset{\text{(a)}}{=} \|T_i G_i z^k - w_i^k\| + \|T_i x_i^k - T_i G_i z^k\| \\ &\overset{\text{(b)}}{\leq} \|T_i G_i z^k - w_i^k\| + L_i \|G_i z^k - x_i^k\| \overset{\text{(c)}}{\to} 0. \end{aligned}$$

Here, (a) uses (23) from Lemma 7, (b) uses the Lipschitz continuity of T_i, and (c) relies on (39) and part 1 of this proof.

<u>Part 3. Subsequential convergence</u>

Consider any increasing sequence of indices $\{q_k\}$ such that $(z^{q_k}, \mathbf{w}^{q_k})$ weakly converges to some point $(z^\infty, \mathbf{w}^\infty) \in \boldsymbol{\mathcal{H}}$. We claim that in any such situation, $(z^\infty, \mathbf{w}^\infty) \in \mathcal{S}$.

By part 1, $z^k - x_n^k \to 0$, so $x_n^{q_k} \rightharpoonup z^\infty$. For any $i = 1, \ldots, n$, part 2 asserts that $y_i^k - w_i^k \to 0$, so $y_i^{q_k} \rightharpoonup w_i^\infty$. Furthermore, part 2, (14), and the boundedness of G_i imply that

$$\sum_{i=1}^n G_i^* y_i^k = \sum_{i=1}^n G_i^* w_i^k + \sum_{i=1}^n G_i^* (y_i^k - w_i^k) \to 0.$$

Finally, part 1 and the boundedness of G_i yield

$$(\forall\, i = 1, \dots, n-1) \qquad x_i^k - G_i x_n^k = x_i^k - G_i z^k - G_i(x_n^k - z^k) \to 0.$$

Next we apply [1, Proposition 2.4] with the following change of notation:

Notation here		**Notation in [1]**
iteration counter k	$\longrightarrow$	iteration counter n
x_n^k	$\longrightarrow$	a_n
$(x_1^k, \dots, x_{n-1}^k)$	$\longrightarrow$	b_n
y_n^k	$\longrightarrow$	a_n^*
$(y_1^k, \dots, y_{n-1}^k)$	$\longrightarrow$	b_n^*
T_n	$\longrightarrow$	A (a maximal monotone operator)
$(x_1, \dots, x_{n-1}) \mapsto T_1 x_1 \times \cdots \times T_{n-1} x_{n-1}$	$\longrightarrow$	B (a maximal monotone operator)
$z \mapsto (G_1 z, \dots, G_{n-1} z)$	$\longrightarrow$	L (a bounded linear operator)
z^∞	$\longrightarrow$	$\bar{x}$
$\mathbf{w}^\infty$	$\longrightarrow$	$\bar{v}^*$.

We then conclude from [1, Proposition 2.4] that $(z^\infty, \mathbf{w}^\infty) \in \mathcal{S}$, and the claim is established.

Invoking Proposition 2(3), we immediately conclude that $\{(z^k, \mathbf{w}^k)\}$ converges weakly to some $(\bar{z}, \overline{\mathbf{w}}) \in \mathcal{S}$. For each $i = 1, \dots, n$, we finally observe that since $G_i z^k - x_i^k \to 0$ and $y_i^k - w_i^k \to 0$, we also have $x_i^k \rightharpoonup G_i \bar{z}$ and $y_i^k \rightharpoonup \overline{w}_i$. □

4 Extensions

4.1 Backtracking Linesearch

This section describes a backtracking linesearch procedure that may be used in the forward steps when the Lipschitz constant is unknown. The backtracking procedure is formalized in Algorithm 3, to be used in place of lines 9-10 of Algorithm 2.

We introduce the following notation: as suggested in line 8 of Algorithm 3, we set $J(i,k)$ to be the number of iterations of the backtracking algorithm for operator $i \in \mathcal{I}_\text{F}$ at outer iteration $k \geq 1$; the subsequent theorem will show that $J(i,k)$ can be upper bounded. As also suggested in line 8, we let $\hat{\rho}_i^{d(i,k)} = \rho_i^{(J(i,k),k)}$ for $i \in \mathcal{I}_\text{F} \cap \text{I}_k$. When using the backtracking procedure for $i \in \mathcal{I}_\text{F}$, it is important to note that the interpretation of $\rho_i^{d(i,k)}$ changes: it is the *initial* trial stepsize value for the i^th operator at iteration k, and the actual stepsize used is $\hat{\rho}_i^{d(i,k)}$. When $i \notin I_k$, we set $J(i,k) = 0$ and $\hat{\rho}_i^{d(i,k)} = \rho_i^{d(i,k)}$.

Assumption 5. *Lines 9-10 of Algorithm 2 are replaced with the procedure in Algorithm 3. Regarding stepsizes, we assume that*

$$\overline{\rho} \triangleq \max_{i=1,\dots,n} \left\{ \sup_k \rho_i^k \right\} < \infty \qquad\qquad \underline{\rho} = \min_{i=1,\dots,n} \left\{ \inf_k \rho_i^k \right\} > 0. \tag{40}$$

Algorithm 3: Backtracking procedure for unknown Lipschitz constants

Input : $i, k, z^{d(i,k)}, w_i^{d(i,k)}, \rho_i^{d(i,k)}, \Delta$

1 $\rho_i^{(1,k)} = \rho_i^{d(i,k)}$
2 $\theta_i^k = G_i z^{d(i,k)}$
3 $\zeta_i^k = T_i \theta_i^k$
4 **for** $j = 1, 2, \ldots$ **do**
5 $\quad \tilde{x}_i^{(j,k)} = \theta_i^k - \rho_i^{(j,k)}(\zeta_i^k - w_i^{d(i,k)})$
6 $\quad \tilde{y}_i^{(j,k)} = T_i \tilde{x}_i^{(j,k)}$
7 $\quad$ **if** $\Delta\|\theta_i^k - \tilde{x}_i^{(j,k)}\|^2 - \langle \theta_i^k - \tilde{x}_i^{(j,k)}, \tilde{y}_i^{(j,k)} - w_i^{d(i,k)} \rangle \leq 0$ **then**
8 $\quad\quad$ **return** $J(i,k) \leftarrow j,\ \hat{\rho}_i^{d(i,k)} \leftarrow \rho_i^{(j,k)},\ x_i^k \leftarrow \tilde{x}_i^{(j,k)},\ y_i^k \leftarrow \tilde{y}_i^{(j,k)}$
9 $\quad \rho_i^{(j+1,k)} = \rho_i^{(j,k)}/2$

Theorem 2. *Suppose Assumptions 1-3, and 5 hold. Then all the conclusions of Theorem 1 follow. Specifically, either the algorithm terminates in a finite number of iterations at point in $\mathcal{S}$, or there exists $(\bar{z}, \overline{\mathbf{w}}) \in \mathcal{S}$ s.t. $(z^k, \mathbf{w}^k) \rightharpoonup (\bar{z}, \overline{\mathbf{w}})$, $x_i^k \rightharpoonup G_i\bar{z}$ and $y_i^k \rightharpoonup \overline{w}_i$ for all $i = 1, \ldots, n-1$, $x_n^k \rightharpoonup \bar{z}$, and $y_n^k \rightharpoonup -\sum_{i=1}^{n-1} G_i^* \overline{w}_i$,*

Proof. The proof of finite termination at an optimal point follows as before, via Lemma 5. From now on, suppose $\pi_k > 0$ for all $k \geq M$ implying that the algorithm runs indefinitely.

The proof proceeds along the following outline: first, we upper bound the number of iterations of the loop in Algorithm 3, implying that the stepsizes $\hat{\rho}_i^k$ are bounded from above and below. We then argue that lemmas 6-10 hold as before. Then we show that lemmas 11-13 essentially still hold, but with different constants. The rest of the proof then proceeds identically to that of Theorem 1.

