

## RESEARCH ARTICLE

# Distributed Solutions for Loosely Coupled Feasibility Problems Using Proximal Splitting Methods\*

Sina Khoshfetrat Pakazad<sup>1</sup>, Martin S. Andersen<sup>2</sup> and Anders Hansson<sup>1</sup>

(Received 00 Month 200x; in final form 00 Month 200x)

In this paper, we consider convex feasibility problems where the underlying sets are loosely coupled, and we propose several algorithms to solve such problems in a distributed manner. These algorithms are obtained by applying proximal splitting methods to convex minimization reformulations of convex feasibility problems. We also put forth distributed convergence tests which enable us to establish feasibility or infeasibility of the problem distributedly, and we provide convergence rate results. Under the assumption that the problem is feasible and boundedly linearly regular, these convergence results are given in terms of the distance of the iterates to the feasible set, which are similar to those of classical projection methods. In case the feasibility problem is infeasible, we provide convergence rate results that concern the convergence of certain error-bounds.

**Keywords:** feasible/infeasible convex feasibility problems; proximal splitting; distributed solution; flow feasibility problem

**AMS Subject Classification:** G.1.6; G.1.10; G.2.2; I.1.2

## 1. Introduction

A convex feasibility problem (CFP), corresponds to the problem of finding an element in the intersection of, say,  $N$  non-empty convex sets  $(\mathcal{C}_i)$ , i.e.,  $x \in \bigcap_{i=1}^N \mathcal{C}_i$ . Such problems appear in many fields of engineering and science, e.g., image reconstruction, best approximation theory, analysis of networked systems [3, 26, 28], and have been studied thoroughly in the literature, e.g., see [1–3, 6, 25]. Many of the algorithms designed for solving CFPs, rely on projections onto the individual sets, and are referred to as projection methods. The behavior and convergence properties of such algorithms have been well-studied, [1–3, 6], both when the CFP is feasible or infeasible. For a thorough survey of such algorithms refer to [3]. In this paper, we focus on CFPs where each constraint defining a set  $\mathcal{C}_i$  in the problem is only dependent on a subset of components of a variable,  $x$ . We also assume that the number of variables that appear jointly in the descriptions of every two constraint sets  $\mathcal{C}_i$  and  $\mathcal{C}_j$  ( $i \neq j$ ) is small. We refer to such CFPs as loosely coupled, and we intend to develop distributed algorithms for solving these problems efficiently. Employing classical projection algorithms for solving such CFPs, while neglecting the underlying structure in the problem, is inefficient and has been shown to be extremely slow, [28]. In order to boost the performance of these algorithms, the structure in the CFP needs to be exploited. This can be done by using ideas from [10], [9] and [29], and it results in a reformulation of the problem as an equivalent feasibility problem in product space. Similar formulations of CFPs are also proposed in [17, 34].

\*This work has been supported by the Swedish Department of Education within the ELLIIT project.

<sup>1</sup> Sina Khoshfetrat Pakazad and Anders Hansson are with the Division of Automatic Control, Department of Electrical Engineering, Linköping University, Sweden. Email: {sina.kh.pa, hansson}@isy.liu.se.

<sup>2</sup> Martin S. Andersen is with the Department of Applied Mathematics and Computer Science, Technical University of Denmark. Email: mskan@dtu.dk.

This product-space formulation can be solved using well-known projection methods, e.g., von Neumann's and Dykstra's alternating projections. Although such algorithms have been shown to be effective and their convergence properties are well-studied, to the best knowledge of the authors, rate of convergence of such methods are mainly available when the CFP is feasible, [6]. Hence, in order to provide convergence rate results for projection-based methods even when the problem is infeasible, we approach the problem in another way.

The product-space reformulation of loosely coupled CFPs can also be written as a convex minimization problem, [6, 11, 12, 18]. Authors in [6], define a convex minimization reformulation of CFPs and consider the use of gradient projection algorithms (GPA) for solving minimization problem. They then utilize the convergence properties of GPA to provide convergence results for the resulting algorithm. Furthermore, they show that in case the problem is infeasible, certain error bounds converge to non-zero values with  $\mathcal{O}(1/\sqrt{k})$  rate of convergence. Prior to [6], such approaches have also been used for similar purposes, using the so-called subgradient methods, [2, 13, 35, 36]. In this paper we also deal with convex minimization reformulations of CFPs. This in turn allows us to employ proximal splitting methods (first order methods) for solving minimization reformulations of CFPs, which result in several distributed projection-based algorithms for solving loosely coupled CFPs. Some of these algorithms are similar to already existing well-known projection methods. However, several of the proposed algorithms can be considered as generalizations of classical projection methods, [12]. Furthermore, if the minimization formulation of a CFP is well-defined even when it is infeasible, the convergence properties, and particularly the rate of convergence of the utilized proximal methods can be used to establish convergence results for the newly generated algorithms. This allows us to provide rate of convergence results for such algorithms even when the CFP is infeasible.

The contributions of this paper are as follows. In this paper, we

- propose several distributed algorithms for solving loosely coupled CFPs;
- establish local convergence tests for these algorithms, which enable us to detect arrival at a feasible solution or infeasibility of the problem in a distributed manner with minimal communication;
- also provide convergence rate results for the proposed algorithms for the case the CFP is either feasible or infeasible. For the case the problem is feasible, these results are given in terms of the distance of the iterates to the feasible set. This enables us to provide a unified treatment of the convergence rate analysis of these algorithms and the classic projection methods. In case of an infeasible problem, the result is a bound of the rate of convergence of a norm of a certain residual to a non-zero constant.

The performance of the proposed algorithms are compared using numerical examples.

## Outline

The paper is organized as follows. In Section 2, we provide a formal description of loosely coupled CFPs and describe different approaches for formulating and solving such problems. Particularly we discuss minimization reformulations of coupled CFPs in Section 2.4. A brief description of the most commonly used proximal splitting methods is given in Section 3. These methods are then applied to convex minimization reformulations of the coupled CFP, and the resulting algorithms are reported in Section 4. In that section, we also provide insights on how to establish convergence to a feasible solution or how to deduce infeasibility of the problem in a distributed manner. The convergence rate results for the proposed algorithms are described in Section 5. We present numerical results in Section 6, and we conclude the paper in Section 7.

## Notation

We denote the set of real  $m \times n$  matrices by  $\mathbb{R}^{m \times n}$ , and  $\mathbb{N}_p$  denotes the set of positive integers  $\{1, 2, \dots, p\}$ . Given a set  $J \subset \{1, 2, \dots, n\}$ , the matrix  $E_J \in \mathbb{R}^{|J| \times n}$  is the 0-1 matrix that is obtained from an identity matrix of order  $n$  by deleting the rows indexed by  $\mathbb{N}_n \setminus J$ . Also,  $|J|$  denotes the number of elements in set  $J$ . This means that  $E_J x$  is a vector with the components of  $x$  that correspond to the elements in  $J$ , and we denote this vector with  $x_J$ . Given a vector  $x$ , we denote the  $i$ th component of this vector with  $x_i$ . The distance from a point  $x \in \mathbb{R}^n$  to a set  $C \subseteq \mathbb{R}^n$  is denoted as  $\text{dist}(x, C)$ , and it is defined as

$$\text{dist}(x, C) = \inf_{y \in C} \|x - y\|. \quad (1)$$

where  $\|\cdot\|$  denotes the 2-norm. Similarly, the distance between two sets  $C_1, C_2 \subseteq \mathbb{R}^n$  is defined as

$$\text{dist}(C_1, C_2) = \inf_{y \in C_1, x \in C_2} \|x - y\|. \quad (2)$$

The relative interior of a set  $C$  is denoted  $\text{rel int}(C)$ , and  $D = \text{diag}(a_1, \dots, a_n)$  is a diagonal matrix of order  $n$  with diagonal entries  $D_{ii} = a_i$ . Given vectors  $x^k$  for  $k = 1, \dots, N$ , the column vector  $(x^1, \dots, x^N)$  is all of the given vectors stacked. We finally denote the so-called effective domain of a convex function,  $f$ , with  $\text{dom } f = \{x \mid f(x) < \infty\}$ .

## 2. Decomposition and convex feasibility

Given  $N$  closed convex sets  $\mathcal{C}_1, \dots, \mathcal{C}_N$ , a general convex feasibility problem is defined as

$$\text{find} \quad v \quad (3a)$$

$$\text{subject to} \quad v \in \mathcal{C}_i, \quad i = 1, \dots, N, \quad (3b)$$

where  $v \in \mathbb{R}^n$ . We are particularly interested in the case where the description of each constraint set  $\mathcal{C}_i$  is only dependent on a small subset of the variables in the vector  $v$ . Let us denote the ordered set of indices of variables that appear in the description of the  $i$ th constraint by  $J_i$ . We also denote the ordered set of indices of constraints for which their description depend on  $v_i$  by  $\mathcal{I}_i$ , i.e.,  $\mathcal{I}_i = \{k \mid i \in J_k\}$ . We call a CFP loosely coupled if  $|\mathcal{I}_i| \ll N$  for all  $i = 1, \dots, n$ . There are different ways of formulating the problem in (3), which allow us to design several new or use various already existing algorithms for solving this problem. In order to unify the analysis of such algorithms, we utilize so-called error bounds, which are the subject of the next subsection.

### 2.1. Error Bounds and Bounded Linear Regularity

Error bounds quantify the distance to the solution set of a problem and they become zero only when we arrive at a solution of the problem. The use of error bounds is common in analysis of iterative algorithms, [32]. For CFPs, authors in [6] and [25] consider the use of

$$T(v) = \max_i \{\text{dist}(v, \mathcal{C}_i)\}, \quad (4)$$

as the error bound, when analyzing projection-based algorithms. Note that  $T(v) = 0$  if and only if  $v \in \bigcap_{i=1}^N \mathcal{C}_i$ . Based on this error bound, the closed convex sets,  $\mathcal{C}_i$ , for  $i = 1, \dots, N$ , are said to be boundedly linearly regular, if for every bounded set  $B$  there exists  $\theta_B > 0$  such that

$$\forall v \in B \quad \text{dist} \left( v, \bigcap_{i=1}^N \mathcal{C}_i \right) \leq \theta_B \max_i \{ \text{dist}(v, \mathcal{C}_i) \}. \quad (5)$$

This allows us to bound the distance to the intersection of these sets, which is very difficult or expensive to compute, by  $T(v)$  which can usually be calculated easily, [6]. It was shown by Bauschke *et al.* [4] and Beck & Teboulle [6] that Slater's condition for a CFP implies bounded linear regularity, i.e., for a general CFP where  $\mathcal{C}_1, \dots, \mathcal{C}_k$  are polyhedral sets and  $\mathcal{C}_{k+1}, \dots, \mathcal{C}_N$  are general closed, convex sets, (5) holds if

$$\left( \bigcap_{i=1}^k \mathcal{C}_i \right) \cap \left( \bigcap_{i=k+1}^N \text{rel int}(\mathcal{C}_i) \right) \neq \emptyset. \quad (6)$$

Bounded linear regularity proves to be essential in the analysis of the proposed algorithms in this paper. Next, we investigate some of the approaches for solving the feasibility problem in (3).

## 2.2. Projection Algorithms and Convex Feasibility Problems

One of the possible approaches for solving the CFP in (3), is to neglect the coupling structure among the constraint sets, and use projection algorithms for finding a solution in the intersection of  $N$  convex sets. Among such projection algorithms, cyclic projection algorithm (CPA), maximum distance projection algorithm (MDPA) and mean projection algorithm (MPA) are some of the most widely used ones, where only MPA is suitable for solving (3) in a distributed manner. This follows from the fact that at each iteration MPA uses

$$v^{(k+1)} := \sum_{i=1}^N \alpha_i^{(k)} P_{\mathcal{C}_i}(v^{(k)}) \quad (7)$$

for updating the iterates, where  $\sum_{i=1}^N \alpha_i^{(k)} = 1$  and  $\alpha_1^{(k)}, \dots, \alpha_N^{(k)} > 0$ . Notice that the updating procedure in (7) is highly parallelizable. That is because the projections can be performed in parallel and simultaneously. Assuming  $N$  computing agents, each agent  $i$  then computes  $P_{\mathcal{C}_i}(v^{(k)})$ , and communicates with all the other agents to update the iterate as in (7). Hence this algorithm, requires global communication among all the agents. In [6], it was shown that in case the sets in (3) are boundedly linearly regular, the algorithm enjoys a linear rate of convergence, where

$$\text{dist} \left( v^{(k+1)}, \bigcap_{i=1}^N \mathcal{C}_i \right) \leq \gamma_B \text{dist} \left( v^{(k)}, \bigcap_{i=1}^N \mathcal{C}_i \right), \quad (8)$$

with

$$\gamma_B = \sqrt{1 - \frac{\min_i \{ \alpha_i^{(k)} \}}{\theta_B^2}}, \quad (9)$$

where  $\theta_B > 0$  depends on the starting point  $x^{(0)}$ . This dependence follows from the fact that  $\theta_B$  should satisfy (5) with  $B = \{v \mid \|v - z\| \leq \|v^{(0)} - z\|\}$  for any  $z \in \mathcal{C}$ . Iusem & De Pierro, [27], have proposed an accelerated variant of this algorithm that takes as the next iterate a convex combination of the projections of  $v^{(k)}$  on only the sets for which  $v^{(k)} \notin \mathcal{C}_i$ . This generally improves the rate of convergence when only a few constraints are violated. However, neglecting the structure in (3) can drastically deteriorate the performance of this algorithm, [28]. Also in case Slater's condition is not satisfied, e.g., when the problem is infeasible, (8) does not hold and this algorithm can perform arbitrarily bad, [6]. In the upcoming subsections, we show how the structure in the coupling among the constraints in (3) can be exploited, which allows us to reformulate the problem in other ways.

