DTU Library

Distributed Semidefinite Programming with Application to Large-scale System Analysis

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

Link back to DTU Orbit

Distributed Semidefinite Programming with Application to Large-scale System Analysis

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

Abstract—**Distributed algorithms for solving coupled semidefinite programs (SDPs) commonly require many iterations to converge. They also put high computational demand on the computational agents. In this paper we show that in case the coupled problem has an inherent tree structure, it is possible to devise an efficient distributed algorithm for solving such problems. This algorithm can potentially enjoy the same efficiency as centralized solvers that exploit sparsity. The proposed algorithm relies on predictor-corrector primal-dual interior-point methods, where we use a message-passing algorithm to compute the search directions distributedly. Message-passing here is closely related to dynamic programming over trees. This allows us to compute the exact search directions in a finite number of steps. Furthermore this number can be computed a priori and only depends on the coupling structure of the problem. We use the proposed algorithm for analyzing robustness of large-scale uncertain systems distributedly. We test the performance of this algorithm using numerical examples.**

Keywords—*SDPs, distributed algorithms, primal-dual methods, robustness analysis, interconnected uncertain systems.*

I. INTRODUCTION

Semidefinite programs are convex optimization problems that include linear matrix inequalities (LMIs) or semidefinite constraints. The computational complexity of solving such problems commonly scales badly with the number of optimization variables and/or the dimension of the semidefinite constraints in the problem. This limits our ability to solve large SDPs. Despite this, large SDPs are appearing more and more in different engineering fields, e.g., in problems related to sensor networks, smart grids and analysis of uncertain systems, e.g., see [2], [4], [7], [29], [31]. This has been the driving force for devising efficient and tailored centralized solvers for such problems. These solvers exploit the structure in the problem to reduce the computational burden of solving the problem in a centralized manner, see e.g., [2], [19], [25], [40], [41]. Despite the success of such approaches for solving medium to large-scale problems, there are still problems that cannot be solved using centralized solvers, see e.g., [1], [8], [14], [36]. This can be due to limited available computational power and/or memory that prohibits us from solving the problem. Also it can be due to certain structural constraints, e.g.,

S. Khoshfetrat Pakazad and A. Hansson are with the Division of Automatic Control, Department of Electrical Engineering, Linköping University, Sweden. Email: {sina.kh.pa, hansson}@isy.liu.se.

M. S. Andersen is with Department of Applied Mathematics and Computer Science, Technical University of Denmark, Denmark. Email: mskan@dtu.dk

A. Rantzer is with Department of Automatic Control, Lund University, Sweden. Email: anders.rantzer@control.lth.se

privacy requirements, that obstructs us from even forming the centralized problem.

For such instances, distributed algorithms may be used for solving the problem. These algorithms facilitate solving the problem using a network of computational agents, without the need for a centralized unit. Due to this, the computational complexity of these algorithms scales better, and they potentially enable us to address structural constraints in the problem. The main approach for designing distributed algorithms consists of two major phases. First the structure in the problem is exploited to decompose the problem or reformulate it as a coupled problem. Then first-order splitting methods are used for solving the resulting problem distributedly, see e.g., [27], [37]. This approach has been used in many applications, e.g., see [14], [23], [36]. In [36] the authors consider a sensor localization problem and use a so-called edge-based decomposition for reformulating the underlying SDP as a coupled one. They then employ alternating direction method of multipliers (ADMM) to solve the problem distributedly. An optimal power flow problem has been considered in [14], where the authors reformulate the problem as a coupled SDP using semidefinite relaxation techniques. They then use ADMM to solve the coupled problem distributedly. In [23] the authors consider robustness analysis of large-scale interconnected uncertain systems. They exploit the sparsity in the interconnections to decompose the underlying SDP and reformulate it as a coupled problem. This problem is then solved distributedly using algorithms that rely on proximal splitting methods.

The algorithms designed using the aforementioned approach, although effective, suffer from some issues. For instance, since these algorithms rely on first-order splitting methods, with convergence rates $\mathcal{O}(1/k)$ or $\mathcal{O}(1/k^2)$ where k is the number of iterations, they require many iterations to converge to an accurate enough solution. Furthermore, exploiting structure and decomposing problems is commonly done through introduction of consensus constraints, which describe the coupling structure in the problem. The number of such constraints is commonly large for SDPs, which can in turn adversely affect the computational and/or convergence properties. Moreover the agents involved in these distributed algorithms need to solve an SDP at every iteration of the algorithm, which can potentially put a considerable computational burden on the agents.

In this paper we propose a distributed algorithm for solving coupled SDPs with a tree structure. These SDPs are defined in Section IV. This algorithm does not suffer from any of the aforementioned issues. We achieve this by avoiding the

use of first-order splitting methods and instead rely on primal-dual interior-point methods, which have superior convergence properties. The proposed algorithm is produced by distributing the computations conducted at each iteration of the primal-dual method. Particularly, we use a message-passing algorithm for computing the search directions. Message passing, here, is closely related to non-serial dynamic programming, [5], [22], [26]. We also present a similar approach for distributing the remaining computations at every iteration. As a consequence, at each iteration of the primal-dual method, the computational burden on each agent is very low. In fact during each iteration, an agent is required to factorize a relatively small matrix once and is required to communicate with its neighbors twelve times.

The proposed algorithm in this paper is closely related to that of [22]. In fact, the authors in [22] use the same approach for devising a distributed algorithm for solving coupled non-conic problems. However, the computation of search directions for SDPs is not as straightforward as for non-conic problems. This is due to introduction of scaling matrices and their inverses in the KKT system, which destroys the structure in the problem. In order to circumvent this issue, we here put forth a novel way for computing the search directions at each iteration. This in turn enables us to use the message-passing algorithm for computing the search directions.

Notice that by using this approach for computing the search directions, we implicitly solve the so-called augmented system. This is done by computing a block LDL^T factorization of its coefficient matrix using a fixed pivoting ordering, where the ordering is enforced by the coupling structure in the problem, [22]. This is in contrast to existing methods that commonly solve the so-called Schur complement system or normal equations. As a result, the proposed algorithm provides us with more stable and accurate implementation, [12], [42]. Solving the augmented system is also considered in [30], where the authors also compute the search directions through solving the augmented system by computing an LDL^T factorization using fixed pivoting ordering. This is particularly done by using regularization and iterative refinement. In this paper, however, a block LDL^T factorization is computed using a fixed pivoting ordering without the use of regularization. Hence, the augmented system is solved without the need for iterative refinement.

We then use the proposed algorithm for analyzing large-scale interconnected uncertain systems, distributedly. This is made possible by exploiting the sparsity in the interconnections, as outlined in [2]. A similar approach was also used in [23]. There, the authors utilized the so-called range-space decomposition for reformulating the analysis problem as a coupled feasibility problem. They then used algorithms that rely on proximal splitting methods for solving it distributedly. We here instead use the so-called domain-space decomposition to reformulate the analysis problem as a coupled SDP. The coupling structure of this coupled problem is less complicated than that of in [23], and has a tree structure. This then enables us to use the presented distributed algorithm for solving the problem efficiently and distributedly. We illustrate the performance of the algorithm using numerical examples.

Outline

Next we first define some notations that are used throughout the paper. In Section II we put forth a definition of coupled and loosely coupled SDPs. We review a predictor-corrector primal-dual interior-point method in Section III and briefly discuss how the structure in coupled problems is reflected in the computations conducted at every iteration of this method. Section IV expresses coupled problems with a tree structure and discusses the use of message-passing algorithm for solving coupled problems with a tree structure. This is then used in Section V where we present the proposed distributed algorithm for solving coupled SDPs with tree structure. In Section VI we discuss a decomposition approach for sparse SDPs. This approach is used in Section VII for reformulating the problem of robustness analysis of large-scale interconnected uncertain systems as coupled SDPs with a tree structure. We test the performance of the proposed distributed algorithm when applied to this problem using numerical experiments in Section VIII. Finally we finish the paper with some concluding remarks in Section IX.

Notation

We denote the set of real and complex numbers with $\mathbb{R}$ and $\mathbb{C}$, and the set of $m \times n$ real and complex matrices with $\mathbb{R}^{m \times n}$ and $\mathbb{C}^{m \times n}$, respectively. The transpose and conjugate transpose of a matrix X is denoted by X^T and X^*, respectively. The null space of a matrix X is denoted by $\mathcal{N}(X)$. With $\mathbb{S}^n$ and $\mathbb{H}^n$ we denote the set of $n \times n$ symmetric and Hermitian matrices. The set of integer numbers $\{1, \ldots, n\}$ is denoted by $\mathbb{N}_n$. Given a set of positive integers $J \subseteq \mathbb{N}_n$, the matrix $E_J \in \mathbb{R}^{|J| \times n}$ is a 0–1 matrix obtained from an $n \times n$ identity matrix with rows indexed by $\mathbb{N}_n \setminus J$ removed, where $|J|$ denotes the number of elements in J. This means that $E_J x$ is a $|J|$-dimensional vector that contains the elements of x indexed by J. We denote this vector by X_J. By $x_l^{i,(k)}$ and $X_{mn}^{i,(k)}$ we denote the lth element of vector x^i and the element at row m and column n of matrix X^i at the kth iteration, respectively. Given matrices X^k for $k = 1, \ldots, N$, blk $\mathrm{diag}(X^1, \ldots, X^N)$ denotes a block-diagonal matrix with blocks specified by the given matrices. Similarly $\mathrm{diag}(x_1, \ldots, x_N)$ is a diagonal matrix with diagonal elements $x_1, \ldots, x_N$. Given vectors x^k for $k = 1, \ldots, N$, the column vector $(x^1, \ldots, x^N)$ is all of the given vectors stacked. The generalized matrix inequality $G \prec H$ ($G \preceq H$) means that $G - H$ is negative (semi)definite. Given a matrix $X \in \mathbb{R}^{m \times n}$, $\mathrm{vec}(X)$ is an mn-dimensional vector that is obtained by stacking all columns of X on top of each other. Given two matrices $X, Y \in \mathbb{R}^{m \times n}$, $X \bullet Y := \mathrm{vec}(X)^T \mathrm{vec}(Y)$. For a symmetric matrix $X \in \mathbb{S}^n$

$$\mathrm{svec}(X) := (X_{11}, \sqrt{2}X_{21}, \ldots, \sqrt{2}X_{n1}, X_{22},$$
$$\sqrt{2}X_{32}, \ldots, \sqrt{2}X_{n2}, \ldots, X_{nn}).$$

