Decomposition and Projection Methods for Distributed Robustness Analysis of Interconnected Uncertain Systems

Sina Khoshfetrat Pakazad, Martin S. Andersen, Anders Hansson, Anders Rantzer

Division of Automatic Control
E-mail: sina.kh.pa@isy.liu.se, martin.andersen@isy.liu.se, hansson@isy.liu.se, anders.rantzer@control.lth.se

2nd December 2011

Report no.: LiTH-ISY-R-3033
Submitted to Technical Report

Address:
Department of Electrical Engineering
Linköpings universitet
SE-581 83 Linköping, Sweden

WWW: http://www.control.isy.liu.se
Abstract

We consider a class of convex feasibility problems where the constraints that describe the feasible set are loosely coupled. These problems arise in robust stability analysis of large, weakly interconnected systems. To facilitate distributed implementation of robust stability analysis of such systems, we propose two algorithms based on decomposition and simultaneous projections. The first algorithm is a nonlinear variant of Cimmino’s mean projection algorithm, but by taking the structure of the constraints into account, we can obtain a faster rate of convergence. The second algorithm is devised by applying the alternating direction method of multipliers to a convex minimization reformulation of the convex feasibility problem. We use numerical results to show that both algorithms require far less iterations than the accelerated nonlinear Cimmino algorithm.

Keywords: Robust stability analysis, convex feasibility problems, projection algorithms, decomposition, distributed computing.
Decomposition and Projection Methods for Distributed Robustness Analysis of Interconnected Uncertain Systems

Sina Khoshfetrat Pakazad, Martin S. Andersen, Anders Hansson and Anders Rantzer

2011-12-02

Abstract

We consider a class of convex feasibility problems where the constraints that describe the feasible set are loosely coupled. These problems arise in robust stability analysis of large, weakly interconnected systems. To facilitate distributed implementation of robust stability analysis of such systems, we propose two algorithms based on decomposition and simultaneous projections. The first algorithm is a nonlinear variant of Cimmino’s mean projection algorithm, but by taking the structure of the constraints into account, we can obtain a faster rate of convergence. The second algorithm is devised by applying the alternating direction method of multipliers to a convex minimization reformulation of the convex feasibility problem. We use numerical results to show that both algorithms require far less iterations than the accelerated nonlinear Cimmino algorithm.
Abstract—We consider a class of convex feasibility problems where the constraints that describe the feasible set are loosely coupled. These problems arise in robust stability analysis of large, weakly interconnected systems. To facilitate distributed implementation of robust stability analysis of such systems, we propose two algorithms based on decomposition and simultaneous projections. The first algorithm is a nonlinear variant of Cimmino’s mean projection algorithm, but by taking the structure of the constraints into account, we can obtain a faster rate of convergence. The second algorithm is devised by applying the alternating direction method of multipliers to a convex minimization reformulation of the convex feasibility problem. We use numerical results to show that both algorithms require far less iterations than the accelerated nonlinear Cimmino algorithm.

Index Terms—Robust stability analysis, convex feasibility problems, projection algorithms, decomposition, distributed computing.

I. INTRODUCTION

DISTRIBUTED and parallel solutions for control problems are of great importance. Prioritizing such approaches over the centralized solutions is mainly motivated by structural or computational reasons, [1]–[5]. Such solutions are common in the control of distributed interconnected systems, which appear in many practical applications, [2], [5]–[10]. Considering the fact that many of the control design methods are model based, they are vulnerable to model uncertainty. Some of these design methods consider the model uncertainty, but many, specially the ones employed in industry, neglect it. As a result, for such methods it is important to address the stability of the closed loop system with respect to the model uncertainty.

Methods for robust stability analysis of uncertain systems has been studied thoroughly, e.g., see [11]–[14]. One of popular methods for robust stability analysis is the so called μ-analysis, an approximation of which solves a Convex Feasibility Problem, CFP, for each frequency on a finite grid of frequencies. For networks of interconnected uncertain systems, each CFP is a global feasibility problem that involves the entire network. In other words, each CFP depends on all the local models, i.e., the models associated with the subsystems in the network, as well as the network topology. In networks where all the local models are available to a central unit and the size of the problem is not computationally prohibitive, each feasibility problem can be solved efficiently in a centralized manner. However distributed or parallel methods are desirable in networks, for instance, where the interconnected system dimension is large or where the local models are either private or available to only a small number of subsystems in the network, [15]–[18]. In [15]–[17], for specific network structures, the system matrix is diagonalized and decoupled using decoupling transformations. This decomposes the analysis and design problem and provides the possibility to produce distributed and parallel solutions. An overview of such algorithms for analysis of systems governed by Partial Differential Equations, PDEs, is given in [19].