### 2.3. Decomposition and Product Space Formulation

Having the structure in the constraints in (3) in mind, we define  $N$  lower-dimensional sets

$$\bar{\mathcal{C}}_i = \{s^i \in \mathbb{R}^{|J_i|} \mid E_{J_i}^T s^i \in \mathcal{C}_i\}, \quad i = 1, \dots, N, \quad (10)$$

such that  $s^i \in \bar{\mathcal{C}}_i$  implies  $E_{J_i}^T s^i \in \mathcal{C}_i$ . This allows us to rewrite the standard form CFP in (3) as

$$\text{find} \quad s^1, s^2, \dots, s^N, v \quad (11a)$$

$$\text{subject to} \quad s^i \in \bar{\mathcal{C}}_i, \quad i = 1, \dots, N \quad (11b)$$

$$s^i = E_{J_i} v, \quad i = 1, \dots, N \quad (11c)$$

where the equality constraints are the so-called coupling or global consensus constraints that ensure that the local variables  $s^1, \dots, s^N$  are consistent with one another. In other words, if the constraints  $v \in \mathcal{C}_i$  and  $v \in \mathcal{C}_j$  ( $i \neq j$ ) both involve  $v_k$ , then the  $k$ th component of  $E_{J_i}^T s^i$  and  $E_{J_j}^T s^j$  must be equal. This formulation decomposes the so-called global variable  $v$  into  $N$  coupled local variables  $s^1, \dots, s^N$ . This allows us to rewrite the problem as a CFP with two sets

$$\begin{aligned} &\text{find} \quad S \\ &\text{subject to} \quad S \in \mathcal{C}, \quad S \in \mathcal{D} \end{aligned} \quad (12)$$

where

$$S = (s^1, \dots, s^l) \in \mathbb{R}^{|J_1|} \times \dots \times \mathbb{R}^{|J_l|}$$

$$\mathcal{C} = \bar{\mathcal{C}}_1 \times \dots \times \bar{\mathcal{C}}_l$$

$$\mathcal{D} = \{\bar{E}v \mid v \in \mathbb{R}^n\}$$

$$\bar{E} = [E_{J_1}^T \dots E_{J_l}^T]^T.$$

The formulation (12) can be thought of as a “compressed” product space formulation of a CFP as described in (3), and it is similar to the consensus optimization problems described in [10, Sec. 7.2], [9, Sec. 3.4]. The problem in (12) can now be solved using von Neumann's and Dykstra's alternating projections (AP) methods, which are methods for finding solutions in the intersection of two sets.

### 2.3.1. Von Neumann's alternating projections

Given the two sets,  $\mathcal{C}$  and  $\mathcal{D}$ , and a starting point  $v^{(0)}$ , von Neumann's AP method computes two sequences

$$S^{(k+1)} = P_{\mathcal{C}} \left( V^{(k)} \right) \quad (13a)$$

$$V^{(k+1)} = P_{\mathcal{D}} \left( S^{(k+1)} \right). \quad (13b)$$

where  $V^{(k)} = \bar{E}v^{(k)}$ . If the CFP in (12) is feasible, i.e.,  $\mathcal{C} \cap \mathcal{D} \neq \emptyset$ , then both sequences converge to a point in  $\mathcal{C} \cap \mathcal{D}$ , [1, 6]. The updates in (13), result in the following iterative algorithm

$$S^{(k+1)} = P_{\mathcal{C}} \left( V^{(k)} \right) \quad (14a)$$

$$= \left( P_{\bar{\mathcal{C}}_1} \left( E_{J_1} v^{(k)} \right), \dots, P_{\bar{\mathcal{C}}_N} \left( E_{J_N} v^{(k)} \right) \right)$$

$$V^{(k+1)} = \bar{E} \underbrace{\left( \bar{E}^T \bar{E} \right)^{-1} \bar{E}^T S^{(k+1)}}_{v^{(k+1)}}, \quad (14b)$$

where (14a) and (14b) are projections onto  $\mathcal{C}$  and onto the column space of  $\bar{E}$ , respectively. Note that the projection onto the set  $\mathcal{C}$  can be computed in parallel by  $N$  computing agents, i.e., agent  $i$  computes  $s_i^{(k)} = P_{\bar{\mathcal{C}}_i} (E_{J_i} v^{(k)})$ , and the second projection can be interpreted as a consensus step that can be solved via distributed averaging. The details of a distributed implementation of (14) are later discussed in Section 4. In case the sets  $\mathcal{C}$  and  $\mathcal{D}$  are boundedly linearly regular, it follows from [6, Cor. 2.1] that

$$\text{dist} \left( S^{(k+1)}, \bigcap_{i=1}^N \mathcal{C}_i \right) \leq \gamma_B \text{dist} \left( S^{(k)}, \bigcap_{i=1}^N \mathcal{C}_i \right) \quad (15)$$

with

$$\gamma_B = \sqrt{1 - \frac{1}{\theta_B^2}} \quad (16)$$

where  $\theta_B > 0$  depends on the starting point as is the case for (8) of MPA. It was shown in [1, 2], that in case the problem in (12) is infeasible

$$V^{(k)} - S^{(k)}, V^{(k)} - S^{(k+1)} \rightarrow d, \quad \|d\| = \text{dist}(\mathcal{C}, \mathcal{D}), \quad (17)$$

where since  $\mathcal{C}$  is assumed to be closed,  $\text{dist}(\mathcal{C}, \mathcal{D})$  is attained. Theoretically, this result provides the possibility to detect infeasibility of (12) by monitoring the sequences in (17). However, to the best knowledge of the authors, the rates of convergence of the sequences  $V^{(k)} - S^{(k)}$  and  $V^{(k)} - S^{(k+1)}$  to  $d$  or  $\|d\|$  to  $\text{dist}(\mathcal{C}, \mathcal{D})$  have not yet been established.

### 2.3.2. Dykstra's alternating projections

The CFP in (12) can also be solved using Dykstra's AP method, where

$$S^{(k+1)} = P_{\mathcal{C}}(V^{(k)} - \bar{\lambda}^{(k)}) \quad (18a)$$

$$V^{(k+1)} = P_{\mathcal{D}}(S^{(k)}) \quad (18b)$$

$$\bar{\lambda}^{(k+1)} = \bar{\lambda}^{(k)} + (S^{(k+1)} - V^{(k+1)}). \quad (18c)$$

and  $\bar{\lambda} = (\bar{\lambda}^1, \dots, \bar{\lambda}^N)$ . Note that this algorithm is a special case of Dykstra's AP method where one of the sets is affine, [2]. Similar to von Neumann's AP method, in case  $\mathcal{C} \cap \mathcal{D} \neq \emptyset$  the iterates  $V^{(k)}$  and  $S^{(k)}$  converge to a point in  $\mathcal{C} \cap \mathcal{D}$ , [2]. The updates in (18), result in the following iterative algorithm

$$S^{(k+1)} = P_{\mathcal{C}}(V^{(k)} - \bar{\lambda}^{(k)}) \quad (19a)$$

$$= \left( P_{\bar{\mathcal{C}}_1} \left( E_{J_1} v^{(k)} - \bar{\lambda}^{1,(k)} \right), \dots, P_{\bar{\mathcal{C}}_N} \left( E_{J_N} v^{(k)} - \bar{\lambda}^{N,(k)} \right) \right)$$

$$V^{(k+1)} = P_{\mathcal{D}}(S^{(k)}) \quad (19b)$$

$$\bar{\lambda}^{(k+1)} = \bar{\lambda}^{(k)} + (S^{(k+1)} - V^{(k+1)}). \quad (19c)$$

As can be seen from (19a), this algorithm is also highly parallelizable. This is discussed in more detail in Section 4. Unlike von Neumann's AP method, the iterative algorithm in (18) does not necessarily converge with a linear rate even when the underlying sets are boundedly linearly regular. Similar to von Neumann's AP method, in case the CFP in (12) is infeasible the sequences  $V^{(k)} - S^{(k)}$  and  $V^{(k)} - S^{(k+1)}$  converge to  $d$ , however, their rates of convergence are not known, [2].

#### 2.4. Convex Minimization Formulation

The problem in (12), can also be reformulated as a convex minimization problem. Let  $\mathcal{I}_{\mathcal{C}}(S)$  and  $\mathcal{I}_{\mathcal{D}}(S)$  be the indicator functions for the sets  $\mathcal{C}$  and  $\mathcal{D}$ , where an indicator function for a set, e.g.,  $\mathcal{A}$ , is defined as

$$\mathcal{I}_{\mathcal{A}}(x) = \begin{cases} \infty & x \notin \mathcal{A} \\ 0 & x \in \mathcal{A} \end{cases}. \quad (20)$$

and hence  $\text{dom}(\mathcal{I}_{\mathcal{A}}) = \mathcal{A}$ . The CFP in (12) can then be equivalently rewritten as the following convex minimization problem

$$\underset{S}{\text{minimize}} \quad \mathcal{I}_{\mathcal{C}}(S) + \mathcal{I}_{\mathcal{D}}(S). \quad (21)$$

Despite the equivalence between the problems in (12) and (21), the minimization problem is not defined in case the CFP in (12) is infeasible, since the effective domain of the cost function would then be empty. This limits our capability to draw conclusions regarding the infeasibility of the corresponding CFP using this formulation. In order to circumvent this issue, we define the following unconstrained minimization problems

$$\underset{S}{\text{minimize}} \quad F_1(S) := \frac{1}{2} \sum_{i=1}^N \|s^i - P_{\bar{\mathcal{C}}_i}(s^i)\|^2 + \mathcal{I}_{\mathcal{D}}(S), \quad (22)$$

and

$$\underset{S}{\text{minimize}} \quad F_2(S) := \frac{1}{2} \sum_{i=1}^N \|s^i - P_{\bar{\mathcal{C}}_i}(s^i)\|^2 + \frac{1}{2} \|S - P_{\mathcal{D}}(S)\|^2, \quad (23)$$

which are well-defined even when the CFP in (12) is infeasible. Note that the problems in (22) and (23) are not equivalent to the CFP in (12). However, these minimization problems always have at least one solution and admit an optimal solution with zero objective value if and only if the problem in (12) is feasible. In fact the optimal solution then constitutes a solution for (12). Similarly, the minimization problems in (22) and (23), yield a non-zero optimal objective value if and only if the CFP in (12) is infeasible. In the upcoming sections, we explain how these minimization problems facilitate the design of distributed algorithms for solving the CFP in (12).

### 3. Proximity Operators and Proximal Splitting

Consider the problem of minimizing a sum of  $p$  closed convex functions

$$\text{minimize } F(x) = f_1(x) + \cdots + f_p(x). \quad (24)$$

Through the use of their so-called proximity operators, [15], proximal splitting algorithms allow us to perform this minimization by considering each of the terms in the cost function separately. Proximity operators are defined as follows.

**DEFINITION 3.1** [15] *Given a closed convex function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ , then for every  $x \in \mathbb{R}^n$ , the proximity operator of the function  $f$ ,  $\text{prox}_f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ , is defined as the unique minimizer of the following optimization problem,*

$$\text{minimize}_y \quad f(y) + \frac{1}{2}\|x - y\|^2.$$

Keeping in mind the problems in (22) and (23), we only consider the case where the cost function consists of two terms, i.e.,  $p = 2$ . Depending on the characteristics of the terms in the cost function, we are allowed to employ different proximal splitting algorithms. Next, we review some of the most widely used of such methods.

#### 3.1. Forward-backward Splitting

The forward-backward proximal splitting algorithm is suitable for cases where at least one of the two terms in the cost function is differentiable with a Lipschitz continuous gradient. Assume that  $f_1$  and  $f_2$  are both closed convex functions and that the problem

$$\text{minimize}_x \quad f_1(x) + f_2(x) \quad (25)$$

has at least one solution. Let  $f_1(x)$  be differentiable. In this case, the problem can be solved using Algorithm 1.