Operators mat and smat are defined as inverses of vec and svec, respectively. Given two matrices X and Y by $X \otimes Y$ we denote the standard Kronecker product. Given $X \in \mathbb{S}^n$, define U as an $n(n+1)/2 \times n^2$ matrix such that $U \mathrm{vec}(X) =$

svec(X). Then for two matrices $X, Y \in \mathbb{R}^{n \times n}$, $\otimes_s$ denotes the symmetrized Kronecker product that is defined as

$$X \otimes_s Y := \frac{1}{2} U(X \otimes Y + Y \otimes X)U^T.$$

For properties of the symmetrized Kronecker product refer to [38]. Given two sets J_1 and J_2, $J_1 \times J_2$ denotes the standard cartesian product and by $J_1 \times_s J_1$ we denote the symmetrized cartesian product defined as

$$J_1 \times_s J_1 := \{(j, k) \in J_1 \times J_1 \mid j \leq k\}.$$

For these two sets $J_1 \setminus J_2$ denotes the standard set minus. By $\min$ we denote the minimum value and with $\arg \min$ we denote the minimizing argument of a function. By $\mathcal{L}_2^n$ we denote the set of n-dimensional square integrable signals, and $\mathcal{RH}_\infty^{m \times n}$ represents the set of real, rational $m \times n$ transfer function matrices with no poles in the closed right half plane. A graph is denoted by $Q(V, \mathcal{E})$ where $V = \{v_1, \ldots, v_n\}$ is its set of vertices or nodes and $\mathcal{E} \subseteq V \times V$ denotes its set of edges. An induced graph by $V' \subseteq V$ on $Q(V, \mathcal{E})$, is a graph $Q_I(V', \mathcal{E}')$ where $\mathcal{E}' = \mathcal{E} \cap (V' \times V')$.

II. COUPLED AND LOOSELY COUPLED SDPs

Let us consider a coupled SDP given as

$$\underset{X}{\text{minimize}} \quad \sum_{i=1}^{N} W^i \bullet X_{J_i J_i} \tag{1a}$$

$$\text{subject to} \quad Q_j^i \bullet X_{J_i J_i} = b_j^i, \quad j = 1, \ldots, m_i, \\ i = 1, \ldots, N, \tag{1b}$$

$$X_{J_i J_i} \succeq 0, \quad i = 1, \ldots, N, \tag{1c}$$

where $Q_j^i, W^i \in \mathbb{S}^{|J_i|}$ such that $\begin{bmatrix} \text{svec}(Q_1^i) & \cdots & \text{svec}(Q_{m_i}^i) \end{bmatrix}$ has full column rank for all $i = 1, \ldots, N$, with the ordered sets $J_i \subseteq \mathbb{N}_n$ such that $\bigcup_{i=1}^{N} J_i = \mathbb{N}_n$, and $X_{J_i J_i} = E_{J_i} X E_{J_i}^T$ with $X \in \mathbb{S}^n$ such that $X_{jk} = 0$ if $(j, k) \notin \mathcal{J}$ and $\mathcal{J} = \bigcup_{i=1}^{N} \mathcal{J}_i := J_i \times_s J_i$. This problem can be seen as a combination of N coupled subproblems, each of which defined by the objective function $W^i \bullet X_{J_i J_i}$ and constraints $Q_j^i \bullet X_{J_i J_i} = b_j^i$ for $j = 1, \ldots, m_i$ and $X_{J_i J_i} \succeq 0$. Let us now define $\mathcal{I}_{(i,j)} = \{k \mid (i, j) \in \mathcal{J}_k\}$, which denotes the set of subproblems that are coupled in that they all depend on the variable X_{ij}. Notice that agents a and b are members of $\mathcal{I}_{(i,j)}$ if and only if $\{i, j\} \subseteq J_a \cap J_b$. It is possible to provide a more explicit description of the coupling among the subproblems by decomposing (1) as

$$\underset{X, \bar{X}^i}{\text{minimize}} \quad \sum_{i=1}^{N} W^i \bullet \bar{X}^i \tag{2a}$$

$$\text{subject to} \quad Q_j^i \bullet \bar{X}^i = b_j^i, \quad i = 1, \ldots, N, \tag{2b}$$

$$\bar{X}^i \succeq 0, \quad i = 1, \ldots, N, \tag{2c}$$

$$\bar{X}^i = E_{J_i} X E_{J_i}^T, \quad i = 1, \ldots, N. \tag{2d}$$

Notice that in (2), the objective function terms and constraints in (2a)–(2c) are decoupled and the coupling in the problem is described using the consensus constraints in (2d). It is also possible to provide a graphical representation of the coupling using undirected graphs. Particularly let $Q_s(\mathcal{J}, \mathcal{E}_s)$

be a graph with vertex set $\mathcal{J}$ as defined above and edge set $\mathcal{E}_s = \{((i, j), (v, t)) \mid \mathcal{I}_{(i,j)} \cap \mathcal{I}_{(v,t)} \neq \emptyset\}$. We refer to this graph as the sparsity graph of the problem. Let us now illustrate the definitions above using an example given as

$$\underset{X}{\text{minimize}} \quad W^1 \bullet X_{\{1,2,4\}\{1,2,4\}} + \\ W^2 \bullet X_{\{1,3,4\}\{1,3,4\}} + W^3 \bullet X_{\{4,5\}\{4,5\}} \tag{3a}$$

$$\text{subject to} \quad \begin{bmatrix} x_{11} & x_{12} & x_{13} & 0 & 0 \\ x_{12} & x_{22} & 0 & x_{24} & 0 \\ x_{13} & 0 & x_{33} & x_{34} & 0 \\ 0 & x_{24} & x_{34} & x_{44} & x_{45} \\ 0 & 0 & 0 & x_{45} & x_{55} \end{bmatrix} \succeq 0. \tag{3b}$$

Notice that the constraint in (3b) can be rewritten as

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & 0 & 0 \\ x_{12} & x_{22} & 0 & x_{24} & 0 \\ x_{13} & 0 & x_{33} & x_{34} & 0 \\ 0 & x_{24} & x_{34} & x_{44} & x_{45} \\ 0 & 0 & 0 & x_{45} & x_{55} \end{bmatrix} = $$

$$E_{J_1}^T \begin{bmatrix} \frac{x_{11}}{2} & x_{12} & 0 \\ x_{12} & x_{22} & x_{24} \\ 0 & x_{24} & \frac{x_{44}}{3} \end{bmatrix} E_{J_1}$$

$$+ E_{J_2}^T \begin{bmatrix} \frac{x_{11}}{2} & x_{13} & 0 \\ x_{13} & x_{33} & x_{34} \\ 0 & x_{34} & \frac{x_{44}}{3} \end{bmatrix} E_{J_2}$$

$$+ E_{J_3}^T \begin{bmatrix} \frac{x_{44}}{3} & x_{45} \\ x_{45} & x_{55} \end{bmatrix} E_{J_3} \succeq 0.$$

with $J_1 = \{1, 2, 4\}$, $J_2 = \{1, 3, 4\}$ and $J_3 = \{4, 5\}$. Then the optimal objective value of

$$\underset{X}{\text{minimize}} \quad W^1 \bullet X_{\{1,2,4\}\{1,2,4\}} + \\ W^2 \bullet X_{\{1,3,4\}\{1,3,4\}} + W^3 \bullet X_{\{4,5\}\{4,5\}} \tag{4a}$$

$$\text{subject to} \quad \begin{bmatrix} \frac{x_{11}}{2} & x_{12} & x_{14} \\ x_{12} & x_{22} & x_{24} \\ x_{14} & x_{24} & \frac{x_{44}}{3} \end{bmatrix} \succeq 0, \quad x_{14} = 0, \tag{4b}$$

$$\begin{bmatrix} \frac{x_{11}}{2} & x_{13} & x_{14} \\ x_{13} & x_{33} & x_{34} \\ x_{14} & x_{34} & \frac{x_{44}}{3} \end{bmatrix} \succeq 0, \quad x_{14} = 0, \tag{4c}$$

$$\begin{bmatrix} \frac{x_{44}}{3} & x_{45} \\ x_{45} & x_{55} \end{bmatrix} \succeq 0, \tag{4d}$$

defines an upperbound for the optimal objective value of (3). The problem in (4) is a coupled SDP with smaller semidefinite constraints. This method for reformulating the problem is commonly used for cases when the original problem is either impossible or very difficult to solve. Notice that this problem is in the same format as (1). The sparsity graph of this problem is illustrated in Figure 1, where for instance there is an edge between the nodes $(1, 1)$ and $(1, 2)$ since the intersection between the sets $\mathcal{I}_{(1,1)} = \{1, 2\}$ and $\mathcal{I}_{(1,2)} = \{2\}$ is nonempty.