In this paper, we use decomposition methods to facilitate distributed robust stability analysis of large networks of weakly interconnected uncertain systems. Decomposition allows us to reformulate a single global feasibility constraint \(x \in C \subset \mathbb{R}^n\) involving \(m\) uncertain systems as a set of \(N < m\) loosely coupled constraints

\[
x \in C_i = \{x \in \mathbb{R}^n \mid f_i(x) \preceq_K 0\}, \quad i = 1, \ldots, N
\]  

where \(f_i(x) \preceq_K 0\) is a linear conic inequality with respect to a convex cone \(K_i\). We will assume that \(f_i\) depends only on a small subset of variables \(x_k \mid k \in J_i\), \(J_i \subseteq \{1, \ldots, n\}\). The number of constraints \(N\) is less than the number of systems \(m\) in the network, and hence the functions \(f_i\) generally involve problem data from more than one subsystem in the network. We will henceforth assume that each \(f_i\) is described in terms of only a small number of local models such that the Euclidean projection \(P_{C_i}(x)\) of \(x\) onto \(C_i\) involves just a small group of systems in the network. This assumption is generally only valid if the network is weakly interconnected.

One algorithm that is suited for distributed solution of the CFP is the nonlinear Cimmino algorithm [20], [21] which is also known as the Mean Projection Algorithm. This algorithm is a fixed-point iteration which takes as the next iterate \(x^{(k+1)}\) a convex combination of the projections \(P_{C_i}(x^{(k)})\), i.e.,

\[
x^{(k+1)} := \sum_{i=1}^{N} \alpha_i P_{C_i}(x^{(k)})
\]

where \(\sum_{i=1}^{N} \alpha_i = 1\) and \(\alpha_1, \ldots, \alpha_N > 0\). Notice that each iteration consists of two steps: a parallel projection step which is followed by a consensus step that can be solved by means of distributed weighted averaging, e.g., [22]–[26]. Iusem &
De Pierro [27] have proposed an accelerated variant of the nonlinear Cimmino algorithm that takes as the next iterate a convex combination of the projections of \( x^{(k)} \) on only the sets for which \( x^{(k)} \notin C_i \). This generally improves the rate of convergence when only a few constraints are violated. However, the nonlinear Cimmino algorithm and its accelerated variant may take unnecessarily conservative steps when the sets \( C_i \) are loosely coupled. We will consider two algorithms that can exploit this type of structure, and both algorithms are closely related to the nonlinear Cimmino in that each iteration consists of a parallel projection step and a consensus step.

The first algorithm that we consider is equivalent to von Neumann Alternating Projection (AP) algorithm [28], [29] in a product space \( E = \mathbb{R}^{|J_1|} \times \cdots \times \mathbb{R}^{|J_N|} \) of dimension \( \sum_{i=1}^{N} |J_i| \). As a consequence, this algorithm converges at a linear rate under mild conditions, and its behavior is well-understood also when the CFP is infeasible [30]. Using the ideas from [31], [32], we also show how this algorithm can be implemented in a distributed manner.

A CFP can also be formulated as a convex minimization problem which can be solved with distributed optimization algorithms; see e.g. [23], [31]–[33]. The second algorithm that we consider is the Alternating Direction Method of Multipliers (ADMM) [31], [32], [34], [35], [36] applied to a convex minimization formulation of the CFP. Unlike AP, ADMM also makes use of dual variables, and when applied to the CFP, it is equivalent to Dykstra’s alternating projection method [30], [36] in the product space \( E \). Although there exist problems for which Dykstra’s method is much slower than the classical AP algorithm, it generally outperforms the latter in practice.

Outline

The paper is organized as follows. In Section II, we present a product space formulation of CFPs with loosely coupled sets, and we propose an algorithm based on von Neumann AP method. In Section III, we consider a convex minimization reformulation of the CFP, and we describe an algorithm based on the ADMM. We discuss distributed implementation of both algorithms in Section IV, and in Section V, we consider distributed solution of the robust stability analysis problem. We present numerical results in Section VI, and we conclude the paper in Section VII.

Notation

We denote with \( \mathbb{N}_p \) the set of positive integers \( \{1, 2, \ldots, p\} \). Given a set \( J \subset \{1, 2, \ldots, n\} \), 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 \( \mathbb{N}_n \setminus J \). This means that \( E_J x \) is a vector with the components of \( x \) that correspond to the elements in \( J \). 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

\[
\text{dist}(x, C) = \inf_{y \in C} \|x - y\|_2.
\]

Similarly, the distance between two sets \( C_1, C_2 \subseteq \mathbb{R}^n \) is defined as

\[
\text{dist}(C_1, C_2) = \inf_{y \in C_1, x \in C_2} \|x - y\|_2.
\]

The relative interior of a set \( C \) is denoted \( \text{relint}(C) \), and \( D = \text{diag}(a_1, \ldots, a_n) \) is a diagonal matrix of order \( n \) with diagonal entries \( D_{ii} = a_i \). The column space of a matrix \( A \) is denoted \( \mathcal{C}(A) \).

II. DECOMPOSITION AND PROJECTION METHODS

A. Decomposition and convex feasibility in product space

Given \( N \) loosely coupled sets \( C_1, \ldots, C_N \), as defined in (1), we define \( N \) lower-dimensional sets

\[
\bar{C}_i = \{ s_i \in \mathbb{R}^{|J_i|} | E^T_J s_i \in C_i \}, \quad i = 1, \ldots, N
\]

such that \( s_i \in \bar{C}_i \) implies \( E^T_J s_i \in C_i \). With this notation, the standard form CFP can be rewritten as

\[
\begin{align*}
\text{find} & \quad s_1, s_2, \ldots, s_N, x \\
\text{subject to} & \quad s_i \in \bar{C}_i, \quad i = 1, \ldots, N \\
& \quad s_i = E_J x, \quad i = 1, \ldots, N
\end{align*}
\]

where the equality constraints are the so called coupling constraints that ensure that the variables \( s_1, \ldots, s_N \) are consistent with one another. In other words, if the sets \( C_i \) and \( C_j \) \((i \neq j)\) depend on \( s_k \), then the \( k \)th component of \( E^T_J s_i \) and \( E^T_J s_j \) must be equal.