---

#### Algorithm 1 Forward-backward method [15, 16]

---

- 1: Given  $\varepsilon \in (0, \min(1, 1/L)]$  and  $x^{(1)}$
  - 2: **for**  $k = 1, 2 \dots$  **do**
  - 3:    $\gamma^{(k)} \in [\varepsilon, 2/L - \varepsilon]$
  - 4:    $\lambda^{(k)} \in [\varepsilon, 1]$
  - 5:    $y^{(k+1)} = x^{(k)} - \gamma^{(k)} \nabla f_1(x^{(k)})$
  - 6:    $x^{(k+1)} = (1 - \lambda^{(k)})x^{(k)} + \lambda^{(k)} \text{prox}_{\gamma^{(k)} f_2}(y^{(k+1)})$
  - 7: **end for**
-



In this algorithm  $\gamma^{(k)}$  is the so-called gradient step,  $\lambda^{(k)}$  is a relaxation parameter and  $L$  is the Lipschitz constant of  $\nabla f_1$ . It was shown in [31], [7, Thm. 3.1], that if  $\gamma^{(k)} < 2/L$  and  $\lambda^{(k)} = 1$ ,

$$F(x^{(k)}) - F(x^*) \leq \frac{L\|x^{(0)} - x^{(k)}\|^2}{2k}, \quad (26)$$

where  $x^*$  is any optimal solution for the problem in (24). It is also possible to obtain better rate of convergence of the objective value by combining this algorithm with the so-called 1-memory accelerated gradient methods, [15]. This comes at the expense of a more complicated algorithm. Let  $l(x; y) = f_1(y) + \langle \nabla f_1(y), x - y \rangle + f_2(x)$  and define

$$D(x, y) = h(x) - h(y) - \langle \nabla h(y), x - y \rangle,$$

where  $h$  is a strictly convex function. The general format for 1-memory accelerated gradient methods can then be presented as in Algorithm 2, [37].

---

**Algorithm 2** 1-memory accelerated gradient method [37]

---

- 1: Given  $\theta^{(1)} \in (0, 1]$  and  $x^{(1)}, g^{(1)}$
  - 2: **for**  $k = 1, 2, \dots$  **do**
  - 3:    $y^{(k+1)} = (1 - \theta^{(k)})x^{(k)} + \theta^{(k)}g^{(k)}$
  - 4:    $g^{(k+1)} = \arg \min_x \{l(x; y^{(k+1)}) + \theta^{(k)}LD(x, g^{(k)})\}$
  - 5:    $\hat{x}^{(k+1)} = (1 - \theta^{(k)})x^{(k)} + \theta^{(k)}g^{(k+1)}$
  - 6:   Choose  $x^{(k+1)}$  to be no worse than  $\hat{x}^{(k+1)}$  in  $l(x; y^{(k+1)}) + L/2\|x - y^{(k+1)}\|^2$
  - 7:   Choose  $\frac{1 - \theta^{(k+1)}}{(\theta^{(k+1)})^2} \leq \frac{1}{(\theta^{(k)})^2}$
  - 8: **end for**
- 

Depending on the choice of function  $h(\cdot)$ , and how we choose to compute  $x^{(k)}$  and  $\theta^{(k)}$ , we end up in different accelerated gradient methods, [8, 37]. In case we choose  $h(x) = \frac{1}{2}\|x - y\|^2$  and merge the fifth and sixth steps of Algorithm 2 by choosing

$$x^{(k+1)} = \arg \min_x \{l(x; Y^{(k+1)}) + \frac{L}{2}\|x - Y^{(k+1)}\|^2\},$$

we can summarize the combination of the forward-backward splitting algorithm with 1-memory accelerated gradient method as Algorithm 3.

---

**Algorithm 3** Accelerated forward-backward method

---

- 1: Given  $\theta_1 \in (0, 1]$  and  $x^{(1)}, g^{(1)}$
  - 2: **for**  $k = 1, 2, \dots$  **do**
  - 3:    $y^{(k+1)} = (1 - \theta^{(k)})x^{(k)} + \theta^{(k)}g^{(k)}$
  - 4:    $b^{(k+1)} = g^{(k)} - \frac{1}{\theta^{(k)}L}\nabla f_1(y^{(k+1)})$
  - 5:    $g^{(k+1)} = \text{prox}_{\frac{1}{\theta^{(k)}L}f_2}(b^{(k+1)})$
  - 6:    $c^{(k+1)} = y^{(k+1)} - \frac{1}{L}\nabla f_1(y^{(k+1)})$
  - 7:    $x^{(k+1)} = \text{prox}_{\frac{1}{L}f_2}(c^{(k+1)})$
  - 8:   Choose  $\frac{1 - \theta^{(k+1)}}{(\theta^{(k+1)})^2} \leq \frac{1}{(\theta^{(k)})^2}$
  - 9: **end for**
- 

There are different convergence results for such algorithms which are dependent on different choices of  $\theta^{(k)}$ , e.g., see [37, Cor. 1] and [7, Thm. 4.4], where all suggest rates

of convergence of order  $\mathcal{O}(1/k^2)$  of the objective value function. In other words

$$F(x^{(k)}) - F(x^*) \leq \mathcal{O}\left(\frac{1}{k^2}\right). \quad (27)$$

### 3.2. Splitting Using Alternating Linearization Methods

The splitting of the problem in (24) for  $p = 2$ , can also be performed by introducing auxiliary constraints as

$$\begin{aligned} & \underset{x, y}{\text{minimize}} && f_1(x) + f_2(y) \\ & \text{subject to} && x = y. \end{aligned} \quad (28)$$

Assume that  $f_1$  and  $f_2$  are both differentiable with Lipschitz continuous gradients. Such linear equality constrained optimization problems can then be solved using the so-called alternating linearization method (ALM), [23]. Let

$$\begin{aligned} Q_\mu^{f_2}(x, y) &= f_1(x) + f_2(y) + \langle \nabla f_2(y), x - y \rangle + \frac{1}{2\mu} \|x - y\|^2 \\ &= f_1(x) + f_2(y) + \frac{1}{2\mu} \|x - (y - \mu \nabla f_2(y))\|^2, \end{aligned}$$

$$\begin{aligned} Q_\mu^{f_1}(y, x) &= f_2(y) + f_1(x) + \langle \nabla f_1(x), y - x \rangle + \frac{1}{2\mu} \|x - y\|^2 \\ &= f_2(y) + f_1(x) + \frac{1}{2\mu} \|x - (y + \mu \nabla f_1(x))\|^2, \end{aligned}$$

and define the following update rules

$$\begin{aligned} x^{(k+1)} &= \arg \min_x Q_{\mu_1}^{f_2}(x, y^{(k)}) \\ &= \text{prox}_{\mu_1 f_1}(y^{(k)} - \mu \nabla f_2(y^{(k)})), \end{aligned} \quad (29)$$

and

$$\begin{aligned} y^{(k+1)} &= \arg \min_y Q_{\mu_2}^{f_1}(y, x^{(k+1)}) \\ &= \text{prox}_{\mu_2 f_2}(x^{(k+1)} - \mu \nabla f_1(x^{(k+1)})). \end{aligned} \quad (30)$$

The ALM scheme for solving the problem in (28) can then be written as in Algorithm 4.

---

#### Algorithm 4 ALM

---

- 1: Given  $\mu_1, \mu_2 > 0$  and  $y^{(1)}$
  - 2: **for**  $k = 1, 2 \dots$  **do**
  - 3:    $x^{(k+1)} = \text{prox}_{\mu_1 f_1}(y^{(k)} - \mu \nabla f_2(y^{(k)}))$
  - 4:    $y^{(k+1)} = \text{prox}_{\mu_2 f_2}(x^{(k+1)} - \mu \nabla f_1(x^{(k+1)}))$
  - 5: **end for**
-

Goldfarb *et al.* in [23, Cor. 2.4] showed that in case  $0 < \mu_1 \leq 1/L_1$  and  $0 < \mu_2 \leq 1/L_2$ , where  $L_1$  and  $L_2$  are Lipschitz constants for the gradients of  $f_1$  and  $f_2$  respectively,

$$F(y^{(k)}) - F(x^*) \leq \frac{\|x^{(1)} - x^*\|^2}{2(\mu_1 + \mu_2)k}, \quad \forall k > 1. \quad (31)$$

In the same paper, the authors also propose an accelerated variant of ALM, which is reported in Algorithm 5.

---

**Algorithm 5** Fast ALM [23]

---

```

1: Given  $\mu_1, \mu_2 > 0$ ,  $t^{(1)} = 1$  and  $z^{(1)} = y^{(1)}$ 
2: for  $k = 1, 2, \dots$  do
3:    $x^{(k+1)} = \text{prox}_{\mu_1 f_1}(z^{(k)} - \mu_1 \nabla f_2(z^{(k)}))$ 
4:    $y^{(k+1)} = \text{prox}_{\mu_2 f_2}(x^{(k+1)} - \mu_2 \nabla f_1(x^{(k+1)}))$ 
5:    $t^{(k+1)} = \frac{1 + \sqrt{1 + t^{(k)2}}}{2}$ 
6:    $z^{(k+1)} = y^{(k+1)} + \frac{t^{(k)} - 1}{t^{(k+1)}}(y^{(k+1)} - y^{(k)})$ 
7: end for

```

---

It was shown in [23, Cor. 3.5], that in case  $\mu_1$  and  $\mu_2$  are chosen in the same manner as for the ALM algorithm,

$$F(y^{(k)}) - F(x^*) \leq \frac{2\|x^{(1)} - x^*\|^2}{(\mu_1 + \mu_2)k^2}, \quad \forall k > 1 \quad (32)$$

Notice that Algorithm 5 is similar to Algorithm 3. This can be particularly observed by comparing the steps 5, 7, 8 and 1 in Algorithm 3 with steps 3, 4, 5 and 6 in Algorithm 5, respectively.

**Remark 1** The alternating linearization method is very similar to the so-called alternating direction method of multipliers (ADMM), [9, 10], and in fact Algorithm 5 is equivalent to a symmetric variant of ADMM, [23]. We have chosen not to investigate ADMM or its other variants, since most of their convergence rate results rely on strong convexity of at least one of the terms in the cost function, [19, 24], which is not the case for neither of the problems in (22) and (23).

### 3.3. Douglas-Rachford Splitting

In case  $\text{ri dom } f_1 \cap \text{ri dom } f_2 \neq \emptyset$  and if the problem in (25) has at least one solution, we can use the so-called Douglas-Rachford splitting algorithm for solving the optimization problem. Note that unlike forward-backward splitting, this method does not require any of the objective function terms to be differentiable. The scheme for solving the optimization problem using this method is described in Algorithm 6.

---

**Algorithm 6** Douglas-Rachford method [14, 15]

---

```

1: Given  $\varepsilon \in (0, 1), \gamma > 0$  and  $y^{(1)}$ 
2: for  $k = 1, 2, \dots$  do
3:    $\lambda^{(k)} \in [\varepsilon, 2 - \varepsilon]$ 
4:    $x^{(k+1)} = \text{prox}_{\gamma f_1}(y^{(k)})$ 
5:    $y^{(k+1)} = y^{(k)} + \lambda^{(k)}(\text{prox}_{\gamma f_2}(2x^{(k+1)} - y^{(k)}) - x^{(k+1)})$ 
6: end for

```

---

**Remark 2** Douglas-Rachford splitting is one of the principle classes of splitting methods and includes many splitting methods as special cases. Particularly, it was shown in [21, 22] that ADMM and ALM fall within this class of splitting methods. The convergence of this splitting method (and its variants) has been studied thoroughly in the literature, [14, 20, 30]. However, due to its generality, the convergence rate of this splitting method in its most general format is yet to be established. Hence, although we utilize this algorithm for solving CFPs, this limits our capability to provide convergence rate results for the resulting algorithm.

Next, we will apply the proximal splitting algorithms described in this section to the problems in (22) and (23).

#### 4. Distributed Solution

In this section, we propose several distributed algorithms that can be used to solve the feasibility problem in (3). In sections 4.1 and 4.2, we describe and discuss the distributed algorithms, and in Section 4.3, we investigate how the convergence of these methods can be established in a distributed manner when the problem in (12) is either feasible or infeasible.