In case for a coupled problem
- $|J_i \cap J_j| \ll n$ for all $i, j \in \mathbb{N}_N$;
- $|\mathcal{I}_{(i,j)} \cap \mathcal{I}_{(v,t)}| \ll N$ for all $(i, j), (v, t) \in \mathcal{J}$,

then we call this problem loosely coupled. As we will see later, it is possible to devise efficient distributed solvers based
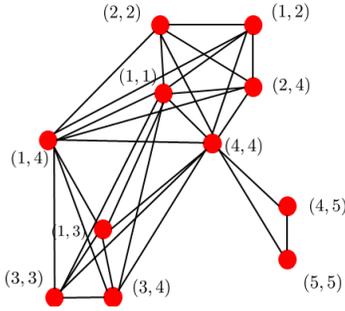
Fig. 1. Sparsity graph for the coupled SDP in (4).

on primal-dual interior-point methods for solving coupled and loosely coupled SDPs. To this end, let us first briefly review primal-dual interior-point methods for solving SDPs.

III. PRIMAL-DUAL INTERIOR-POINT METHODS FOR SOLVING SDPS

It is possible to iteratively solve a standard-form SDP, given as

$$
\begin{aligned}
\underset{X}{\text{minimize}} \quad & C \bullet X \\
\text{subject to} \quad & A_i \bullet X = b_i, \quad i = 1, \dots, m, \\
& X \succeq 0,
\end{aligned}
\tag{5}
$$

where $b \in \mathbb{R}^m$ and $X, A_i, C \in \mathbb{S}^n$ such that $[\text{svec}(A_1) \ \dots \ \text{svec}(A_m)]$ has full column rank, using primal-dual interior-point methods. Particularly, given the iterates $(X^{(k)} \succ 0, S^{(k)} \succ 0, v^{(k)})$, a primal-dual interior-point method generates the next iterates $(X^{(k+1)}, S^{(k+1)}, v^{(k+1)})$ by taking a single Newton step applied to the perturbed KKT conditions

$$
A_i \bullet X = b_i, \quad i = 1, \dots, m,
\tag{6a}
$$

$$
\sum_{i=1}^m v_i A_i + S = C,
\tag{6b}
$$

$$
XS = \delta I,
\tag{6c}
$$

together with $S \succ 0$ and $X \succ 0$ where $\delta > 0$. Specifically this Newton step can be computed by solving the following linear system of equations

$$
A_i \bullet \Delta X = b_i - A_i \bullet X^{(k)}, \quad i = 1, \dots, m,
\tag{7a}
$$

$$
\sum_{i=1}^m \Delta v_i A_i + \Delta S = C - S^{(k)} - \sum_{i=1}^m v_i^{(k)} A_i,
\tag{7b}
$$

$$
H_D(\Delta X S^{(k)} + X^{(k)} \Delta S) = \delta I - H_D(X^{(k)} S^{(k)}),
\tag{7c}
$$

where $H_D(M) = 1/2(DMD^{-1} + D^{-T}MD^T)$, $\delta = \sigma\mu$ is the perturbation parameter with $\mu = X^{(k)} \bullet S^{(k)}/n$ denoting the surrogate duality gap and $\sigma \in [0, 1]$, and where (7c) is a modified linearization of (6c) that ensures that the computed directions ΔS and ΔX are symmetric. There are different choices for the scaling matrix D in (7c), e.g., see [38] and references therein. For the sake of brevity, we limit our

discussion to the choices presented in [32], [33], that is we choose $D = G^{-1}$ with $W = GG^T$ where

$$
\begin{aligned}
W :=& (X^{(k)})^{\frac{1}{2}} \left((X^{(k)})^{\frac{1}{2}} S^{(k)} (X^{(k)})^{\frac{1}{2}} \right)^{-\frac{1}{2}} (X^{(k)})^{\frac{1}{2}} \\
=& (S^{(k)})^{-\frac{1}{2}} \left((S^{(k)})^{\frac{1}{2}} X^{(k)} (S^{(k)})^{\frac{1}{2}} \right)^{\frac{1}{2}} (S^{(k)})^{-\frac{1}{2}}.
\end{aligned}
\tag{8}
$$

This scaling is referred to as the Nesterov-Todd or NT scaling. In order to make the notation less complicated, from now on we drop the iteration index k, and we use lowercase notation for denoting vectorized variables or residuals, e.g., we use Δx as $\text{svec}(\Delta X)$ or r_{dual} as $\text{svec}(R_{\text{dual}})$. Using symmetrized Kronecker product we can then rewrite (7) more compactly as

$$
\begin{bmatrix} 0 & A & 0 \\ A^T & 0 & I \\ 0 & U & F \end{bmatrix} \begin{bmatrix} \Delta v \\ \Delta x \\ \Delta s \end{bmatrix} = \begin{bmatrix} r_{\text{primal}} \\ r_{\text{dual}} \\ r_{\text{cent}} \end{bmatrix},
\tag{9}
$$

where $A = [\text{svec}(A_1) \ \dots \ \text{svec}(A_m)]^T$, $U = D \otimes_s D^{-T} S$, $F = DX \otimes_s D^{-T}$ and

$$
r_{\text{primal}} = b - Ax
$$

$$
R_{\text{dual}} = C - S - \sum_{i=1}^m v_i A_i,
\tag{10}
$$

$$
R_{\text{cent}} = \delta I - H_D(XS),
$$

see [38]. One way of solving (9), is to first solve for Δs as in

$$
\Delta s = F^{-1} \left(r_{\text{cent}} - U\Delta x \right),
\tag{11}
$$

and then solve

$$
\begin{bmatrix} -F^{-1}U & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta v \end{bmatrix} = \begin{bmatrix} r \\ r_{\text{primal}} \end{bmatrix}
\tag{12}
$$

for ΔX and Δv, where $r = r_{\text{dual}} - F^{-1}r_{\text{cent}}$. Notice that since $F^{-1}U$ is positive definite, [38, Thm. 3.2], (12) also describes the optimality condition for the following convex optimization problem

$$
\begin{aligned}
\underset{\Delta x}{\text{minimize}} \quad & \frac{1}{2}\Delta x^T F^{-1} U \Delta x + r^T \Delta x \\
\text{subject to} \quad & A\Delta x = r_{\text{primal}}.
\end{aligned}
\tag{13}
$$

So it is possible to compute ΔX and Δv by either solving the system of equations in (12) or the problem in (13). In this paper we focus on predictor-corrector primal-dual methods that rely on modified Newton directions. In order to compute these directions, at each iteration, we need to solve (12) or (13) twice with different choices of r. We lay out a predictor-corrector primal-dual interior-point method in Algorithm 1, based on the work in [38].

Remark 1: Algorithm 1 can detect infeasibility of the problem if either the primal or dual iterates diverge. This means that this algorithm is unable to detect weak infeasibility, [15], [28], and generally in such cases converges to a near-feasible solution, [39].

The major computational burden of each iteration of this primal-dual method concerns the computation of the predictor and corrector directions. Next we will investigate how the

Algorithm 1 Predictor-corrector Primal-dual Interior-point Method, [38]

1: Given $k = 0$, $\tau \in (0, 1)$, $a \in \{1, 2, 3\}$, $\epsilon > 0$, $\epsilon_{\text{feas}} > 0$, initial iterates $(X^{(0)}, S^{(0)}, v^{(0)})$ such that $X^{(0)} \succ 0$ and $S^{(0)} \succ 0$ and $\mu = X^{(0)} \bullet S^{(0)}/n$.

2: **repeat**

3:     Compute D.

4:     Predictor step: Set $\sigma = 0$ and compute the search directions ΔX_{pred}, Δv_{pred} by either solving (12) or (13) and ΔS_{pred} using (11).

5:     Compute primal and dual step sizes as

$$\alpha_p := \min\left(1, \frac{-\tau}{\lambda_{\min}\left((X^{(k)})^{-1}\Delta X_{\text{pred}}\right)}\right),$$
$$\alpha_d := \min\left(1, \frac{-\tau}{\lambda_{\min}\left((S^{(k)})^{-1}\Delta S_{\text{pred}}\right)}\right).$$

6:     Set $\sigma = \left(\frac{(X^{(k)}+\alpha_p\Delta X_{\text{pred}})\bullet(S^{(k)}+\alpha_d\Delta S_{\text{pred}})}{X^{(k)}\bullet S^{(k)}}\right)^a$.

7:     Corrector step: Having computed σ compute the search directions ΔX_{corr}, Δv_{corr} by either solving (12) or (13) with

$$r^{(k)} = r_{\text{dual}}^{(k)} - (F^{(k)})^{-1}r_{\text{cent}}^{(k)} +$$
$$(F^{(k)})^{-1}\,\text{svec}(H_D(\Delta X_{\text{pred}}\Delta S_{\text{pred}})),$$

    and ΔS_{corr} using

$$\Delta s_{\text{corr}} = (F^{(k)})^{-1}\left(r_{\text{cent}}^{(k)} -\right.$$
$$\left.\text{svec}(H_D(\Delta X_{\text{pred}}\Delta S_{\text{pred}})) - U^{(k)}\Delta x_{\text{corr}}\right).$$

8:     Compute primal and dual step sizes as above, though using ΔX_{corr} and ΔS_{corr}.

