A Simplified Form of Block-Iterative Operator Splitting, and an Asynchronous Algorithm Resembling the Multi-Block ADMM*

Jonathan Eckstein †

July 8, 2016

Abstract

This paper develops what is essentially a simplified version of the block-iterative operator splitting method already proposed by the author and P. Combettes, but with more general initialization conditions. It then describes one way of implementing this algorithm asynchronously under a computing model inspired by modern HPC environments, which consist of interconnected nodes each having multiple processor cores sharing a common local memory. The asynchronous implementation framework is then applied to derive an asynchronous algorithm which resembles the ADMM with an arbitrary number of blocks of variables. Unlike earlier proposals for asynchronous ADMM-like methods, the algorithm relies neither on probabilistic control nor on restrictive assumptions about the problem instance, instead making only standard convex-analytic regularity assumptions. It also allows the proximal parameter to vary freely between arbitrary positive bounds, an unusual feature in an ADMM-like method.

1 Introduction

Consider an optimization problem of the form

$$\begin{array}{ll} \min & \sum_{i=1}^{n} f_i(x_i) \\ \text{ST} & \sum_{i=1}^{n} M_i x_i = b, \end{array} \tag{1}$$

where $n \geq 2$, $f_i : \mathbb{R}^{n_i} \to \mathbb{R} \cup \{+\infty\}$ is closed proper convex and M_i is an $m \times n_i$ matrix for $i = 1, \ldots, n$, and $b \in \mathbb{R}^m$. For the case $n = 2$, a currently popular decomposition method for such problems is the alternating direction method of multipliers (ADMM) [12, 13, 8, 11]. There has also been considerable interest in the ADMM-class algorithms for the case $n > 2$,

*This work was supported in part by NSF grant CCF-1115638, Computing and Communications Foundations, CISE directorate. The author would also like to thank Patrick Combettes, as this work grew out the same discussions with him that led to [7].

†Department of Management Science and Information Systems and RUTCOR, 100 Rockafeller Road, Livingston Campus, Rutgers University, Piscataway NJ 08854 USA, jeckstei@rci.rutgers.edu.

although the ADMM's most obvious generalization to $n > 2$ does not converge in the general convex case [5].

The end goal of this paper is to develop an ADMM-like algorithm for (1) that not only can accomodate any value of $n \geq 2$, but can operate asynchronously in a parallel environment. Asynchronism may be important for efficient parallel implementation in cases in which subproblems related to the various formulation blocks $(f_i, M_i)_{i=1,\ldots,n}$ need to be overlapped in parallel but have run times that may vary widely with i or are otherwise unpredictable. There have been a number of prior proposals for asynchronous parallel variants of the ADMM, but they have a number of features not present in the original ADMM, namely:

- Reliance on randomized algorithm elements and probabilistic convergence analyses [14, 18, 19]
- Restrictive problem assumptions beyond those present in (1) above [15, 18, 19, 4]
- Processing patterns linked to a given communication graph topology [15, 18].

This paper develops an asynchronous ADMM-like method that is innovative in that it does not rely on randomization and uses only standard convex-analytic regularity assumptions on (1) — see Assumptions 16 and 17 below. These assumptions simply ensure that an optimal solution exists and that standard optimality conditions apply there. Further, instead of assuming processors communicate using a fixed graph — a style of computation appropriate for distributed sensor networks, for example — it uses a computational setting modeled after current high-performance computing (HPC) environments. Such systems are usually made up of multiple processing nodes interconnected by a powerful all-to-all connected network; each node consists of multiple processor cores operating from a locally shared memory.

Unlike prior work on asynchronous ADMM variants, this paper takes a "top-down" approach: the key to the development is to specialize a more general monotone operator splitting algorithm which is in turn in essence a special case of the block-iterative monotone operator splitting scheme proposed in [7]. This algorithm solves a monotone inclusion problem — see (2) below — that is more general than (1). Section 2 develops the convergence theory of this method which is shown as Algorithm 1 below. While the algorithm in nearly a special case of the weakly convergent method proposed in [7], it has much less restrictive initialization conditions than can be derived from [7], a property that necessitates analyzing its convergence "from scratch". As discussed in Section 2, these proofs have additional potential benefits.

Section 3 next analyzes a possible way of implementing the algorithm of Section 2 in an environment modeled after a modern HPC system. The mathematical problem is still an abstract monotone inclusion, but the implementation details are more specific.

Finally, Section 4 further specializes the algorithm of Section 3 to produce an asynchronous procedure for solving (1). This procedure does not have the probabilistic algorithmic elements or problem-instance restrictions of earlier proposed asynchronous ADMM-like methods. Furthermore, it allows the proximal parameter to be varied with both the iteration and the block index i, a novel property for an ADMM-like method. The "price" paid for these advantages is that the calculations required to coordinate the various subsystem calculations are somewhat complicated, involving orthogonal projection onto a halfspace within

a certain primal-dual space. However, as noted in Remarks 1 and 4 below, it should in some cases be possible to distribute these coordination calculations among multiple processors.

2 A Form of Block-Iterative Operator Splitting

Suppose $n \geq 2$ and

- $\mathcal{H}_0, \mathcal{H}_1, \ldots, \mathcal{H}_n$ are real Hilbert spaces
- $T_i : \mathcal{H}_i \rightrightarrows \mathcal{H}_i$ are maximal monotone set-valued operators, $i = 1, \ldots, n$
- $L_i : \mathcal{H}_0 \to \mathcal{H}_i$ are continuous linear operators, $i = 1, \ldots, n$.

Consider the problem of finding $x \in \mathcal{H}_0$ such that

$$0 \in \sum_{i=1}^{n} L_i^* T_i(L_i x), \tag{2}$$

where L_i^* denotes the adjoint of L_i. Consider the linear space

$$\mathcal{W} = \left\{ (w_1, \ldots, w_n) \in \mathcal{H}_1 \times \cdots \times \mathcal{H}_n \;\middle|\; \sum_{i=1}^{n} L_i^* w_i = 0 \right\}, \tag{3}$$

and let $\mathrm{proj}_{\mathcal{W}}$ denote the orthogonal projection map $\mathcal{H}_1 \times \cdots \times \mathcal{H}_n \to \mathcal{W}$ using the norms induced by those of $\mathcal{H}_1, \ldots, \mathcal{H}_n$, that is, given some $x = (x_1, \ldots, x_n) \in \mathcal{H}_1 \times \cdots \times \mathcal{H}_n$,

$$\mathrm{proj}_{\mathcal{W}}(x) = \mathrm{proj}_{\mathcal{W}}(x_1, \ldots, x_n) = \operatorname*{arg\,min}_{(u_i \in \mathcal{H}_i)_{i=1}^n} \left\{ \sum_{i=1}^{n} \|u_i - x_i\|^2 \;\middle|\; \sum_{i=1}^{n} L_i^* u_i = 0 \right\}.$$

We will proceed to develop a decomposition method (Algorithm 1 below) applicable to situations in which $\mathrm{proj}_{\mathcal{W}}$ is easily evaluated. In the general case, however, $\mathrm{proj}_{\mathcal{W}}$ may not be easy to evaluate or even to explicitly characterize, in which case the more complicated algorithms proposed in [7] may be preferable. The overall approach also builds on earlier work in [9, 10] and [1].

As with most operator-splitting methods, a basic building block of our algorithm is the "prox" operation, for which we use the following slightly unconventional notation:

Definition 1. *Let T be a maximal monotone set-value operator on a real Hilbert space $\mathcal{H}$, let $t \in \mathcal{H}$, and let λ be any positive scalar. Then $\mathrm{Prox}_T^\lambda(t)$ denotes the unique $(x, y) \in \mathrm{Gph}\, T$ such that $x + \lambda y = t$.*

In terms of more customary notation involving the resolvent mapping $J_{\lambda T} = (I + \lambda T)^{-1}$, setting $(x, y) = \mathrm{Prox}_T^\lambda(t)$ is equivalent to setting

$$x = J_{\lambda T}(t) = (I + \lambda T)^{-1}(t) \qquad\qquad y = \frac{1}{\lambda}(t - x). \tag{4}$$

This section's main algorithm has the following parameters:

Algorithm 1: Abstract algorithm for problem (2)

1 Start with any $z^0 \in \mathcal{H}^0$, $w^0 \in \mathcal{W}$ and $(x_i^{-1}, y_i^{-1} \in \mathcal{H}_i)_{i=1,\ldots,n}$
2 **for** $k = 0, 1, 2, \ldots$ **do**
3 **for** $i = 1, \ldots, n$ **do**
4 **if** $i \in I_k$ **then**
5 $(x_i^k, y_i^k) = \mathrm{Prox}_{T_i}^{\mu_{i,d(i,k)}}\Big(L_i z^{d(i,k)} + \mu_{i,d(i,k)} w_i^{d(i,k)} + e_i^k\Big)$, for some $e_i^k \in \mathcal{H}_i$
6 or, if $k < \hat{k}$, then $(x_i^k, y_i^k) \in \mathcal{H}_i^2$ may be chosen arbitrarily
7 **else**
8 $(x_i^k, y_i^k) = (x_i^{k-1}, y_i^{k-1})$
9 $u^k = \mathrm{proj}_{\mathcal{W}}(x_1^k, \ldots, x_n^k)$
10 $v^k = \sum_{i=1}^n L_i^* y_i^k$
11 $\tau_k = \|u^k\|^2 + \|v^k\|^2$
12 **if** $\tau^k > 0$ **then**
13 Choose some $\rho_k : 0 < \rho_k < 2$
14 $\theta_k = \dfrac{\rho_k}{\tau_k} \max\left\{0, \sum_{i=1}^n \langle L_i z^k - x_i^k, y_i^k - w_i^k \rangle\right\}$
15 **else**
16 $\theta_k = 0$
17 $z^{k+1} = z^k - \theta_k v^k$
18 $w^{k+1} = w^k - \theta_k u^k$

- An integer $\hat{k} \geq 0$
- For each iteration $k \geq 0$, a subset $I_k \subseteq \{1, \ldots, n\}$
- For each iteration $k \geq 0$ and $i \in \{1, \ldots, n\}$, a positive scalar $\mu_{i,k}$
- For each iteration $k \geq 0$ and $i \in \{1, \ldots, n\}$, a "delay" value $d(i,k) \in \{0, \ldots, k\}$
- For each iteration $k \geq 0$, an overrelaxation parameter $\rho_k \in (0, 2)$.

These parameters are subject to some further restrictions that will be given in Assumption 5 below. The algorithm is stated in Algorithm 1. In line 5, we interpret $L_i z^{d(i,k)} + \mu_{i,d(i,k)} w_i^{d(i,k)}$ as being the desired or "target" value of the Prox operation, while e_i^k represents some kind of calculation error. This calculation error will be constrained in Assumption 5(5) below.

To give some background motivation, Algorithm 1 is separator-projection method for identifying a point in (2)'s *Kuhn-Tucker set*

$$Z \stackrel{\text{def}}{=} \left\{ (z, w_1, \ldots, w_n) \in \mathcal{H}_0 \times \cdots \times \mathcal{H}_n \;\middle|\; \sum_{i=1}^n L_i^* w_i = 0, (\forall\, i = 1, \ldots, n) : w_i \in T_i(z) \right\}, \quad (5)$$

which is a generalization of the "extended solution set" introduced in [10] for the case $\mathcal{H}_0 = \mathcal{H}_1 = \cdots = \mathcal{H}_n$ and all the L_i being the identity mapping on $\mathcal{H}_0$. It is also a special case of

the Kuhn-Tucker sets defined in [1, 3]. It is easily seen that $z \in \mathcal{H}_0$ solves (2) if and only if there exists $w_1 \in \mathcal{H}_1, \ldots, w_n \in \mathcal{H}_n$ such that $(z, w_1, \ldots, w_n) \in Z$.