##### 4.1. Proximal splitting and distributed implementation

In order to facilitate providing a distributed solution for the feasibility problem in (3) and due to the similarity between the problems in (22), (23) and (24), we employ proximal splitting algorithms. We apply algorithms 1 and 3 for solving the minimization problem in (22). To be able to use these algorithms, we identify  $f_1(S)$  as  $\frac{1}{2} \sum_{i=1}^N \|s^i - P_{\bar{C}_i}(s^i)\|^2$  and  $f_2(S)$  as  $\mathcal{I}_{\mathcal{D}}(h)$ . The proximity operators for these functions are given as

$$\text{prox}_{f_1}(S) = \frac{S + P_{\mathcal{C}}(S)}{2} \quad (33a)$$

$$\text{prox}_{f_2}(S) = P_{\mathcal{D}}(S), \quad (33b)$$

Note that the proximity operator computation of  $f_1(S)$  is highly parallelizable. Assume that a network of  $N$  computing agents is available. Then  $\text{prox}_{f_1}$  can be computed in a distributed manner where each of the  $N$  agents calculates  $(s^i + P_{\bar{C}_i}(s^i)) / 2$  individually. Considering the definition of the set  $\mathcal{D}$ , the proximity operator of  $f_2$  is merely a linear projection and is given as

$$\text{prox}_{f_2}(S) = \bar{E}(\bar{E}^T \bar{E})^{-1} \bar{E}^T S. \quad (34)$$

Note that

$$\bar{E}^T \bar{E} = \text{diag}(|\mathcal{I}_1|, \dots, |\mathcal{I}_N|),$$

and hence,  $\text{prox}_{f_2}$  describes the required communication and interaction between the agents in the network. For instance, define  $b = (\bar{E}^T \bar{E})^{-1} \bar{E}^T$ , then each of the components of  $b$  can be expressed as

$$b_j = \frac{1}{|\mathcal{I}_j|} \sum_{q \in \mathcal{I}_j} \left( E_{J_q}^T s^{q,(k+1)} \right)_j. \quad (35)$$

As a result in order to compute this quantity, the agents in the set  $\mathcal{I}_j$  must interact with one another. In other words, this requires each agent  $i$  to communicate with all the agents in

$$\text{Ne}(i) = \{j \mid J_i \cap J_j \neq \emptyset\}, \quad (36)$$

which are referred to as neighbours of agent  $i$ . This interpretation of  $\text{prox}_{f_2}$  is later used in the description of the proposed algorithms for distributed feasibility analysis. In order to solve the minimization problem in (23) we use algorithms 4–6, where in this case,  $f_2(S)$  is identified as  $\frac{1}{2}\|S - P_{\mathcal{D}}(S)\|^2$ . The proximity operator for this function is given as

$$\text{prox}_{f_2}(S) = \frac{S + P_{\mathcal{D}}(S)}{2}, \quad (37)$$

The proximity operator of  $f_2$  for this case, is also dependent on computing projections onto the consensus set. Hence, similar to the previous case,  $\text{prox}_{f_2}$  will also describe the communication and interaction among agents.

#### 4.1.1. Forward-backward algorithm

Considering the description of the functions  $f_1$  and  $f_2$  in problem (22),  $f_1$  is differentiable and

$$\nabla f_1(S) = S - P_{\mathcal{C}}(S),$$

which is Lipschitz continuous with Lipschitz constant  $L = 1$ . Applying the forward-backward method to this problem results in the following update rules

$$\begin{aligned} Y^{(k+1)} &= S^{(k)} - \gamma^{(k)} \left( S^{(k)} - P_{\mathcal{C}}(S^{(k)}) \right) \\ &= (1 - \gamma^{(k)})S^{(k)} + \gamma^{(k)}P_{\mathcal{C}}(S^{(k)}) \end{aligned}$$

$$\begin{aligned} S^{(k+1)} &= (1 - \lambda^{(k)})S^{(k)} + \lambda^{(k)}P_{\mathcal{D}}(Y^{(k+1)}) \\ &= (1 - \lambda^{(k)})S^{(k)} + \lambda^{(k)}\bar{E}(\bar{E}^T \bar{E})^{-1} \bar{E}^T Y^{(k+1)}, \end{aligned}$$

where  $Y^{(k+1)} = (y^{1,(k+1)}, \dots, y^{N,(k+1)})$ . Note that, if we choose  $S^{(1)} = \bar{E}v^{(1)}$  then

---

#### Algorithm 7 Forward-backward method

---

```

1: Given  $\varepsilon \in (0, 1]$ ,  $v^{(1)}$  and  $S^{(1)} = \bar{E}v^{(1)}$ 
2: for  $k = 1, 2 \dots$  do
3:    $\gamma^{(k)} \in [\varepsilon, 2 - \varepsilon]$ 
4:    $\lambda^{(k)} \in [\varepsilon, 1]$ 
5:   for  $i = 0, 1 \dots, N$  do
6:      $y^{i,(k+1)} = (1 - \gamma^{(k)})s^{i,(k)} + \gamma^{(k)}P_{\bar{\mathcal{C}}_i}(s^{i,(k)})$ 
7:     Communicate with all agents  $r$  belonging to  $\text{Ne}(i)$ 
8:     for all  $j \in J_i$  do
9:        $v_j^{(k+1)} = \frac{1}{|J_j|} \sum_{q \in J_j} \left( E_{J_q}^T y^{q,(k+1)} \right)_j$ 
10:    end for
11:     $s^{i,(k+1)} = (1 - \lambda^{(k)})s^{i,(k)} + \lambda^{(k)}v_{J_i}^{(k+1)}$ 
12:  end for
13: end for
```

---

$S^{(k)} \in \mathcal{D}$  for all  $k > 1$ . The resulting distributed solution based on the forward-backward algorithm is summarized in Algorithm 7. Notice that, for a constant  $\lambda = 1$ , this algorithm is equivalent to the two point projection method in [6]. Furthermore if  $\gamma^{(k)} = 1$ , this algorithm is von Neumann's AP method.

#### 4.1.2. Accelerated forward-backward algorithm

It is possible to obtain faster convergence rates, by employing the accelerated forward-backward splitting to solve the problem in (22). Let  $Y^{(k)} = (y^{1,(k)}, \dots, y^{N,(k)})$  and  $G^{(k)}$ ,  $U^{(k)}$ ,  $Z^{(k)}$  be defined similarly. By applying this algorithm to the problem in (22), we arrive at the following update rules

$$Y^{(k+1)} = (1 - \theta^{(k)})S^{(k)} + \theta^{(k)}G^{(k)} \quad (38a)$$

$$U^{(k+1)} = G^{(k)} - \frac{1}{\theta^{(k)}} \left( Y^{(k+1)} - P_{\mathcal{C}}(Y^{(k+1)}) \right) \quad (38b)$$

$$G^{(k+1)} = P_{\mathcal{D}}(U^{(k+1)}) \quad (38c)$$

$$Z^{(k+1)} = P_{\mathcal{C}}(Y^{(k+1)}) \quad (38d)$$

$$S^{(k+1)} = P_{\mathcal{D}}(Z^{(k+1)}) \quad (38e)$$

Note that, similar to the forward-backward algorithm, in case we choose  $S^{(1)} = G^{(1)} = \bar{E}v^{(1)}$ , we will have  $Y^{(k)}, G^{(k)}, S^{(k)} \in \mathcal{D}$  for all  $k \geq 1$ . Substituting (38b) into (38c) and by using (38a), we can then simplify these update rules as follows

$$\begin{aligned} Y^{(k+1)} &= (1 - \theta^{(k)})S^{(k)} + \theta^{(k)}G^{(k)} \\ G^{(k+1)} &= \bar{E}(\bar{E}^T \bar{E})^{-1} \bar{E}^T U^{(k+1)} \\ &= \bar{E}(\bar{E}^T \bar{E})^{-1} \bar{E}^T \left( G^{(k)} - \frac{1}{\theta^{(k)}} \left( Y^{(k+1)} - P_{\mathcal{C}}(Y^{(k+1)}) \right) \right) \\ &= G^{(k)} - \frac{1}{\theta^{(k)}} \bar{E}(\bar{E}^T \bar{E})^{-1} \bar{E}^T \\ &\quad \left( (1 - \theta^{(k)})S^{(k)} + \theta^{(k)}G^{(k)} - P_{\mathcal{C}}(Y^{(k+1)}) \right) \\ &= \frac{\theta^{(k)} - 1}{\theta^{(k)}} S^{(k)} + \frac{1}{\theta^{(k)}} \bar{E}(\bar{E}^T \bar{E})^{-1} \bar{E}^T P_{\mathcal{C}}(Y^{(k+1)}) \\ S^{(k+1)} &= \bar{E}(\bar{E}^T \bar{E})^{-1} \bar{E}^T P_{\mathcal{C}}(Y^{(k+1)}) \end{aligned}$$

The resulting distributed algorithm can then be summarized as in Algorithm 8.

**Algorithm 8** Accelerated proximal gradient method

---

```

1: Given  $\theta^{(0)} \in (0, 1]$ ,  $v^{(0)}$  and  $G^{(1)} = S^{(1)} = \bar{E}v^{(0)}$ 
2: for  $k = 1, 2, \dots$  do
3:   for  $i = 1, 2, \dots, N$  do
4:      $y^{i,(k+1)} = (1 - \theta^{(k)})s^{i,(k)} + \theta^{(k)}g^{i,(k)}$ 
5:     Communicate with all agents  $r$  belonging to  $\text{Ne}(i)$ 
6:     for all  $j \in J_i$  do
7:        $v_j^{(k+1)} = \frac{1}{|J_j|} \sum_{q \in J_j} \left( E_{J_q}^T P_{\bar{C}_q}(y^{q,(k+1)}) \right)_j$ 
8:     end for
9:      $g^{i,(k+1)} = \frac{\theta^{(k)} - 1}{\theta^{(k)}} s^{i,(k)} + \frac{1}{\theta^{(k)}} v_{J_i}^{(k+1)}$ 
10:     $s^{i,(k+1)} = v_{J_i}^{(k+1)}$ 
11:  end for
12:  Choose  $\theta^{(k+1)}$  such that  $\frac{1 - \theta^{(k+1)}}{\theta^{(k+1)2}} \leq \frac{1}{\theta^{(k)2}}$ 
13: end for

```

---

**4.1.3. ALM**

It is also possible to devise a distributed feasibility analysis algorithm by applying ALM to the formulation in (23). Define  $\nu^{(k)} = (\nu^{1,(k)}, \dots, \nu^{N,(k)}) = \nabla f_1(S^{(k)})$  and  $\xi^{(k)} = (\xi^{1,(k)}, \dots, \xi^{N,(k)}) = \nabla f_2(Y^{(k)})$ . From the optimality conditions for (29) and (30), we have

$$\begin{aligned} \nabla f_1(S^{(k+1)}) + \frac{1}{\mu_1}(S^{(k+1)} - Y^{(k)}) + \nabla f_2(Y^{(k)}) &= 0 \\ \nabla f_2(Y^{(k+1)}) - \frac{1}{\mu_2}(S^{(k+1)} - Y^{(k+1)}) + \nabla f_1(S^{(k+1)}) &= 0 \end{aligned}$$

which results in the following update rules for  $\nu^{(k)}$  and  $\xi^{(k+1)}$

$$\begin{aligned} \nu^{(k+1)} &= -\xi^{(k)} - \frac{1}{\mu_1}(S^{(k+1)} - Y^{(k)}) \\ \xi^{(k+1)} &= -\nu^{(k+1)} + \frac{1}{\mu_2}(S^{(k+1)} - Y^{(k+1)}) \end{aligned} \tag{39}$$

Applying Algorithm 4 to the problem in (23) then results in

$$\begin{aligned} S^{(k+1)} &= \text{prox}_{\mu_1 f_1}(Y^{(k)} - \mu_1 \xi^{(k)}) \\ &= \frac{1}{\mu_1 + 1} \left( Y^{(k)} - \mu_1 \xi^{(k)} + \mu_1 P_{\bar{C}}(Y^{(k)} - \mu_1 \xi^{(k)}) \right) \end{aligned} \tag{40a}$$

$$\nu^{(k+1)} = -\xi^{(k)} - \frac{1}{\mu_1}(S^{(k+1)} - Y^{(k)}) \tag{40b}$$

$$\begin{aligned} Y^{(k+1)} &= \text{prox}_{\mu_2 f_2}(S^{(k+1)} - \mu_2 \nu^{(k+1)}) \\ &= \frac{1}{\mu_2 + 1} \left( S^{(k+1)} - \mu_2 \nu^{(k+1)} + \mu_2 P_{\mathcal{D}}(S^{(k+1)} - \mu_2 \nu^{(k+1)}) \right) \\ &= \frac{1}{\mu_2 + 1} \left( S^{(k+1)} - \mu_2 \nu^{(k+1)} + \mu_2 \bar{E}(\bar{E}^T \bar{E})^{-1} \bar{E}^T (S^{(k+1)} - \mu_2 \nu^{(k+1)}) \right) \end{aligned} \tag{40c}$$

$$\xi^{(k+1)} = -\nu^{(k+1)} + \frac{1}{\mu_2}(S^{(k+1)} - Y^{(k+1)}), \tag{40d}$$

which is obtained by combining the update rules in (29), (30) and (39). Since the Lipschitz constants for both  $f_1$  and  $f_2$  are equal to 1, we can also choose  $\mu_1 = \mu_2 = 1$ . The resulting distributed feasibility algorithm can then be summarized in Algorithm 9.