The formulation (6) decomposes the global variable \( x \) into \( N \) coupled variables \( s_1, \ldots, s_N \). This allows us to rewrite the problem as a CFP with two sets

\[
\begin{align*}
\text{find} & \quad S \\
\text{subject to} & \quad S \in C, \ S \in D
\end{align*}
\]

where

\[
\begin{align*}
S &= (s_1, \ldots, s_N) \in \mathbb{R}^{|J_1|} \times \cdots \times \mathbb{R}^{|J_N|} \\
C &= \bar{C}_1 \times \cdots \times \bar{C}_N \\
D &= \{ E_J x | x \in \mathbb{R}^n \} \\
\bar{E} &= [E^T_{J_1} \cdots E^T_{J_N}]^T
\end{align*}
\]

The formulation (7) can be thought of as a “compressed” product space formulation of a CFP with the constraints in (1), and it is similar to the consensus optimization problems described in [31, Sec. 7.2].

B. Von Neumann’s alternating projection in product space

The problem (7) can be solved using the von Neumann’s AP method. Given a CFP with two sets

\[
\begin{align*}
\text{find} & \quad x \\
\text{subject to} & \quad x \in A, \ x \in B
\end{align*}
\]

and a starting point \( y^{(0)} = x_0 \), von Neumann’s AP method computes two sequences [37]

\[
\begin{align*}
z^{(k+1)} &= P_A \left(y^{(k)}\right) \\
y^{(k+1)} &= P_B \left(z^{(k+1)}\right)
\end{align*}
\]

If the CFP is feasible, i.e., \( A \cap B \neq \emptyset \), then both sequences converge to a point in \( A \cap B \). We discuss infeasible problems at the end of this section. If we define \( X = \bar{E} x \), applying
the von Neumann AP algorithm to the CFP (7) results in the update rule

\[ S^{(k+1)} = P_C \left( X^{(k)} \right) \]

\[ = \left( P_{C_1} \left( E_J x^{(k)} \right), \ldots, P_{C_N} \left( E_J x^{(k)} \right) \right) \]

\[ X^{(k+1)} = \bar{E} \left( E^T \bar{E} \right)^{-1} E^T S^{(k+1)}, \]

where (10a) is the projection onto \( C \), and (10b) is the projection onto the column space of \( E \). The projection onto the set \( C \) can be computed in parallel by \( N \) computing agents, i.e., agent \( i \) computes \( s_i^{(k)} = P_{C_i} (E_J x^{(k)}) \), and the second projection can be interpreted as a consensus step that can be solved via distributed averaging.

\[ \text{dist}(x, C) \leq \max_{1 \leq i \leq N} \{ \text{dist}(x, C_i) \}, \quad \forall x \in B. \]

It was later shown by Bauschke et al. [39] and Beck & Teboulle [40] that Slater’s condition for the CFP implies bounded linear regularity, i.e., (11) holds and hence the cyclic projection algorithm converges linearly if

\[ \left( \bigcap_{i=1}^{k} C_i \right) \bigcap \left( \bigcap_{i=k+1}^{N} \text{relint}(C_i) \right) \neq \emptyset \]

where \( C_1, \ldots, C_k \) are polyhedral sets and \( C_{k+1}, \ldots, C_N \) are general closed, convex sets. For two sets, cyclic projections is equivalent to alternating projections, and when \( \{ A, B \} \) is boundedly linearly regular, it follows from [40, Thm. 2.2] that

\[ \text{dist}(x^{(k+1)}, C) \leq \gamma_B \text{dist}(x^{(k)}, C) \]

with

\[ \gamma_B = \sqrt{1 - \frac{1}{\theta_B^2}} \]

where \( C = A \cap B \) and \( \theta_B \) is a positive constant that depends on the starting point \( x^{(0)} \) and that satisfies (11) with \( B = \{ x \mid \| x - z \| \leq \| x^{(0)} - z \| \} \) for any \( z \in C \).

For infeasible problems (i.e., \( A \cap B = \emptyset \)), the iterates satisfy

\[ g^{(k)} - z^{(k)} \rightarrow v, \quad \| v \|_2 = \text{dist}(A, B). \]

Moreover, the sequences \( y^{(k)} \) and \( z^{(k)} \) converge if the distance \( \text{dist}(A, B) \) is attained, and otherwise \( \| y^{(k)} \|_2 \rightarrow \infty \) and \( \| z^{(k)} \| \rightarrow \infty \) [30, 37]. This means that we can use the difference \( v^{(k)} = y^{(k)} - z^{(k)} \) to detect infeasibility. If the set \( C \) in the CFP in (7) is a closed set, it is possible to use the statement in (14) for detecting the infeasibility of the CFP in (6). Note that this is the case for the problem described in Section V.