9:     Update

$$X^{(k+1)} = X^{(k)} + \alpha_p \Delta X_{\text{corr}},$$
$$S^{(k+1)} = S^{(k)} + \alpha_d \Delta S_{\text{corr}},$$
$$v^{(k+1)} = v^{(k)} + \alpha_d \Delta v_{\text{corr}}.$$

10:     Set $k = k + 1$.

11:     $\mu = X^{(k)} \bullet S^{(k)}/n$.

12: **until** $\left\|r_{\text{primal}}^{(k)}\right\|^2$, $\left\|\text{svec}\left(R_{\text{dual}}^{(k)}\right)\right\|^2 \leq \epsilon_{\text{feas}}$ and $\mu \leq \epsilon$.

---

structure in coupled problems is reflected in (13) and how this structure can be used to our advantage.

Let us apply the primal-dual method in Algorithm 1 to the coupled SDP in (2). The perturbed KKT optimality conditions for this problem can be written as

$$Q_j^i \bullet \bar{X}^i = b_j^i, \quad j = 1, \ldots, m_i, \quad (14\text{a})$$

$$\sum_{j=1}^{m_i} v_j^i Q_j^i - \text{smat}(\bar{v}^i) + S^i = W^i, \quad (14\text{b})$$

$$\bar{X}^i S^i = \delta I, \quad (14\text{c})$$

$$\bar{X}^i - X_{J_i J_i} = 0, \quad (14\text{d})$$

for $i = 1, \ldots, N$, together with

$$\sum_{i=1}^{N}(E_{J_i} \otimes_s E_{J_i})^T \bar{v}^i = 0, \quad (15)$$

and $\bar{X}^i, S^i \succ 0$ for $i = 1, \ldots, N$. Define $\mathcal{Q}^i = \left[\text{svec}(Q_1^i) \quad \cdots \quad \text{svec}(Q_{m_i}^i)\right]^T$. Similar to (7), given iterates X and $\bar{X}^i \succ 0$ such that they satisfy (2d), $S^i \succ 0$, v^i and $\bar{v}^i$ such that $\sum_{i=1}^{N}(E_{J_i} \otimes_s E_{J_i})^T \bar{v}^i = 0$ for all $i = 1, \ldots, N$, the Newton step corresponding to the above system of equations can be computed by solving

$$\mathcal{Q}^i \Delta \bar{x}^i = b^i - \mathcal{Q}^i \bar{x}^i, \quad (16\text{a})$$

$$\sum_{j=1}^{m_i} \Delta v_j^i Q_j^i - \text{smat}(\Delta \bar{v}^i) + \Delta S^i =$$
$$W^i - \sum_{j=1}^{m_i} v_j^i Q_j^i + \text{smat}(\bar{v}^i) - S^i, \quad (16\text{b})$$

$$H_{D^i}(\Delta \bar{X}^i S^i + \bar{X}^i \Delta S^i) = \delta I - H_{D^i}(\bar{X}^i S^i), \quad (16\text{c})$$

$$\Delta \bar{X}^i - \Delta X_{J_i J_i} = 0, \quad (16\text{d})$$

for $i = 1, \ldots, N$, together with $\sum_{i=1}^{N}(E_{J_i} \otimes_s E_{J_i})^T \Delta \bar{v}^i = 0$, where the scaling matrices D^i are computed as discussed above and in (8), though based on the given local iterates $\bar{X}^{i,(k)}$ and $S^{i,(k)}$. This system of equations can be rewritten in a more compact manner as

$$\begin{bmatrix} 0 & 0 & \mathcal{Q} & 0 & 0 \\ 0 & 0 & I & -\bar{\mathcal{E}} & 0 \\ \mathcal{Q}^T & I & 0 & 0 & I \\ 0 & -\bar{\mathcal{E}}^T & 0 & 0 & 0 \\ 0 & 0 & \mathbf{U} & 0 & \mathbf{F} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{v} \\ \Delta\bar{\mathbf{v}} \\ \Delta\bar{\mathbf{x}} \\ \Delta x \\ \Delta\mathbf{s} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_{\text{primal}} \\ 0 \\ \mathbf{r}_{\text{dual}} \\ 0 \\ \mathbf{r}_{\text{cent}} \end{bmatrix} \quad (17)$$

where $\mathcal{Q}$, $\mathbf{U}$ and $\mathbf{F}$ are block-diagonal with diagonal blocks $\mathcal{Q}^i$ and

$$U^i = D^i \otimes_s (D^i)^{-T} S^i,$$
$$F^i = D^i \bar{X}^i \otimes_s (D^i)^{-T},$$

and $\bar{\mathcal{E}}^T = \left[(E_{J_1} \otimes_s E_{J_1})^T \quad \cdots \quad (E_{J_N} \otimes_s E_{J_N})^T\right]$. Also here $\Delta\mathbf{v}$, $\Delta\bar{\mathbf{v}}$, $\Delta\bar{\mathbf{x}}$ and $\Delta\mathbf{s}$ denote all the corresponding variables stacked, e.g., $\Delta\mathbf{v} = (\Delta v^1, \ldots, \Delta v^N)$. Similarly $\mathbf{r}_{\text{primal}}$, $\mathbf{r}_{\text{dual}}$ and $\mathbf{r}_{\text{cent}}$ denote all the primal, dual and centering residuals stacked, where each of the stacked terms in the residual vectors are based on

$$r_{\text{primal}}^i = b^i - \mathcal{Q}^i x^i, \quad (18\text{a})$$

$$R_{\text{dual}}^i = W^i - \sum_{j=1}^{m_i} v_j^i Q_j^i + \text{smat}(\bar{v}^i) - S^i, \quad (18\text{b})$$

$$R_{\text{cent}}^i = \delta I - H_{D^i}(\bar{X}^i S^i). \quad (18\text{c})$$

Similar to before, we compute the primal-dual directions by first solving for $\Delta\mathbf{s}$ as

$$\Delta s^i = (F^i)^{-1}\left(r_{\text{cent}}^i - U^i \Delta \bar{x}^i\right), \quad (19)$$

or equivalently as

$$\Delta s^i = (F^i)^{-1}\left(r_{\text{cent}}^i - U^i \operatorname{svec}(E_{J_i}^T \Delta X E_{J_i})\right), \qquad (20)$$

for $i = 1, \ldots, N$. Then we solve

$$\begin{bmatrix} -\mathbf{F}^{-1}\mathbf{U} & 0 & \mathcal{Q}^T & I \\ 0 & 0 & 0 & -\bar{\mathcal{E}}^T \\ \mathcal{Q} & 0 & 0 & 0 \\ I & -\bar{\mathcal{E}} & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta\bar{\mathbf{x}} \\ \Delta x \\ \Delta\mathbf{v} \\ \Delta\bar{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} \mathbf{r} \\ 0 \\ \mathbf{r}_{\text{primal}} \\ 0 \end{bmatrix}, \quad (21)$$

where $\mathbf{r} = (r^1, \ldots, r^N)$ with $r^i = r_{\text{dual}}^i - (F^i)^{-1}r_{\text{cent}}^i$. Notice that the system of equations in (21) also describes the optimality conditions for

$$\underset{\Delta\bar{\mathbf{x}}, \Delta x}{\text{minimize}} \quad \sum_{i=1}^{N} \frac{1}{2}(\Delta\bar{x}^i)^T (F^i)^{-1} U^i \Delta\bar{x}^i + (r^i)^T \Delta\bar{x}^i \quad (22a)$$

$$\text{subject to} \quad \mathcal{Q}^i \Delta\bar{x}^i = r_{\text{primal}}^i, \quad i = 1, \ldots, N, \qquad (22b)$$

$$\Delta\bar{X}^i - \Delta X_{J_i J_i} = 0, \quad i = 1, \ldots, N. \qquad (22c)$$

where $(F^i)^{-1}U^i \succ 0$ for $i = 1, \ldots, N$. So the predictor and corrector directions can also be computed by solving (22). To be more precise, for the predictor directions, we solve (22), with $\sigma = 0$, for $\Delta\bar{\mathbf{x}}_{\text{pred}}, \Delta x_{\text{pred}}, \Delta\mathbf{v}_{\text{pred}}$ and $\Delta\bar{\mathbf{v}}_{\text{pred}}$, and compute $\Delta\mathbf{s}_{\text{pred}}$ using (19). For the corrector directions, using the updated σ, we compute the directions $\Delta\bar{\mathbf{x}}_{\text{corr}}, \Delta x_{\text{corr}}, \Delta\mathbf{v}_{\text{corr}}$ and $\Delta\bar{\mathbf{v}}_{\text{corr}}$ by solving (22) with

$$\begin{aligned} r^i = r_{\text{dual}}^i \\ - (F^i)^{-1}\left(r_{\text{cent}}^i - \operatorname{svec}(H_{D^i}(\Delta\bar{X}_{\text{pred}}^i \Delta S_{\text{pred}}^i))\right) \end{aligned} \quad (23)$$

and compute $\Delta\mathbf{s}_{\text{corr}}$ as

$$\begin{aligned} \Delta s_{\text{corr}}^i = (F^i)^{-1}\big(r_{\text{cent}}^i - \\ \operatorname{svec}(H_{D^i}(\Delta\bar{X}_{\text{pred}}^i \Delta S_{\text{pred}}^i)) - U^i\Delta\bar{x}_{\text{corr}}^i\big), \end{aligned} \quad (24)$$