Regarding upper bounding the inner loop iterations, fix any $i \in \mathcal{I}_F$. For any $k \geq 1$ such that $i \in I_k$ and for any $j \geq 1$, substituting the values just assigned to θ_i^k and ζ_i^k allows us to expand the forward step on line 5 of Algorithm 3 into

$$\tilde{x}_i^{(j,k)} = G_i z^{d(i,k)} - \rho_i^{(j,k)}(T_i G_i z^{d(i,k)} - w_i^{d(i,k)}).$$

Following the arguments used to derive the $\mathcal{I}_F$ terms in (34), we have

$$\left(\left(\rho_i^{(j,k)}\right)^{-1} - L_i\right)\|G_i z^{d(i,k)} - \tilde{x}_i^{(j,k)}\|^2 - \langle G_i z^{d(i,k)} - \tilde{x}_i^{(j,k)}, \tilde{y}_i^{(j,k)} - w_i^{d(i,k)} \rangle \leq 0. \tag{41}$$

Using that $\rho_i^{(j,k)} = 2^{1-j}\rho_i^{d(i,k)}$, some elementary algebraic manipulations establish that once

$$j \geq \left\lceil 1 + \log_2\left((\Delta + L_i)\rho_i^{d(i,k)}\right)\right\rceil,$$

one must have $\Delta \leq \left(\rho_i^{(j,k)}\right)^{-1} - L_i$, and by (41) the condition triggering the return statement in Algorithm 3 must be true. Therefore, for any $k \geq 1$ we have

$$J(i,k) \leq \max\left\{\left\lceil 1 + \log_2\left((\Delta + L_i)\rho_i^{d(i,k)}\right)\right\rceil, 1\right\}$$

$$\leq \max\left\{2+\log_2\left((\Delta+L_i)\rho_i^{d(i,k)}\right),1\right\}. \tag{42}$$

By the condition $\overline{\rho}<\infty$ in (40), we may now infer that $\{J(i,k)\}_{k\in\mathbb{N}}$ is bounded. Furthermore, by substituting (42) into $\hat{\rho}_i^{d(i,k)} = 2^{1-J(i,k)}\rho_i^{d(i,k)}$, we may infer for all $k\geq 1$ that

$$\hat{\rho}_i^k \geq \min\left\{\frac{1}{2(L_i+\Delta)},\rho_i^{d(i,k)}\right\} \geq \min\left\{\frac{1}{2(L_i+\Delta)},\underline{\rho}\right\}. \tag{43}$$

Since $\hat{\rho}_i^k \leq \rho_i^{d(i,k)}$ for all $k\geq 1$, we must have $\limsup_{k\to\infty}\{\hat{\rho}_i^k\}\leq\bar{\rho}$. Since the choice of $i\in\mathcal{I}_\text{F}$ was arbitrary, we know that $\{\hat{\rho}_i^k\}_{k\in\mathbb{N}}$ is bounded for all $i\in\mathcal{I}_\text{F}$, and the first phase of the proof is complete.

We now turn to lemmas 6-10. First, Lemma 6 still holds, since it remains true that $y_i^k = T_i x_i^k$ for all $i\in\mathcal{I}_\text{F}$ and $k\geq M$. Next, a result like that of Lemma 7 holds, but with $\rho_i^{l(i,k)}$ replaced by $\hat{\rho}_i^{l(i,k)}$ for all $i\in\mathcal{I}_\text{F}$. The arguments of Lemmas 8-10 remain completely unchanged.

Next we show that Lemma 11 holds with a different constant. The derivation leading up to (32) continues to apply if we incorporate the substitution in Lemma 7 specified in the previous paragraph. Therefore, we replace ρ_i^k by $\hat{\rho}_i^k$ in (32) for $i\in\mathcal{I}_\text{F}$. Using (43), $\limsup_{k\to\infty}\{\hat{\rho}_i^k\}\leq\bar{\rho}$, and Assumption 5, we conclude that Lemma 11 still holds, with the constant ξ_1 adjusted in light of (43).

Now we show that Lemma 12 holds with a different constant. For $k\geq M$, we may use Lemma 7 and the termination criterion for Algorithm 3 to write

$$\begin{aligned}\psi_k &= \sum_{i\in\mathcal{I}_\text{B}}\left\langle G_i z^{l(i,k)}-x_i^k, y_i^k - w_i^{l(i,k)}\right\rangle + \sum_{i\in\mathcal{I}_\text{F}}\left\langle G_i z^{l(i,k)}-x_i^k, y_i^k - w_i^{l(i,k)}\right\rangle \\ &\geq (1-\sigma)\sum_{i\in\mathcal{I}_\text{B}}(\rho_i^k)^{-1}\|x_i^k - G_i z^{l(i,k)}\|^2 + \Delta\sum_{i\in\mathcal{I}_\text{F}}\|x_i^k - G_i z^{l(i,k)}\|^2.\end{aligned}$$

Here, the terms involving $\mathcal{I}_\text{B}$ are dealt with the same way as before in Lemma 12. We conclude that Lemma 12 holds with ξ_2 replaced by $\xi_2' = \min\{(1-\sigma)\overline{\rho}^{-1},\Delta\}$.

Now we show that Lemma 13 holds with a different constant. The derivation up to (36) proceeds as before, but replacing $\rho_i^{l(i,k)}$ with $\hat{\rho}_i^{l(i,k)}$ for $i\in\mathcal{I}_\text{F}$. Using (43) and Assumption 3, it is clear that the conclusion of Lemma 13 follows with the constant ξ_3 adjusted in light of (43).

Finally, the rest of the proof now follows in the same way as in the proof of Theorem 1. □

4.2 Backtracking is Unnecessary for Affine Operators

When $i\in\mathcal{I}_\text{F}$ and T_i affine, it is not necessary to iteratively backtrack to find a valid stepsize. Instead, it is possible to directly solve for a stepsize $\rho=\rho_i^{(j,k)}$ such that the condition on line 7 of Algorithm 3 is immediately satisfied. Thus, one can process an affine operator with only two forward steps, even without having estimated its Lipschitz constant.

From here on, we continue to use the notation $\theta_i^k = G_i z^{d(i,k)}$ and $\zeta_i^k = T_i\theta_i^k$ introduced in Algorithm 3. Fix $i\in\mathcal{I}_\text{F}$ and suppose that $T_i x = T_i^l x + c_i$ where $c_i\in\mathcal{H}_i$ and T_i^l is linear. The loop termination condition on line 7 of Algorithm 3 may be written

$$\langle\theta_i^k - \tilde{x}_i^{(j,k)}, \tilde{y}_i^{(j,k)} - w_i^{d(i,k)}\rangle \geq \Delta\|\theta_i^k - \tilde{x}_i^{(j,k)}\|^2. \tag{44}$$

Substituting the expressions for $\tilde{x}_i^{(j,k)}$ and $\tilde{y}_i^{(j,k)}$ from lines 5-6 of Algorithm 3 into the left-hand side of (44), replacing $\rho_i^{(i,j)}$ with ρ for simplicity, and using the linearity of T_i^l yields

$$\begin{aligned}
& \rho \left\langle \zeta_i^k - w_i^{d(i,k)}, T_i^l\Big(\theta_i^k - \rho\big(T_i G_i z^{d(i,k)} - w_i^{d(i,k)}\big)\Big) + c_i - w_i^{d(i,k)} \right\rangle \\
= \;& \rho \left\langle \zeta_i^k - w_i^{d(i,k)}, T_i^l \theta_i^k - \rho T_i^l\big(\zeta_i^k - w_i^{d(i,k)}\big) + c_i - w_i^{d(i,k)} \right\rangle \\
= \;& \rho \left\langle \zeta_i^k - w_i^{d(i,k)}, \zeta_i^k - w_i^{d(i,k)} - \rho T_i^l\big(\zeta_i^k - w_i^{d(i,k)}\big) \right\rangle \\
= \;& \rho \left(\|\zeta_i^k - w_i^{d(i,k)}\|^2 - \rho \left\langle \zeta_i^k - w_i^{d(i,k)}, T_i^l\big(\zeta_i^k - w_i^{d(i,k)}\big) \right\rangle \right). \qquad (45)
\end{aligned}$$