---

**Algorithm 9** Alternating linearization method
 

---

```

1: Given  $Y^{(1)}$  and  $\xi^{(1)} = Y^{(1)} - P_{\mathcal{D}}(Y^{(1)})$ 
2: for  $k = 1, 2, \dots$  do
3:   for  $i = 1, 2, \dots, N$  do
4:      $s^{i,(k+1)} = \frac{1}{2} \left( y^{i,(k)} - \xi^{i,(k)} + P_{\bar{\mathcal{C}}_i}(y^{i,(k)} - \xi^{i,(k)}) \right)$ 
5:      $\nu^{i,(k+1)} = -\xi^{i,(k)} - (s^{i,(k+1)} - y^{i,(k)})$ 
6:     Communicate with all agents  $r$  belonging to  $\text{Ne}(i)$ 
7:     for all  $j \in J_i$  do
8:        $v_j^{(k+1)} = \frac{1}{|J_j|} \sum_{q \in J_j} \left( E_{J_q}^T (s^{q,(k+1)} - \nu^{q,(k+1)}) \right)_j$ 
9:     end for
10:     $y^{i,(k+1)} = \frac{1}{2} \left( s^{i,(k+1)} - \nu^{i,(k+1)} + v_{J_i}^{(k+1)} \right)$ 
11:     $\xi^{i,(k+1)} = -\nu^{i,(k+1)} + (s^{i,(k+1)} - y^{i,(k+1)})$ 
12:   end for
13: end for
```

---

**Remark 3** The authors in [23], propose another variant of Algorithm 4 that allows for non-smooth terms in the cost function, with similar convergence results. This enables us to use ALM for solving the formulation in (23) of the CFP. However, doing so recovers von Neumann's AP method.

#### 4.1.4. Fast ALM

Following the ideas from the derivation of Algorithm 9, we can also apply fast ALM to the problem in (23) as follows. Define  $\nu^{(k)} = \nabla f_1(S^{(k)})$ ,  $\xi^{(k)} = \nabla f_2(Y^{(k)})$  and similarly  $\beta^{(k)} = \nabla f_2(Z^{(k)})$ . From the optimality conditions of the 3<sup>rd</sup> and 4<sup>th</sup> steps of Algorithm 5, we have

$$\begin{aligned} \nabla f_1(S^{(k+1)}) + \frac{1}{\mu_1}(S^{(k+1)} - Z^{(k)}) + \nabla f_2(Z^{(k)}) &= 0 \\ \nabla f_2(Y^{(k+1)}) - \frac{1}{\mu_2}(S^{(k+1)} - Y^{(k+1)}) + \nabla f_1(S^{(k+1)}) &= 0. \end{aligned}$$

Also recall that  $\nabla f_2(Z^{(k)}) = Z^{(k)} - P_{\mathcal{D}}(Z^{(k)}) = Z^{(k)} - \bar{E}(\bar{E}^T \bar{E})^{-1} \bar{E}^T Z^{(k)}$  is linear with respect to the input argument, and hence, by the 6th step of Algorithm 5, we arrive at the following update rules

$$\begin{aligned} \nu^{(k+1)} &= \beta^{(k)} - \frac{1}{\mu_1}(S^{(k+1)} - Z^{(k)}) \\ \xi^{(k+1)} &= -\nu^{(k+1)} + \frac{1}{\mu_2}(S^{(k+1)} - Y^{(k+1)}) \\ \beta^{(k+1)} &= \xi^{(k+1)} + \frac{t^{(k)} - 1}{t^{(k+1)}}(\xi^{(k+1)} - \xi^{(k)}) \end{aligned} \tag{41}$$

Combining these with the resulting update rules obtained from applying Algorithm 5 to the problem in (23), yields

$$\begin{aligned} S^{(k+1)} &= \text{prox}_{\mu_1 f_1}(Z^{(k)} - \mu_1 \beta^{(k)}) \\ &= \frac{1}{\mu_1 + 1} \left( Z^{(k)} - \mu_1 \beta^{(k)} + \mu_1 P_{\mathcal{C}}(Z^{(k)} - \mu_1 \beta^{(k)}) \right) \end{aligned} \tag{42a}$$



$$\nu^{(k+1)} = -\beta^{(k)} - \frac{1}{\mu_1}(S^{(k+1)} - Z^{(k)}) \quad (42b)$$

$$\begin{aligned} Y^{(k+1)} &= \text{prox}_{\mu_2 f_2}(S^{(k+1)} - \mu_2 \nu^{(k+1)}) \\ &= \frac{1}{\mu_2 + 1} \left( S^{(k+1)} - \mu_2 \nu^{(k+1)} + \mu_2 P_{\mathcal{D}}(S^{(k+1)} - \mu_2 \nu^{(k+1)}) \right) \\ &= \frac{1}{\mu_2 + 1} \left( S^{(k+1)} - \mu_2 \nu^{(k+1)} + \mu_2 \bar{E}(\bar{E}^T \bar{E})^{-1} \bar{E}^T (S^{(k+1)} - \mu_2 \nu^{(k+1)}) \right) \end{aligned} \quad (42c)$$

$$\xi^{(k+1)} = -\nu^{(k+1)} + \frac{1}{\mu_2}(S^{(k+1)} - Y^{(k+1)}), \quad (42d)$$

$$t^{(k+1)} = \frac{1 + \sqrt{1 + t^{(k)2}}}{2} \quad (42e)$$

$$Z^{(k+1)} = Y^{(k+1)} + \frac{t^{(k)} - 1}{t^{(k+1)}}(Y^{(k+1)} - Y^{(k)}) \quad (42f)$$

$$\beta^{(k+1)} = \xi^{(k+1)} + \frac{t^{(k)} - 1}{t^{(k+1)}}(\xi^{(k+1)} - \xi^{(k)}), \quad (42g)$$

where similar to the previous algorithm, we can choose  $\mu_1 = \mu_2 = 1$ . This algorithm is summarized in Algorithm 10.

---

**Algorithm 10** Fast alternating linearization method

---

```

1: Given  $Z^{(1)} = Y^{(1)}$ ,  $\beta^{(1)} = Z^{(1)} - P_{\mathcal{D}}(Z^{(1)})$  and  $t^{(1)} = 1$ 
2: for  $k = 1, 2, \dots$  do
3:   for  $i = 1, 2, \dots, N$  do
4:      $s^{i,(k+1)} = \frac{1}{2} \left( z^{i,(k)} - \beta^{i,(k)} + P_{\bar{\mathcal{C}}_i}(z^{i,(k)} - \beta^{i,(k)}) \right)$ 
5:      $\nu^{i,(k+1)} = -\beta^{i,(k)} - (s^{i,(k+1)} - z^{i,(k)})$ 
6:     Communicate with all agents  $r$  belonging to  $\text{Ne}(i)$ 
7:     for all  $j \in J_i$  do
8:        $v_j^{(k+1)} = \frac{1}{|J_j|} \sum_{q \in J_j} \left( E_{J_q}^T (s^{q,(k+1)} - \nu^{q,(k+1)}) \right)_j$ 
9:     end for
10:     $y^{i,(k+1)} = \frac{1}{2} \left( s^{i,(k+1)} - \nu^{i,(k+1)} + v_{J_i}^{(k+1)} \right)$ 
11:     $\xi^{i,(k+1)} = -\nu^{i,(k+1)} + (s^{i,(k+1)} - y^{i,(k+1)})$ 
12:     $t^{(k+1)} = \frac{1 + \sqrt{1 + t^{(k)2}}}{2}$ 
13:     $z^{i,(k+1)} = y^{i,(k+1)} + \frac{t^{(k)} - 1}{t^{(k+1)}}(y^{i,(k+1)} - y^{i,(k)})$ 
14:     $\beta^{i,(k+1)} = \xi^{i,(k+1)} + \frac{t^{(k)} - 1}{t^{(k+1)}}(\xi^{i,(k+1)} - \xi^{i,(k)})$ 
15:   end for
16: end for
```

---

Remark 4 In [23], another variant of Algorithm 5 is suggested that can handle non-differentiable terms in the cost function and can deliver similar convergence rate results. This variant of the algorithm includes a skipping step which does not allow an efficient distributed implementation and requires global communication of iterates among all agents.

#### 4.1.5. Douglas-Rachford algorithm

We now apply the Douglas-Rachford algorithm to the minimization problem in (23). This results in the following update rules,

$$S^{(k+1)} = \frac{1}{\gamma + 1} \left( Y^{(k)} + \gamma P_{\mathcal{C}}(Y^{(k)}) \right)$$

$$\begin{aligned}
Y^{(k+1)} &= Y^{(k)} + \lambda^{(k)} \left( \text{prox}_{\gamma f_2}(2S^{(k+1)} - Y^{(k)}) - S^{(k+1)} \right) \\
&= Y^{(k)} + \lambda^{(k)} \\
&\quad \left( \frac{1}{\gamma + 1} \left( 2S^{(k+1)} - Y^{(k)} + \gamma P_{\mathcal{D}}(2S^{(k+1)} - Y^{(k)}) \right) - S^{(k+1)} \right) \\
&= Y^{(k)} + \lambda^{(k)} \\
&\quad \left( \frac{2}{\gamma + 1} S^{(k+1)} - \frac{1}{\gamma + 1} Y^{(k)} + \frac{\gamma}{\gamma + 1} P_{\mathcal{D}}(2S^{(k+1)} - Y^{(k)}) - S^{(k+1)} \right) \\
&= Y^{(k)} + \lambda^{(k)} \left( \frac{1 - \gamma}{\gamma + 1} S^{(k+1)} - \frac{1}{\gamma + 1} Y^{(k)} + \frac{\gamma}{\gamma + 1} P_{\mathcal{D}}(2S^{(k+1)} - Y^{(k)}) \right)
\end{aligned}$$

The resulting iterative algorithm is reported in Algorithm 11.

---

**Algorithm 11** Douglas-Rachford method

---

```

1: Given  $\varepsilon \in (0, 1)$ ,  $\gamma > 0$  and  $Y^{(1)}$ 
2: for  $k = 1, 2, \dots$  do
3:    $\lambda^{(k)} \in [\varepsilon, 2 - \varepsilon]$ 
4:   for  $i = 1, 2, \dots, N$  do
5:      $s^{i,(k+1)} = \frac{1}{\gamma+1} \left( y^{i,(k)} + \gamma P_{\bar{C}_i}(y^{i,(k)}) \right)$ 
6:      $z^{i,(k+1)} = 2s^{i,(k+1)} - y^{i,(k)}$ 
7:     Communicate with all agents  $r$  belonging to  $\text{Ne}(i)$ 
8:     for all  $j \in J_i$  do
9:        $v_j^{(k+1)} = \frac{1}{|J_j|} \sum_{q \in J_j} \left( E_{J_q}^T z^{q,(k+1)} \right)_j$ 
10:    end for
11:     $y^{i,(k+1)} = y^{i,(k)} + \lambda^{(k)} \left( \frac{1-\gamma}{\gamma+1} s^{i,(k+1)} - \frac{1}{\gamma+1} y^{i,(k)} + \frac{\gamma}{\gamma+1} v_{J_i}^{(k+1)} \right)$ 
12:  end for
13: end for

```

---

Note that this algorithm is similar to the method proposed method in [33] for large-scale distributed learning.

#### 4.2. Distributed Implementation of von Neumann's and Dykstra's AP method

Similar to the algorithms presented in sections 4.1.1–4.1.5, it is also possible to implement von Neumann's and Dykstra's AP methods in a distributed manner. The distributed version of these algorithms are presented in algorithms 12 and 13.

---

**Algorithm 12** Von Neumann's AP method

---

```

1: Given  $x^{(1)}$ 
2: for  $k = 1, 2, \dots$  do
3:   for  $i = 1, 2, \dots, N$  do
4:      $s^{i,(k+1)} = P_{\bar{C}_i} \left( v_{J_i}^{(k)} \right)$ 
5:     Communicate with all agents  $r$  belonging to  $\text{Ne}(i)$ 
6:     for all  $j \in J_i$  do
7:        $v_j^{(k+1)} = \frac{1}{|J_j|} \sum_{q \in J_j} \left( E_{J_q}^T s^{q,(k+1)} \right)_j$ 
8:     end for
9:   end for
10: end for

```

---

**Algorithm 13** Dykstra's AP method

---

```

1: Given  $x^{(1)}$  and  $\bar{\lambda}^{(1)} = 0$ 
2: for  $k = 1, 2, \dots$  do
3:   for  $i = 1, 2, \dots, N$  do
4:      $s^{i,(k+1)} = P_{\bar{C}_i} \left( v_{J_i}^{(k)} - \bar{\lambda}^{i,(k)} \right)$ 
5:     Communicate with all agents  $r$  belonging to  $\text{Ne}(i)$ 
6:     for all  $j \in J_i$  do
7:        $v_j^{(k+1)} = \frac{1}{|J_j|} \sum_{q \in J_j} \left( E_{J_q}^T s^{q,(k+1)} \right)_j$ 
8:     end for
9:      $\bar{\lambda}^{i,(k+1)} = \bar{\lambda}_i^{(k)} + \left( s^{i,(k+1)} - v_{J_i}^{(k+1)} \right)$ 
10:   end for
11: end for

```

---

**4.3. Local convergence tests**

In case (12) is feasible, algorithms 7–13 converge to a feasible solution of the problem. Local convergence tests check the convergence of the iterates to a feasible solution or detect infeasibility of the problem in a distributed manner with minimal communication. Unlike the so-called global tests, which require transmission of the local variables to a central unit, local methods only demand each agent to merely declare its local variables feasibility or convergence status with respect to its local constraints and/or objective value. Recall that for feasible problems applying the proposed algorithms to the formulations in (22) and (23) will generate sequences that converge to an optimal solution, [5, 14–16, 37], which yield zero objective value. Also for the case (12) is infeasible, all the proposed algorithms converge to a solution of the problems in (22) and (23), however, this solution does not result in zero objective value. Based on the aforementioned observation, we propose an approach for establishing convergence to a feasible solution or infeasibility of the problem. This approach is based on monitoring the convergence of the objective function value and the satisfaction of local constraints.