for $i = 1, \ldots, N$. As a result having computed predictor or corrector versions of the directions $\Delta\bar{\mathbf{x}}, \Delta x, \Delta\mathbf{v}$ and $\Delta\bar{\mathbf{v}}$, computing Δs_{pred}^i and Δs_{corr}^i can be done independently by N computing agents in parallel. Also notice that the coupling structure in (22) is the same as in (2). This allows us to employ distributed computational algorithms to distributedly solve for the search directions using N collaborating agents. To illustrate this, note that the problem in (22) can be written as

$$\begin{aligned} \underset{\bar{x}, x}{\text{minimize}} \quad & F_1(\bar{x}) \\ \text{subject to} \quad & A\bar{x} + Bx = c, \end{aligned} \quad (25)$$

with $\bar{x} = (\Delta\bar{x}^1, \ldots, \Delta\bar{x}^N)$ and $x = \Delta x$. This problem can be solved distributedly using proximal splitting methods, e.g., ADMM, [6], [10], [34]. The use of proximal splitting methods for computing the primal-dual directions has been considered in [3], [21]. Devising distributed algorithms for solving coupled SDPs that also rely on this approach can be seen as an extension of the use of the algorithm proposed in [3] to SDPs. Even though distributed algorithms based on proximal splitting are effective for non-conic problems, they suffer from certain issues when used for solving SDPs. Particularly, notice that the computed search directions using
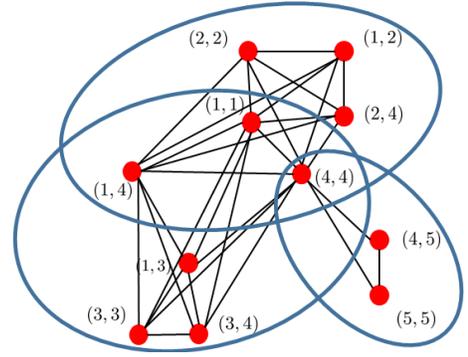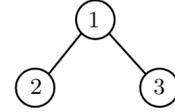

Fig. 2. Clustered sparsity graph.

$$K_1 = \{(1,1),(1,2),(1,4),(2,2),(2,4),(4,4)\}$$

$$K_2 = \{(1,1),(1,3),(1,4),(3,3),(3,4),(4,4)\} \quad K_3 = \{(4,4),(4,5),(5,5)\}$$

Fig. 3. Tree representation of the sparsity graph .

this approach are inexact and first-order splitting methods generally require many iterations to compute accurate enough search directions. Furthermore, the number of consensus constraints in (22c) are generally large for coupled SDPs which can in turn adversely affect the performance and numerical properties of such splitting methods. Also notice that for a predictor-corrector primal-dual method the search directions are computed through solving a system of the form (22) twice. This means that the iterative scheme for solving (22) needs to be run twice at each iteration of the primal-dual method. Hence, distributed algorithms that rely on proximal or first-order splitting for computing the search directions, potentially, require many iterations to converge to the solution. Despite all such issues, in many cases such splitting methods are among the only resorts for distributedly solving coupled or loosely coupled SDPs. However for coupled problems that have an inherent tree structure, which is common in loosely coupled SDPs, we can devise an efficient algorithm for solving coupled SDPs. This is the focus of the upcoming sections. But first we express what we mean by the tree structure.

IV. TREE STRUCTURE IN COUPLED PROBLEMS AND MESSAGE PASSING

Let us reconsider the coupled SDP in (4). Notice that for this problem it is possible to cluster the variables or the nodes in its sparsity graph as shown in Figure 2. As can be seen from the figure, each of the clusters induce a complete subgraph on the sparsity graph. We can then provide a more compact representation of the sparsity graph using the tree in Figure 3. Each node in this tree corresponds to each of the clusters of variables denoted by K_i. Furthermore, for this problem, the tree is such that for every two nodes i and j in the tree, $K_i \cap K_j$

is contained in all the clusters in the path connecting the two nodes in the tree. We refer to problems that enjoy this inherent structure as coupled with a tree structure. Next we lay out an approach for exploiting this structure in coupled problems.

Let us start by describing some definitions relating to graphs. Consider a graph $Q(V, \mathcal{E})$. A clique C_i of this graph is a maximal subset of V that induces a complete subgraph on Q, i.e., no clique is properly contained in another clique, [9]. Assume that all cycles of length at least four of $Q(V, \mathcal{E})$ have a chord, where a chord is an edge between two non-consecutive vertices in a cycle. This graph is then called chordal [17, Ch. 4]. It is possible to make a non-chordal graph chordal by adding edges to the graph. The resulting graph is then referred to as a chordal embedding. Let $\mathbf{C}_Q = \{C_1, \dots, C_q\}$ denote the set of its cliques, where q is the number of cliques of the graph. Then there exists a tree defined on $\mathbf{C}_Q$ such that for every $C_i, C_j \in \mathbf{C}_Q$ where $i \neq j$, $C_i \cap C_j$ is contained in all the cliques in the path connecting the two cliques in the tree. This property is called the clique intersection property, [9], and trees with this property are referred to as clique trees. As a result it is possible to represent chordal graphs using clique trees. This means that in case the sparsity graph is chordal, it is possible to use algorithms for generating clique trees for chordal graphs, to extract the aforementioned tree structure in the problem. In fact this has been used for the coupled example in (4). Notice that the sparsity graph for this example is chordal, and the clusters marked in Figure 2 are its cliques. Their corresponding clique tree is depicted in Figure 3. Also notice that in case the sparsity graph is not chordal, the same procedure can be used on its chordal embedding for extracting the tree structure. Coupled problems with a tree structure can be solved using a message-passing algorithm. Consider the following coupled convex optimization problem

$$\underset{x}{\text{minimize}} \quad f_1(x) + f_2(x) + \cdots + f_N(x), \quad (26)$$

where $f_i : \mathbb{R}^n \to \mathbb{R}$ for $i = 1, \dots, N$. This problem can be seen as a combination of N subproblems, each of which is defined by a term in the cost function and depends only on a few elements of x. Let us describe the coupling structure in this problem in a similar manner as we did for the coupled SDP in (1). That is we denote the ordered set of indices of x that each subproblem i depends on by J_i, and we denote the ordered set of indices of functions that depend on x_j by $\mathcal{I}_j$. We can equivalently rewrite this problem as

$$\underset{x}{\text{minimize}} \quad \bar{f}_1\left(x_{J_1}\right) + \cdots + \bar{f}_N(x_{J_N}), \quad (27)$$

where the functions $\bar{f}_i : \mathbb{R}^{|J_i|} \to \mathbb{R}$ are lower dimensional descriptions of f_is such that $f_i(x) = \bar{f}_i(E_{J_i} x)$ for all $x \in \mathbb{R}^n$ and $i = 1, \dots, N$. Let us assume that the sparsity graph of this problem, $Q_s(V_s, \mathcal{E}_s)$, has an inherent tree structure with a set of cliques $\mathbf{C}_{Q_s} = \{C_1, \dots, C_q\}$ and a clique tree, $T(V_t, \mathcal{E}_t)$. This problem can be solved distributedly using the message-passing algorithm that utilizes the clique tree as its computational graph. This means that the nodes $V_t = \{1, \dots, q\}$ act as computational agents that communicate or collaborate with their neighbors defined by the edge set $\mathcal{E}_t$. In order to describe the message-passing algorithm, we first need to assign each

subproblem in (27) to each of the agents. We can assign a subproblem or function $\bar{f}_i$ to an agent j if $J_i \subseteq C_j$. Let us denote the set of indices of the subproblems assigned to agent j by ϕ_j. Then we can rewrite (26) as

$$\underset{x}{\text{minimize}} \quad \sum_{i=1}^{q} F_i\left(x_{C_i}\right), \quad (28)$$

where $F_i\left(x_{C_i}\right) := \sum_{j \in \phi_i} \bar{f}_j\left(x_{J_i}\right)$. The message-passing algorithm, much the same as dynamic programming, solves (28) by performing an upward-downward pass through the clique tree, see e.g., [22, Sec. 4], [26] and references therein. Next we show how the message-passing algorithm can be used for devising distributed solvers for coupled SDPs with a tree structure.

V. DISTRIBUTED PRIMAL-DUAL INTERIOR-POINT METHODS FOR COUPLED SDPS

Let us reconsider the coupled SDP in (1), and assume that the sparsity graph of this problem, $Q_s(V_s, \mathcal{E}_s)$, has an inherent tree structure with clique set $\mathbf{C}_{Q_s} = \{C_1, \dots, C_q\}$ and clique tree $T(V_t, \mathcal{E}_t)$. Here we propose a method that allows us to solve this problem distributedly over the clique tree. To this end, we first need to assign the constituent subproblems of (1) to each of the agents in the tree. Firstly define $\bar{C}_i \subseteq \mathbb{N}_n$ such that $\bar{C}_i \times_s \bar{C}_i = C_i$. Then we can assign a subproblem i to agent j if $J_i \subseteq \bar{C}_j$. As in Section IV, let us denote the set of indices of subproblems assigned to agent j by ϕ_j. The proposed algorithm in this section relies on primal-dual interior-point methods. As was discussed in Section III, the most computationally demanding stage within the primal-dual method in Algorithm 1 concerns the computation of the predictor and corrector directions. Hence the first step for devising a distributed algorithm for solving coupled SDPs is to distribute the computation of these directions, which is discussed next.