Substituting the expression for $\tilde{x}_i^{(i,j)}$ from line 5 of Algorithm 3, the right-hand side of (44) may be written

$$\Delta \rho^2 \|\zeta_i^k - w_i^{d(i,k)}\|^2. \qquad (46)$$

Substituting (45) and (46) into (44) and solving for ρ yields that the loop exit condition holds when

$$\rho \leq \tilde{\rho}_i^k \triangleq \frac{\|\zeta_i^k - w_i^{d(i,k)}\|^2}{\Delta \|\zeta_i^k - w_i^{d(i,k)}\|^2 + \left\langle \zeta_i^k - w_i^{d(i,k)}, T_i^l\big(\zeta_i^k - w_i^{d(i,k)}\big) \right\rangle}. \qquad (47)$$

If $\zeta_i^k - w_i^{d(i,k)} = 0$, then (47) is not defined, but in this case the step acceptance condition (44) holds trivially and lines 5-6 of the backtracking procedure yield $\tilde{x}_i^{(j,k)} = \theta_i^k$ and $\tilde{y}_i^{(j,k)} = \zeta_i^k$ for any stepsize $\rho_i^{(j,k)}$.

We next show that $\tilde{\rho}_i^k$ as defined in (47) will behave in a bounded manner even as $\zeta_i^k - w_i^{d(i,k)} \to 0$. Temporarily letting $\xi = \zeta_i^k - w_i^{d(i,k)}$, we note that as long as $\xi \neq 0$, we have

$$\tilde{\rho}_i^k = \frac{\|\xi\|^2}{\Delta \|\xi\|^2 + \langle \xi, T_i^l \xi \rangle} = \frac{1}{\Delta + \frac{\langle \xi, T_i^l \xi \rangle}{\|\xi\|^2}} \in \left[\frac{1}{\Delta + L_i}, \frac{1}{\Delta} \right], \qquad (48)$$

where the inclusion follows because T_i is monotone and thus T_i^l is positive semidefinite, and because T_i is L_i-Lipschitz continuous and therefore so is T_i^l. Thus, choosing $\tilde{\rho}_i^k$ to take some arbitary fixed value $\bar{\rho} > 0$ whenever $\zeta_i^k - w_i^{d(i,k)} = 0$, the sequence $\{\tilde{\rho}_i^k\}$ is bounded from both above and below, and all of the arguments of Theorem 2 apply if we use $\tilde{\rho}_i^k$ in place of the results of the backtracking line search.

In order to calculate (47), one must compute $\zeta_i^k = T_i G_i z^{d(i,k)}$ and $T_i^l\big(\zeta_i^k - w_i^{d(i,k)}\big)$. Then x_i^k can be obtained via $x_i^k = \theta_i^k - \rho\big(\zeta_i^k - w_i^{d(i,k)}\big)$ and

$$y_i^k = \zeta_i^k - \rho T_i^l\big(\zeta_i^k - w_i^{d(i,k)}\big). \qquad (49)$$

In total, this procedure requires one application of G_i and two applications of T_i^l.

4.3 Greedy Block Selection

We now introduce a greedy block selection strategy which may be useful in some implementations of Algorithm 2. We have found this stategy to improve performance on several empirical tests.

For simplicity, consider Algorithm 2 with $D = 0$ (no asynchronicity delays), $|I_k| = 1$ for all k (only one subproblem activated per iteration), and $\beta_k = 1$ for all k (no overrelaxation of the projection step). Consider some particular iteration $k \geq M$ and assume $\|\nabla\varphi_k\| > 0$ (otherwise the algorithm terminates at a solution). Ideally, one might like to maximize the length of the step $p^{k+1} - p^k$ toward the solution set $\mathcal{S}$, and $\left\|p^{k+1} - p^k\right\|_\gamma = \varphi_k(p^k)/\left\|\nabla\varphi_k\right\|_\gamma$.

Assuming that $\beta_k = 1$, the current point p^k computed on lines 25-26 of Algorithm 2 is the projection of p^{k-1} onto the halfspace $\{p : \varphi_{k-1}(p) \leq 0\}$. If p^{k-1} was not already in this halfspace, that is, $\varphi_{k-1}(p^{k-1}) > 0$, then after the projection we have $\varphi_{k-1}(p^k) = 0$. Using the notation $G_n = I$ and w_n^k defined in (14), $\varphi_{k-1}(p^k) = 0$ is equivalent to

$$\sum_{i=1}^{n} \langle G_i z^k - x_i^{k-1}, y_i^{k-1} - w_i^k \rangle = 0. \tag{50}$$

Suppose we select operator i to be processed next, that is, $I_k = \{i\}$. After updating (x_i^k, y_i^k), the corresponding term in the summation in (50) becomes either $\frac{1}{\rho_i}\|G_i z^k - x_i^k\|^2$ for $i \in \mathcal{I}_\text{B}$, or bounded below by $\Delta\|G_i z^k - x_i^k\|^2$ for $i \in \mathcal{I}_\text{F}$ with backtracking. Either way, processing operator i will cause this term to become nonnegative while the other terms remain unchanged, so if we select an i with $\langle G_i z^k - x_i^{k-1}, y_i^{k-1} - w_i^k \rangle < 0$, then the sum in (50) must increase by at least $-\langle G_i z^k - x_i^{k-1}, y_i^{k-1} - w_i^k \rangle$, meaning that after processing subproblem i we will have

$$\varphi_k(p^k) \geq -\langle G_i z^k - x_i^k, y_i^k - w_i^k \rangle > 0.$$

Choosing the i for which $\langle G_i z^k - x_i^{k-1}, y_i^{k-1} - w_i^k \rangle$ is the most negative maximizes the above lower bound on $\varphi_k(p^k)$ and would thus seem a promising heuristic for selecting i.

Note that this "greedy" procedure is only heuristic because it does not take into account the denominator in the projection operation, nor how much $\langle G_i z^k - x_i^k, y_i^k - w_i^k \rangle$ might exceed zero after processing block i. Predicting this quantity for every block, however, might require essentially the same computation as evaluating a proximal or forward step for all blocks, after which we might as well update all blocks, that is, set $I_k = \{1, \ldots, n\}$.

In order guarantee convergence under this block selection heuristic, we must build in a safeguard: if a block has not been processed for more than $M > 0$ iterations, we must process it immediately regardless of the value of $\langle G_i z^k - x_i^{k-1}, y_i^{k-1} - w_i^k \rangle$. This provision forces conformance to Assumption 2(1), so that convergence is still assured by Theorem 1.

4.4 Variable Metrics

Looking at Lemmas 12 and 13, it can be seen that the update rules for (x_i^k, y_i^k) can be abstracted. In fact any procedure that returns a pair (x_i^k, y_i^k) in the graph of T_i satisfying, for some $\xi_4 > 0$,

$$(\forall i = 1, \ldots, n) \quad \langle G_i z^{l(i,k)} - x_i^k, y_i^k - w_i^{l(i,k)} \rangle \geq \xi_4 \|G_i z^{l(i,k)} - x_i^k\|^2 \tag{51}$$

$$(\forall i \in \mathcal{I}_\text{B}) \quad \langle G_i z^{l(i,k)} - x_i^k, y_i^k - w_i^{l(i,k)} \rangle \geq \xi_4 \|y_i^k - w_i^{l(i,k)}\|^2 \tag{52}$$

$$(\forall i \in \mathcal{I}_\text{F}) \quad \langle G_i z^{l(i,k)} - x_i^k, y_i^k - w_i^{l(i,k)} \rangle + L_i\|G_i z^{l(i,k)} - x_i^k\|^2 \geq \xi_4 \|T_i G_i z^{l(i,k)} - w_i^{l(i,k)}\|^2 \tag{53}$$

yields a convergent algorithm. As with lemmas 12 and 13, these inequalities need only hold in the limit.