### III. Convex Minimization Reformulation

The CFP (6) is equivalent to a convex minimization problem

\[ \text{minimize} \quad \sum_{i=1}^{N} g_i(s_i) \]

\[ \text{subject to} \quad s_i = E_J x, \quad i = 1, \ldots, N, \]

with variables \( S \) and \( x \), where

\[ g_i(s_i) = \begin{cases} \infty & s_i \notin \bar{C}_i \\ 0 & s_i \in \bar{C}_i \end{cases} \]

is the indicator function for the set \( \bar{C}_i \). In the following, we apply the ADMM to the problem (15).

#### A. Solution via Alternating Direction Method of Multipliers

Consider the following equality constrained convex optimization problem

\[ \text{minimize} \quad F(x) \]

\[ \text{subject to} \quad Ax = b \]

with variable \( x \in \mathbb{R}^n \), and where \( A \in \mathbb{R}^{m \times n}, B \in \mathbb{R}^m \), and \( F : \mathbb{R}^n \rightarrow \mathbb{R} \). The augmented Lagrangian for this problem can be expressed as

\[ L_\rho(x, \lambda) = F(x) + \lambda^T (Ax - b) + \frac{\rho}{2} \| Ax - b \|^2, \]

where \( \rho > 0 \) and \( \lambda \in \mathbb{R}^m \). This can also be rewritten in normalized form as

\[ \bar{L}_\rho(x, \bar{\lambda}) = F(x) + \frac{\rho}{2} \| Ax - b + \bar{\lambda} \|^2 - \frac{\rho}{2} \| \bar{\lambda} \|^2 \]

where \( \bar{L}_\rho(x, \bar{\lambda}) \) is referred to as the normalized augmented Lagrangian, and \( \lambda = \lambda / \rho \) is the so-called normalized Lagrange variable.

A general convex optimization problem of the form

\[ \text{minimize} \quad F(S) \]

\[ \text{subject to} \quad S \in A \]

can be cast as a problem of the form (17) by means of the indicator function \( g(S) \) for the set \( A \), i.e.,

\[ \text{minimize} \quad F(S) + g(q) \]

\[ \text{subject to} \quad S - q = 0 \]

where \( q \) is an auxiliary variable. The normalized augmented Lagrangian for this problem is given by

\[ \bar{L}_\rho(S, q, \bar{\lambda}) = F(S) + g(q) \]

\[ + \frac{\rho}{2} \| S - (q - \bar{\lambda}) \|^2 - \frac{1}{2\rho} \| \bar{\lambda} \|^2 \]

The problem in (21) can be solved by applying the ADMM to (22); see e.g. [31] and [32, Sec. 3.4]. This results in the following iterative algorithm

\[ S^{(k+1)} = \arg \min_S \left\{ F(S) + \frac{\rho}{2} \| S - (q^{(k)} - \bar{\lambda}^{(k)}) \|^2 \right\} \]

\[ q^{(k+1)} = \arg \min_q \left\{ g(q) + \frac{\rho}{2} \| S^{(k+1)} - (q - \bar{\lambda}^{(k)}) \|^2 \right\} \]

\[ = P_A (S^{(k+1)} + \bar{\lambda}^{(k)}), \]