Lemma 1. *Z is a closed convex set.*

Proof. This result is essentially a special case of [3, Proposition 2.8(i)]; see also [7]. In the notation of [3], we define maximal monotone operators $A : \mathcal{H}_0 \rightrightarrows \mathcal{H}_0$ and $B : \mathcal{H}_1 \times \cdots \times \mathcal{H}_n \rightrightarrows \mathcal{H}_1 \times \cdots \times \mathcal{H}_n$, along with a linear map $L : \mathcal{H}_0 \to \mathcal{H}_1 \times \cdots \times \mathcal{H}_n$, as follows:

$$\begin{aligned} A: &\qquad x \mapsto \{0\} \\ B: &\qquad (w_1, \ldots, w_n) \mapsto T_1(w_1) \times \cdots \times T_n(w_n) \\ L: &\qquad x \mapsto (L_1 x, \ldots, L_n x). \end{aligned}$$

Note that B is maximal monotone by the maximal monotonicity of the T_i. Next, [3, Proposition 2.8(i)] asserts that the set $\{(x, w) \mid Ax + L^* w \ni 0, B^{-1} w - Lx \ni 0\}$ is closed and convex. This set is in fact identical to Z, as we now show: first, using the definitions of A and L, we have that

$$Ax + L^* w \ni 0 \quad \Leftrightarrow \quad \{0\} + \sum_{i=1}^{n} L_i^* w_i \ni 0 \quad \Leftrightarrow \quad \sum_{i=1}^{n} L_i^* w_i = 0.$$

Second, using the definitions of B and L, we also have that

$$\begin{aligned} B^{-1} w - Lx \ni 0 \quad &\Leftrightarrow \quad (\forall\, i = 1, \ldots, n): \;\; T_i^{-1}(w_i) - L_i x \ni 0 \\ &\Leftrightarrow \quad (\forall\, i = 1, \ldots, n): \;\; w_i \in T_i(L_i x). \end{aligned}$$

Therefore, $Z = \{(x, w) \mid Ax + L^* w \ni 0, B^{-1} w - Lx \ni 0\}$, and is therefore must be closed and convex. □

We now analyze the convergence of Algorithm 1. It should be stressed that the algorithm is in essence a special case of the weakly convergent algorithm already proposed by the author and P. Combettes [7] in which (in the notation of that paper) one has $m = 1$, A_1 being the zero operator, and $\mathcal{K} = \mathcal{W}$. This paper provides its own analysis of the algorithm for the following reasons:

- Algorithm 1 has far less restrictive initialization conditions than the methods in [7]. Unfortunately, there appears to be no way to formally apply the results in [7] to algorithms with even slightly less restrictive initialization conditions, and there are number of applications in which looser initialization conditions should be beneficial. These applications include the one given in Section 4 below, as well as other anticipated future uses.

- In the less complicated setting used here, the basic structure of the convergence proof should be easier to comprehend than in [7]. This gives the proofs here some additional tutorial value.

- Approximate resolvent calculations (that is, $e_i^k \neq 0$) are fully integrated into the analysis here.

- One can prove by induction that an initialization-restricted version of Algorithm 1 is indeed a special case of one of the Algorithms in [7]. However, that proof is still quite lengthy, so the extra space consumed by giving a direct proof of Algorithm 1's convergence is only a few pages.

We begin by establishing some properties of inner product expressions of the kind found in line 14 of the algorithm:

Lemma 2. *Given $(x_i, y_i) \in T_i$ for $i = 1, \cdots, n$, define $c : \mathcal{H}_0 \times \mathcal{W} \to \mathbb{R}$ via*

$$c(z, w_1, \ldots, w_n) \stackrel{\text{def}}{=} \textstyle\sum_{i=1}^n \langle L_i z - x_i, y_i - w_i \rangle. \tag{6}$$

Then the following hold:

1. *c is an affine function on $\mathcal{H}_0 \times \mathcal{W}$*
2. *$\nabla c = \Big(\sum_{i=1}^n L_i^* y_i, \text{proj}_{\mathcal{W}}(x_1, \ldots, x_n) \Big)$.*
3. *$c(z, w_1, \ldots, w_n) \leq 0$ for all $(z, w_1, \ldots, w_n) \in Z$*
4. *$\nabla c = 0$ if and only if there exists $z \in \mathcal{H}_0$ such that $(z, y_1, \ldots, y_n) \in Z$ and $L_i z = x_i$ for all i.*

Proof. For the first claim, we note that for any $(z, w_1, \ldots, w_n) \in \mathcal{H}_0 \times \mathcal{W}$,

$$\begin{aligned}
c(z, w_1, \ldots, w_n) &= \textstyle\sum_{i=1}^n \langle L_i z - x_i, y_i - w_i \rangle \\
&= \textstyle\sum_{i=1}^n \langle L_i z, y_i - w_i \rangle - \sum_{i=1}^n \langle x_i, y_i \rangle + \sum_{i=1}^n \langle x_i, w_i \rangle \\
&= \textstyle\sum_{i=1}^n \langle z, L_i^* y_i - L_i^* w_i \rangle - \sum_{i=1}^n \langle x_i, y_i \rangle + \sum_{i=1}^n \langle x_i, w_i \rangle \\
&= \textstyle\langle z, \sum_{i=1}^n L_i^* y_i - \sum_{i=1}^n L_i^* w_i \rangle - \sum_{i=1}^n \langle x_i, y_i \rangle + \sum_{i=1}^n \langle x_i, w_i \rangle \\
&= \textstyle\langle z, \sum_{i=1}^n L_i^* y_i \rangle - \sum_{i=1}^n \langle x_i, y_i \rangle + \sum_{i=1}^n \langle x_i, w_i \rangle,
\end{aligned} \tag{7}$$

where the last equality holds because $(w_1, \ldots, w_n) \in \mathcal{W}$ means that $\sum_{i=1}^n L_i^* w_i = 0$. From (7), it is clear that c is affine on $\mathcal{H}_0 \times \mathcal{W}$. It is also clear from (7) that the z-component of c's gradient must be $\sum_{i=1}^n L_i^* w_i$. Letting $w = (w_1, \ldots, w_n)$ and $x = (x_1, \ldots, x_n)$, we further have

$$\begin{aligned}
c(z, w_1, \ldots, w_n) &= \textstyle\langle z, \sum_{i=1}^n L_i^* y_i \rangle - \sum_{i=1}^n \langle x_i, y_i \rangle + \langle x, w \rangle \\
&= \textstyle\langle z, \sum_{i=1}^n L_i^* y_i \rangle - \sum_{i=1}^n \langle x_i, y_i \rangle + \langle \text{proj}_{\mathcal{W}}(x), w \rangle,
\end{aligned}$$

where the second equality follows because $w \in \mathcal{W}$. Therefore, the w-component of ∇c is $\text{proj}_{\mathcal{W}}(x) \in \mathcal{W}$, which completes the proof of the second assertion.

Next, consider the third assertion. If $(z, w_1, \ldots, w_n) \in Z$, then we have $(L_i z, w_i) \in T_i$ for each i, and since $(x_i, y_i) \in T_i$, we have $\langle L_i z - x_i, w_i - y_i \rangle \geq 0$. Each term in the definition of c is just the negation of one of these inner products, so we conclude that $\varphi(z, w_1, \ldots, w_n) \leq 0$, proving the third assertion.

For the last assertion, let $L : \mathcal{H}_0 \to \mathcal{H}_1 \times \cdots \mathcal{H}_n$ denote the linear map $x \mapsto (L_1 x, \ldots, L_n x)$, as in the proof of Lemma 1. We note from the second assertion that if $\nabla c = 0$, then

$\sum_{i=1}^n L_i^* y_i = 0$ and $\operatorname{proj}_{\mathcal{W}}(x_1, \ldots, x_n) = 0$. If $\operatorname{proj}_{\mathcal{W}}(x_1, \ldots, x_n) = 0$, then $x \in \mathcal{W}^\perp = (\ker L^*)^\perp = \operatorname{im} L$. Therefore, x is of the form $(L_1 z, \ldots, L_n z)$ for some $z \in \mathcal{H}_0$. We then have $(L_i z, y_i) = (x_i, y_i) \in T_i$ for all i, while $\sum_{i=1}^n L_i^* y_i = 0$, which means that $(z, y_1, \ldots, y_n) \in Z$. □

To give a preview of the analysis, the basic idea behind Algorithm 1 is that $(x_i^k, y_i^k) \in \operatorname{Gph} T_i$ for all i and sufficiently large k, which means that Lemma 2(3) assures that the function

$$c_k : (z, w_1, \ldots, w_n) \mapsto \textstyle\sum_{i=1}^n \langle L_i z - x_i^k, y_i^k - w_i \rangle \tag{8}$$

provides a valid cut for the "target" set Z. Lines 9-18 compute $(z^{k+1}, w_1^{k+1}, \ldots, w_n^{k+1})$ as the projection, with an overrelaxation factor ρ_k, of $(z^k, w_1^k, \ldots, w_n^k)$ onto the halfspace $\{(v, w) \in \mathcal{H}_0 \times \mathcal{W} \mid c_k(v, w) \leq 0\}$. We now review the basic properties of recursions of this type. Generically, we will use an algorithm of the following form:

Algorithm 2 (Generic separator-projection algorithm)**.** *Let $\mathcal{P}$ be a real Hilbert space, let $p^0 \in \mathcal{P}$ be an arbitrary point, and suppose the subset $P^* \subset \mathcal{P}$ is nonempty, closed, and convex. For $k = 0, 1, \ldots$, iterate as follows:*

1. *Select some affine function $c_k : \mathcal{P} \to \mathbb{R}$ such that $c_k(p) \leq 0$ for all $p \in P^*$ and let H_k denote the closed convex set $\{p \mid c_k(p) \leq 0\}$ (which will be a halfspace unless $\nabla c_k = 0$).*
2. *For some $\rho_k \in (0, 2)$, compute $p^{k+1} = \rho_k \operatorname{proj}_{H_k}(p^k) + (1 - \rho_k) p^k$, where $\operatorname{proj}_{H_k}$ denotes orthogonal projection onto H_k with $\mathcal{P}$.*

Using the standard formula for orthogonal projection onto a halfspace, Algorithm 3 gives a more explicit description of exactly the same framework.

Algorithm 3: Generic separator-projection algorithm with explicit projection formula

```
1 Start with an p^0 ∈ P
2 for k = 0, 1, 2, ... do
3     Select an affine function c_k : P → R such that H_k ≝ {p ∈ P | c_k(p) ≤ 0} ⊇ P*
4     if ∇c_k ≠ 0 then
5         Pick some ρ_k ∈ (0, 2)
6         p^{k+1} = p^k − ρ_k ( max{0, c_k(p^k)} / ‖∇c_k‖^2 ) ∇c_k
7     else
8         p^{k+1} = p^k
```

Note that if $\nabla c_k = 0$ then we must have $H_k = \mathcal{P}$ (it may only be empty or all of $\mathcal{P}$, but the former option is excluded because we have assumed that $H_k \supset P^* \neq \emptyset$). In this case, $\operatorname{proj}_{H_k}(p^k) = p^k$ and so $p^{k+1} = p^k$, explaining the "null step" taken in line 8 when $\nabla c_k = 0$. If $\nabla c_k = 0$ is encountered in practice, a possibly preferable option might be to use Lemma 2(4) to attempt to "jump" immediately to a solution by identifying some $z^* \in \mathcal{H}_0$ such that $x_i^k = L_i z^*$ for all $i = 1, \ldots, n$. It is easily seen that such a z^* would necessarily solve (2).

The practicality of this approach depends on the linear operators L_i, and we avoid cluttering subsequent proofs with it, since it adds considerable complexity to the algorithm statements and we expect $\nabla c_k = 0$ to be a rare event in practice.