An obvious way to make use of this abstraction is to introduce variable metrics. To simplify the following, we will ignore the error terms e_i^k and assume no delays, i.e. $d(i,k) = k$. The updates on lines 5–7 and 8–10 of Algorithm 2 can be replaced with

$$(\forall i \in \mathcal{I}_{\mathrm{B}}) \qquad x_i^k + \rho_i^k U_i^k y_i^k = G_i z^k + \rho_i^k U_i^k w_i^k, \qquad y_i^k \in T_i x_i^k, \tag{54}$$

$$(\forall i \in \mathcal{I}_{\mathrm{F}}) \qquad x_i^k = z^k - \rho_i^k U_i^k (T_i G_i z^k - w_i^k), \qquad y_i^k = T_i x_i^k, \tag{55}$$

where $\{U_i^k : \mathcal{H}_i \to \mathcal{H}_i\}$ are a sequence of bounded linear self-adjoint operators such that

$$\forall i = 1, \ldots, n, x \in \mathcal{H}_i : \quad \inf_{k \geq 1} \langle x, U_i^k x \rangle \geq \underline{\lambda} \|x\|^2 \text{ and } \sup_{k \geq 1} \|U_i^k\| \leq \overline{\lambda} \tag{56}$$

where $0 < \underline{\lambda}, \overline{\lambda} < \infty$. In the finite dimensional case, (56) simply states that the eigenvalues of the set of matrices $\{U_i^k\}$ can be uniformly bounded away from 0 and $+\infty$. It can be shown that using (54)–(55) leads to the desired inequalities (51)–(53).

The new update (54) can be written as

$$x_i^k = (I + \rho_i^k U_i^k T_i)^{-1} (G_i z^k + \rho_i^k U_i^k w_i^k). \tag{57}$$

It was shown in [10, Lemma 3.7] that this is a proximal step with respect to $U_i^k T_i$ and that this operator is maximal monotone under an appropriate inner product. Thus the update (57) is single valued with full domain and hence well-defined. In the optimization context where $T_i = \partial f_i$ for closed convex proper f_i, solving (57) corresponds to the subproblem

$$\min_{x \in \mathcal{H}_i} \left\{ \rho_i^k f_i(x) + \frac{1}{2} \langle (U_i^k)^{-1} (x - a), x - a \rangle \right\}$$

where $a = G_i z^k + \rho_i^k U_i^k w_i^k$. For the variable-metric forward step (55), the stepsize constraint (28) must be replaced by $\rho_i^k < 1/\|U_i^k\| L_i$.

5 Prelimary Numerical Experiments

5.1 Overview

We now present some preliminary numerical experiments with variants of Algorithm 2. Due to the extensive effort required to build and test an efficient highly parallel implementation, our experiments are based on a prototype serial implementation and do not reflect true parallelism or asynchronicity; however, we do try to assess the likely impact of asynchronicity delays and of imposing block structure like (13) onto superficially more monolithic problems.

A ubiquitous optimization problem with a plethora of currently popular applications is ℓ_1-regularized least squares, or lasso. We consider the problem

$$\min_{x \in \mathbb{R}^d} \left\{ \frac{1}{2} \|Qx - b\|_2^2 + \lambda \|x\|_1 \right\}, \tag{58}$$

where $Q \in \mathbb{R}^{m\times d}$, $b \in \mathbb{R}^m$, and $\lambda \geq 0$. In the experiments we present here, we split problem (58) in the following way: let $\mathcal{R} = \{R_1, .., R_r\}$ be an arbitrary partition of $\{1, \ldots, m\}$. For $i = 1, \ldots, r$, let $Q_i \in \mathbb{R}^{|R_i|\times d}$ be the submatrix of Q with rows corresponding to indices in R_i and similarly let $b_i \in \mathbb{R}^{|R_i|}$ be the corresponding subvector of b. Then a problem equivalent to (58) is

$$\min_{x\in\mathbb{R}^d} \left\{ \frac{1}{2}\sum_{i=1}^{r} \|Q_i x - b_i\|_2^2 + \lambda\|x\|_1 \right\},$$

which is in the form of problem (2) and satisfies the constraint qualification of [2, Theorem 16.47(i)]. Therefore, it is a special case of (13) with $n = r + 1$, $G_i = I$ for $i = 1, \ldots, r + 1$, $T_i(x) = Q_i^\top(Q_i x - b_i)$ for $i = 1, \ldots, r$, and $T_{r+1}(x) = \partial\{\lambda\|x\|_1\}$. It is also possible to consider a formulation in which the elements of $\mathcal{R}$ overlap, but for simplicity we do not consider this situation here. In a distributed-memory parallel implementation, one may imagine that the block matrices Q_i could be stored in different processors.

There are other possible ways to model problem (58) in the form (13): for example, one could set $G_i = Q_i$ and $T_i(t) = \frac{1}{2}\|t - b_i\|^2$ for $i = 1, \ldots, r$. We will consider such formulations in future work.

The operator T_n is not Lipschitz, so we always use backward steps to process it. The backward step for this operator is the well-known shrinkage and soft thresholding operation, whose calculations are very simple and take time linear in d. Because of its simplicity, we process the operator T_n at every iteration, that is, $n \in I_k$ for all $k \geq 1$; in a distributed implementation, processing of T_n could be considered as essentially part of the coordination process. Furthermore note that the entire coordination/projection process conducted at each iteration from lines 13-27 of Algorithm 2, has $\mathrm{O}(d)$ computational complexity, and it remains $\mathrm{O}(d)$ even if we include the processing of T_n at every step.

On the other hand, the operators $T_i = Q_i^\top(Q_i(\cdot) - b_i)$, $i = 1, \ldots, r = n - 1$, are Lipschitz and could be processed by either forward or backward steps. Since they are affine, we can furthermore use the stepsize rule in Section 4.2, which only requires two forward steps per iteration, that is, two multiplies by Q_i and two by $Q_i^\top$. We do not need to estimate the largest eigenvalue of Q or Q_i, a potentially costly computation. We will refer to the approach of setting $\mathcal{I}_{\mathrm{F}} = \{1, \ldots, r\}$ and using the stepsize rule from Section 4.2 as "Projective Splitting with Forward Steps" (PSFor).

On the other hand, we could also process $T_1, \ldots, T_r$ with backward steps. The corresponding proximal map is

$$\mathrm{prox}_{\rho\nabla\{\frac{1}{2}\|Q_i(\cdot)-b_i\|_2^2\}}(t) = (I + \rho Q_i^\top Q_i)^{-1}(t + \rho_i b). \tag{59}$$

A standard way to approach this calculation is to employ a matrix factorization to solve the system of equations $(I + \rho Q_i^\top Q_i)x = t + \rho b$. If $|R_i| < d$, it may be preferable to use the matrix inversion lemma as in [4] to factor a matrix of dimension $|R_i| \times |R_i|$ instead of the $d \times d$ dimensional matrix $I + \rho Q_i^\top Q_i$. But if both d and $|R_i|$ are large, then neither of these options may be attractive.

Another option is to use an iterative solver to compute (59); this option is particularly attractive since our method tolerates errors in the computation of backward steps. So, we choose some $\sigma \in [0,1)$ and apply a standard conjugate gradient algorithm to the linear system of equations $(I + \rho Q_i^\top Q_i)x = t + \rho b$ until the relative error criteria (26) and (27) are met.