A. Distributed Computation of Primal-dual Directions Using Message-passing

Recall that we can compute the predictor and corrector directions by solving the problem in (22) for different choices of $r^{i,(k)}$. Firstly notice that the problem in (22) is equivalent to the problem

$$\underset{\Delta x}{\text{minimize}} \quad \sum_{i=1}^{N} \frac{1}{2}(\Delta x^i)^T (F^i)^{-1} U^i \Delta x^i + (r^i)^T \Delta x^i \quad (29a)$$

$$\text{subject to} \quad \mathcal{Q}^i \Delta x^i = r^i_{\text{primal}}, \ i = 1, \dots, N, \quad (29b)$$

with $\Delta x^i := \text{svec}(\Delta X_{J_i J_i})$, that is achieved by eliminating the constraints in (22c). It is then possible to compute the search directions by solving the problem in (29). Particularly, by solving this problem we compute primal variables direction $\text{svec}(\Delta X)$ and dual variables directions $\Delta \mathbf{v}$. Then we can construct the remaining primal and dual directions as

$$\begin{aligned}
\Delta \bar{X}^i &= \Delta X_{J_i J_i}, \\
\Delta \bar{v}^i &= (F^i)^{-1} U^i \mathcal{Q}^i \Delta \bar{x}^i + r^i - (\mathcal{Q}^i)^T \Delta v^i.
\end{aligned} \quad (30)$$

for $i = 1, \ldots, N$. Next theorem shows that these directions in fact satisfy the system of equations in (21).

Theorem 1: The primal-dual directions computed by solving (29) and using (30) satisfy the system of equations in (21).

Proof: Notice that any solution of (29) satisfies

$$\bar{\mathcal{E}}^T \left(\mathbf{F}^{-1}\mathbf{U}\bar{\mathcal{E}}\Delta x - \mathcal{Q}^T\Delta \mathbf{v} \right) = -\bar{\mathcal{E}}^T\mathbf{r}, \qquad (31a)$$

$$\mathcal{Q}\bar{\mathcal{E}}\Delta x = \mathbf{r}_{\text{primal}}. \qquad (31b)$$

By choosing $\Delta\bar{X}^i = \Delta X_{J_i J_i}$, the primal directions, Δx and $\Delta\bar{\mathbf{x}}$, will satisfy the third and fourth block equations in (21). Furthermore, notice that by (31a) we have that

$$\mathbf{F}^{-1}\mathbf{U}\Delta\bar{\mathbf{x}} - \mathcal{Q}^T\Delta\mathbf{v} + \mathbf{r} \in \mathcal{N}(\bar{\mathcal{E}}^T).$$

So if we set

$$\Delta\bar{v}^i = (F^i)^{-1}U^i\mathcal{Q}^i\Delta\bar{x}^i + r^i - (\mathcal{Q}^i)^T\Delta v^i. \qquad (32)$$

for $i = 1, \ldots, N$, not only the primal-dual iterates satisfy the first block equation in (21), but also we have $\bar{\mathcal{E}}^T\Delta\bar{\mathbf{v}} = 0$. This completes the proof. ∎

Consequently, we can construct the primal-dual solutions for the problem in (22) by first solving the problem in (29) and constructing the remainder of the solution as outlined in (30). Notice that the coupling structure of (29) is the same as that of (1). This means that both problems have the same sparsity graph and tree representation of the coupling structure. We can equivalently rewrite (29) as

$$\text{minimize} \quad \sum_{i=1}^{N} \bar{f}_i\left(\Delta x^i\right), \qquad (33)$$

where $\bar{f}_i\left(\Delta x^i\right) := f_i\left(\Delta x^i\right) + \mathcal{I}_{\mathcal{C}_i}\left(\Delta x^i\right)$, with

$$f_i\left(\Delta x^i\right) = (\Delta x^i)^T(F^i)^{-1}U^i\Delta x^i + (r^i)^T\Delta x^i, \qquad (34)$$

for $i = 1, \ldots, N$, and functions $\mathcal{I}_{\mathcal{C}_i}$ for $i = 1, \ldots, N$, are the indicator functions for the constraints in (29b), i.e.,

$$\mathcal{I}_{\mathcal{C}_i}\left(\Delta x^i\right) = \begin{cases} 0 & \mathcal{Q}^i\Delta x^i = r^i_{\text{primal}} \\ \infty & \text{Otherwise} \end{cases}.$$

This problem is in the same format as (27), and due to its coupling structure, can be solved distributedly using message passing, see [22, Sec. 6.2].

So far we have described how to distribute the computation of the search directions using message passing. However, it remains to discuss how to distributedly compute the primal and dual step sizes, update the perturbation parameter and decide on terminating the algorithm. We discuss these next.

B. Distributed Step-size Computation and Termination Check

The clique tree used for computing the search directions can also be used for performing the remaining computations in Algorithm 1 distributedly. Notice that the computations described in this section are different than that of presented in [22]. This is because here we rely on a predictor-corrector method and we are concerned with SDPs. Let us first focus on step size computation. Similar to the message-passing

algorithm, in order to compute the primal and dual step sizes we need to perform an upward-downward pass over the clique tree. We start the computation from the agents at the leaves of the tree, where every such agent first computes

$$\underline{\lambda}_p^i = \min_{j \in \phi_i} \left(\lambda_{\min}\left((\bar{X}^j)^{-1}\Delta\bar{X}_{\text{pred}}^j \right) \right), \qquad (35a)$$

$$\underline{\lambda}_d^i = \min_{j \in \phi_i} \left(\lambda_{\min}\left((S^j)^{-1}\Delta S_{\text{pred}}^j \right) \right), \qquad (35b)$$

and communicates them to its corresponding parent. Each agent i that has received these quantities from their children, will then compute

$$\underline{\lambda}_p^i = \min\left(\min_{j \in \text{ch}(i)} \left(\underline{\lambda}_p^j \right), \min_{j \in \phi_i} \left(\lambda_{\min}\left((\bar{X}^j)^{-1}\Delta\bar{X}_{\text{pred}}^j \right) \right) \right), \qquad (36a)$$

$$\underline{\lambda}_d^i = \min\left(\min_{j \in \text{ch}(i)} \left(\underline{\lambda}_d^j \right), \min_{j \in \phi_i} \left(\lambda_{\min}\left((S^j)^{-1}\Delta S_{\text{pred}}^j \right) \right) \right), \qquad (36b)$$

and will communicate them to its parent. This procedure is then continued until we arrive at the root of the tree. At this point, the agent at the root computes the primal and dual step sizes as

$$\alpha_p := \min\left(1, \frac{-\tau}{\underline{\lambda}_p^r} \right), \quad \alpha_d := \min\left(1, \frac{-\tau}{\underline{\lambda}_d^r} \right), \qquad (37)$$

where $\underline{\lambda}_p^r$ and $\underline{\lambda}_d^r$ are calculated as in (36). These quantities are then communicated downwards through the tree until they reach the agent at the leaves. At this point, all agents will know the primal and dual step sizes. So the step sizes computation can be done by an upward-downward pass over the tree. Notice that the need for computing primal and dual step sizes also appear in Step 7 of Algorithm 1. We can use the same procedure for computing the step sizes at this step by simply replacing the predictor directions with corrector ones.

As can be seen from Algorithm 1, in order to compute the corrector directions we first need to update the parameter σ in Step 6 of the algorithm. We can use a similar approach to perform this update distributedly over the clique tree. Let us start the computations from the leaves of the tree. Every agent i at the leaves will then compute and communicate

$$\sigma_1^i = \sum_{j \in \phi_i} (\bar{X}^j + \alpha_p\Delta\bar{X}_{\text{pred}}^j) \bullet (S^j + \alpha_d\Delta S_{\text{pred}}^j), \qquad (38a)$$

$$\sigma_2^i = \sum_{j \in \phi_i} \bar{X}^j \bullet S^j, \qquad (38b)$$

to its corresponding parent. Then every agent i that has received these quantities from its children computes and communicates

$$\sigma_1^i = \sum_{j \in \text{ch}(i)} \sigma_1^j + \sum_{j \in \phi_i} (\bar{X}^j + \alpha_p\Delta\bar{X}_{\text{pred}}^j) \bullet (S^j + \alpha_d\Delta S_{\text{pred}}^j), \qquad (39a)$$

$$\sigma_2^i = \sum_{j \in \text{ch}(i)} \sigma_2^j + \sum_{j \in \phi_i} \bar{X}^j \bullet S^j, \qquad (39b)$$

to its parent. This procedure is then continued until we reach the agent at the root. Then this agent also computes the quantities σ_1^r and σ_2^r as in (39) and calculates the update for σ as

$$\sigma = \left(\frac{\sigma_1^r}{\sigma_2^r}\right)^a. \tag{40}$$

This quantity is then communicated downwards through the tree until it reaches the leaves of the tree. Hence, at every iteration of the primal-dual method all agents will have an update of σ after an upward-downward pass over the clique tree.