The following standard result summarizes some basic properties of Algorithms 2 and 3:

Proposition 3 (see *e.g.* [6, Proposition 2])**.** *The sequence $\{p^k\}$ generated by Algorithms 2 and 3 obeys the following inequality:*

$$(\forall\, p^* \in P^*)\ (\forall\, k \geq 0) \quad \left\|p^{k+1} - p^*\right\| \leq \left\|p^k - p^*\right\|^2 - \rho_k(2-\rho_k)\left\|p^{k+1} - p^k\right\|^2. \tag{9}$$

If there exist $\underline{\rho}, \overline{\rho} \in \mathbb{R}$ such that $0 < \underline{\rho} \leq \rho_k \leq \overline{\rho} < 2$ for all k, the following properties hold:

1. *$\{\|p^k - p^*\|\}$ is nonincreasing for all $p^* \in P^*$*
2. *$\{p^k\}$ is bounded*
3. *$p^k - p^{k+1} \to 0$*
4. *If all weak limit points of $\{p^k\}$ are in P^*, then $\{p^k\}$ converges weakly to a point in P^*.*

Using Lemma 2, we now establish that Algorithm 1 is essentially a special case of Algorithms 2/3.

Proposition 4. *In Algorithm 1 suppose that either*

$$\hat{k} = 0 \ \text{ and } \ (\exists\, \bar{k} \geq 0) : \ \left\{ i \in \{1,\dots,n\} \ \middle|\ (x_i^{-1}, y_i^{-1}) \in \operatorname{Gph} T_i \right\} \cup \left(\bigcup_{j=0}^{\bar{k}} I_j \right) = \{1,\dots,n\}, \tag{10}$$

or

$$(\exists\, \bar{k} \geq \hat{k}) \ : \ \left(\bigcup_{j=\hat{k}}^{\bar{k}} I_j \right) = \{1,\dots,n\}. \tag{11}$$

Then for all $k \geq \bar{k}$, each iteration of Algorithm 1 is an instance of the iteration of Algorithms 2 and 3 with $\mathcal{P} = \mathcal{H}_0 \times \mathcal{W}$, $P^* = Z$, and c_k selected according to (8)*, which yields the desired property that $c_k(z, w_1, \dots, w_m) \leq 0$ for all $(z, w_1, \dots, w_n) \in Z = P^*$.*

Proof. Examining lines 3-8 of Algorithm 1, one may observe for each $k \geq \hat{k}$ and $i = 1, \dots, n$, the pair (x_i^k, y_i^k) is updated only from the result of a Prox operation or by "recycling" its previous value. Therefore, once (x_i^k, y_i^k) is in $\operatorname{Gph} T_i$, it will remain in T_i for all subsequent iterations. For the case $\hat{k} = 0$, condition (10) means that for each i, either $(x_i^{-1}, y_i^{-1}) \in \operatorname{Gph} T_i$ to begin with, or (x_i^j, y_i^j) was set by the result of a Prox operation at some iteration $j \in \{0, \dots, \bar{k}\}$. This means that for iteration $\bar{k}$ and therefore all subsequent iterations, we have $(x_i^k, y_i^k) \in \operatorname{Gph} T_i$. For general $\hat{k}$, condition (11) guarantees that by iteration $\bar{k} \geq \hat{k}$, all the pairs (x_i^k, y_i^k) have been set by the result of a Prox operation and are thus in the corresponding $\operatorname{Gph} T_i$. In either case, it then follows immediately from Lemma 2(1, 3) that for all $k \geq \bar{k}$, the function c_k defined in (8) is affine on $\mathcal{H}_0 \times \mathcal{W}$ and $H_k \stackrel{\text{def}}{=} \{p \in \mathcal{P} \mid c_k(p) \leq 0\} \supseteq P^*$. Thus, with $P^* = Z$, this choice of c_k meets the selection criteria of Algorithms 2 and 3. From Lemma 2(2), it may then be readily confirmed that, for $\mathcal{P} = \mathcal{H}_0 \times \mathcal{W}$, the computations on lines 9-18 of Algorithm 1 implement the projection operation in lines 4-8 of Algorithm 3. □

For iterations $\bar{k}$ before (10) or (11) holds, $c_{\bar{k}}$ is not guaranteed to be a valid cut for Z, and Algorithm 1 may "wander" farther from this target set. But once (10) or (11) holds, Algorithm 1 will approach Z in a Fejér-monotone manner, as guaranteed by Proposition 3. If we require that $\hat{k} = 0$ and $I_0 = \{1, \ldots, n\}$, which is essentially the initialization condition of [7], or simply that $\hat{k} = 0$ and (x_i^0, y_i^0) be some arbitrary point in $\operatorname{Gph} T_i$ for all $i = 1, \ldots, n$, then (10) is satisfied for $\bar{k} = 0$ and thus Algorithm 1 becomes a special case of Algorithms 2/3 from the first iteration. These are the most natural ways to operate the algorithm, but to provide maximum flexibility for possible future applications that might behave differently, we allow for arbitrary initialization of the (x_i^0, y_i^0) and for the possibility that $\hat{k} > 0$. The latter may be useful in situations in which early iterations use only some approximation of the operators T_i.

As stated, Algorithm 1 is so general that it may easily "stall" without converging to a point in Z. We now state operating conditions under which we will prove that it converges weakly to a point in Z:

Assumption 5. *Algorithm 1 is operated in the following manner:*

1. *There exist $\underline{\rho}, \bar{\rho} \in]0, 2[$ such that $\underline{\rho} \leq \rho_k \leq \bar{\rho}$ for all k.*

2. *For some scalars $0 < \underline{\mu} \leq \bar{\mu}$, we have $\underline{\mu} \leq \mu_{ij} \leq \bar{\mu}$ for all $j \geq 0$ and $i = 1, \ldots, n$.*

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

$$(\forall\, j \geq 0) \quad : \quad \left(\bigcup_{k=j}^{j+M-1} I_k \right) = \{1, \ldots, n\}. \tag{12}$$

4. *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 $w_i^{d(i,k)}$ in line 5 may be out of date.*

5. *The error vectors e_i^k in the* Prox *calculations obey the following conditions for all i and k, for some fixed scalars $B \geq 0$ and $\sigma \in [0, 1[$:*

$$\left\| e_i^k \right\| \leq B \tag{13}$$

$$\langle e_i^k, L_i z^{d(i,k)} - x_i^k \rangle \geq -\sigma \left\| L_i z^{d(i,k)} - x_i^k \right\|^2 \tag{14}$$

$$\langle e_i^k, y_i^k - w_i^{d(i,k)} \rangle \leq \sigma \mu_{i,d(i,k)} \left\| y_i^k - w_i^{d(i,k)} \right\|^2. \tag{15}$$

Clearly, Assumption 5(5) holds if $e_i^k = 0$ for all $i = 1, \ldots, n$ and $k \geq 0$, that is, all the Prox operations are computed exactly. We will now proceed to analyze the convergence of Algorithm 1 under Assumption 5. The basic strategy is to show that all weak limit points of the sequence $\{p^k\} \stackrel{\text{def}}{=} \left\{(z^k, w^k)\right\} = \left\{(z^k, w_1^k, \ldots, w_n^k)\right\}$ must be in Z, after which we may

invoke Proposition 3(4) to obtain weak convergence. Since the analysis is fairly involved, it is broken up into a sequence of "building-block" lemmas. To begin with, define

$$(\forall\, i = 1, \ldots, n)\ (\forall\, k \geq 0): \qquad S(i,k) \stackrel{\text{def}}{=} \{j \in \mathbb{N} \mid\ j \leq k, i \in I_j\}$$
$$s(i,k) \stackrel{\text{def}}{=} \begin{cases} \max S(i,k), & \text{when } S(i,k) \neq \emptyset, \\ 0, & \text{when } S(i,k) = \emptyset. \end{cases}$$

In words, $s(i,k)$ is the most recent iteration up to and including k at which the index-i information in the separator was updated, or 0 if no index-i information has been processed by iteration k. Note that if $i \in I_k$, then $s(i,k) = k$. Part 3 of Assumption 5 guarantees that $0 \leq k - s(i,k) < M$. Next, we also define

$$\big(\forall\, i \in \{1, \ldots, n\} \backslash I_0\big): \qquad d(i,0) \stackrel{\text{def}}{=} -1$$
$$(\forall\, i = 1, \ldots, n)\ (\forall\, k \geq 0): \qquad \ell(i,k) \stackrel{\text{def}}{=} d\big(i, s(i,k)\big).$$

Lemma 6. *Under parts 3 and 4 of Assumption 5, we have $k - \ell(i,k) \leq M + D$ for all $i = 1, \ldots, n$ and $k \geq 1$.*

Proof. Fix any i and k. Part 3 of the assumption guarantees that $0 \leq k - s(i,k) \leq D$. Assumption 4 with k replaced by $s(i,k)$ implies that $s(i,k) - \ell\big(i, s(i,k)\big) \leq M$. Adding these two inequalities and using the definition of $\ell(i,k)$ yields the desired result. □

Lemma 7. *In Algorithm 1,*

$$(\forall\, i = 1, \ldots, n)\ (\forall\, k \geq 0): \qquad x_i^k = x_i^{s(i,k)} \qquad\qquad y_i^k = y_i^{s(i,k)}.$$

For each $i = 1, \ldots, n$, define $\hat{k}_i = \min\{k \geq \hat{k} \mid i \in I_k\}$. Then Assumption 5(3) implies $\hat{k}_i$ is finite for all $i = 1, \ldots, n$, and we have

$$(\forall\, i = 1, \ldots, n)\ (\forall\, k \geq \hat{k}_i): \qquad x_i^k + \mu_{i,\ell(i,k)} y_i^k = L_i z^{\ell(i,k)} + \mu_{i,\ell(i,k)} w_i^{\ell(i,k)} + e_i^{\ell(i,k)}. \tag{16}$$

Proof. The result follows immediately from the definitions of the Prox operation, $s(i,k)$, and $\ell(i,k)$, together with an examination of lines 3-8 of the algorithm. □

Lemma 8. *Parts 1 and 3 of Assumption 5 imply that the conclusions of Proposition 4 hold, that is, the iterations Algorithm 1 are a special case of those of Algorithms 2/3 for sufficiently large k. Furthermore, the following apply to $\{p^k\} = \big\{(z^k, w^k)\big\} = \big\{(z^k, w_1^k, \ldots, w_n^k)\big\}$:*

1. *$\{\|p^k - p^*\|\}$ is nonincreasing for all sufficiently large k and all $p^* \in Z$*
2. *$\{p^k\}$ is bounded*
3. *$p^k - p^{k+1} \to 0$*
4. *If all weak limit points of $\{p^k\}$ are in P^*, then $\{p^k\}$ converges weakly to a point in Z.*

Proof. Substituting $j = \hat{k}$ into (12), we conclude that (11) holds with $\bar{k} = \hat{k} + M$. Therefore the assumptions of Proposition 4 are met and its conclusions hold. Using also that part 1 of the assumption gives the desired bounds on $\{\rho_k\}$, we may apply the conclusions of Proposition 3 with $P^* = Z$ to $\{p^k\} = \left\{(z^k, w_1^k, \ldots, w_n^k)\right\}$ for sufficiently large k, proving the rest of the result. □

Lemma 9. *Suppose a solution to* (2) *exists and parts 1, 3, and 4 of Assumption 5 hold in Algorithm 1. Then*

$$(\forall\, i = 1, \ldots, n): \qquad z^k - z^{\ell(i,k)} \to 0 \qquad\qquad w_i^k - w_i^{\ell(i,k)} \to 0. \tag{17}$$

Proof. Fix any i and consider the first assertion, $z^k - z^{\ell(i,k)} \to 0$. Using a standard "telescoping" sequence, we have

$$\left\|z^k - z^{\ell(i,k)}\right\| = \left\| \sum_{j=1}^{k-\ell(i,k)} \left(z^{k-j+1} - z^{k-j}\right) \right\| \leq \sum_{j=1}^{k-\ell(i,k)} \left\|z^{k-j+1} - z^{k-j}\right\| \leq \sum_{j=1}^{M+D} \left\|z^{k-j+1} - z^{k-j}\right\|,$$

where in the last inequality we use the bound $k - \ell(i,k) \leq M + D$ from Lemma 6. Since part 1 of Assumption 5 guarantees that $0 < \underline{\rho} \leq \rho_k \leq \overline{\rho} < 2$ for all k, each term in the last summation above converges to zero as $k \to \infty$ because Proposition 3(3) assures us that $z^k - z^{k+1} \to 0$. Since the number of terms in the last expression is fixed, their sum converges to zero as well. The proof for each sequence $\{w_i^k - w_i^{\ell(i,k)}\}$ is essentially identical. □