For a given candidate solution x_i^k, we have $y_i^k = Q_i^\top(Q_i x_i^k - b_i)$, and the error can be explicitly computed as $e_i^k = x_i^k + \rho_i y_i^k - (z^{d(i,k)} + \rho_i w_i^{d(i,k)})$. Every iteration of the conjugate gradient algorithm requires two matrix multiplies, one by Q_i and one by $Q_i^\top$. We "warm-start" the conjugate gradient algorithm by initializing it at x_i^{k-1}. We will refer to this approach as "Projective Splitting with Backward Steps" (PSBack).

We compared PSFor and PSBack with two more established methods: FISTA [3] and the relative error inexact version of ADMM proposed in [19, Algorithm 4], which we will call RE-ADMM. For FISTA, we used the backtracking linesearch variant. For RE-ADMM, we experimented with different values for the relative error constant σ and found $\sigma = 0.9$ typically worked best for our test datasets. The stepsize c had to be tuned for each dataset and its values are given in Table 1.

To compare the methods fairly, we counted each algorithm's number of "Q-equivalent" matrix multiplies — multiplication by a matrix the size of Q or $Q^\top$. In applications with large data matrices, matrix multiplications should be the dominant computation at each iteration. For Algorithm 2, the number of "Q-equivalent" multiplies depends on the number of blocks $i \in I_k$ at each iteration and the size of each block, corresponding to the number of rows in each matrix Q_i. Note that Q_i is a smaller matrix than Q, which we take into account as follows: since Q_i has $|R_i|$ rows and all the matrices in our testing are dense, one Q_i multiply counts as $|R_i|/m$ Q-equivalent multiplies.

We implemented serial versions of PSFor and PSBack in Python using the `numpy` package. Our present implementation is strictly serial, but able to simulate some asynchronous and block-iterative effects. We ran the algorithm with various values of r with each block chosen to have the same number of rows (or nearly so when m is not divisible by r). At each iteration, we selected one block from among $1, \ldots, r = n-1$ for a forward step in PSFor or backward step with CG in PSBack, and block $n = r+1$ for a backward step. Thus, I_k always has the form $\{i, n\}$, with $1 \leq i < n$. To select this i, we tested the greedy block selection scheme described in Section 4.3, as well as choosing blocks at random. For the greedy scheme, we did not use the safeguard parameter M as in practice we found that every block was updated fairly regularly. We also considered various values for the maximum delay D as defined in Assumption 2: for each activated block we simulated asynchonicity by uniformly randomly generating a delay $d(i,k) \in \{k-D, .., k\}$ while ensuring that the point $(w_i^{d(i,k)}, z^{d(i,k)})$ used is newer than the last point used for that block.

5.2 Experimental Details

We considered three real datasets taken from the UCI Machine Learning Repository [14], as well as one synthetic dataset. All the data matrices were dense and we normalized their columns to have unit norm. The datasets consisted of the gene expression cancer RNA-Seq dataset[1] [34] of size 3204×20531, which we will refer to as "*gene*", the "*drivFace*" dataset[2]

[1]`https://archive.ics.uci.edu/ml/datasets/gene+expression+cancer+RNA-Seq`

[2]`https://archive.ics.uci.edu/ml/datasets/DrivFace`

[15] of size 606×6400, the sEMG hand gesture data set[3], [29] of size 1800×3000, which we refer to as "*hand*", and finally a synthetic dataset of size 1000×10000 with all entries of A and b drawn i.i.d. $\mathcal{N}(0,1)$. For *gene*, the original dataset has 801 examples. We augmented each example with three "noisy" copies, each being equal to the original feature vector plus an i.i.d. zero-mean Gaussian random vector with standard deviation equal to 0.1 times the original population standard deviation.

For PSFor we used $\Delta = 1$ and for PSBack we used $\sigma = 0.9$ for all experiments. For the forward steps, instead of (47), we used the modified rule

$$\rho_i^k = \min\left\{\frac{\tilde{\rho}_i^k}{2}, \rho_i^{s(i,k-1)}\right\}, \quad i = 1, \ldots, r, \quad k \geq 1, \tag{60}$$

which performed in a more stable manner in our experiments; recall that $\rho_i^{s(i,k-1)}$ is the most recently used stepsize for the operator T_i. To handle the case $k = 1$, we set $\rho_i^0 = +\infty$.

To set ρ_{r+1}^k for PSFor, we used a heuristic inspired by the forward-backward method. Specifically, we set ρ_{r+1}^k to be the average of the most recently used stepsizes across all $i \in \mathcal{I}_{\mathrm{F}}$, that is, $\rho_{r+1}^k = \frac{1}{r}\sum_{i=1}^r \rho_i^{s(i,k)}$. (If an operator has not yet been updated with a forward step then it is not included in the average).

Table 1 displays the chosen values for γ (which were the same for PSFor and PSBack), the stepsizes for PSBack, and the parameter c for RE-ADMM. For simplicity, the stepsizes in PSBack were fixed across iterations. These parameters were chosen by running the algorithms for a small number of iterations using a small selection of candidate values and seeing which one worked best. The table also shows the model paramater λ, which in each case was chosen so that the solution had approximately 10% of its entries nonzero.

	gene	drivFace	hand	random
γ	100	100	1	1
$\rho_1 \ldots \rho_r$ (PSBack)	10^{-3}	1	0.1	0.1
ρ_{r+1} (PSBack)	10^{-3}	1	0.1	0.1
c (RE-ADMM)	100	10	1	1
λ	10	10	1	1

Table 1: Parameter settings for PSFor, PSBack, and RE-ADMM for the four lasso datasets.

Let $F(x)$ be the lasso objective defined in (58). We test the algorithms by comparing:

1. the objective function residual: $(F(x_{alg}^m) - F^*)/F^*$, and
2. the subgradient residual: $\min\{\|g\| : g \in \partial F(x_{alg}^m)\}$,

where x_{alg}^m is an appropriate output iterate from each algorithm, F^* is the optimal value of the problem estimated by running all algorithms for 10^5 iterations and taking the minimum objective value, and m counts the number of "Q-equivalent multiplies". For PSFor and

[3] `https://archive.ics.uci.edu/ml/datasets/sEMG+for+Basic+Hand+movements`. We combined all 5 subjects into one dataset.

PSBack, $x^m_{alg} = x^m_{r+1}$, and for RE-ADMM $x^m_{alg} = z^m$ computed on the second-to-last line of [19, Algorithm 4].

In the subsequent results section we will use the following notation: PSFor(r, D) means PSFor with r blocks and a maximum delay of D, and similarly for PSBack(r, D). PSFor(r, G) means PSFor with r blocks and the greedy block selection strategy of Section 4.3.

5.3 Results

PSFor$(10, G)$ and PSBack$(10, G)$ were the best-performing variants of PSFor and PSBack, respectively. Figure 2 compares these two algorithms with FISTA and RE-ADMM on each of the four datasets. On all datasets except *drivFace*, PSFor$(10, G)$ performs the best. PSFor outperforms PSBack on all datasets. Figure 3 plots the effect of increasing delay on PSFor and of not using the greedy block selection strategy. Both of these adversely effect the performance on the algorithm, but the degradation is fairly graceful. In a distributed setting this mild degradation may be outweighed by the considerable advantages and speedups offered by an asynchronous implementation. Figure 3 also demostrates that PSFor$(1, 0)$ is far slower than any of the variants with 10 blocks. This runs counter to the typical behavior of decomposition methods in optimization, which tend to converge more slowly the more subsystems they must coordinate.

5.4 Further Experiments: Blocks Processed Per Iteration

In the previous experiments, the set of blocks to be processed at each iteration was $I_k = \{i, r+1\}$, where $i \in \{1, \ldots, r\}$ was selected either at random or by the greedy procedure of Section 4.3. We now consider the effect of choosing more than one least-squares block per iteration, that is, setting $I_k = \{i_1, i_2, \ldots, i_b, r+1\}$ where $\{i_j\}_{j=1}^b$ may be selected randomly or taken to be the "best" b blocks according to the greedy criterion. We refer to these variants as PSFor(r, R, b) and PSFor(r, G, b). For simplicity, we will not consider delays in these additional experiments.