**4.3.1. Convergence of the objective value**

One of the ways to establish convergence of the proposed algorithms is through monitoring the so-called relative change of objective value (they intend to minimize), at each iteration. In case this quantity falls below a certain threshold, we can deduce convergence of the algorithm to a solution. Particularly, given a sequence  $\{d^{(k)}\}$ , the relative change of this sequence at iteration  $k + 1$  can be defined as

$$R^{(k+1)} = \frac{\|d^{(k+1)} - d^{(k)}\|}{\|d^{(k)}\|}. \quad (43)$$

For algorithms 7 and 8 that concern the problem in (22), we then monitor the following quantity

$$R_1^{(k+1)} = \frac{\left| \|S^{(k+1)} - P_C(S^{(k+1)})\|^2 - \|S^{(k)} - P_C(S^{(k)})\|^2 \right|}{\|S^{(k)} - P_C(S^{(k)})\|^2}. \quad (44)$$

This is because  $S^{(k)} \in \mathcal{D}$ ,  $\forall k \geq 1$ , for both algorithms. Monitoring this quantity locally, however, requires that all agents communicate their iterates to all other agents in the network. In order to alleviate this issue, we instead consider monitoring an upper bound

for  $R_1^{(k+1)}$ . Notice that

$$\begin{aligned}
 R_1^{(k+1)} &\leq \frac{\sum_{i=1}^N \left| \left\| s^{i,(k+1)} - P_{\bar{C}_i}(s^{i,(k+1)}) \right\|^2 - \left\| s^{i,(k)} - P_{\bar{C}_i}(s^{i,(k)}) \right\|^2 \right|}{\left\| S^{(k)} - P_C(S^{(k)}) \right\|^2} \\
 &\leq \sum_{i=1}^N \frac{\left| \left\| s^{i,(k+1)} - P_{\bar{C}_i}(s^{i,(k+1)}) \right\|^2 - \left\| s^{i,(k)} - P_{\bar{C}_i}(s^{i,(k)}) \right\|^2 \right|}{\left\| s^{i,(k)} - P_{\bar{C}_i}(s^{i,(k)}) \right\|^2} \\
 &=: \sum_{i=1}^N R_1^{i,(k+1)}.
 \end{aligned} \tag{45}$$

As can be seen from (45), this upper bound can now be monitored in a distributed manner. Then if all the local relative changes, i.e.,  $R_1^{i,(k+1)}$ , fall below a certain threshold, we can infer convergence of the algorithm to a solution. Hence, we can deduce convergence with little communication.

For algorithms 9–11, which concern the problem in (23), the monitored relative change takes the following form

$$R_2^{(k+1)} = \frac{|F_2(Y^{(k+1)}) - F_2(Y^{(k)})|}{F_2(Y^{(k)})} \tag{46}$$

where  $F_2(\cdot)$  is defined as in (23). The relative change  $R_2^{(k+1)}$  can also be bounded in a similar manner as (45) as follows

$$\begin{aligned}
 R_2^{(k+1)} &\leq \sum_{i=1}^N \frac{\left| \left\| y^{i,(k+1)} - P_{\bar{C}_i}(y^{i,(k+1)}) \right\|^2 - \left\| y^{i,(k)} - P_{\bar{C}_i}(y^{i,(k)}) \right\|^2 \right|}{\left\| y^{i,(k)} - P_{\bar{C}_i}(y^{i,(k)}) \right\|^2 + \left\| y^{i,(k)} - E_{J_i}C^{(k)} \right\|^2} \\
 &\quad + \frac{\left| \left\| y^{i,(k+1)} - E_{J_i}C^{(k+1)} \right\|^2 - \left\| y^{i,(k)} - E_{J_i}C^{(k)} \right\|^2 \right|}{\left\| y^{i,(k)} - P_{\bar{C}_i}(y^{i,(k)}) \right\|^2 + \left\| y^{i,(k)} - E_{J_i}C^{(k)} \right\|^2} \\
 &=: \sum_{i=1}^N R_2^{i,(k+1)}.
 \end{aligned} \tag{47}$$

where  $C^{(k)} = (E^T E)^{-1} E^T Y^{(k)}$ . As a result the convergence of algorithms 9–11 can also be established distributedly and with little communication. However, notice that for each agent to compute its local relative change at each iteration, i.e.,  $R_2^{i,(k+1)}$ , additional communication among agents is required. This additional communication is required for computation of  $C^{(k)}$ .

#### 4.3.2. Feasibility of local constraints

In case the CFP is feasible, all the proposed algorithms converge to a feasible solution. We can detect arrival at a feasible solution distributedly, by checking the feasibility of local constraints. If at a certain iteration, the local iterates of all agents satisfy their corresponding local constraints and if furthermore we have global consensus over the network, i.e., (11c) is satisfied, we can infer that we have converged to a feasible solution.

For algorithms 7 and 8, the iterate  $S^{(k)}$  already satisfy the global consensus constraints. Hence, the feasibility detection at each iteration (for these algorithms) requires

each agent  $i$  to check whether  $s^{i,(k)} \in \bar{\mathcal{C}}_i$ . In case this is satisfied for all  $i = 1, \dots, N$ , we can then infer arrival at a feasible solution. For algorithms 9–11, however, this test is slightly more complicated. This is because the iterate  $Y^{(k)}$  does not necessarily satisfy the global consensus constraints. Consequently, when  $y^{i,(k)} \in \bar{\mathcal{C}}_i$  for all  $i = 1, \dots, N$ , the agents would still need to communicate with their neighbors to check whether global consensus is reached or not. Then, in case local constraints for all agents and global consensus constraints are both satisfied, we can infer arrival at a feasible solution.

By combining the two methods for detecting convergence of the objective value and arrival at a feasible solution, we can now describe a distributed framework for establishing convergence to a solution as follows. At each iteration, each agent should

- (1) check the feasibility of its local iterates with respect to their corresponding local constraints. If all agents are locally feasible, communicate with neighbors to check the satisfaction of the global consensus constraint (this only applies to algorithms 9–11);
- (2) check whether the local relative change has fallen below the predefined threshold.

Then,

- if condition (1) is satisfied for all agents, the algorithm has converged to a feasible solution;
- if condition (2) is satisfied for all agents, the algorithm has converged and in case there exists an agent with non-zero local objective value, the CFP is infeasible.

**Remark 5** The convergence of algorithms 12 and 13, can also be established in a similar manner. We refer to [28], for details of the corresponding convergence detection framework.

## 5. Convergence Rate

In this section, we investigate the convergence results for the algorithms presented in Section 4. We are particularly interested in the possibility of unifying the convergence rate results for proximal methods with the existing results for projection methods, which were discussed in Section 2. The convergence rate results for projection methods, see (8) and (15), are based on the distance of the iterates to the feasible set and are proven under the assumption that the underlying sets are boundedly linearly regular, (or that Slater's conditions are satisfied). In order to unify these results with convergence rate results for proximal splitting methods, we study the convergence of the algorithms presented in Section 4 by investigating the feasible and infeasible cases separately.

### 5.1. Feasible problem

Throughout this section we assume that the CFP in (12) is feasible and its underlying constraint sets are boundedly linearly regular, i.e., they satisfy (5).

#### 5.1.1. Forward-backward splitting

In this subsection we focus on algorithms 7 and 8, which are obtained by applying forward-backward splitting to (22). As was mentioned in sections 4.1.1 and 4.1.2, in these algorithms, the iterate  $S^{(k)} \in \mathcal{D}$  for all  $k \geq 1$ . Hence,  $\text{dist}(S^{(k)}, \mathcal{D}) = 0$  and

$$F(S^{(k)}) = \frac{1}{2} \left\| S^{(k)} - P_{\mathcal{C}}(S^{(k)}) \right\|^2 + \mathcal{I}_{\mathcal{D}}(S^{(k)}) \quad (48)$$

$$= \frac{1}{2} \left\| S^{(k)} - P_{\mathcal{C}}(S^{(k)}) \right\|^2. \quad (49)$$

Assuming bounded linear regularity of the problem in (12), we then have

$$\begin{aligned} \text{dist}\left(S^{(k)}, \mathcal{C} \cap \mathcal{D}\right) &\leq \theta_B \max\left\{\text{dist}\left(S^{(k)}, \mathcal{C}\right), \text{dist}\left(S^{(k)}, \mathcal{D}\right)\right\} \\ &= \theta_B \text{dist}\left(S^{(k)}, \mathcal{C}\right) \end{aligned}$$

Consequently and by (48), for algorithms 7 and 8 we have that

$$\text{dist}\left(S^{(k)}, \mathcal{C} \cap \mathcal{D}\right) \leq \mathcal{O}\left(\frac{1}{\sqrt{k}}\right), \quad (50)$$

and

$$\text{dist}\left(S^{(k)}, \mathcal{C} \cap \mathcal{D}\right) \leq \mathcal{O}\left(\frac{1}{k}\right), \quad (51)$$

respectively.

### 5.1.2. ALM splitting

Recall that algorithms 9 and 10 are obtained by applying ALM and fast ALM to the problem in (23). Assuming bounded linear regularity of the CFP, we then arrive at

$$\begin{aligned} \text{dist}^2\left(Y^{(k)}, \mathcal{C} \cap \mathcal{D}\right) &\leq \theta_B^2 \max\left\{\text{dist}^2\left(Y^{(k)}, \mathcal{C}\right), \text{dist}^2\left(Y^{(k)}, \mathcal{D}\right)\right\} \\ &\leq \left\|Y^{(k)} - P_{\mathcal{C}}(Y^{(k)})\right\|^2 + \left\|Y^{(k)} - P_{\mathcal{D}}(Y^{(k)})\right\|^2 \\ &= 2F(Y^{(k)}). \end{aligned}$$

Note that in case the problem in (12) is feasible  $F(S^*) = 0$  and hence by (31) and (32), we have convergence rate results

$$\text{dist}\left(Y^{(k)}, \mathcal{C} \cap \mathcal{D}\right) \leq \mathcal{O}\left(\frac{1}{\sqrt{k}}\right), \quad (52)$$

and

$$\text{dist}\left(Y^{(k)}, \mathcal{C} \cap \mathcal{D}\right) \leq \mathcal{O}\left(\frac{1}{k}\right), \quad (53)$$

for algorithms 9 and 10, respectively.

## 5.2. Infeasible problem

The convergence results for algorithms 7–10, are not affected by the feasibility or infeasibility of the CFP in (12). Hence, we can consider the cost function of the problems in (22) and (23), as a measure for detecting infeasibility. This is similar to the measures used for detecting infeasibility for von Neumann's and Dykstra's AP methods, where the sequences  $\|V^{(k)} - S^{(k)}\|$  and  $\|V^{(k)} - S^{(k+1)}\|$  converge to  $\text{dist}(\mathcal{C}, \mathcal{G})$ . However, recall that the rate of convergence of the mentioned sequences is not established. This is in contrast to algorithms 7–10, where we can use the existing results on convergence rate of the objective value of algorithms 1–5, to provide convergence results on certain residuals that can assist us in detecting infeasibility of the problem.

### 5.2.1. Forward-backward splitting

Since for algorithms 7 and 8, the iterate  $S^{(k)} \in \mathcal{D}$  for all  $k \geq 1$ , even when  $\mathcal{C} \cap \mathcal{D} = \emptyset$ , we have

$$F(S^{(k)}) = \frac{1}{2} \|S^{(k)} - P_{\mathcal{C}}(S^{(k)})\|^2.$$

Recall that if the problem in (12) is infeasible, the optimal objective value for the problem in (22) will be nonzero. Hence we can establish the infeasibility of the problem, by monitoring the residual  $\|S^{(k)} - P_{\mathcal{C}}(S^{(k)})\|^2$ , which will converge to  $\text{dist}^2(S^*, \mathcal{C}) = \text{dist}^2(\mathcal{C}, \mathcal{D})$ . By the convergence results in (26) and (27), we know that for algorithms 7 and 8, the rates of convergence of this residual are of  $\mathcal{O}(1/k)$  and  $\mathcal{O}(1/k^2)$ , respectively.