It now remains to discuss distributed computation of terms in the stopping criteria. This concerns the computation of primal and dual residuals norms together with the surrogate duality gap. These quantities can also be computed distributedly over the clique tree using an analogous approach as above. Similarly as before let us start the computations from the leaves of the tree where every such agent computes

$$r_d^i = \sum_{j \in \phi_i} \left\| r_{\text{dual}}^j \right\|^2, \quad r_p^i = \sum_{j \in \phi_i} \left\| r_{\text{primal}}^j \right\|^2, \tag{41a}$$

$$\mu^i = \sum_{j \in \phi_i} \bar{X}^j \bullet S^j, \tag{41b}$$

based on the updated iterates, and communicates them to its parent. Then every agent i that has received the necessary information from its children will compute

$$r_d^i = \sum_{j \in \text{ch}(i)} r_d^j + \sum_{j \in \phi_i} \left\| r_{\text{dual}}^j \right\|^2, \tag{42a}$$

$$r_p^i = \sum_{j \in \text{ch}(i)} r_p^j + \sum_{j \in \phi_i} \left\| r_{\text{primal}}^j \right\|^2, \tag{42b}$$

$$\mu^i = \sum_{j \in \text{ch}(i)} \mu^j + \sum_{j \in \phi_i} \bar{X}^j \bullet S^j, \tag{42c}$$

based on the updated iterates, and communicates them to the respective parent. This procedure is then continued until we reach the agent at the root, which will compute the primal and dual residuals as

$$\left\| r_{\text{primal}}^{(k)} \right\|^2 = \sum_{j \in \text{ch}(r)} r_p^j + \sum_{j \in \phi_r} \left\| r_{\text{primal}}^j \right\|^2, \tag{43a}$$

$$\left\| r_{\text{dual}} \right\|^2 = \sum_{j \in \text{ch}(r)} r_d^j + \sum_{j \in \phi_r} \left\| r_{\text{dual}}^j \right\|^2, \tag{43b}$$

and the surrogate duality gap as

$$\mu = \frac{1}{\sum_{j=1}^N |J_j|} \left(\sum_{j \in \text{ch}(r)} \mu^j + \sum_{j \in \phi_r} \bar{X}^j \bullet S^j \right). \tag{44}$$

This agent will then check the stopping criteria as in Step 12 of Algorithm 1. If these criteria are satisfied, then the agent at the root will communicate the decision to terminate the algorithm downwards through the tree. Otherwise, this agent will instead communicate the surrogate duality gap. Agents will need this parameter for updating the perturbation parameter for the next iteration of the primal-dual method.

So far we have expressed how to distribute the computations in every iteration of the primal-dual method. Next we summarize the outlined distributed algorithm in this section.

C. Summary of the Algorithm and Its Computational Properties

Let us reconsider the coupled SDP in (1). Given such a problem and its corresponding sparsity graph, $Q_s(V_s, \mathcal{E}_s)$, we extract its tree structure based on clique set $C_{Q_s} = \{\bar{C}_1, \ldots, \bar{C}_q\}$. Having done so we have the computational graph for our algorithm and it is possible to assign the constituent subproblems to each of the agents using the guidelines in Section IV or at the beginning of Section V. We can now summarize our proposed distributed algorithm as below.

Given $k = 0$, $\tau \in (0,1)$, $a \in \{1,2,3\}$, $\epsilon > 0$, $\epsilon_{\text{feas}} > 0$, initial iterates $X^{(0)}$, $\bar{X}^{i,(0)} = X_{J_i J_i}^{(0)} \succ 0$, $S^{i,0} \succ 0$, $v^{i,0}$, $\bar{v}^{i,0}$ such that $\sum_{i=1}^N (E_{J_i} \otimes_s E_{J_i})^T \bar{v}^{i,(0)} = 0$, for $i = 1, \ldots, N$, and $\mu = \sum_{i=1}^N \bar{X}^{i,(0)} \bullet S^{i,(0)} / \left(\sum_{j=1}^N |J_j| \right)$

repeat
  for $i = 1, \ldots, q$ **do**
    Agent i, given $\bar{X}^{j,(k)}$, $S^{j,(k)}$, $\bar{v}^{j,(k)}$, $v^{j,(k)}$, $\sigma = 0$, and $r^{j,(k)} = r_{\text{dual}}^{j,(k)} - (F^{j,(k)})^{-1} r_{\text{cent}}^{j,(k)}$, form $\bar{f}_i\left(\Delta x^i\right)$ as in (33) for $j \in \phi_i$.
  end for

Perform an upward-downward pass and compute the predictor directions using message-passing, and (19) and (30).

Compute the primal and dual step sizes, by performing an upward-downward pass through the tree as discussed in Section V-B.

Update σ by performing an upward-downward pass through the tree as discussed in Section V-B.

  for $i = 1, \ldots, q$ **do**
    Agent i reforms their subproblems with $r^{j,(k)}$ as in (23) for $j \in \phi_i$.
  end for

Perform an upward-downward pass and compute the corrector directions using message-passing, and (24) and (30).

Compute the primal and dual step sizes, by performing an upward-downward pass through the tree as discussed in Section V-B.

  for $i = 1, \ldots, q$ **do**
    Agent i updates

$$X_{J_j J_j}^{(k+1)} := X_{J_j J_j}^{(k)} + \alpha_p (\Delta X_{J_j J_j})_{\text{corr}},$$
$$\bar{X}^{j,(k+1)} := \bar{X}^{j,(k)} + \alpha_p \Delta \bar{X}_{\text{corr}}^j,$$
$$S^{j,(k+1)} := S^{j,(k)} + \alpha_d \Delta S_{\text{corr}}^j,$$
$$v^{j,(k+1)} := v^{j,(k)} + \alpha_d \Delta v_{\text{corr}}^j,$$
$$\bar{v}^{j,(k+1)} := \bar{v}^{j,(k)} + \alpha_d \Delta \bar{v}_{\text{corr}}^j,$$

    for $j \in \phi_i$.
  end for
  $k = k + 1$.

Evaluate μ and the termination criteria by performing

an upward-downward pass through the tree and decide whether to terminate the algorithm.

until the algorithm is terminated

From the outlined algorithm, we can observe that each iteration of the primal-dual method is accomplished within six upward-downward passes through the tree. Namely, two passes for computing the predictor and corrector directions, two for computing primal and dual step sizes, one for updating σ and one for evaluating the stopping criteria and computing the surrogate duality gap. Let the height of the tree, that is the maximum number of edges in a path from the root to a leaf, be h. As a result, each iteration of the primal-dual method is accomplished in $6 \times 2 \times h$ steps. Furthermore, among these passes the ones required for computing the predictor and corrector directions are by far the most computationally demanding ones. This is mainly because during the upward message-passing for these passes, every agent i needs to factorize a matrix, see [22, Sec. 6.2]. However, notice that at every iteration of the primal-dual method, this matrix is the same for the predictor and corrector directions computations. This means that if each agent pre-caches the factorization of this matrix during predictor directions computations, it can reuse it for corrector directions computation, see [22, Remark 8]. This significantly reduces the computational burden of the upward-downward pass for computing corrector directions. Let us assume that the primal-dual method converges within p iterations. Then the major computational burden for each agent concerns the computation of p factorizations of a matrix, that is commonly of comparatively small size for loosely coupled problems. This is in stark contrast to distributed algorithms that purely rely on first-order splitting methods, as at every iteration of such algorithms each agent is required to solve an SDP.

Remark 2: The algorithm presented in this section, can distributedly detect infeasibility in the sense discussed in Remark 1, by monitoring their local primal and dual variables. In case any agent detects divergence of these variables, it can then communicate the occurrence through the tree to terminate the algorithm.

Next we discuss a class of sparse SDPs, that appear in robustness analysis of large-scale interconnected uncertain systems, and we will describe how such problems can be reformulated as coupled SDPs with an inherent tree structure.

VI. CHORDAL SPARSITY AND DOMAIN-SPACE DECOMPOSITION

In order to describe sparsity in SDPs, we first briefly discuss the use of graphs for expressing sparsity patterns of symmetric matrices.

A. Sparsity and Semidefinite Matrices

Consider a symmetric matrix $X \in \mathbb{S}^n$, and an undirected graph $H(V, \mathcal{E})$ with $V = \{1, \ldots, n\}$ and $\mathcal{E} = \{(i, j) \in (V \times V) \mid X_{ij} \neq 0, i \neq j\}$. We refer to this graph as the sparsity pattern graph of X. It is also possible to use undirected graphs to describe partial symmetric matrices. A

partial symmetric matrix is a symmetric matrix where only a subset of its elements are specified and the rest are free. For the symmetric matrix X this structure can be expressed using $H(V, \mathcal{E})$ with $V = \{1, \ldots, n\}$ and $\mathcal{E} \subseteq (V \times V)$. Particularly, the edge set is such that we can express the set of indices of specified elements using $\mathbf{I}_s = \mathcal{E} \cup \{(i, i) \mid i = 1, \ldots, n\}$. We denote the set of partial symmetric matrices over $H(V, \mathcal{E})$ by $\mathbb{S}_H^n$. A matrix $X \in \mathbb{S}_H^n$ is then said to be positive semidefinite completable if by choosing its free elements, i.e., elements with indices in $\mathbf{I}_f = (V \times V) \setminus \mathbf{I}_s$, it is possible to produce a positive semidefinite matrix. Such matrices play a central role in the upcoming discussions. Let us review a fundamental result concerning semidefinite completable matrices.

Theorem 2: [18, Thm. 7] Let $H(V, \mathcal{E})$ be a chordal graph with clique set $\mathbf{C}_H = \{\bar{C}_1, \ldots, \bar{C}_l\}$ such that clique intersection property holds. Then $X \in \mathbb{S}_H^n$ is positive semidefinite completable, if and only if

$$X_{\bar{C}_i \bar{C}_i} \succeq 0, \quad i = 1, \ldots, l. \tag{45}$$

We will next discuss how this theorem can be used for reformulating sparse SDPs.