Increasing b increases the per-iteration complexity of the algorithm in a straightforward way. Processing a block via forward steps requires two multiplies by Q_i and two by $Q_i^\top$. Hence, each additional block requires $\mathrm{O}(md/r)$ more flops per iteration.

Figures 4 and 5 respectively show results for PSFor$(10, G, b)$ and PSFor$(10, R, b)$, with $b = 1, 2, 4, 8$. Note that the x-axis is now the number of iterations of Algorithm 2, not the number of Q-equivalent multiplies. Rather suprisingly, PSFor$(10, G, 1)$ performs almost as well as PSFor$(10, G, b)$ for $b = 2, 4, 8$, even though it only processes one block per iteration and hence has far lower per-iteration complexity. It seems that most of the information required to create a "good" separating hyperplane is contained in just one block. For randomly chosen blocks, there is greater benefit to using more than one block, especially for the *random* and *gene* datasets. However, on the *hand* and *drivFace* datasets there is little benefit. In all experiments there is little or no advantage to employing more than two blocks per iteration.

These experiments suggest that processing many blocks per iteration may well be undesirable. A similar convergence rate can be achieved with much lower per-iteration complexity by processing just one or two blocks per iteration, expecially if using the greedy selection criterion.

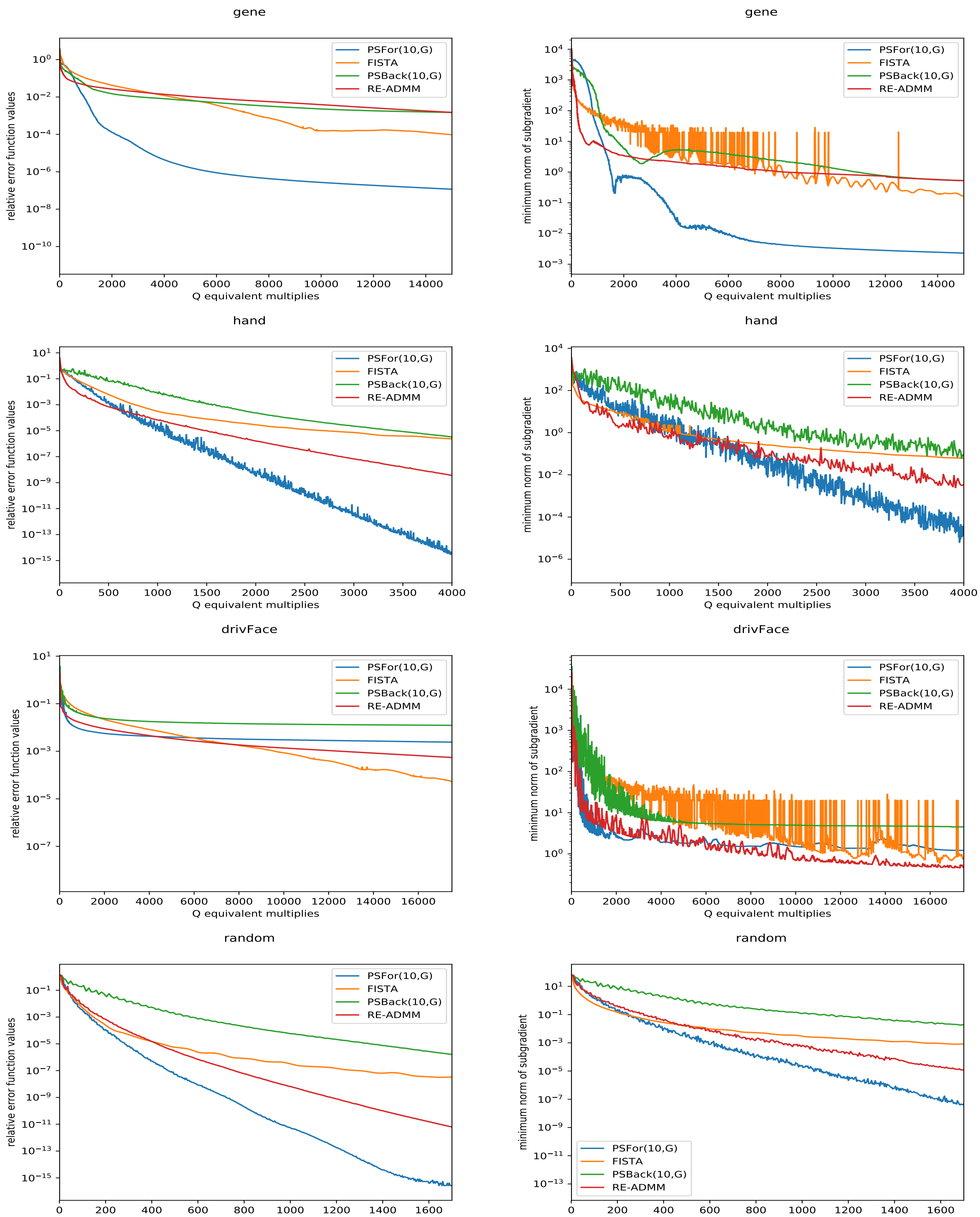

Figure 2: Results on the four datasets. The left column plots function values in terms of relative error and the right column plots the minimum subgradient norm.