### 5.2.2. ALM splitting

For algorithms 9 and 10, we can also draw similar conclusions. Recall that in case the problem in (12) is infeasible, the optimal objective value for the problem in (23) will be nonzero. Hence, we can deduce infeasibility of the problem by monitoring the convergence of the objective value of the problem in (23), i.e.,

$$\|Y^{(k)} - P_{\mathcal{C}}(Y^{(k)})\|^2 + \|Y^{(k)} - P_{\mathcal{D}}(Y^{(k)})\|^2,$$

which, by (31) and (32), is known to converge with  $\mathcal{O}(1/k)$  and  $\mathcal{O}(1/k^2)$  convergence rates, respectively.

**Remark 6** Among algorithms 7–11, the ones based on the accelerated forward-backward and fast ALM, i.e., algorithms 8 and 10, have better convergence properties. However, Algorithm 8 has a practical advantage over the other. This is because, in order to detect convergence and arrival at a feasible solution, Algorithm 10 requires more communication with neighbors and also a more sophisticated approach to do so. Hence, and purely based on the discussions in sections 4.3 and 5, we expect Algorithm 8 to outperform the rest of the described methods.

## 6. Numerical Results

In this section, we apply algorithms 7–13 to a class of flow feasibility problems, and compare their performance when solving these problems. In section 6.1, we first describe the considered flow feasibility problem and we then apply the algorithms to feasible and infeasible flow problems in sections 6.2 and 6.3, respectively.

### 6.1. Flow Feasibility Problem

Let  $G(\mathcal{V}, \mathcal{E})$  be a directed graph, where  $\mathcal{V} = \{1, \dots, N\}$  is the set of its vertices and  $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$  is the set of its edges. Two nodes  $i$  and  $j$  are adjacent if  $(i, j) \in \mathcal{E}$ , and the set of the  $i$ th node's adjacent nodes are denoted as  $\text{adj}(i)$ . Also let the nodes  $u, o \in \mathcal{V}$  be the so-called source and sink nodes of the graph, respectively. Assume that we inject a flow  $U$  to the source node. The flow feasibility problem then corresponds to the problem of assessing whether it is possible to relay  $U$  from the source node to the sink node, by assigning flows to different edges in the graph and without violating flow constraints. These constraints mainly describe how different nodes in the network are allowed to relay the input flow from the source node to the sink node, by assigning flows to their

edges. Let  $f_j^i$  denote the flow assigned to the edge  $(i, j) \in \mathcal{E}$ . Notice that since  $f_j^i$  and  $f_i^j$  correspond to the flow within the same edge, we have

$$f_j^i = f_i^j, \quad \forall (i, j) \in \mathcal{E}. \quad (54)$$

The constraints in the flow feasibility problem can then be expressed as follows.

- (1) The flow within each edge should be nonnegative and should not exceed its maximum capacity, i.e.,

$$0 \leq f_j^i \leq c_{ij}, \quad \forall (i, j) \in \mathcal{E},$$

where  $c_{ij}$  denotes the maximum capacity of the  $(i, j)$  edge, and naturally  $c_{ij} = c_{ji}$ .

- (2) All nodes should satisfy the conservation of flow at all time. In other words, the sum of flows entering a node should be equal to the sum of flows leaving a node, i.e., for all nodes  $i \in \mathcal{V} \setminus \{u, o\}$

$$\sum_{j \in \text{adj}(i) \setminus \mathcal{O}(i)} f_j^i = \sum_{j \in \mathcal{O}(i)} f_j^i,$$

where  $\mathcal{O}(i)$  denotes the set of  $i$ th nodes' adjacent nodes that receive flow from this node. For the nodes  $u$  and  $o$ , this entails

$$\sum_{j \in \text{adj}(u) \setminus \mathcal{O}(u)} f_j^u + U = \sum_{j \in \mathcal{O}(u)} f_j^u,$$

and

$$\sum_{j \in \text{adj}(o) \setminus \mathcal{O}(o)} f_j^o = \sum_{j \in \mathcal{O}(o)} f_j^o + U,$$

respectively.

- (3) The sum of flows leaving a node should not exceed an internal nodal capacity (which could be private to the node), i.e., for all nodes  $i \in \mathcal{V} \setminus \{o\}$ ,

$$\sum_{j \in \mathcal{O}(i)} f_j^i \leq n_i,$$

where  $n_i$  is the  $i$ th node nodal capacity, and for the node  $o$ , this entails

$$\sum_{j \in \mathcal{O}(o)} f_j^o + U \leq n_o.$$

Having described the constraints of the flow feasibility problem, for  $i \in \mathcal{V} \setminus \{u, o\}$ , define

$$\bar{C}_i = \left\{ s^i \left| \begin{array}{l} \sum_{j \in \text{adj}(i) \setminus \mathcal{O}(i)} f_j^i = \sum_{j \in \mathcal{O}(i)} f_j^i \\ \sum_{j \in \mathcal{O}(i)} f_j^i \leq n_i \\ 0 \leq f_j^i \leq c_{ij} \quad \forall j \in \text{adj}(i) \end{array} \right. \right\}. \quad (55)$$



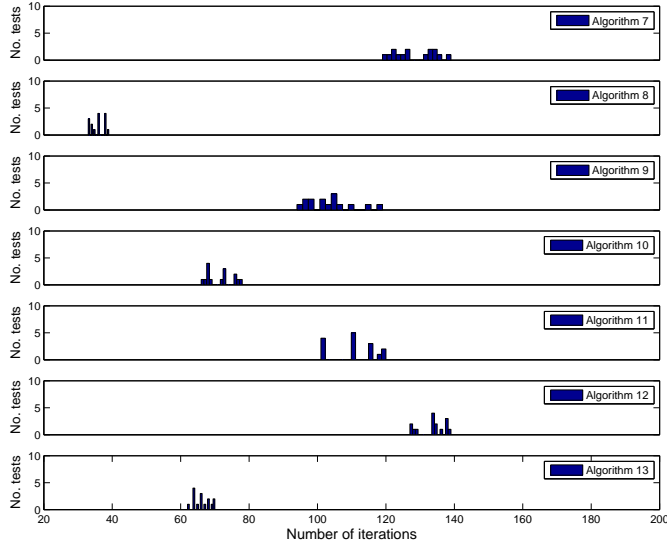


Figure 1. Number of required iterations for each algorithm to converge to a feasible solution. The figure illustrates the results achieved for all the 15 randomly generated examples.

Similarly for  $i \in \{u, o\}$ , define

$$\bar{C}_u = \left\{ s^u \left| \begin{array}{l} \sum_{j \in \text{adj}(u) \setminus \mathcal{O}(u)} f_j^u + U = \sum_{j \in \mathcal{O}(u)} f_j^u \\ \sum_{j \in \mathcal{O}(u)} f_j^u \leq n_u \\ 0 \leq f_j^u \leq c_{uj} \quad \forall j \in \text{adj}(u) \end{array} \right. \right\}, \quad (56)$$

and

$$\bar{C}_o = \left\{ s^o \left| \begin{array}{l} \sum_{j \in \text{adj}(o) \setminus \mathcal{O}(o)} f_j^o = \sum_{j \in \mathcal{O}(o)} f_j^o + U \\ \sum_{j \in \mathcal{O}(o)} f_j^o + U \leq n_o \\ 0 \leq f_j^o \leq c_{oj} \quad \forall j \in \text{adj}(o) \end{array} \right. \right\}, \quad (57)$$

where for each  $i \in \mathcal{V}$ ,  $s^i$  is the vector of flows of all the edges that are connected to the  $i$ th node. Notice that the sets  $\bar{C}_i$ , for  $i = 1, \dots, N$ , are decoupled, and (54) describes the coupling among these constraints. In other words, (54) defines the global consensus constraints. With this definition of  $s^i$ s and  $\bar{C}_i$ s, the flow feasibility problem has the same format as the problem in (12), where  $\mathcal{D}$  is given by (54). Next, we apply algorithms 7–13, to this class of flow feasibility problems.

## 6.2. Feasible Flow Problem

In this section, we will study and compare the performance of algorithms 7–13, when they are applied to a set of flow feasibility problems. To this end, we pose flow feasibility problems over 15 connected directed graphs with 60 nodes. The source and sink nodes

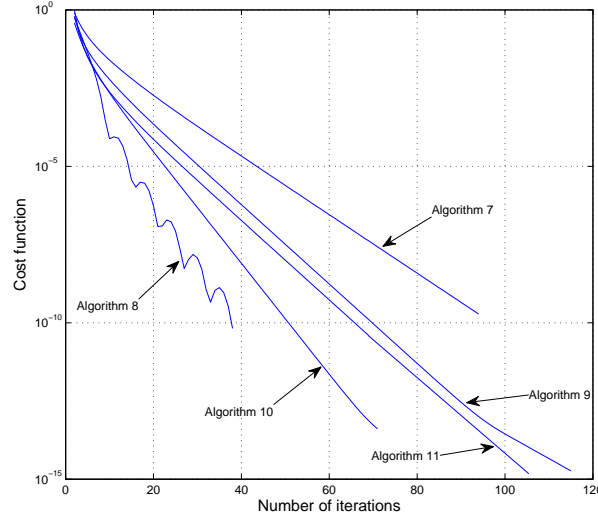


Figure 2. The evolution of the cost function being minimized for algorithms 7–11.

of each of these graphs have been chosen to be the nodes with the most number of outgoing and ingoing edges, respectively. These graphs have been generated using the algorithm presented in Appendix A. The number of variables in the feasibility problems that correspond to the generated flow problems, i.e., the number of local and global variables, varies within the range of [936, 1146]. Also for the sake of simplicity, we have chosen the edge capacities to be equal for all edges and also we have chosen the nodal capacities to be a proportional to the number of outgoing edges from each node, i.e., given a graph  $G(\mathcal{V}, \mathcal{E})$ ,  $c_{ij} = \bar{c}$  for all  $(i, j) \in \mathcal{E}$ , and  $n_i = |\mathcal{O}(i)|\bar{n}$  for all  $i \in \mathcal{V}$ . In order to assure feasibility of the generated problems, we have chosen the input flow to the source node, edge capacities and nodal capacities such that the network is capable of relaying the input flow to the sink node. Particularly, for the examples considered in this section, we have used the following procedure for finding suitable input flow, edge and nodal capacities. We first set the initial values for the input flow and the capacities as  $U = 100$ ,  $\bar{c} = 10$  and  $n_i = |\mathcal{O}(i)|\bar{c}/2$  for all  $i = 1, \dots, N$ . If the resulting flow problem is feasible, then the initial values are deemed to be suitable. Otherwise, we

- (1) set  $U := U/2$ .
- (2) then check if the resulting flow problem is feasible; in which case we consider the current values of  $\bar{c}$ ,  $n_i$ s and  $U$  as the chosen ones. In case the problem was still infeasible, we
  - set  $\bar{c} := 2\bar{c}$ .
  - set  $n_i := |\mathcal{O}(i)|\bar{c}/2$  for all  $i = 1, \dots, N$ .
  - again check whether the resulting flow problem is feasible or not. In case the problem is feasible we have found the suitable values for  $U$ ,  $\bar{c}$  and  $\bar{n}$ . Otherwise, we continue by going back to step (1) of the scheme.

For algorithms 8–10 that do not have any tuning parameters (except for  $\theta^0$  in Algorithm 8 which is chosen to be 1) we have applied the algorithms as they are. For algorithms 7 and 11 that do contain tuning parameters, these parameters are chosen such that the algorithms achieve their best performance for each specific example. However, we have not considered time varying parameters in these algorithms. Figure 1 illustrates the obtained results from this experiment. The figure shows the number of required iterations for each algorithm to converge to a feasible solution. In order to detect convergence to a feasible solution, we have utilized the proposed approach in Section 4.3,

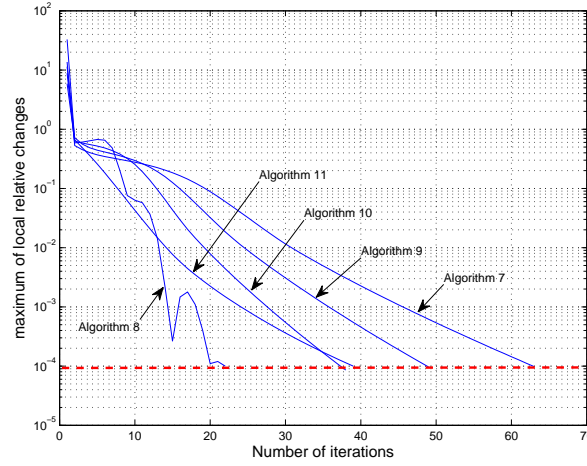


Figure 3. The maximum of local relative changes over the network for algorithms 7–11 when the flow feasibility problem is infeasible.