\[ \bar{\lambda}^{(k+1)} = \bar{\lambda}^{(k)} + \left( S^{(k+1)} - q^{(k+1)} \right). \]
The problem (15) can be solved using a similar approach. The normalized augmented Lagrangian for this problem is given by
\[
L_P(S, x, \bar{\lambda}) = \sum_{i=1}^{N} \{ g_i(s_i) + \frac{\rho}{2} \| s_i - (E_j x - \bar{\lambda}_i) \|_2^2 - \frac{\rho}{2} \| \bar{\lambda}_i \|_2^2 \} \tag{23}
\]
where \( \bar{\lambda}_i \in \mathbb{R}^{|J_i|} \) and \( \bar{\lambda} = (\bar{\lambda}_1, \ldots, \bar{\lambda}_N) \). If we apply the ADMM to (23), we obtain the following iterative method, i.e.,
\[
S^{(k+1)} = \arg\min_S \left\{ \sum_{i=1}^{N} \left( g_i(s_i) + \frac{\rho}{2} \| s_i - (E_i x^{(k)} - \bar{\lambda}_i^{(k)}) \|_2^2 \right) \right\}
\]
\[
x^{(k+1)} = \arg\min_x \left\{ \sum_{i=1}^{N} \frac{\rho}{2} \| s_i^{(k+1)} - (E_i x - \bar{\lambda}^{(k)}) \|_2^2 \right\}
\]
\[
\bar{\lambda}_i^{(k+1)} = \bar{\lambda}_i^{(k)} + (s_i^{(k+1)} - E_i x^{(k+1)}).
\tag{24}
\]
The update of \( S \) can be decomposed into \( N \) subproblems
\[
s_i^{(k+1)} = \arg\min_{s_i} \left\{ g_i(s_i) + \frac{\rho}{2} \| s_i - (E_i x^{(k)} - \bar{\lambda}_i^{(k)}) \|_2^2 \right\} = P_C \left( E_i x^{(k)} - \bar{\lambda}_i^{(k)} \right),
\tag{25}
\]
and the update of \( x \) can be computed as
\[
x^{(k+1)} = \left( E^T \right)^{-1} E^T \left( S^{(k+1)} + \bar{\lambda}_i^{(k)} \right).
\tag{26}
\]
Using the definition \( X = E x \), we can write the update of \( X \) as
\[
X^{(k+1)} = \left( E^T \right)^{-1} E^T \left( S^{(k+1)} + \bar{\lambda}_i^{(k)} \right).
\tag{27}
\]
Furthermore, given the starting point \( x^{(0)} = x_0 \) and \( \bar{\lambda}^{(0)} = 0 \), we obtain
\[
\bar{\lambda}^{(k)} = \sum_{i=1}^{k} \left( S^{(i)} - X^{(i)} \right).
\tag{28}
\]
The matrix \( \bar{E} = \left( E^T \right)^{-1} E^T \) in (27) defines an orthogonal projection onto the column space of the matrix \( E \), and hence \( \bar{E} \left( E^T \right)^{-1} E^T \bar{\lambda}^{(k)} = 0 \) for all \( k \geq 0 \). This allows us to simplify (27) as
\[
X^{(k+1)} = \bar{E} \left( E^T \bar{E} \right)^{-1} E^T S^{(k+1)}.
\tag{29}
\]
To summarize, the update rules for the ADMM, applied to (15), are given by
\[
S^{(k+1)} = P_C \left( X^{(k)} - \bar{\lambda}^{(k)} \right)
\tag{30a}
\]
\[
X^{(k+1)} = \bar{E} \left( E^T \bar{E} \right)^{-1} E^T S^{(k+1)}
\tag{30b}
\]
\[
\bar{\lambda}^{(k+1)} = \bar{\lambda}^{(k)} + (S^{(k+1)} - X^{(k+1)}).
\tag{30c}
\]
We remark that this algorithm is a special case of the algorithm developed in [31, Sec. 7.2] for consensus optimization. We also note that this algorithm is equivalent to Dykstra’s alternating projection method for two sets where one of the sets is affine [30]. Moreover, it is possible to detect infeasibility in the same way that we can detect infeasibility for the AP method, i.e., \( X^{(k)} - S^{(k)} \rightarrow v \) where \( \| v \|_2 = \text{dist}(C, D) \) [30].

**Algorithm 1** Alternating projection algorithm in product space

1. Given \( x^{(0)} \) and for all \( i = 1, \ldots, N \), each agent \( i \) should
2. repeat
3. \( s_i^{(k+1)} \leftarrow P_C \left( x_i^{(k)} \right) \).
4. Communicate with all agents \( r \) belonging to \( \text{Ne}(i) \).
5. for all \( j \in J_i \) do
6. \( x_j^{(k+1)} = \frac{1}{|J_j|} \sum_{q \in J_j} \left( E_j^T x_q^{(k+1)} \right) \).
7. end for
8. \( k \leftarrow k + 1 \)
9. until forever

Note that, unlike the AP method, the iteration (30) does not necessarily converge with a linear rate when the feasible sets satisfy (11).

**IV. DISTRIBUTED IMPLEMENTATION**

In this section, we describe how the algorithms can be implemented in a distributed manner using techniques that are similar to those described in [31] and [32, Sec. 3.4]. Specifically, the parallel projection steps in the algorithms expressed by the update rules in (10) and (30) are amenable to distributed implementation. We will henceforth assume that a network of \( N \) computing agents is available.

Let \( J_i = \{ k \mid i \in J_k \} \subseteq \mathbb{N}_n \) denote the set of constraints that depend on \( x_i \). Then it is easy to verify that
\[
\bar{E}^T \bar{E} = \text{diag}(|J_1|, \ldots, |J_n|)
\tag{31}
\]
and consequently, the \( j \)-th component of \( x \) in the update rules (10b) and (30b) is of the form
\[
x_j^{(k+1)} = \frac{1}{|J_j|} \sum_{q \in J_j} \left( E_j^T x_q^{(k+1)} \right).
\tag{32}
\]
In other words, the agents in the set \( J_j \) must solve a distributed averaging problem to compute \( x_j^{(k+1)} \). Let \( x_j = E_j x_i \). Since the set \( C_i \), associated with agent \( i \) involves the variables index by \( J_i \), agent \( i \) should update \( x_j \) by performing the update in (32) for all \( j \in J_i \). This requires agent \( i \) to communicate with all agents in the set
\[
\text{Ne}(i) = \left\{ j \mid J_i \bigcap J_j \neq \emptyset \right\},
\tag{33}
\]
which we call the neighbors of agent \( i \). Each agent \( j \in \text{Ne}(i) \) shares one or more variables with agent \( i \). Distributed variants of the algorithms presented in the previous sections are summarized in Algorithms 1 and 2.

**A. Feasibility Detection**

For strictly feasible problems, the algorithms 1 and 2 converge to a feasible solution. We now discuss different techniques for checking feasibility.