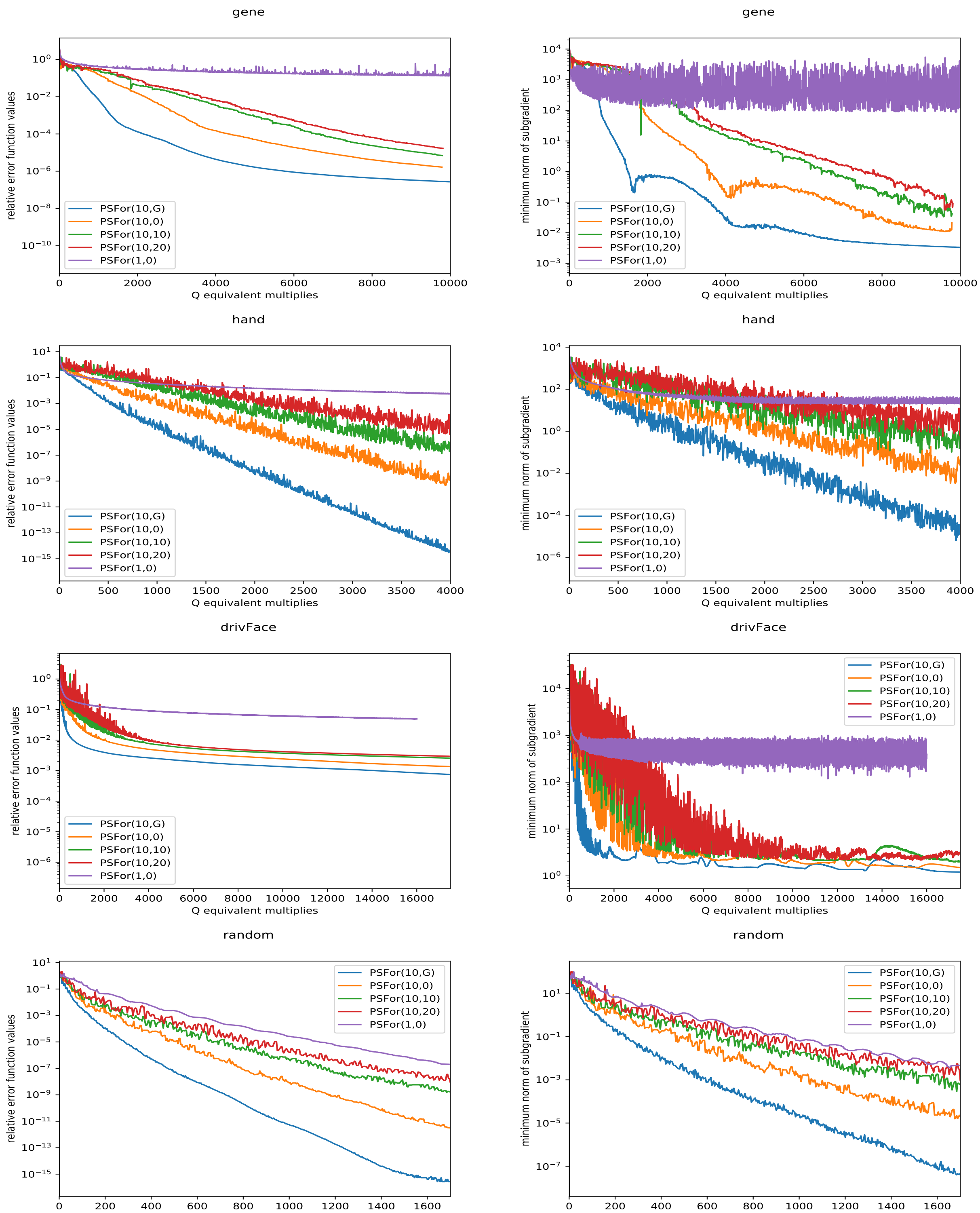

Figure 3: Effect of D on the performance of PSFor. The left column plots function values in terms of relative error and the right column plots the minimum subgradient norm.

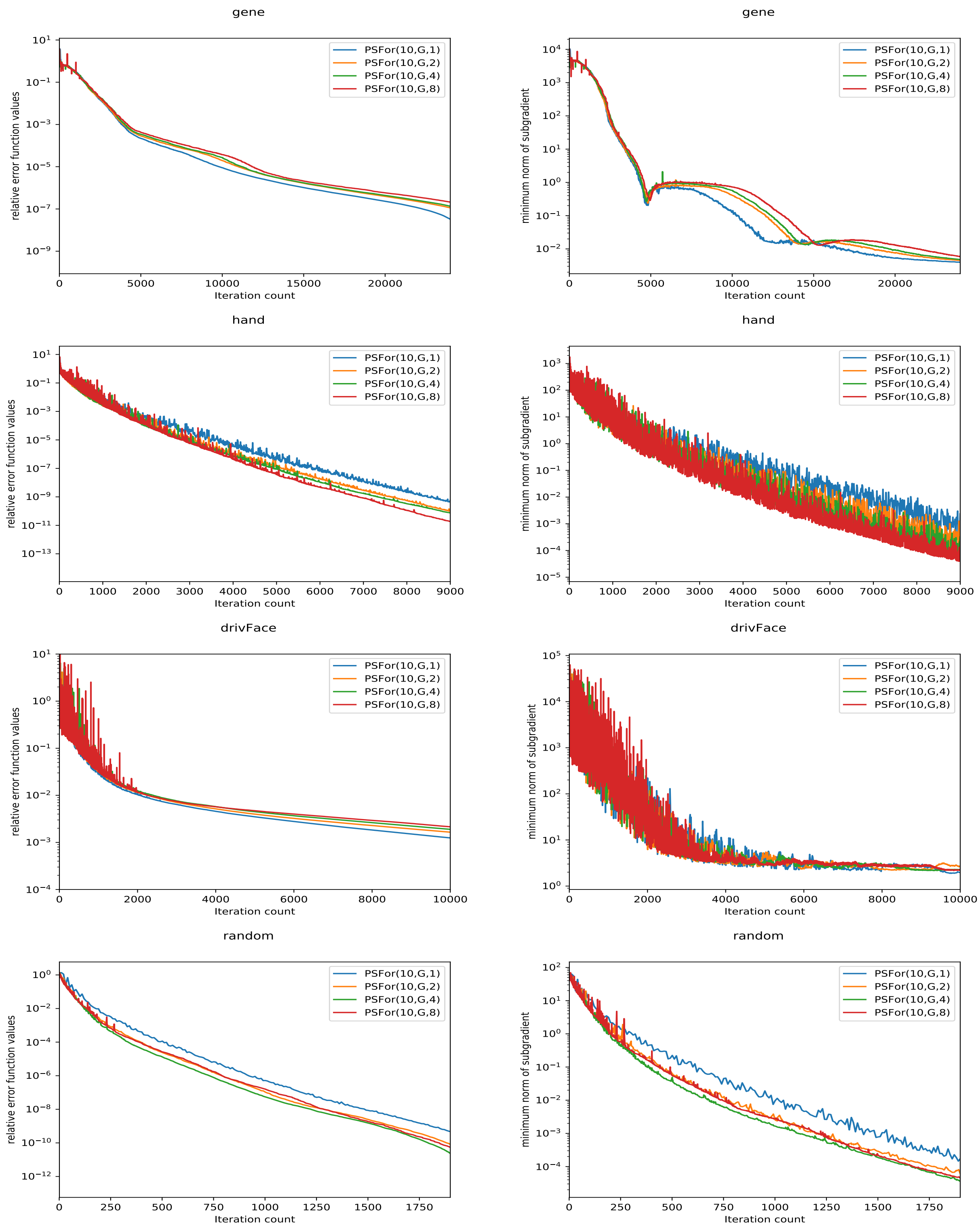

Figure 4: Effect of the number of blocks processed per iteration with greedy block selection. The left column plots function values in terms of relative error and the right column plots the minimum subgradient norm.