Lemma 10. *Suppose a solution to* (2) *exists and parts 1, 2, 3, and 5 of Assumption 5 hold in Algorithm 1. Then the sequences* $\{z^k\}$, $\{w^k\} = \left\{(w_1^k, \ldots, w_n^k)\right\}$, $\{u^k\}$, $\{v^k\}$ $\{x^k\} = \left\{(x_1^k, \ldots, x_n^k)\right\}$ *and* $\{y^k\} = \left\{(y_1^k, \ldots, y_n^k)\right\}$ *are all bounded.*

Proof. By Lemma 8(2), the sequences $\{z^k\}$ and $\{w^k\}$ are bounded. Now, fix any i, and consider the right-hand side of (16), $L_i z^{\ell(i,k)} + \mu_{i,\ell(i,k)} w_i^{\ell(i,k)} + e_i^{s(i,k)}$. We now argue that this vector is bounded: first, since $\{z^k\}$ is bounded and the operator the linear operator L_i is continuous and hence bounded, $\{L_i z^{\ell(i,k)}\}$ is bounded. We also know that $\{w_i^k\}$ is bounded, and since there is an upper bound on $\mu_{i,\ell(i,k)}$ from part 2 of the assumption, we know that $\{\mu_{i,\ell(i,k)} w_i^{\ell(i,k)}\}$ is bounded. Finally, (13) from part 5 of the assumption guarantees that $\{e_i^{s(i,k)}\}$ is bounded. We conclude that the right-hand side of (16) is bounded, so its left-hand side $x_i^k + \mu_{i,\ell(i,k)} y_i^k$ is bounded. Next, fix any element $(\tilde{x}, \tilde{y}) \in T_i$. We then observe that for all sufficiently large k we have

$$\begin{aligned}
&\left\|x_i^k + \mu_{i,\ell(i,k)} y_i^k - \left(\tilde{x} + \mu_{i,\ell(i,k)} \tilde{y}\right)\right\|^2 \\
&\qquad = \left\|x_i^k - \tilde{x} + \mu_{i,\ell(i,k)} \left(y_i^k - \tilde{y}\right)\right\|^2 \\
&\qquad\qquad = \left\|x_i^k - \tilde{x}\right\|^2 + 2\mu_{i,\ell(i,k)} \langle x_i^k - \tilde{x}, y_i^k - \tilde{y}\rangle + \mu_{i,\ell(i,k)}^2 \left\|y_i^k - \tilde{y}\right\|^2 \\
&\qquad\qquad\qquad \geq \left\|x_i^k - \tilde{x}\right\|^2 + \mu_{i,\ell(i,k)}^2 \left\|y_i^k - \tilde{y}\right\|^2,
\end{aligned}$$

where the inequality follows from the monotonicity of T_i and $(x_i^k, y_i^k), (\tilde{x}, \tilde{y}) \in T_i$. Since $\{x_i^k + \mu_{i,\ell(i,k)} y_i^k\}$ is bounded and part 2 of the assumption bounds the value of $\mu_{i,\ell(i,k)}$, the

sequence at the beginning of the above chain of relations is bounded. It immediately follows that $\{x_i^k\}$ is bounded, and, since assumption 2 also asserts a lower bound on all $\mu_{i,\ell(i,k)}$, $\{y_i^k\}$ must also be bounded.

Since the above logic holds for all i, it immediately follows that $\{x^k\}$ and $\{y^k\}$ are bounded. Since $\text{proj}_{\mathcal{W}}$ is a projection map, it is nonexpansive, that is, $\|\text{proj}_{\mathcal{W}}(x)\| \leq \|x\|$ for any x, so the sequence $\{u^k\} = \{\text{proj}_{\mathcal{W}}(x^k\}$ must also be bounded. Also, since the linear operators L_i are contiuous and hence bounded, their adjoints L_i^* are also bounded, and therefore since all the $\{y_i^k\}$ are bounded, so is $\{v^k\} = \left\{\sum_{i=1}^n L_i^* y_i^k\right\}$. □

To make the notation in the remainder of the analysis more manageable, we now define, for all $k \geq 0$ and $i = 1, \ldots, n$,

$$\phi_{ik} \stackrel{\text{def}}{=} \langle L_i z^k - x_i^k, y_i^k - w_i^k \rangle \qquad \phi_k \stackrel{\text{def}}{=} \sum_{i=1}^n \phi_{ik} = c_k(p^k)$$

$$\psi_{ik} \stackrel{\text{def}}{=} \langle L_i z^{\ell(i,k)} - x_i^k, y_i^k - w_i^{\ell(i,k)} \rangle \qquad \psi_k \stackrel{\text{def}}{=} \sum_{i=1}^n \psi_{ik}.$$

Lemma 11. *Suppose that a solution to* (2) *exists and parts 1, 2, 3, and 5 of Assumption 5 hold in Algorithm 1. Then* $\limsup_{k\to\infty} \phi_k \leq 0$.

Proof. From lines 9-18 of Algorithm 1, we have for all $k \geq 0$ that

$$\begin{aligned} \left\|p^k - p^{k+1}\right\| &= \left\|(z^k, w^k) - (z^{k+1}, w^{k+1})\right\| \\ &= \theta_k \left\|(v^k, u^k)\right\| \\ &= \left(\frac{\rho_k}{\tau_k} \max\left\{0, \textstyle\sum_{i=1}^n \langle L_i z^k - x_i^k, y_i^k - w_i^k \rangle\right\}\right) \sqrt{\tau_k} \\ &= \frac{\rho_k \max\{0, \phi_k\}}{\sqrt{\tau_k}}. \end{aligned} \tag{18}$$

Lemma 8(3) states that $\left\|p^k - p^{k+1}\right\| \to 0$. Therefore, since Lemma 10 implies that the denominator $\sqrt{\tau_k} = \left\|(v^k, u^k)\right\|$ of (18) is bounded, its numerator must converge to 0. Since part 1 of the assumption states that $\rho_k \geq \underline{\rho} > 0$ for all k, we conclude $\max\{0, \phi_k\} \to 0$ and hence that $\limsup_{k\to\infty} \phi_k \leq 0$. □

Lemma 12. *Suppose that a solution to* (2) *exists and parts 1, 3, and 4 of Assumption 5 hold in Algorithm 1. Then* $\phi_{ik} - \psi_{ik} \to 0$ *for* $i = 1, \ldots, n$ *and* $\phi_k - \psi_k \to 0$.

Proof. Fixing any i and k, we calculate

$$\begin{aligned} \phi_{ik} - \psi_{ik} &= \langle L_i z^k - x_i^k, y_i^k - w_i^k \rangle - \langle L_i z^{\ell(i,k)} - x_i^k, y_i^k - w_i^{\ell(i,k)} \rangle \\ &= \langle L_i z^k - L_i z^{\ell(i,k)} + L_i z^{\ell(i,k)} - x_i^k, y_i^k - w_i^{\ell(i,k)} + w_i^{\ell(i,k)} - w_i^k \rangle \\ &\quad - \langle L_i z^{\ell(i,k)} - x_i^k, y_i^k - w_i^{\ell(i,k)} \rangle \\ &= \langle L_i z^k - L_i z^{\ell(i,k)}, y_i^k - w_i^{\ell(i,k)} \rangle + \langle L_i z^{\ell(i,k)} - x_i^k, w_i^{\ell(i,k)} - w_i^k \rangle \\ &\quad + \langle L_i z^k - L_i z^{\ell(i,k)}, w_i^{\ell(i,k)} - w_i^k \rangle. \end{aligned}$$

Consider the first term in this last expression. Lemma 9 asserts that $z^k - z^{\ell(i,k)} \to 0$, hence $L_i z^k - L_i z^{\ell(i,k)} \to 0$ because L_i is continous. On the other hand, Proposition 3 establishes that $\{w_i^k\}$ and hence $\{w_i^{\ell(i,k)}\}$ is bounded. Furthermore, Lemma 10 asserts that $\{y_i^k\}$ is bounded. Hence, $\langle L_i z^k - L_i z^{\ell(i,k)}, y_i^k - w_i^{\ell(i,k)} \rangle$ is the inner product of a vector that converges to 0 with another vector that is bounded. Thus, it converges to zero. A similar argument establishes that the second term, $\langle L_i z^{\ell(i,k)} - x_i^k, w_i^{\ell(i,k)} - w_i^k \rangle$, also converges to zero. Finally, the third term in the above expression, $\langle L_i z^k - L_i z^{\ell(i,k)}, w_i^{\ell(i,k)} - w_i^k \rangle$, must also converge to zero since both $L_i z^k - L_i z^{\ell(i,k)} \to 0$ and $w_i^k - w_i^{\ell(i,k)} \to 0$ by Lemma 9 and the continuity of L_i. Therefore, $\phi_{ik} - \psi_{ik} \to 0$. Summing this limit over $i = 1, \ldots, n$ yields $\phi_k - \psi_k \to 0$. □

The following result is essentially a special case of [1, Proposition 2.4], which in turn generalizes earlier results such as [2, Corollary 3]:

Lemma 13. *Given $(T_i, L_i)_{i=1,\ldots,n}$ as specified in problem (2), suppose there are sequences $\{z^k\} \subset \mathcal{H}_0$, $\big(\{(x_i^k, y_i^k)\} \subset \operatorname{Gph} T_i\big)_{i=1,\ldots,n}$ and points $z^\infty \in \mathcal{H}_0$ and $(w_i^\infty \in \mathcal{H}_i)_{i=1,\ldots,n}$ such that*

$$\begin{aligned} & \sum_{i=1}^n L_i^* y_i^k \to 0 && z^k \rightharpoonup z^\infty \\ (\forall\, i = 1, \ldots, n): \quad & L_i z^k - x_i^k \to 0 && y_i^k \rightharpoonup w_i^\infty, \end{aligned}$$

where $\rightharpoonup$ denotes weak convergence as $k \to \infty$. Then $(z^\infty, w_1^\infty, \ldots, w_n^\infty) \in Z$.

Proof. Define A, B, and L as in the proof Lemma 1. For all k, let t^k be the zero vector in $\mathcal{H}_0$, from which we immediately obtain that $(z^k, t^k) \in A$ for all k. Letting $x^k = (x_1^k, \ldots, x_n^k)$ and $y^k = (y_1^k, \ldots, y_n^k)$, the assumption that $(x_i^k, y_i^k) \in T_i$ for all i and k implies that $(x^k, y^k) \in B$ for all k. Furthermore, we have $t^k + L^* y^k = 0 + \sum_{i=1}^n L_i^* y_i^k \to 0$ due to the assumption that $\sum_{i=1}^n L_i^* y_i^k \to 0$, and furthermore $Lz^k - x^k = (L_1 z^k - x_1^k, \ldots, L_n z^k - x_n^k) \to 0$ because of the assumption that $L_i z^k - x_i^k \to 0$ for all i. Letting $w^\infty = (w_1^\infty, \ldots, w_n^\infty)$, we therefore have the situation that

$$\begin{array}{lll} \{(z^k, t^k)\} \subset A & z^k \rightharpoonup z^\infty & Lz^k - x^k \to 0 \\ \{(x^k, y^k)\} \subset B & y^k \rightharpoonup w^\infty & t^k + L^* y^k \to 0. \end{array}$$

After some notation transformations, we then conclude from [1, Proposition 2.4] that we must have $Az + L^* w \ni 0$ and $B^{-1} w - Lz \ni 0$. As in the proof of Lemma 1, these conditions are equivalent to $(z^\infty, w^\infty) = (z^\infty, w_i^\infty, \ldots, w_n^\infty) \in Z$. □

We now have the necessary ingredients to prove convergence of Algorithm 1.