1) **Global Feasibility Test:** Perhaps the easiest way to check feasibility is to directly check the feasibility of \( x^{(k)} \) with respect to all the constraints. This can be accomplished by explicitly forming \( x^{(k)} \). This requires a centralized unit that recieves the local variables from the individual agents.
Algorithm 2 ADMM based algorithm

1: Given $x^{(0)}$, $\lambda^{(0)} = 0$, for all $i = 1, \ldots, N$, each agent $i$ should
2: repeat
3: \[ s_i^{(k+1)} \leftarrow P_{C_i} \left( x_i^{(k)} - \bar{\lambda}_i^{(k)} \right). \]
4: Communicate with all agents $r$ belonging to $\mathcal{N}(i)$.
5: for all $j \in J_i$ do
6: \[ x_j^{(k+1)} = \frac{1}{|J_i|} \sum_{\eta \in J_i} \left( E_j^T s_i^{(k+1)} + \lambda_i^{(k+1)} \right). \]
7: end for
8: \[ \bar{\lambda}_i^{(k+1)} \leftarrow \bar{\lambda}_i^{(k)} + \left( s_i^{(k+1)} - x_i^{(k+1)} \right). \]
9: $k \leftarrow k + 1$
10: until forever

2) Local Feasibility Test: It is also possible to check feasibility locally. Instead of sending the local variables to a central unit, each agent declares its feasibility status with respect to its local constraint. This type of feasibility detection method is based on the following Lemmas.

Lemma 1: If $x_j^{(k)} \in \bar{C}_i$, for all $i = 1, \ldots, N$ then using these vectors a feasible solution, $x$, can be constructed for the original problem, i.e., $x \in \bigcap_{i=1}^N \bar{C}_i$.

Proof: The update rule for each of the components of the variable $x$ is merely the average of all the corresponding local iterates from other agents. Hence, all iterates of each of the components of the $x$ variable are equal. As a result, based on the definition of the $\bar{C}_i$, by having $x_j^{(k)} \in \bar{C}_i$ for all $i = 1, \ldots, N$, it is possible to construct a feasible vector $x^{(k)}$ from the iterates $x_j^{(k)}$.

Lemma 2: If $\|X^{(k+1)} - X^{(k)}\|_2 = 0$ and $\|S^{(k+1)} - X^{(k+1)}\|_2 = 0$, then a feasible solution, $x$, can be constructed for the original problem, i.e., $x \in \bigcap_{i=1}^N \bar{C}_i$.

Proof: Consider the update rules for Algorithm 1. The conditions stated in the lemma imply that $s_i^{(k+1)} = x_j^{(k)}$. As a result for all $i = 1, \ldots, N$

\[ s_i^{(k+1)} = P_{C_i} \left( x_i^{(k)} \right) = x_i^{(k)}, \]

which implies that $x_j^{(k)} \in \bar{C}_i$. Therefore, similar to Lemma 1, it is possible to generate a feasible solution out of vectors $x_j^{(k)}$.

Consider the update rules in Algorithm 2 for the ADMM based algorithm. Then the conditions stated above imply that

\[ S^{(k+1)} = P_C \left( X^{(k)} - \bar{\lambda}_i^{(k)} \right) = X^{(k)}, \]

\[ X^{(k+1)} = \bar{E} \left( \bar{E}^T \bar{E} \right)^{-1} \bar{E}^T X^{(k+1)} = X^{(k)} \]

\[ \bar{\lambda}^{(k+1)} = \bar{\lambda}^{(k)} + \left( S^{(k+1)} - X^{(k+1)} \right) = \bar{\lambda}^{(k)}. \]

Hence any solution that satisfies the conditions in the lemma, constitutes an equilibrium point for the update rule in (30), and as a result

\[ P_{\bar{C}_i} \left( x_i^{(k)} \right) = x_i^{(k)}, \]

which in turn provides the possibility to construct a feasible solution.

Remark 1: For Algorithm 1, conditions in lemmas 1 and 2 are equivalent. However, this is not the case for Algorithm 2. For this algorithm, satisfaction of the conditions in Lemma 2 implies the conditions in Lemma 1.

Remark 2: Feasibility detection using Lemma 1, requires additional computations for Algorithm 2. These computations include local feasibility check of the iterates $x_j$. Note that this check does not incur any additional cost for Algorithm 1.

B. Infeasibility Detection

Recall from Section II that if the CFP is infeasible, then the sequence $\|S^{(k)} - X^{(k)}\|_2$ will converge to a nonzero constant $\|v\|_2 = \text{dist}(C, D)$. Therefore, in practice, it is possible to detect infeasible problems by limiting the number of iterations, i.e., if $\|S^{(k)} - X^{(k)}\|_2$ is not sufficiently small after the maximum number of iterations, the problem is considered to be infeasible.

V. ROBUST STABILITY ANALYSIS

Robust stability of uncertain large scale weakly interconnected systems with structured uncertainty can be analyzed through the so-called $\mu$-analysis framework [41]. This leads to CFP which is equivalent to a semidefinite programming (SDP) problem involving the system matrix.

Consider the following system description

\[ Y(s) = M(s)U(s), \]

where $M(s)$ is a $m \times m$ transfer function matrix, and let

\[ U(s) = \Delta Y(s), \]

where $\Delta = \text{diag}(\delta_i)$, with $\delta_i \in \mathbb{R}$, $|\delta_i| \leq 1$ for $i = 1, \ldots, m$, represent the uncertainties in the system. The system is said to be robustly stable if there exists a diagonal positive definite $X(\omega)$ and $0 < \mu < 1$ such that

\[ M(j\omega)^* X(\omega) M(j\omega) - \mu^2 X(\omega) < 0, \]

for all $\omega \in \mathbb{R}$. Note that this problem is infinite dimensional, and in practice, it is often solved approximately by discretizing the frequency variable.