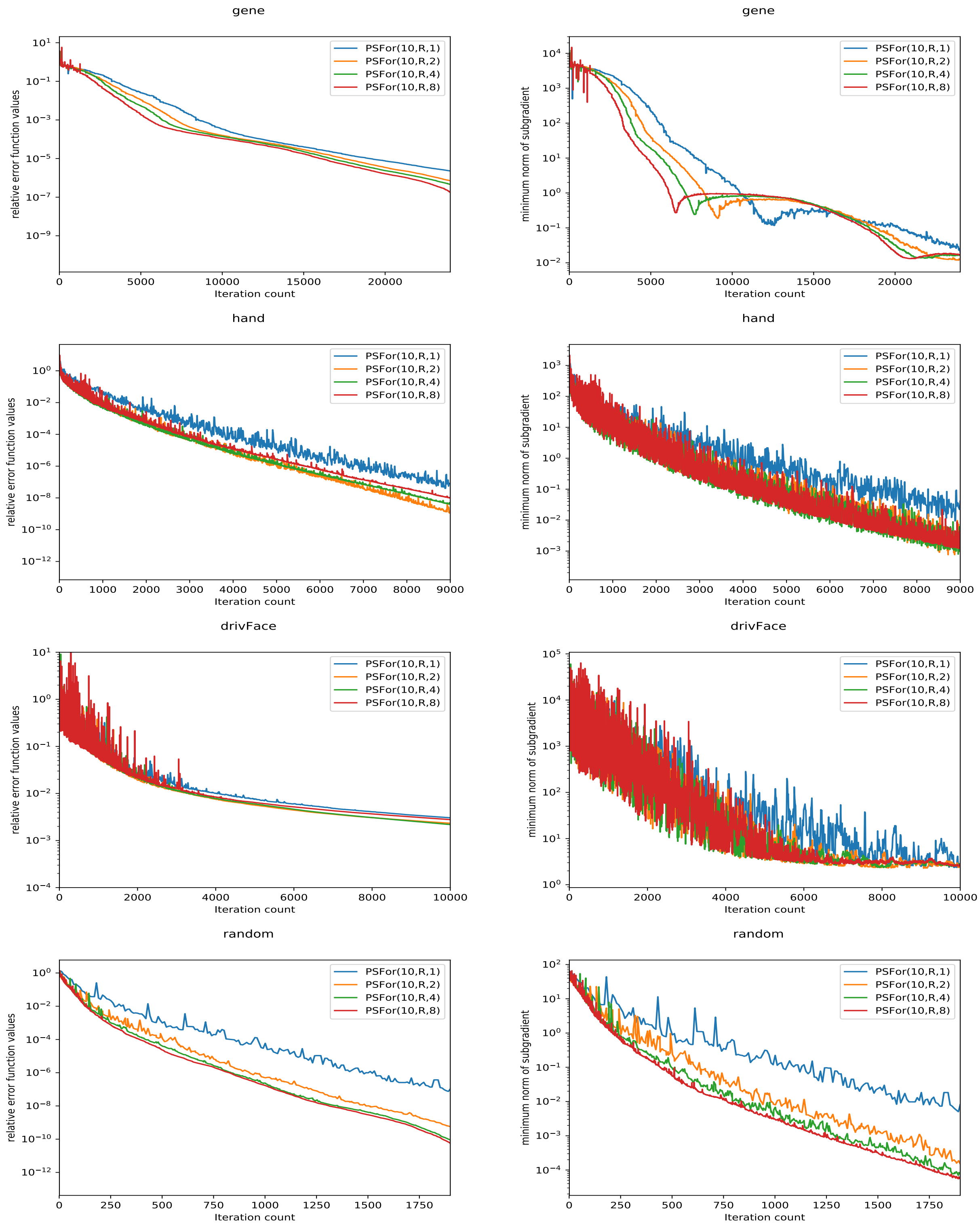

Figure 5: Effect of the number of blocks processed per iteration with random block selection. The left column plots function values in terms of relative error and the right column plots the minimum subgradient norm.

Acknowledgements This material is based upon work supported by the National Science Foundation under Grant No. 1617617.

References

[1] Alotaibi, A., Combettes, P.L., Shahzad, N.: Solving coupled composite monotone inclusions by successive Fejér approximations of their Kuhn-Tucker set. SIAM Journal on Optimization **24**(4), 2076–2095 (2014)

[2] Bauschke, H.H., Combettes, P.L.: Convex analysis and monotone operator theory in Hilbert spaces, 2nd edn. Springer (2017)

[3] Beck, A., Teboulle, M.: Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems. IEEE Transactions on Image Processing **18**(11), 2419–2434 (2009)

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

[5] Briceño-Arias, L.M., Combettes, P.L.: A monotone+ skew splitting model for composite monotone inclusions in duality. SIAM Journal on Optimization **21**(4), 1230–1250 (2011)

[6] Chambolle, A., Pock, T.: A first-order primal-dual algorithm for convex problems with applications to imaging. Journal of Mathematical Imaging and Vision **40**(1), 120–145 (2011)

[7] Combettes, P.L.: Fejér monotonicity in convex optimization. In: Encyclopedia of optimization, vol. 2, pp. 106–114. Springer Science & Business Media (2001)

[8] Combettes, P.L., Eckstein, J.: Asynchronous block-iterative primal-dual decomposition methods for monotone inclusions. Mathematical Programming **168**(1-2), 645–672 (2018)

[9] Combettes, P.L., Pesquet, J.C.: Proximal splitting methods in signal processing. In: Fixed-point algorithms for inverse problems in science and engineering, pp. 185–212. Springer (2011)

[10] Combettes, P.L., Vũ, B.C.: Variable metric forward–backward splitting with applications to monotone inclusions in duality. Optimization **63**(9), 1289–1318 (2014)

[11] Combettes, P.L., Wajs, V.R.: Signal recovery by proximal forward-backward splitting. Multiscale Modeling & Simulation **4**(4), 1168–1200 (2005)

[12] Condat, L.: A primal–dual splitting method for convex optimization involving Lipschitzian, proximable and linear composite terms. Journal of Optimization Theory and Applications **158**(2), 460–479 (2013)

[13] Davis, D., Yin, W.: A three-operator splitting scheme and its optimization applications. Set-Valued and Variational Analysis **25**(4), 829–858 (2017)

[14] Dheeru, D., Karra Taniskidou, E.: UCI Machine Learning Repository (2017). URL `http://archive.ics.uci.edu/ml`

[15] Diaz-Chito, K., Hernández-Sabaté, A., López, A.M.: A reduced feature set for driver head pose estimation. Applied Soft Computing **45**, 98–107 (2016)

[16] Eckstein, J.: A simplified form of block-iterative operator splitting and an asynchronous algorithm resembling the multi-block alternating direction method of multipliers. Journal of Optimization Theory and Applications **173**(1), 155–182 (2017)

[17] Eckstein, J., Svaiter, B.F.: A family of projective splitting methods for the sum of two maximal monotone operators. Mathematical Programming **111**(1), 173–199 (2008)

[18] Eckstein, J., Svaiter, B.F.: General projective splitting methods for sums of maximal monotone operators. SIAM Journal on Control and Optimization **48**(2), 787–811 (2009)

[19] Eckstein, J., Yao, W.: Relative-error approximate versions of Douglas-Rachford splitting and special cases of the ADMM. Mathematical Programming (2017). DOI 10.1007/s10107-017-1160-5

[20] Iusem, A., Svaiter, B.: A variant of Korpelevich's method for variational inequalities with a new search strategy. Optimization **42**(4), 309–321 (1997)

[21] Komodakis, N., Pesquet, J.C.: Playing with duality: An overview of recent primal-dual approaches for solving large-scale optimization problems. IEEE Signal Processing Magazine **32**(6), 31–54 (2015)

[22] Korpelevich, G.: The extragradient method for finding saddle points and other problems. Matecon **12**, 747–756 (1976)

[23] Krasnosel'skii, M.A.: Two remarks on the method of successive approximations. Uspekhi Matematicheskikh Nauk **10**(1), 123–127 (1955)

[24] Lions, P.L., Mercier, B.: Splitting algorithms for the sum of two nonlinear operators. SIAM Journal on Numerical Analysis **16**(6), 964–979 (1979)

[25] Mann, W.R.: Mean value methods in iteration. Proceedings of the American Mathematical Society **4**(3), 506–510 (1953)

[26] Mercier, B., Vijayasundaram, G.: Lectures on topics in finite element solution of elliptic problems. Tata Institute of Fundamental Research, Bombay (1979)

[27] Peng, Z., Xu, Y., Yan, M., Yin, W.: ARock: an algorithmic framework for asynchronous parallel coordinate updates. SIAM Journal on Scientific Computing **38**(5), A2851–A2879 (2016)

[28] Recht, B., Re, C., Wright, S., Niu, F.: Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In: Advances in neural information processing systems, pp. 693–701 (2011)

[29] Sapsanis, C., Georgoulas, G., Tzes, A., Lymberopoulos, D.: Improving EMG based classification of basic hand movements using EMD. In: Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE, pp. 5754–5757. IEEE (2013)

[30] Solodov, M.V., Svaiter, B.F.: A hybrid projection-proximal point algorithm. Journal of convex analysis **6**(1), 59–70 (1999)

[31] Solodov, M.V., Svaiter, B.F.: A new projection method for variational inequality problems. SIAM Journal on Control and Optimization **37**(3), 765–776 (1999)

[32] Tseng, P.: A modified forward-backward splitting method for maximal monotone mappings. SIAM Journal on Control and Optimization **38**(2), 431–446 (2000)

[33] Vũ, B.C.: A splitting algorithm for dual monotone inclusions involving cocoercive operators. Advances in Computational Mathematics **38**(3), 667–681 (2013)

[34] Weinstein, J.N., Collisson, E.A., Mills, G.B., Shaw, K.R.M., Ozenberger, B.A., Ellrott, K., Shmulevich, I., Sander, C., Stuart, J.M., Network, C.G.A.R., et al.: The Cancer Genome Atlas Pan-Cancer analysis project. Nature genetics **45**(10), 1113 (2013)