Proposition 14. *Suppose that a solution to (2) exists and Assumption 5 holds in Algorithm 1. Then $\{z^k\}$ converges weakly to some solution z^∞ of (2), and for each $i = 1, \ldots, n$, the following hold:*

1. *$\{w_i^k\}$ converges weakly to some $w_i^\infty \in T_i(L_i z^\infty)$*
2. *$x_i^k \rightharpoonup L_i z^\infty$*

3. $y_i^k \rightharpoonup w_i^\infty$

Finally, we also have $\sum_{i=1}^n L_i^* w_i^\infty = 0$.

Proof. We begin by showing that all weak limit points of $\{p^k\} = \left\{(z^k, w_1^k, \ldots, w_n^k)\right\}$ are in Z. To do so, we combine the previously established results: first, combining Lemmas 11 and 12 yields $\limsup_{k\to\infty} \psi_k \leq 0$. Next, referring to Lemma 7, we may rearrange (16) into either of

$$y_i^k - w_i^{\ell(i,k)} = \frac{1}{\mu_{i,\ell(i,k)}} \left(L_i z^{\ell(i,k)} - x_i^k + e_i^{s(i,k)} \right) \tag{19}$$

$$L_i z^{\ell(i,k)} - x_i^k = \mu_{i,\ell(i,k)} \left(y_i^k - w_i^{\ell(i,k)} \right) - e_i^{s(i,k)}. \tag{20}$$

Recalling (14) from part 5 of the assumption, we next calculate

$$\begin{aligned}
\psi_{ik} &= \langle L_i z^{\ell(i,k)} - x_i^k, y_i^k - w_i^{\ell(i,k)} \rangle && \text{by definition} \\
&= \left\langle L_i z^{\ell(i,k)} - x_i^k, \tfrac{1}{\mu_{i,\ell(i,k)}} \left(L_i z^{\ell(i,k)} - x_i^k + e_i^{s(i,k)} \right) \right\rangle && \text{substituting (19)} \\
&= \tfrac{1}{\mu_{i,\ell(i,k)}} \left\| L_i z^{\ell(i,k)} - x_i^k \right\|^2 + \tfrac{1}{\mu_{i,\ell(i,k)}} \langle L_i z^{\ell(i,k)} - x_i^k, e_i^{s(i,k)} \rangle \\
&\geq \tfrac{1}{\mu_{i,\ell(i,k)}} \left\| L_i z^{\ell(i,k)} - x_i^k \right\|^2 - \tfrac{\sigma}{\mu_{i,\ell(i,k)}} \left\| L_i z^{\ell(i,k)} - x_i^k \right\|^2 && \text{by (14)} \\
&= \tfrac{1-\sigma}{\mu_{i,\ell(i,k)}} \left\| L_i z^{\ell(i,k)} - x_i^k \right\|^2. && (21)
\end{aligned}$$

Since $\sigma < 1$, we conclude that $\psi_{ik} \geq 0$ for all i and k. However, we have already established that $\limsup_{k\to\infty} \sum_{i=1}^n \psi_{ik} = \limsup_{k\to\infty} \psi_k \leq 0$, so we must have $\psi_{ik} \to 0$ for all i.

Using that $\sigma < 1$ and $\{\mu_{i,\ell(i,k)}\}$ is bounded above by part 2 of the assumption, we conclude from (21) and $\psi_{ik} \to 0$ that $L_i z^{\ell(i,k)} - x_i^k \to 0$ for all i. Adding this limit to $L_i z^k - L_i z^{\ell(i,k)} \to 0$, which we obtain from Lemma 9 and the continuity of L_i, we conclude that $L_i z^k - x_i^k \to 0$.

Next, we substitute (20) instead of (19) into the definition of ψ_{ik}, obtaining

$$\begin{aligned}
\psi_{ik} &= \langle L_i z^{\ell(i,k)} - x_i^k, y_i^k - w_i^{\ell(i,k)} \rangle \\
&= \left\langle \mu_{i,\ell(i,k)} \left(y_i^k - w_i^{\ell(i,k)} \right) - e_i^{s(i,k)}, y_i^k - w_i^{\ell(i,k)} \right\rangle \\
&= \mu_{i,\ell(i,k)} \left\| y_i^k - w_i^{\ell(i,k)} \right\|^2 - \langle e_i^{s(i,k)}, y_i^k - w_i^{\ell(i,k)} \rangle \\
&\geq \mu_{i,\ell(i,k)} \left\| y_i^k - w_i^{\ell(i,k)} \right\|^2 - \sigma \mu_{i,\ell(i,k)} \left\| y_i^k - w_i^{\ell(i,k)} \right\|^2 \\
&= (1-\sigma) \mu_{i,\ell(i,k)} \left\| y_i^k - w_i^{\ell(i,k)} \right\|^2, && (22)
\end{aligned}$$

where the inequality is a result of (15), also from part 5 of the assumption. Using $\sigma < 1$, the lower bound on $\{\mu_{i,\ell(i,k)}\}$ from part 2 of the assumption, and $\psi_{ik} \to 0$, we conclude that $y_i^k - w_i^{\ell(i,k)} \to 0$ for all i. Since we already established that $w_i^k - w_i^{\ell(i,k)} \to 0$ in Lemma 9, we may infer that $y_i^k - w_i^k \to 0$ for all i. From the continuity of the adjoint operators L_i^*, it follows that $L_i^* y_i^k - L_i^* w_i^k \to 0$ for all i. Adding $L_i^* y_i^k - L_i^* w_i^k \to 0$ for $i = 1, \ldots, n$

produces $\sum_{i=1}^n L_i^* y_i^k - \sum_{i=1}^n L_i^* w_i^k \to 0$. However, since we know $\sum_{i=1}^n L_i^* w_i^k = 0$ for all k since the iterates $p^k = (z^k, w^k)$ of Algorithm 1 are in $\mathcal{P} = \mathcal{H}_0 \times \mathcal{W}$, it immediately follows that $\sum_{i=1}^n L_i^* y_i^k \to 0$.

Now consider any such limit point $p^\infty = (z^\infty, w_1^\infty, \ldots, w_n^\infty)$ of $\{p^k\}$, along with some $\mathcal{K} \subseteq \mathbb{N}$ such that $(z^k, w_1^k, \ldots, w_n^k) \to_{\mathcal{K}} (z^\infty, w_1^\infty, \ldots, w_n^\infty)$. Since $y_i^k - w_i^k \to 0$ and $w_i^k \to_{\mathcal{K}} w_i^\infty$, we have $y_i^k \to_{\mathcal{K}} w_i^\infty$ for all i. Since we have already established that $L_i z^k - x_i^k \to 0$ for all i and $\sum_{i=1}^n L_i^* y_i^k \to 0$, we have

$$\begin{array}{lll} & \displaystyle\sum_{i=1}^n L_i^* y_i^k \to 0 & z^k \to_{\mathcal{K}} z^\infty \\ (\forall\, i = 1, \ldots, n): & L_i z^k - x_i^k \to 0 & y_i^k \to_{\mathcal{K}} w_i^\infty. \end{array}$$

Furthermore, we have $(x_i^k, y_i^k) \in T_i$ for all i and sufficiently large k. Therefore, applying Lemma 13 over the subsequence $\mathcal{K}$ with $(z, w_1, \ldots, w_n) = (z^\infty, w_1^\infty, \ldots, w_n^\infty)$, we conclude that we must have $p^\infty = (z^\infty, w_1^\infty, \ldots, w_n^\infty) \in Z$. Since the choice of weak limit point p^∞ was arbitrary, all limit points of $\{p^k\}$ are in Z.

In view of Lemma 8(4), we may now conclude that $\{p^k\}$ weakly converges to a point in Z, establishing all the claims except for the weak convergence of $\{x_i^k\}$ and $\{y_i^k\}$. These remaining convergence results follow immediately from the already proven claims because we have already established, for all $i = 1, \ldots, n$, that $L_i z^k - x_i^k \to 0$ and $y_i^k - w_i^k \to 0$. $\square$

3 One Approach to Asynchronous Implementation

We now present a specific class of asynchronous implementations of Algorithm 1, which we express as the coupled Algorithms 4 and 5 below. We model a partititioned-memory computing environment in which any data needed to represent the n operators T_i are partitioned into P nonempty "pools": specifically we let $\mathcal{S} = \{S_1, \ldots, S_P\}$ be a partition of $\{1, \ldots, n\}$, and suppose that for each $p = 1, \ldots, P$ and $i \in S_p$ the data describing T_i are stored in memory pool p. We further suppose that there is a set W of processing elements, which we call "workers", partitioned into nonempty subsets $W_1, \ldots, W_P$, one for each pool. The workers in W_p are capable of performing Prox operations on the operators T_i for which $i \in S_p$, and execute a "worker" loop as show in in Algorithm 5. This structure allows us to model the typical organization of current HPC systems, which consist of compute nodes each consisting of multiple processor cores sharing a local memory, with the nodes interconnected by a communication network. In a fully shared-memory environment, we could have $P = 1$ and $S_1 = \{1, \ldots, n\}$, that is, a single global pool of memory. At the other extreme, we could have $P = n$ and $S_p = \{p\}$ for $p = 1, \ldots, n$, that is, each T_i is stored in its own memory pool. In between these extremes, the pools could correspond to processing nodes on the communication network, each with multiple workers corresponding to individual processor cores (although some other cores might be reserved for coodination tasks).

While the "worker" process given in Algorithm 5 is straightforward, the "control" process embodied in Algorithm 4 is fairly complicated, so we provide some interpretation here: $\overline{W}_{k,p}$ is the set of idle workers in pool p at the outset of iteration k. The loop in lines 4-8 makes sure that each pool with an idle worker at iteration k assigns at least one of them to a Prox

Algorithm 4: A possible class of asynchronous implementations of Algorithm 1.

Start with arbitrary $z^0 \in \mathcal{H}$, $(w_1^0, \ldots, w_n^0) \in \mathcal{W}$ and $\left((x_i^{-1}, y_i^{-1}) \in \mathcal{H}_i^2\right)_{i=1,\ldots,n}$
for p=1,...,P **do** $\overline{W}_{p,0} = W_p$
for $k = 0, 1, 2, \ldots$ **do**
  foreach $p \in \{1, \ldots, P\}$ **do**
    if $\overline{W}_{p,k} \neq \emptyset$ **then**
      Select some nonempty $\overline{J}_{p,k} \subseteq \overline{W}_{p,k}$ **foreach** $j \in \overline{J}_{p,k}$ **do**
        Select $i_{k,p,j} \in S_p$ and $\mu_{k,p,j} \in [\underline{\mu}, \overline{\mu}]$
        Assign task $(i_{k,p,j}, \mu_{k,p,j}, z^k, w^k_{i_{k,p,j}})$ to worker j
    else
      $\overline{J}_{p,k} = \emptyset$
  wait for at least one worker signals a completed its task
  Let $C_k \neq \emptyset$ be the set of workers who have signaled completed tasks
  for $p = 1, \ldots, P$ **do**
    $\overline{W}_{p,k+1} = (\overline{W}_{p,k} \backslash \overline{J}_{p,k}) \cup (C_k \cap W_p)$
    foreach $j \in C_k \cap W_p$ **do**
      Let (i, x, y) be the data returned by worker j
      $x_i^k = x$
      $y_i^k = y$
    foreach value of $i \in S_p$ not encountered in the preceding loop **do**
      $x_i^k = x_i^{k-1}$
      $y_i^k = y_i^{k-1}$
  $v^k = \sum_{i=1}^n L_i^* y_i^k$
  $u^k = Q(x^k)$
  $\Delta_k = \left\|v^k\right\|^2 + \left\|u^k\right\|^2$
  if $\Delta_k > 0$ **then**
    Choose some $\rho_k \in [\underline{\rho}, \bar{\rho}]$
    $\theta_k = (\rho_k / \Delta_k) \sum_{i=1}^n \langle L_i z^k - x_i^k, y_i^k - w_i^k \rangle$
  else
    $\theta_k = 0$
  $z^{k+1} = z^k - \theta_k v^k$
  $w^{k+1} = w^k - \theta_k u^k$

Algorithm 5: Loop for each worker $j \in W_p$ controlled by Algorithm 4.

repeat indefinitely
  wait to be assigned a task (i, μ, z, w_i) (where $i \in S_p$)
  $(x, y) = \mathrm{Prox}_{T_i}^{\mu}(L_i z + \mu w_i)$
  Signal a completed task and return (i, x, y)

operation: $\overline{J}_{p,k} \subseteq \overline{W}_{k,p}$ is the set workers in pool p given such assignments. If $|S_p| \geq |W_p|$, one would ordinarily like to occupy all idle workers in a pool and thus set $\overline{J}_{p,k} = \overline{W}_{k,p}$. To prove convergence, however, it is only necessary for $\overline{J}_{p,k}$ to be nonempty whenever $\overline{W}_{k,p}$ is.