In the following, we investigate only a single frequency point such that the dependence on the frequency is dropped. Moreover, for the sake of simplicity, we assume that $M$ is real-valued. The extension to complex valued $M$ is straightforward.

As a result, feasibility of the following CFP is a sufficient condition for robust stability of the system

\[ \text{find } X \]

subject to

\[ M'XM - X \preceq -\epsilon I \]

\[ x_i \geq \epsilon, \quad i = 1, \ldots, m \]

for $\epsilon > 0$ and where $x_i$ are the diagonal elements of $X$.

A large scale network of weakly interconnected uncertain systems can also be represented in the form (37)-(38). In the case of weakly interconnected system, the system matrix $M$ that relates the input and output signals is sparse. As an
example, we consider a chain of systems which leads to a
tri-diagonal system matrix
\[
M = \begin{bmatrix}
g_1 & h_1 & 0 & 0 & 0 \\
g_2 & h_2 & 0 & 0 & 0 \\
0 & f_3 & \ddots & \ddots & 0 \\
0 & 0 & \ddots & g_{m-1} & h_{m-1} \\
0 & 0 & 0 & f_m & g_m
\end{bmatrix}.
\] (41)

This system matrix is obtained if the input-output relation for
the underlying systems are given by
\[
\begin{align*}
p_1 &= g_1 q_1 + h_1 z_2 \\
z_1 &= q_1 \\
q_1 &= \delta_1 p_1, \\
p_m &= g_m q_m + f_m z_{m-1} \\
z_m &= q_m \\
q_m &= \delta_1 p_m,
\end{align*}
\] (42)
and
\[
\begin{align*}
p_i &= g_i q_i + f_i z_{i-1} + h_i z_{i+1} \\
z_i &= q_i \\
q_i &= \delta_i p_i,
\end{align*}
\] (44)
for \(i = 2, \ldots, m - 1\). The tri-diagonal structure in the system
matrix implies that the LMI defined in (40) becomes banded.
This is a special case of a so-called chordal sparsity pattern,
and these have been exploited in SDP by several authors; see
[42]–[45].

Positive semidefinite matrices with chordal sparsity patterns
can be decomposed into a sum of matrices that are positive
semidefinite [44], [46, Sec. 5.1]. For example, a positive
semidefinite band matrix can be decomposed into a sum of
positive semidefinite matrices, as illustrated in Figure 4. Note
that the nonzero blocks marked in the matrices on the right-
hand side in Figure 4 are structurally equivalent to the block
in the left-hand side, but the numerical values are generally
different. Using this decomposition technique, the problem
in (40) can be reformulated as in (7).

By introducing auxiliary variables \(w, y, z \in \mathbb{R}^{m-3}\)
and letting \(q = (x, w, y, z)\), the five-diagonal matrix \(M'XM - X\)
can be decomposed as
\[
\begin{align*}
f_1(q) &= \begin{bmatrix} g_1 & f_2 \\
h_1 & g_2 & h_2 \\
0 & 0 & 0 \end{bmatrix}
\begin{bmatrix} x_1 \\
x_2 \end{bmatrix} - \begin{bmatrix} x_1 \\
x_2 - w_1 - y_1 \\
0 - y_1 \end{bmatrix} \quad (45a)
\end{align*}
\]
\[
\begin{align*}
f_{m-1}(q) &= \begin{bmatrix} h_{m-1} & 0 \\
0 & g_m \\
0 & 0 \end{bmatrix}
\begin{bmatrix} x_{m-1} \\
x_m \end{bmatrix} - \begin{bmatrix} f_{m-1} \\
0 \end{bmatrix} \quad (45b)
\end{align*}
\]
and
\[
\begin{align*}
f_i(q) &= \begin{bmatrix} f_{i+1} \\
h_{i+1} \\
g_{i+1} \end{bmatrix}
\begin{bmatrix} x_{i+1} \\
x_m \end{bmatrix} - \begin{bmatrix} f_i \\
h_i \end{bmatrix} \quad (45c)
\end{align*}
\] for \(i = 2, \ldots, m - 3\). Notice that (45a) and (45b) depend
on data from two subsystems whereas (45c) depends on data
from only one subsystem. This dependence is also illustrated
in Figure 1. The right-hand side of the LMI in (40) can be
decomposed in a similar manner

\[ F_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad F_{m-2} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad F_i = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad i = 2, \ldots, m-3. \]

With this decomposition, we can reformulate the LMI in (40) as a set of \( m - 2 \) LMIs

\[ q \in C_i = \{ q \mid f_i(q) \succeq F_i \}, \quad i = 1, \ldots, m-2, \]

or equivalently, as the constraints

\[ s_i \in \tilde{C}_i = \{ E_J q \mid f_i(q) \succeq F_i \} \subseteq \mathbb{R}^{|J_i|}, \quad s_i = E_J q, \quad i = 1, \ldots, m-2. \]

for \( i = 1, \ldots, m-2 \). Here \( J_i \) is the set of indices of the entries of \( q \) that are required to evaluate \( f_i \). Notice that (46) is of the form (6). The CFP defined by the constraints in (46) can be solved over a network of \( m - 2 \) agents. The connectivity of this network and the coupling variables among the agents are illustrated in Figure 1.