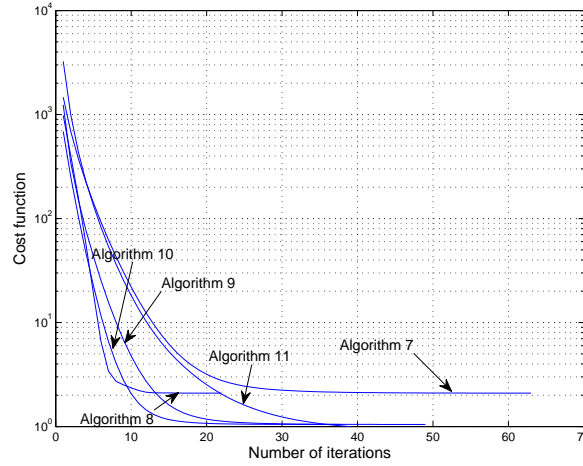


Figure 4. The evolution of the cost function being minimized for algorithms 7–11 when the flow feasibility problem is infeasible.

where the threshold for convergence detection based on local relative changes was set to  $10^{-4}$ . However, for all examples, the convergence to feasible solutions was established using the condition concerning the feasibility of local constraints. This was because this condition was satisfied prior to the convergence of the objective value. As can be seen from Figure 1, Algorithm 8 clearly outperforms the rest of the algorithms, followed by algorithms 13 and 10. Figure 2, illustrates the performance of algorithms 7–11 in minimizing their corresponding cost functions, for one specific example. As was expected, algorithms 8 and 10, outperform the rest of the algorithms.

### 6.3. Infeasible Flow Problem

In this experiment, we consider the case when the flow feasibility problem is infeasible, where we also use a similar setup as in Section 6.2. Particularly, we randomly generate a connected directed graph with 60 nodes, however, we design the problem such that the resulting flow problem is infeasible. To be more specific, we reduce the capacity of the edges and increase the input flow to the network, up to a point that we exceed the

relaying capabilities of the network. The procedure that we used for finding suitable values for  $U$ ,  $\bar{c}$  and  $n_i$ s for this purpose, is similar to the approach discussed in Section 6.2. In fact, the only difference is in the steps when we change the values for  $U$  and  $\bar{c}$ , where we instead set these values as  $U := 2U$  and  $\bar{c} := \bar{c}/2$ .

Note that, since the flow feasibility problem is infeasible, the convergence can only be established through monitoring the local relative changes. Figure 3 depicts the maximum of local relative changes across the network, for algorithms 7–11. The dashed line in the figure, illustrates the threshold for convergence detection. As can be seen from the figure, Algorithm 8 outperforms the other algorithms and converges within 22 iterations. This is followed by algorithms 10 and 11 which converged within 38 and 39 iterations, respectively. It is worth mentioning that, Dykstra’s AP method also converged to a solution in 39 iterations, however, von Neumann’s AP method required 892 iterations to converge. Figure 4 depicts the behavior of algorithms 7–11 when applied to an infeasible problem. Unlike the behavior that was observed in Figure 2, the cost function sequence, now, converges to a nonzero constant.

## 7. Conclusions

In this paper, we presented several algorithms for solving loosely coupled convex feasibility problems distributedly and efficiently. These algorithms were the result of application of proximal splitting methods to convex minimization reformulations of product-space formulation of such CFPs. We also proposed a distributed feasibility/infeasibility detection scheme that require little communication among the agents. Furthermore, through the use of convergence rate results for proximal splitting methods, we provided a unified treatment of the convergence rate analysis of the proposed algorithms and of classical projection methods. We also studied the performance of the proposed algorithms using numerical experiments which illustrated that Algorithm 8 outperforms the rest of the algorithms.

A possible shortcoming of the presented algorithms can be in handling infeasible problems where  $\text{dist}(\mathcal{C}, \mathcal{D})$  is extremely small. In such cases and due to sub-linear convergence properties of these algorithms, they can require many iteration to converge (though, this was not observed in any of the 15 examples). As future research direction, we intend to further investigate this problem and devise possible remedies, e.g., by using more advanced (primal-dual) splitting methods.

## Appendix A. An Algorithm for Random Generation of Connected Directed Graphs

In this appendix, we present an algorithm for generating connected and directed graphs,  $G(\mathcal{V}, \mathcal{E})$ , with  $N$  vertices and with adjacency matrix  $A \in \mathbb{R}^{N \times N}$  given by

$$A_{ij} = \begin{cases} 0 & (i, j) \notin \mathcal{E} \\ 1 & (i, j) \in \mathcal{E} \text{ the edge is leaving the } i\text{th node} \\ -1 & (i, j) \in \mathcal{E} \text{ the edge is entering the } i\text{th node} \end{cases}. \quad (\text{A1})$$

Notice that, by this definition,  $A$  is skew-symmetric. Next, we describe an algorithm, that allows us to randomly generate the adjacency matrix of connected and directed graphs.

**Algorithm A1** Random Generation of Connected Directed Graphs

---

```

1: Given  $N$ ,  $F_1 = 1$ ,  $Iter_m = 1000$  and  $A$  a  $N \times N$  zero matrix
2: for  $i = 1 : Iter_m$  do
3:   for  $i = N - 1 : -1 : 1$  do
4:     Set  $F_2 = 1$  and  $F_3 = 1$ 
5:     while  $F_2 == 1$  do
6:       Generate a random  $1 \times i$  0-1 vector,  $x$ .
7:       if Number of nonzero elements in  $[A(N - i, 1 : N - i) x]$  is larger than 2 then
8:         Set  $F_2 = 0$ 
9:       end if
10:    end while
11:    while  $F_3 == 1$  do
12:      Randomly assign a sign to nonzero elements in  $x$ .
13:      if There exists both positive and negative elements in  $[A(N - i, 1 : N - i) x]$  then
14:        Set  $F_3 = 0$ 
15:         $A(N - i, N - i + 1 : N) = x$ 
16:         $A(N - i + 1 : N, N - i) = x^T$ 
17:      end if
18:    end while
19:  end for
20:  if There exists both positive and negative elements in  $A(N, 1 : N)$  then
21:    break
22:  end if
23: end for

```

---

Note that this algorithm can fail to generate a suitable adjacency matrix at each run, and one must continue on running the algorithm until it satisfies the conditions in the algorithm. The proposed algorithm is not an efficient method for generating connected directed graphs, and it is merely a simple methodology that we used in Section 6 for generating such graphs.

**References**

- [1] H.H. Bauschke and J.M. Borwein, *On the convergence of von Neumann's alternating projection algorithm for two sets*, Set-Valued Analysis 1 (1993), pp. 185–212.
- [2] H.H. Bauschke and J.M. Borwein, *Dykstra's alternating projection algorithm for two sets*, Journal of Approximation Theory 79 (1994), pp. 418–443.
- [3] H.H. Bauschke and J.M. Borwein, *On projection algorithms for solving convex feasibility problems*, SIAM Rev. 38 (1996), pp. 367–426.
- [4] H.H. Bauschke, J.M. Borwein, and W. Li, *Strong conical hull intersection property, bounded linear regularity, jameson's property (g), and error bounds in convex optimization*, Mathematical Programming 86 (1999), pp. 135–160.
- [5] H.H. Bauschke and P.L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, Springer, 2011.
- [6] A. Beck and M. Teboulle, *Convergence rate analysis and error bounds for projection algorithms in convex feasibility problems*, Optimization Methods and Software 18 (2003), pp. 377–394.
- [7] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM J. Img. Sci. 2 (2009), pp. 183–202.
- [8] S.R. Becker, E.J. Candes, and M.C. Grant, *Templates for convex cone problems with applications to sparse signal recovery*, Mathematical Programming Computation 3 (2011), pp. 165–218.
- [9] D.P. Bertsekas and J.N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Athena Scientific, 1997.
- [10] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Foundations and Trends in Machine Learning 3 (2011), pp. 1–122.
- [11] D. Butnariu, A.N. Iusem, and R.S. Burachik, *Iterative methods of solving stochastic convex feasibility problems and applications*, Computational Optimization and Applications 15 (2000), pp. 269–307.
- [12] Y. Censor and T. Elfving, *A multiprojection algorithm using bregman projections in a product space*, Numerical Algorithms 8 (1994), pp. 221–239.
- [13] Y. Censor and A. Lent, *Cyclic subgradient projections*, Mathematical Programming 24 (1982), pp. 233–235.

- [14] P. Combettes and J.C. Pesquet, *A Douglas-Rachford splitting approach to nonsmooth convex variational signal recovery*, IEEE Journal of Selected Topics in Signal Processing 1 (2007), pp. 564–574.
- [15] P.L. Combettes and J.C. Pesquet, *Proximal splitting methods in signal processing*, in *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, Springer Optimization and Its Applications, Vol. 49, Springer New York, 2011, pp. 185–212.
- [16] P.L. Combettes and V.R. Wajs, *Signal recovery by proximal forward-backward splitting*, Multiscale Modeling and Simulation 4 (2005), pp. 1168–1200.
- [17] A.R. De Pierro, *From parallel to sequential projection methods and vice versa in convex feasibility: Results and conjectures*, in *Inherently Parallel Algorithms in Feasibility and Optimization and their Applications*, D. Butnariu, Y. Censor, and S. Reich, eds., Studies in Computational Mathematics, Vol. 8, Elsevier, 2001, pp. 187–201.
- [18] A.R. De Pierro and A.N. Iusem, *A parallel projection method for finding a common point of a family of convex sets*, Pesquisa Operacional 5 (1985), pp. 1–20.
- [19] W. Deng and W. Yin, *On the global and linear convergence of the generalized alternating direction method of multipliers*, Tech. Rep. technical report 12-14, Rice University, CAAM, 2012.
- [20] J. Eckstein and D.P. Bertsekas, *On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators*, Mathematical Programming 55 (1992), pp. 293–318.
- [21] D. Gabay, *Applications of the method of multipliers to variational inequalities*, in *Augmented Lagrangian Methods: Applications to the Solution of Boundary-Value Problems*, M. Fortin and R. Glowinski, eds., North-Holland, 1983.
- [22] R. Glowinski and P. Le Tallec, *Augmented Lagrangian and Operator-Splitting Methods in Nonlinear Mechanics*, Society for Industrial and Applied Mathematics, 1989.
- [23] D. Goldfarb, S. Ma, and K. Scheinberg, *Fast alternating linearization methods for minimizing the sum of two convex functions*, Mathematical Programming (2012), pp. 1–34.
- [24] T. Goldstein, B. ODonoghue, and S. Setzer, *Fast alternating direction optimization methods*, Tech. Rep., Stanford university, 2012.
- [25] L.C. Gubin, B.T. Polyak, and E.V. Raik, *The method of projections for finding the common point of convex sets*, USSR Computational Math. Math. Phys. 7 (1967), pp. 1–24.
- [26] G. Herman, *Image reconstruction from projections*, Real-Time Imaging 1 (1995), pp. 3–18.
- [27] A.N. Iusem and A.R. De Pierro, *Convergence results for an accelerated nonlinear Cimmino algorithm*, Numerische Mathematik 49 (1986), pp. 367–378.
- [28] S. Khoshfetrat Pakazad, M.S. Andersen, A. Hansson, and A. Rantzer, *Decomposition and Projection Methods for Distributed Robustness Analysis of Interconnected Uncertain Systems*, in *Proceedings of the 13th IFAC Symposium on Large Scale Complex Systems: Theory and Applications*, 2013.
- [29] K. Kiwiel, C. Rosa, and A. Ruszczynski, *Proximal decomposition via alternating linearization*, SIAM Journal on Optimization 9 (1999), pp. 668–689.
- [30] J. Lawrence and J. Spingarn, *On fixed points of non-expansive piecewise isometric mappings*, in *Proceedings of the London Mathematical Society*, Vol. 55, 1987, pp. 605–624.
- [31] E. Levitin and B. Polyak, *Constrained minimization methods*, Zhurnal Vychislitel’noi Matematiki i Matematicheskoi Fiziki 6 (1966), pp. 787–823.
- [32] J. Pang, *Error bounds in mathematical programming*, Math. Program. 79 (1997), pp. 299–332.
- [33] N. Parikh and S. Boyd, *Block splitting for large-scale distributed learning*, in *Neural Information Processing Systems (NIPS), Workshop on Big Learning*, 2011.
- [34] G. Pierra, *Decomposition through formalization in a product space*, Mathematical Programming 28 (1984), pp. 96–115.
- [35] B. Polyak, *Minimization of unsmooth functionals*, USSR Computational Mathematics and Mathematical Physics 9 (1969), pp. 14–29.
- [36] L.T.D. Santos, *A parallel subgradient projections method for the convex feasibility problem*, Journal of Computational and Applied Mathematics 18 (1987), pp. 307–320.
- [37] P. Tseng, *On accelerated proximal gradient methods for convex-concave optimization*, Submitted to SIAM Journal on Optimization (2008).