Line 11 then waits until at least one worker signals that it has completed a task and becomes idle. Line 14 then computes the new set of idle processors $\overline{W}_{p,k+1}$ in each pool p for the next iteration. This set must be nonempty for at least one p because line 11 waited for at least one worker to complete a task, meaning that $C_k \neq 0$. Therefore, at least one Prox task is assigned in each iteration k.

The pseudocode on lines 15-18 incorporates the pair $(x_i^k, y_i^k) \in \operatorname{Gph} T_i$ from each recently completed Prox operation into the separator calculations. If the algorithm is configured in such a way that two Prox operations involving the same operator T_i may be active simultaneously, it is conceivable that two or more such operations could complete at the same iteration k. In this case, we may consider the loop at lines 15-18 to select one of them arbitrarily, overwriting the values resulting from the others. In practice, however, it may not be desirable to operate the algorithm so that such a situation can occur. If such a situation does occur, it would seem natural to use the data from most recently initiated computation.

Lines 19-21 express the recycling of existing pairs for indices i that were not just encountered in lines 15-18. The algorithm is written with most of its variables indexed by k to facilitate expressing proofs, but in an actual computer implementation there would be no k indices and variable values would simply be periodically overwritten. With the algorithm expressed in this alternative form, lines 19-21 would be unnecessary.

Finally, the pseudocode on lines 22-31 performs the same separator-projection calculations as in lines 9-18 at the end of Algorithm 1.

Remark 1. For simplicity, Algorithm 4 is presented as if performed in serial by a single coordinating processor. However, depending on the calculations required to evaluate the linear projection $\operatorname{proj}_{\mathcal{W}}$, it should in many cases be possible to implement Algorithm 4 in a distributed manner. In this case, the memory required to describe the linear operators L_i could be partitioned in the same way as the memory for the T_i.

Remark 2. Once one or more workers become idle, the strategy of Algorithm 4 is to incorporate the results of their just-completed calculations into the iterates (z^{k+1}, w^{k+1}) before assigning more Prox tasks, in order to have newly assigned tasks use the most up-to-date information possible. One would expect this approach to be appropriate when the time for a worker to complete an assigned task is relatively large compared to that needed for the coordination calculations in lines 22-31, so that the worker idleness incurred by waiting for them is not signficant. In some applications, however, it may be desirable to consider alternative strategies that prioritize keeping workers busy and maximizing the overlap between worker and coordination calculations. Such a goal should be achievable by reorganizing the functional blocks of the algorithm.

Remark 3. For simplicity, we have formulated the "worker" algorithm (Algorithm 5) so that it computes Prox mappings exactly, which is equivalent to letting $e_i^k = 0$ for all encountered values of i and k in Algorithm 1. It is a straightforward generalization to modify the worker algorithm to compute resolvents inexactly, as in part 5 of Assumption 5.

We now proceed to analyze Algorithm 4 and give some conditions under which Algorithm 4 operates as a special case of Algorithm 1 conforming to the assumptions of Proposition 14.

Proposition 15. *Suppose Algorithm 4 is operated with parameters* $\underline{\mu}, \overline{\mu}, \underline{\rho}, \overline{\rho} \in \mathbb{R}$ *such that* $0 < \underline{\mu} \leq \overline{\mu}$ *and* $0 < \underline{\rho} \leq \overline{\rho} < 2$*, and under the following conditions:*

1. *There exists a positive integer* $\overline{M}$ *such that for every* $\overline{M}$ *executions of line 7 that have a given value of* p*, each element of* S_p *is selected to be* $i_{k,p,j}$ *at least once.*

2. *There are respective upper and lower bounds* $t_{\max} \geq t_{\min} > 0$ *on the elapsed time between assignment of a* Prox *task in line 8 and its completion.*

Then the sequences $\{z^k\}$*,* $\{w^k\}$*,* $\{x^k\}$*, and* $\{y^k\}$ *converge to a solution of* (2) *in the same manner as in Proposition 14.*

Proof. We first establish that all parts of Assumption 5 hold in this situation. The assumptions on the parameter bounds $\underline{\mu}, \overline{\mu}, \underline{\rho}, \overline{\rho}$ guarantee that parts 1 and 2 of Assumption 5 hold. Also, Algorithm 5 computes its Prox operations exactly and we make take $e_i^k = 0$ for all i and k, meaning that part 5 of the assumption holds trivially. To verify the remainder of Assumption 5, it therefore it suffices to prove two claims, letting I_k denote the set of values of i encountered by lines 15-18 of Algorithm 4 at iteration k:

- There is some nonnegative integer D such that the following holds: consider any iteration k, $i \in I_k$, and let p be such that $i \in S_p$. Then the return value (i, x, y) obtained in line 16 of Algorithm 5 is the result of some task assigned at some earlier iteration $d(i,k)$. Then $k - d(i,k) \leq D$. In other words, part 4 of the assumption holds.

- Part 3 of the assumption holds, that is, there exists a positive integer M such that for all $q \geq 1$, $\left(\bigcup_{k=q}^{q+M} I_k\right) = \{1, \ldots, n\}$.

We start with the first claim. Consider any iteration k and $i \in I_k$, initiated by assignment to some worker j at some earlier iteration $d(i,k)$. By hypothesis 2, this must have occurred at some time $t \leq t_{\max}$ before the detection of the finished task in line 11. Again, by hypothesis 2, the remaining $|W| - 1$ workers could each have completed at most $\lfloor t/t_{\min} \rfloor \leq \lfloor t_{\max}/t_{\min} \rfloor$ tasks during that time. If each completion event were detected in a separate execution of line 11, there could be at most $(|W|-1)\lfloor t_{\max}/t_{\min} \rfloor$ intervening iterations; if any executions of line 11 detect multiple completions, there will be fewer intervening iterations. It follows that the claim holds if we set $D = (|W|-1)\lfloor t_{\max}/t_{\min} \rfloor$.

We now consider the second claim. First, we call any iteration in which line 6 is executed for a given value of $\bar{p}$ of p an *activation of pool* $\bar{p}$. Since an activation of a given pool occurs as soon as it is detected to have an idle worker, we can use the same argument as used in proving the first claim to assert that the maximum possible number of iterations between any two activations of the same pool $\bar{p}$ is $D+1$ (the extra single iteration is necessary since activations occur at the beginning of each iteration, whereas detection of completed tasks happens after activation). Take any iteration $q \geq 0$ and index $i \in \{1, \ldots, n\}$, and let $\bar{p}$ be such that $i \in S_{\bar{p}}$. As just discussed, at most $D+1$ iterations after iteration q can be executed

until pool $\bar{p}$ is activated. Each activation corresponds to at least one execution of lines 7-8, and such executions can occur at most $\overline{M}$ times until index i is selected as $i_{k,p,j}$, according to hypothesis 1. Therefore, at most $(D+1)\overline{M}$ iterations beyond iteration q can occur before Prox task i is dispatched to some worker. This task will be completed and incorporated into some I_k within at most D additional iterations. Since the choices of q of i were arbitrary, the second claim holds with $M = (D+1)\overline{M} + D$.

We have now established that Algorithm 4 is as a special case of Algorithm 1 conforming to Assumption 5, meaning that Proposition 14 applies and its conclusions hold. □

4 An Asynchronous ADMM-Like Algorithm

We now return to problem (1). We will make the following standard assumption:

Assumption 16. *Problem* (1) *possesses at least one KKT point, that is, there exist* $\lambda^* \in \mathbb{R}^m$ *and* $(x_i^* \in \mathbb{R}^{n_i})_{i=1,\dots,n}$ *such that*

$$\sum_{i=1}^{n} M_i x_i^* = b \qquad\qquad (\forall\, i = 1, \dots, n) : -M_i^\top \lambda^* \in \partial f_i(x_i^*). \tag{23}$$

That (23) holds is sufficient for $(x_1^*, \dots, x_n^*)$ to solve (1) and λ^* to solve the dual problem

$$\min_{\lambda \in \mathbb{R}^m} \left(\sum_{i=1}^{n} f_i^*(-M_i^\top \lambda) \right) + \langle b, \lambda \rangle, \tag{24}$$

where f_i^* denotes the convex conjugate of f_i. Furthermore, Assumption 16 guarantees that the optimal values of (1) and (24) are equal. The dual problem (24) may be derived by manipulating (23) or by applying Lagrangian, Fenchel, or parametric duality. By selecting any $b_1, \dots, b_n \in \mathbb{R}^m$ such that $\sum_{i=1}^n b_i = b$, this dual may also be written

$$\min_{\lambda \in \mathbb{R}^m} \sum_{i=1}^{n} \big(f_i^*(-M_i^\top \lambda) + \langle b_i, \lambda \rangle \big). \tag{25}$$

Each term in this last sum may also be written

$$\begin{aligned} q_i(\lambda) = f_i^*(-M_i^\top \lambda) + \langle b_i, \lambda \rangle &= -\left(\inf_{x_i \in \mathbb{R}^{n_i}} \big\{ f_i(x_i) + \langle M_i^\top \lambda, x_i \rangle \big\} - \langle b_i, \lambda \rangle \right) \\ &= -\left(\inf_{x_i \in \mathbb{R}^{n_i}} \big\{ f_i(x_i) + \langle \lambda, M_i x_i - b_i \rangle \big\} \right), \end{aligned} \tag{26}$$

meaning that the dual problem may be expressed as

$$\min_{\lambda \in \mathbb{R}^m} \sum_{i=1}^{n} q_i(\lambda). \tag{27}$$

We also make the following blanket regularity assumption:

Assumption 17. *For $i = 1, \ldots, n$, the monotone operator $-M \circ \partial f^* \circ (-M^\top)$ is maximal.*

For each i, [16, Theorem 23.9] asserts that $\mathrm{Gph}\Big(\partial\big(f_i^* \circ (-M_i^\top)\big)\Big) \supseteq \mathrm{Gph}\,\big(-M_i \circ \partial f_i^* \circ (-M_i^\top)\big)$. The function $\partial f_i^* \circ (-M_i^\top)$ is convex, meaning that $\partial\big(f_i^* \circ (-M_i^\top)\big)$ is a monotone operator. Therefore Assumption 17 implies that $\partial\big(f_i^* \circ (-M_i^\top)\big) = -M_i \circ \partial f_i^* \circ (-M_i^\top)$. Invoking [16, Theorems 23.5 and 23.9], a sufficient condition to imply Assumption 17 is $(\ker M_i)^\perp \cap \mathrm{ri}\,\mathrm{rge}\,\partial f_i \neq \emptyset$ for $i = 1, \ldots, n$. This condition is trivially satisfied, for example, if $\ker M_i = \{0\}$.