Remark 3: As was mentioned above, the CFP in (40) is often solved over a frequency grid. If these frequency points are chosen sufficiently close, it is probable that the CFPs for adjacent frequencies are similar. As a result, it is very likely that a solution to a CFP for one frequency is either a solution to or close to a solution to the CFP for the adjacent frequencies. In this case, warm starting the projection-based algorithms using solutions to CFPs at adjacent frequency points may significantly reduce the computational cost.

VI. NUMERICAL RESULTS

In this section, we apply Algorithms 1 and 2 to a family of random problems involving chains of uncertain systems. These problems have the same band structure as described in Section V. We use the local feasibility detection method introduced in Section IV-A. Note that in order to avoid numerical problems and unnecessarily many projections, the projections are performed for slightly tighter bounds than those used for feasibility checks of the local iterates. This setup is the same for all the experiments presented in this section.

Figures 5 and 6 show the behavior of Algorithm 1 for 50 randomly generated problems with \( m = 52 \). These problems decompose into 50 subproblems which 50 agents solve collaboratively. Figure 5 shows the number of agents that are required to evaluate \( f_i \) with respect to the iteration number. This figure illustrates the results for 50 randomly generated problems.

Figures 5 and 6 show the behavior of Algorithm 1 for 50 randomly generated problems with \( m = 52 \). These problems decompose into 50 subproblems which 50 agents solve collaboratively. Figure 5 shows the number of agents that are required to evaluate \( f_i \) with respect to the iteration number. This figure illustrates the results for 50 randomly generated problems.

Figures 5 and 6 show the behavior of Algorithm 1 for 50 randomly generated problems with \( m = 52 \). These problems decompose into 50 subproblems which 50 agents solve collaboratively. Figure 5 shows the number of agents that are required to evaluate \( f_i \) with respect to the iteration number. This figure illustrates the results for 50 randomly generated problems.

Figures 5 and 6 show the behavior of Algorithm 1 for 50 randomly generated problems with \( m = 52 \). These problems decompose into 50 subproblems which 50 agents solve collaboratively. Figure 5 shows the number of agents that are required to evaluate \( f_i \) with respect to the iteration number. This figure illustrates the results for 50 randomly generated problems.
detect convergence to a feasible solution, respectively. In our experiments, Algorithm 2 is faster when feasibility detection is based on the conditions in Lemma 1, and Algorithm 1 is faster when the condition in Lemma 2 is used for feasibility detection. It is also worth mentioning that with the accelerated nonlinear Cimmino algorithm, more than 1656 iterations were needed to obtain a feasible solution.

In the next experiment, we investigate the performance of the algorithms as a function of the number of systems in the chain. The results in Figures 9 and 10 indicate that the problem parameter $m$ does not affect the performance of Algorithm 1 much. The number of iterations required to reach consensus is only increased slightly. Figures 11 and 12 verifies that Algorithm 2 behaves in a similar manner.

VII. CONCLUSION

In this paper, we have shown that it is possible to solve CPFs with loosely coupled constraints efficiently in a distributed manner. We have proposed two algorithms. One is based on von Neumann’s AP method, and hence it enjoys the linear convergence rate that characterizes this algorithms when applied to strictly feasible problems. The other method is based on the ADMM, and it generally outperforms the AP method in practice in terms of the number of iterations required to obtain a feasible solution. Both methods can detect infeasible problems. For structured problems that arise in robust stability analysis of a large-scale weakly interconnected uncertain systems, our numerical results show that both algorithms outperform the classical projection-based algorithms.

ACKNOWLEDGMENT

This research is supported by the Swedish department of education within the ELLIIT research program.

REFERENCES

Fig. 12. Algorithm 2: $\|X^{(k)} - X^{(k-1)}\|_2^2$ and $\|G^{(k)} - X^{(k)}\|_2^2$ with respect to the iteration number. This figure illustrates the results for problems with different number of constraints.


Title  
**Decomposition and Projection Methods for Distributed Robustness Analysis of Interconnected Uncertain Systems**

Author  
**Sina Khoshfetrat Pakazad, Martin S. Andersen, Anders Hansson, Anders Rantzer**

Abstract  
We consider a class of convex feasibility problems where the constraints that describe the feasible set are loosely coupled. These problems arise in robust stability analysis of large, weakly interconnected systems. To facilitate distributed implementation of robust stability analysis of such systems, we propose two algorithms based on decomposition and simultaneous projections. The first algorithm is a nonlinear variant of Cimmino's mean projection algorithm, but by taking the structure of the constraints into account, we can obtain a faster rate of convergence. The second algorithm is devised by applying the alternating direction method of multipliers to a convex minimization reformulation of the convex feasibility problem. We use numerical results to show that both algorithms require far less iterations than the accelerated nonlinear Cimmino algorithm.

Keywords  
Robust stability analysis, convex feasibility problems, projection algorithms, decomposition, distributed computing.