We will attempt to solve (27) by applying Algorithms 4-5 with $\mathcal{H}_0 = \mathcal{H}_1 = \cdots = \mathcal{H}_n = \mathbb{R}^m$, and for all $i = 1, \ldots, n$, $L_i = \mathrm{Id}$ and $T_i = \partial q_i$. With this setting of the L_i, we have

$$\mathcal{W} = \left\{ (w_1, \cdots, w_n) \in (\mathbb{R}^m)^n \;\middle|\; \sum_{i=1}^n w_i = 0 \right\},$$

and the Kuhn-Tucker set Z of (5) is identical to the "extended solution set" introduced in [10]. This setting is an example of a situation in which the projection operator $\mathrm{proj}_{\mathcal{W}}$ is easily evaluated. Specifically, we have

$$\mathrm{proj}_{\mathcal{W}}(x_1, \ldots, x_n) = (x_1 - \bar{x}, \ldots, x_n - \bar{x}), \quad \text{where } \bar{x} = \frac{1}{n}\sum_{i=1}^n x_i. \tag{28}$$

The only further information needed to apply Algorithms 4-5 is an explicit way to compute the Prox operation in Algorithm 5 to the the subgradient mappings ∂q_i of the functions defined in (26). This information is contained in the following result:

Proposition 18. *For any $i \in \{1, \ldots, n\}$, let $T_i = \partial q_i$, where q_i is defined as in* (26)*, and consider any $t \in \mathbb{R}^m$. If $\hat{x}_i$ is a solution to the problem*

$$\min_{x_i \in \mathbb{R}^{n_i}} \big\{ f_i(x_i) + \langle t, M_i x_i - b_i \rangle + \tfrac{\mu}{2} \|M_i x_i - b_i\|^2 \big\}, \tag{29}$$

one has

$$\mathrm{Prox}_{T_i}^{\mu}(t) = \Big(t + \mu(M_i \hat{x}_i - b_i), b_i - M_i \hat{x}_i\Big). \tag{30}$$

If Assumption 17 holds, a solution to (29) *must exist for all $t \in \mathbb{R}^m$.*

Proof. In essence, the proof of the first claim dates back to [17]. However, it is stated explicitly (in slightly different notation) as an isolated result in [11, Proposition 10].

It remains to prove the second claim. Fix any $i = 1, \ldots, n$. Assumption 17 implies that $\partial\big(f_i^* \circ (-M_i^\top)\big) = (-M_i) \circ (\partial f_i^*) \circ (-M_i^\top)$, and hence that, for any $r \in \mathbb{R}^m$, all $s \in \partial q_i(r) = \partial\big(f_i^* \circ (-M_i^\top)\big)(r) + b_i$ must be of the form $b_i - M_i \hat{x}_i$, where $\hat{x}_i \in \partial f_i^*(-M_i^\top r)$. The last inclusion is equivalent to $-M_i^\top r \in \partial f_i(\hat{x}_i)$ and thus $0 \in \partial f_i(\hat{x}_i) + M_i^\top r$. Take any $t \in \mathbb{R}^m$ and let $(r, s) = \mathrm{Prox}_{T_i}^{\mu}(t) \in \mathrm{Gph}\, T_i$. We then have $r + \mu s = t$, which means $s = (1/\mu)(t - r)$. By the preceding discussion, there must also exist some $\hat{x}_i \in \mathbb{R}^{n_i}$ such that $s = b_i - M_i \hat{x}_i$. Therefore,

$$(1/\mu)(t - r) = b_i - M_i \hat{x}_i \qquad \Leftrightarrow \qquad r = t + \mu(M_i \hat{x}_i - b_i).$$

substituting this last expression into $0 \in \partial f_i(\hat{x}_i) + M_i^\top r$, we obtain

$$0 \in \partial f_i(\hat{x}_i) + M_i^\top\big(t + \mu(M_i \hat{x}_i - b_i)\big) = \partial f_i(\hat{x}_i) + M_i^\top t + \mu M_i^\top (M_i \hat{x}_i - b_i),$$

which is equivalent to $\hat{x}_i$ solving (29). □

If the regularity condition $(\ker M_i)^\perp \cap \operatorname{ri}\operatorname{rge}\partial f_i \neq \emptyset$ is violated, there may be values of t for which the calculation of $\operatorname{Prox}_{T_i}^{\mu}(t)$ is not straightforward, but such issues typically do not arise in practice.

Now consider what occurs for $T_i = \partial q_i$ and $L_i = \mathrm{Id}$ when a task tuple (i, μ, z, w_i) is passed into Algorithm 5. Since it is customary to use x for primal variables in problems such as (1), we will denote the returned pair in $\operatorname{Prox}_{T_i}^{\mu} \operatorname{Gph} T_i$ as (λ, y) instead of (x, y). To compute this pair, Proposition 18 indicates that we should find $\hat{x}_i$ such that

$$\hat{x}_i \in \operatorname*{Arg\,min}_{x_i \in \mathbb{R}^{n_i}} \left\{ f_i(x_i) + \langle z + \mu w_i, M_i x_i - b_i \rangle + \tfrac{\mu}{2} \left\| M_i x_i - b_i \right\|^2 \right\} \tag{31}$$

$$\Leftrightarrow \quad 0 \in \partial f_i(\hat{x}_i) + M^\top (z + \mu w_i) + \mu M^\top (M_i \hat{x}_i - b_i)$$

$$\Leftrightarrow \quad 0 \in \partial f_i(\hat{x}_i) + M^\top z + \mu M^\top \big(M_i \hat{x}_i - (b_i - w_i) \big)$$

$$\Leftrightarrow \quad \hat{x}_i \in \operatorname*{Arg\,min}_{x_i \in \mathbb{R}^{n_i}} \left\{ f_i(x_i) + \langle z, M_i x_i \rangle + \tfrac{\mu}{2} \left\| M_i x_i - (b_i - w_i) \right\|^2 \right\}. \tag{32}$$

Note that the b_i term may be dropped from the inner product term in (31), and the net effect of w_i is to shift the "target" value of $M_i x_i$ from b_i to $b_i - w_i$.

Again referring to Proposition 18, the returned pair $(\lambda, y) = \operatorname{Prox}_{T_i}^{\mu} \in \operatorname{Gph} T_i$ is given by

$$\begin{aligned} \lambda &= (z + \mu w_i) + \mu (M_i \hat{x}_i - b_i) \\ &= z + \mu \big(M_i \hat{x}_i - (b_i - w_i) \big) \end{aligned} \qquad\qquad y = b_i - M_i \hat{x}_i.$$

In view of this development and (28), Algorithms 4-5 specialized to the setting $\mathcal{H}_0 = \mathcal{H}_1 = \cdots = \mathcal{H}_n = \mathbb{R}^m$, $L_i = \mathrm{Id}$, $T_i = \partial q_i$ take the form given in Algorithms 6-7. Note that to avoid notation conflicts, we relabel the variables x_i^k of Algorithm 4 to λ_i^k. These variables are dual variables with respect to the problem (1) because we are applying Algorithms 4 to (1)'s dual formulation (27). To recover estimates of the optimal solution to the original problem (1), the "worker" operation in Algorithm 7 also returns the value of the minimizer $\hat{x}_i$, which the "master" process in Algorithm 6 tracks as the sequences $\{x_i^k\}_{k \in \mathbb{N}}$ for $i = 1, \ldots, n$ (the original x_i^k having been relabeled λ_i^k).

It is easily seen by induction that Algorithm 6 has $y_i^k = b_i - M_i x_i^k$ for all $i = 1, \ldots, n$ and sufficient large k. If the operators M_i are easy to evaluate, it would therefore be possible to drop the sequences $\{y_i^k\}$ from the implementation of the algorithm and substitute the corresponding $b_i - M_i x_i^k$ wherever y_i^k appears. Using this substitution, the calculation of v^k in line 20 may be expressed as

$$v^k = \sum_{i=1}^n y_i^k = \sum_{i=1}^n (b_i - M_i x_i^k) = b - \sum_{i=1}^n M_i x_i^k, \tag{33}$$

that is, v^k is simply the overall constraint violation vector. Using this relation, the update to the (dual) solution estimate z^k in line 29 may therefore be written

$$z^{k+1} = z^k - \theta_k \left(b - \sum_{i=1}^n M_i x_i^k \right) = z^k + \theta_k \left(\sum_{i=1}^n M_i x_i^k - b \right),$$

which is the standard multiplier update of the augmented Lagrangian method and ADMM, but with θ_k playing the role of the stepsize. The role of the w_i^k and the associated step

Algorithm 6: Asynchronous algorithm for (1) resembling the n-block ADMM.

Start with arbitrary $z^0, w_1^0, \ldots, w_n^0, \lambda_1^{-1}, \ldots, \lambda_n^{-1}, y_1^{-1}, \ldots, y_n^{-1} \in \mathbb{R}^m$, with $\sum_{i=1}^n w_i^0 = 0$
for $p = 1, \ldots, P$ **do** $\overline{W}_{p,0} = W_p$
for $k = 0, 1, 2, \ldots$ **do**
  foreach $p \in \{1, \ldots, P\}$ **do**
    if $\overline{W}_{p,k} \neq \emptyset$ **then**
      Select some nonempty $\overline{J}_{p,k} \subseteq \overline{W}_{p,k}$ **foreach** $j \in \overline{J}_{p,k}$ **do**
        Select $i_{k,p,j} \in S_p$ and $\mu_{k,p,j} \in [\underline{\mu}, \overline{\mu}]$
        Assign task $(i_{k,p,j}, \mu_{k,p,j}, z^k, w^k_{i_{k,p,j}})$ to worker j
    else
      $\overline{J}_{p,k} = \emptyset$
  wait for at least one worker signals a completed its task
  Let $C_k \neq \emptyset$ be the set of workers who have signaled completed tasks
  for $p = 1, \ldots, P$ **do**
    $\overline{W}_{p,k+1} = (\overline{W}_{p,k} \backslash \overline{J}_{p,k}) \cup (C_k \cap W_p)$
    foreach $j \in C_k \cap W_p$ **do**
      Let (i, x, λ, y) be the data returned by worker j
      $(x_i^k, \lambda_i^k, y_i^k) = (x, \lambda, y)$
    foreach value of $i \in S_p$ not encountered in the preceding loop **do**
      $(x_i^k, \lambda_i^k, y_i^k) = (x_i^{k-1}, \lambda_i^{k-1}, y_i^{k-1})$
  $v^k = \sum_{i=1}^n y_i^k$
  $\bar{\lambda}^k = \frac{1}{n} \sum_{i=1}^n \lambda_i^k$
  for $i = 1, \ldots, n$ **do** $u_i^k = \lambda_i^k - \bar{\lambda}^k$
  $\Delta_k = \left\|v^k\right\|^2 + \sum_{i=1}^n \left\|u_i^k\right\|^2$
  if $\Delta_k > 0$ **then**
    Choose some $\rho_k \in [\underline{\rho}, \bar{\rho}]$
    $\theta_k = (\rho_k / \Delta_k) \sum_{i=1}^n \langle z^k - \lambda_i^k, y_i^k - w_i^k \rangle$
  else
    $\theta_k = 0$
  $z^{k+1} = z^k - \theta_k v^k$
  for $i = 1, \ldots, n$ **do** $w_i^{k+1} = w_i^k - \theta_k u_i^k$

Algorithm 7: Loop for each worker $j \in W_p$ controlled by Algorithm 6.

repeat indefinitely
  wait to be assigned a task (i, μ, z, w_i) (where $i \in S_p$)
  Find $\hat{x}_i \in \operatorname{Arg\,min}_{x_i \in \mathbb{R}^{n_i}} \left\{ f_i(x_i) + \langle z, M_i x_i \rangle + \frac{\mu}{2} \left\| M_i x_i - (b_i - w_i) \right\|^2 \right\}$
  $\lambda = z + \mu\big(M_i \hat{x}_i - (b_i - w_i)\big)$
  $y = b_i - M_i \hat{x}_i$
  Signal a completed task and return $(i, \hat{x}_i, \lambda, y)$

directions u_i^k has a less familiar character; as mentioned above, the function of w_i^k is to make $b_i - w_i^k$ a "target" value for $M_i x_i$. The tentative new multipliers λ_i^k are calculated by looking at only one block of variables, and the adjustments u_i^k to the w_i^k are based on how much each tentative new multiplier varies from their average (over i).

We now state a convergence result for Algorithms 6-7, most of which, in view of the development above, follows essentially immediately from Proposition 15.

Proposition 19. *Let Assumption 16 hold, and suppose Algorithm 6 is operated with parameters $\underline{\mu}, \overline{\mu}, \underline{\rho}, \overline{\rho} \in \mathbb{R}$ such that $0 < \underline{\mu} \leq \overline{\mu}$ and $0 < \underline{\rho} \leq \overline{\rho} < 2$, and under the following conditions:*

1. *There exists a positive integer $\overline{M}$ such that for every $\overline{M}$ executions of line 7 that have a given value of p, each element of S_p is selected to be $i_{k,p,j}$ at least once.*
2. *There are respective upper and lower bounds $t_{\max} \geq t_{\min} > 0$ on the elapsed time between assignment of a block minimization task in line 8 and its completion.*

Then the sequences $\{z^k\}, \{\lambda_1^k\}, \ldots, \{\lambda_n^k\}$ generated by Algorithms 6-7 all converge to same limit z^*, which is an optimal solution of the dual problem (24) of (1). The primal sequence $\{(x_1^k, \ldots, x_n^k)\}$ generated by the algorithms is asymptotically feasible for (1) in the sense that $\lim_{k\to\infty} \left\{\sum_{i=1}^n M_i x_i^k\right\} = b$. If Assumption 17 also holds, then the minimization operation in Algorithm 7 always has a solution, the sequence $\{(x_1^k, \ldots, x_n^k)\}$ is asymptotically optimal for (1) in the sense that $\limsup_{k\to\infty} \{\sum_{i=1}^n f_i(x_i)\} \leq \zeta^*$, where ζ^* denotes the optimal value of (1), and all limit points of $\{(x_1^k, \ldots, x_n^k)\}$ are optimal solutions of (1).

Proof. The preceding discussion has already established that Algorithms 6-7 are equivalent to Algorithms 4-5 applied to the situation $\mathcal{H}_0 = \mathbb{R}^m$ and, for all $i = 1, \ldots, n$, $\mathcal{H}_i = \mathbb{R}^m$, $L_i = \mathrm{Id}$, and $T_i = \partial q_i$, where q_i is as defined in (26). Assumption 16 implies that the dual problem (24) has a solution, and therefore that a solution to the inclusion (2) exists. Proposition 15 then implies, since weak and strong convergence are identical in finite dimension, that $\{z^k\}$ converges to some optimal solution z^* of the dual problem, and that $\{p_1^k\}, \ldots, \{p_n^k\}$ all converge to the same limit. This establishes the first claim. We further know from Proposition 15 that $v^k = \sum_{i=1}^n L_i^* y_i^k \to 0$. In this particular setting, we have from (33) that $v^k = b - \sum_{i=1}^n M_i x_i^k$, so $v^k \to 0$ is equivalent to the claimed asymptotic feasibility.

Now suppose that Assumption 17 holds in addition to Assumtion 16. Consider the sequences $\{w_i^k\}$ and $\{y_i^k\} = \{b_i - M_i x_i^k\}$ generated by Algorithm 6. Proposition 15 guarantees that for each $i = 1, \ldots, n$, $\{w_i^k\}$ and $\{y_i^k\}$ converge to a common limit $w_i^* \in T_i(z^*)$ such that $\sum_{i=1}^n w_i^* = 0$. By Assumption 17, $T_i = \partial q_i = \big(-M_i \circ \partial f_i^* \circ (-M_i^\top)\big) + b_i$. Therefore there must exist some $x_i^* \in \partial f_i^*(-M_i^\top z^*)$ such that $w_i^* = b_i - M_i x_i^*$. Since f_i is a closed proper convex function, having $x_i^* \in \partial f_i^*(-M_i^\top z^*)$ is equivalent to $-M_i^\top z^* \in \partial f_i(x_i^*)$. It follows that $(\lambda^*, x_1^*, \ldots, x_n^*)$ satify the KKT conditions (23), $(x_1^*, \ldots, x_n^*)$ is an optimal solution of (1), and $\sum_{i=1}^n f_i(x_i^*) = \zeta^*$.

For each $k \geq 0$ and $i = 1, \ldots, n$, let $\ell(i,k)$ denote the iteration at which task resulting in the most recently incorporated tuple for subsystem i was initiated, as in the analysis in Section 2. Then we have, for all $i = 1, \ldots, n$ and all sufficiently large k that

$$x_i^k = x_i^{\ell(i,k)} \in \operatorname*{Arg\,min}_{x_i \in \mathbb{R}^{n_i}} \left\{ f_i(x_i) + M_i^\top z^{\ell(i,k)} + \frac{\mu_{i,\ell(i,k)}}{2} \left\| M_i x_i - b_i + w_i^{\ell(i,k)} \right\|^2 \right\}.$$

Therefore, for all $i = 1, \ldots, n$ and all sufficiently large k we have

$$f_i(x_i^k) + M_i^\top z^{\ell(i,k)} + \frac{\mu_{\ell(i,k)}}{2} \left\| M_i x_i^k - b_i + w_i^{\ell(i,k)} \right\|^2 \\ \leq f_i(x_i^*) + M_i^\top z^{\ell(i,k)} + \frac{\mu_{\ell(i,k)}}{2} \left\| M_i x_i^* - b_i + w_i^{\ell(i,k)} \right\|^2.$$

Define M and D as in the proof of Proposition 15, which means that parts 3 and 4 of Assumption 5 hold and Lemma 6 guarantees that $\ell(i,k) \geq k - (M + D)$. It then follows that $\lim_{k\to\infty} z^{\ell(i,k)} = \lim_{j\to\infty} z^j = z^*$, and similarly $\lim_{k\to\infty} w_i^{\ell(i,k)} = \lim_{j\to\infty} w_i^j = w_i^*$ for $i = 1, \ldots, n$. Since $M_i x_i^* - b_i = -w_i^*$ by construction, $M_i x_i^k - b_i = -y_i^k \to -w_i^*$, and the parameters μ_k are bounded above, the quadratic terms on both sides of the above inequality vanish in the limit. Taking limits, we obtain

$$(\forall\, i = 1 \ldots, n) \qquad \limsup_{k\to\infty} f_i(x_i^k) + M_i^\top z^* \leq f_i(x_i^*) + M_i^\top z^*.$$

Cancelling the identical terms $M_i^\top z^*$ from both sides and summing over $i = 1, \ldots, n$, we obtain

$$\limsup_{k\to\infty} \left\{ \sum_{i=1}^n f_i(x_i^k) \right\} \leq \sum_{i=1}^n f_i(x_i^*) = \zeta^*, \tag{34}$$

and the asymptotic optimality claim is established. To establish the last claim, consider any limit point $(x_1^\infty, \ldots, x_n^\infty)$ of $\{(x_1^k, \ldots, x_n^k)\}$. Taking limits over some sequence of indices $\mathcal{J}$ such that $(x_1^k, \ldots, x_n^k) \to_{\mathcal{J}} (x_1^\infty, \ldots, x_n^\infty)$, we obtain

$$\sum_{i=1}^n f_i(x_i^\infty) \leq \liminf_{k\to\infty,\, k\in\mathcal{J}} \left\{ \sum_{i=1}^n f_i(x_i^k) \right\} \leq \limsup_{k\to\infty,\, k\in\mathcal{J}} \left\{ \sum_{i=1}^n f_i(x_i^k) \right\} \leq \limsup_{k\to\infty} \left\{ \sum_{i=1}^n f_i(x_i^k) \right\} \leq \zeta^*,$$

where the first inequality follows from f_i being closed and thus lower semicontinuous, and the last inequality comes from (34). But since $\lim_{k\to\infty} \left\{ \sum_{i=1}^n M_i x_i^k \right\} = b$, we must have $\sum_{i=1}^n M_i x_i^\infty = b$, which means that $(x_1^\infty, \ldots, x_n^\infty)$ is feasible for (1) and furthermore that $\sum_{i=1}^n f_i(x_i^\infty) \geq \zeta^*$, since ζ^* is the optimal value of (1). Therefore, $\sum_{i=1}^n f_i(x_i^\infty) = \zeta^*$ and $(x_1^\infty, \ldots, x_n^\infty)$ is an optimal solution to (1). □

Remark 4. Remark 1, referring to parallelizing the "control" algorithm as well as the distribution of subproblem calculations, applies equally to Algrithm 6 as it does to Algorithm 4.

Remark 5. Operators T_i whose Prox operations are very quickly computed, for example in time linear in the subproblem dimension n_i, might most profitably be processed in every iteration and essentially considered part of the coordination process rather than dispatched as asynchronously processed subproblems.

Remark 6. Because Algorithms 6-7 are based on a block-iterative projective splitting framework in the family of [9, 10, 1, 7], the penalty parameters $\mu_{i,k}$ need not be fixed with respect to either the subsysem i or the iteration k, but only bounded above and below. This is a novel property for an ADMM-like algorithm.

References

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

[2] H. H. Bauschke. A note on the paper by Eckstein and Svaiter on "General projective splitting methods for sums of maximal monotone operators". *SIAM J. Control Optim.*, 48(4):2513–2515, 2009.

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

[4] T. H. Chang, M. Hong, W. C. Liao, and X. Wang. Asynchronous distributed ADMM for large-scale optimization — part I: Algorithm and convergence analysis. *IEEE Trans. Signal Proc.*, 64(12):3118–3130, 2016.

[5] C. Chen, R. Sun, and Y. Ye. On convergence of the multi-block alternating direction method of multipliers. In *Proceedings of the 8th International Congress on Industrial and Applied Mathematics*, pages 3–15. Higher Ed. Press, Beijing, 2015.

[6] P. L. Combettes. Féjer monotonicity in convex optimization. In C. A. Floudas and P. M. Pardalos, editors, *Encyclopedia of Optimization*, pages 106–114. Springer-Verlag, 2001.

[7] P. L. Combettes and J. Eckstein. Asynchronous block-iterative primal-dual decomposition methods for monotone inclusions. Preprint 1507.03291, ArXiv, 2015. Accepted to *Mathematical Programming*.

[8] J. Eckstein and D. P. Bertsekas. On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Math. Program.*, 55(3):293–318, 1992.

[9] J. Eckstein and B. F. Svaiter. A family of projective splitting methods for the sum of two maximal monotone operators. *Math. Program.*, 111(1-2, Ser. B):173–199, 2008.

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

[11] J. Eckstein and W. Yao. Understanding the convergence of the alternating direction method of multipliers: theoretical and computational perspectives. *Pac. J. Optim.*, 11(4):619–644, 2015.

[12] M. Fortin and R. Glowinski. On decomposition-coordination methods using an augmented Lagrangian. In M. Fortin and R. Glowinski, editors, *Augmented Lagrangian methods: Applications to the numerical solution of boundary-value problems*, volume 15 of *Studies in Mathematics and its Applications*, pages 97–146. North-Holland Publishing Co., Amsterdam, 1983.

[13] D. Gabay. Applications of the method of multipliers to variational inequalities. In M. Fortin and R. Glowinski, editors, *Augmented Lagrangian Methods: Applications to the Solution of Boundary Value Problems*, chapter IX, pages 299–340. North-Holland, Amsterdam, 1983.

[14] F. Iutzeler, P. Bianchi, P. Ciblat, and W. Hachem. Asynchronous distributed optimization using a randomized alternating direction method of multipliers. In A. Astolfi, editor, *52nd IEEE Conference on Decision and Control*, pages 3671–3676, 2013.

[15] J. F. C. Mota, J. M. F. Xavier, P. M. Q. Aguiar, and M. P uschel. D-ADMM: A communication-efficient distributed algorithm for separable optimization. *IEEE Transactions on Signal Processing*, 61(10):2718–2723, 2013.

[16] R. T. Rockafellar. *Convex analysis*. Princeton Mathematical Series, No. 28. Princeton University Press,, Princeton, N.J., 1970.

[17] R. T. Rockafellar. Augmented Lagrangians and applications of the proximal point algorithm in convex programming. *Math. Oper. Res.*, 1(2):97–116, 1976.

[18] E. Wei and A. Ozdaglar. On the $O(1/k)$ convergence of asynchronous distributed alternating direction method of multipliers. In A. Tewfik, editor, *2013 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 551–554, 2013.

[19] R. Zhang and J. Kwok. Asynchronous distributed ADMM for consensus optimization. In T. Jebara and E. P. Xing, editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1701–1709, 2014.