B. Domain-space Decomposition

Consider the following inequality-form SDP

$$\underset{y}{\text{minimize}} \quad c^T y \tag{46a}$$

$$\text{subject to} \quad \sum_{i=1}^{g} E_{J_i}^T Q^i E_{J_i} y_i + \sum_{i=1}^{g} E_{J_i}^T M^i E_{J_i} \preceq 0 \tag{46b}$$

where $y \in \mathbb{R}^g$, $Q^i, M^i \in \mathbb{S}^{|J_i|}$ and $J_i \subset \mathbb{N}_n$ for $i = 1, \ldots, g$. Let us denote the sparsity pattern graph for the matrix $\sum_{i=1}^{g} E_{J_i}^T E_{J_i}$ with $H(V, \mathcal{E})$. Assume that this graph is chordal, or that we can produce a chordal embedding by adding a few edges, with clique set $\mathbf{C}_H = \{\bar{C}_1, \ldots, \bar{C}_l\}$. The dual problem for (46) is given as

$$\underset{Z}{\text{minimize}} \quad -\sum_{i=1}^{g} Z_{J_i J_i} \bullet M^i \tag{47a}$$

$$\text{subject to} \quad Z_{J_i J_i} \bullet Q^i = -c_i, \quad i = 1, \ldots, g, \tag{47b}$$

$$Z \succeq 0. \tag{47c}$$

We can observe that the only elements that affect the equality constraints and the cost function are the ones specified by $\mathbf{I}_s$. The rest are only used in the semidefinite constraint. This in turn implies that $Z \in \mathbb{S}_H^n$, and using Theorem 2, allows us to equivalently rewrite (47) as

$$\underset{Z_{\bar{C}_1 \bar{C}_1}, \ldots, Z_{\bar{C}_l \bar{C}_l}}{\text{minimize}} \quad -\sum_{i=1}^{g} Z_{J_i J_i} \bullet M^i \tag{48a}$$

$$\text{subject to} \quad Z_{J_i J_i} \bullet Q^i = -c_i, \quad i = 1, \ldots, g, \tag{48b}$$

$$Z_{\bar{C}_i \bar{C}_i} \succeq 0, \quad i = 1, \ldots, l. \tag{48c}$$

This method of reformulating (47) as (48) is referred to as the domain-space decomposition [16], [24]. Notice that for every J_i there exists a $\bar{C}_j$ such that $J_i \subseteq \bar{C}_j$. This is because every

set J_i induces a complete subgraphs on $H(V, \mathcal{E})$, and hence based on the definitions of cliques, it is either a subset of a clique or a clique itself. Let us denote the set of indices of sets J_i that are a subset of $\bar{C}_j$ by ϕ_j. We can then group the equality constraints in (48b) and rewrite the problem in (48) as

$$\underset{Z_{\bar{C}_1 \bar{C}_1}, \dots, Z_{\bar{C}_l \bar{C}_l}}{\text{minimize}} \quad -\sum_{i=1}^{g} Z_{J_i J_i} \bullet M^i \tag{49a}$$

$$\text{subject to} \quad Z_{J_j J_j} \bullet Q^j = -c_j, \; j \in \phi_i, \quad i = 1, \dots, l, \tag{49b}$$

$$Z_{\bar{C}_i \bar{C}_i} \succeq 0, \quad i = 1, \dots, l. \tag{49c}$$

which is in the same format as (1). This problem comprises l subproblems. Furthermore, due to its construction has a chordal sparsity graph with q cliques and a clique tree that has the same structure as the clique tree for $H(V, \mathcal{E})$, where instead of $\bar{C}_i$, the cliques are given as $\bar{C}_i \times_s \bar{C}_i$. In fact the chordality of the sparsity graph follows, since the ordering defined by the clique tree is also a perfect elimination ordering for this graph, see [17] for more details.

Remark 3: Notice that the discussion in this section also extends to matrices in positive semidefinite Hermitian cones, [18]. This means that the decomposition scheme described here, can also be used for problems with complex data matrices.

Next we will discuss robustness analysis of interconnected uncertain systems and will show how the approach described here can be used for reformulating this problem as a coupled SDP.

VII. ROBUSTNESS ANALYSIS OF INTERCONNECTED UNCERTAIN SYSTEMS

In this section, we discuss robustness analysis of interconnected uncertain systems using integral quadratic constraints (IQCs). We start this discussion by first reviewing the IQC analysis framework.

A. Robustness Analysis using IQCs

Consider the following uncertain system

$$p = Gq, \quad q = \Delta(p), \tag{50}$$

where $G \in \mathcal{RH}_{\infty}^{m \times m}$ is the system transfer function matrix, and $\Delta : \mathbb{R}^m \to \mathbb{R}^m$ is a bounded and causal operator representing the uncertainty in the system. We can characterize the uncertainty in the system using IQCs. Particularly it is said that Δ satisfies the IQC defined by Π, i.e., $\Delta \in \text{IQC}(\Pi)$, if

$$\int_0^\infty \begin{bmatrix} v \\ \Delta(v) \end{bmatrix}^T \Pi \begin{bmatrix} v \\ \Delta(v) \end{bmatrix} dt \geq 0, \quad \forall v \in \mathcal{L}_2^d, \tag{51}$$

where Π is a bounded and self-adjoint operator. This constraint can also be written in frequency domain as

$$\int_{-\infty}^{\infty} \begin{bmatrix} \widehat{v}(j\omega) \\ \widehat{\Delta(v)}(j\omega) \end{bmatrix}^* \Pi(j\omega) \begin{bmatrix} \widehat{v}(j\omega) \\ \widehat{\Delta(v)}(j\omega) \end{bmatrix} d\omega \geq 0, \tag{52}$$

where $\hat{v}$ and $\widehat{\Delta(v)}$ are the Fourier transforms of the signals [20], [31]. The uncertain system is then said to be robustly stable if the interconnection between G and Δ remains stable for all $\Delta \in \text{IQC}(\Pi)$. This can be established using the following theorem.

Theorem 3: The uncertain system in (50) is robustly stable, if

1) for all $\tau \in [0, 1]$ the interconnection described in (50), with $\tau\Delta$, is well-posed;
2) for all $\tau \in [0, 1]$, $\tau\Delta \in \text{IQC}(\Pi)$;
3) there exists $\epsilon > 0$ such that

$$\begin{bmatrix} G(j\omega) \\ I \end{bmatrix}^* \Pi(j\omega) \begin{bmatrix} G(j\omega) \\ I \end{bmatrix} \preceq -\epsilon I, \; \forall \omega \in [0, \infty]. \tag{53}$$

Proof: See [20], [31]. ∎

Satisfaction of the conditions in this theorem is a sufficient condition for robustness of the uncertain system. As a result, for robustness analysis of this system it is required to find a multiplier Π such that $\Delta \in \text{IQC}(\Pi)$ and that it satisfies the semi-infinite LMI in (53). The condition $\Delta \in \text{IQC}(\Pi)$ commonly imposes structural constraints on Π, and hence the analysis problem is then to find Π with a particular structure such that it satisfies (53). It is possible to do this using either the KYP lemma, [20], [35], or approximately using frequency-gridding, which establishes satisfaction of (53) over a finite frequencies. We utilize the latter approach later as it preserves the structure in the problem. Next we describe how this framework can be used for analyzing interconnected uncertain systems.

B. Robustness Analysis of Interconnected Uncertain Systems using IQCs

An interconnected uncertain system can be viewed as a network of N uncertain subsystems. We describe each of these subsystems as

$$\begin{aligned} p^i &= G_{pq}^i q^i + G_{pw}^i w^i \\ z^i &= G_{zq}^i q^i + G_{zw}^i w^i \\ q^i &= \Delta^i(p^i), \end{aligned} \tag{54}$$

where $G_{pq}^i \in \mathcal{RH}_{\infty}^{d_i \times d_i}$, $G_{pw}^i \in \mathcal{RH}_{\infty}^{d_i \times m_i}$, $G_{zq}^i \in \mathcal{RH}_{\infty}^{l_i \times d_i}$, $G_{zw}^i \in \mathcal{RH}_{\infty}^{l_i \times m_i}$, and $\Delta^i : \mathbb{R}^{d_i} \to \mathbb{R}^{d_i}$. It is possible to describe the interconnection among the subsystems using a 0–1 matrix Γ as

$$\begin{bmatrix} w^1 \\ w^2 \\ \vdots \\ w^N \end{bmatrix} = \underbrace{\begin{bmatrix} \Gamma_{11} & \Gamma_{12} & \cdots & \Gamma_{1N} \\ \Gamma_{21} & \Gamma_{22} & \cdots & \Gamma_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \Gamma_{N1} & \Gamma_{N2} & \cdots & \Gamma_{NN} \end{bmatrix}}_{\Gamma} \begin{bmatrix} z^1 \\ z^2 \\ \vdots \\ z^N \end{bmatrix}, \tag{55}$$

where each Γ_{ij} describes which components of z^j is connected to which components of w^i. Let us define $p = (p^1, \dots, p^N)$, $q = (q^1, \dots, q^N)$, $w = (w^1, \dots, w^N)$ and $z = (z^1, \dots, z^N)$.

Then we can compactly describe the entire interconnected uncertain system as

$$
\begin{aligned}
p &= G_{pq}q + G_{pw}w \\
z &= G_{zq}q + G_{zw}w \\
q &= \Delta(p) \\
w &= \Gamma z,
\end{aligned}
\tag{56}
$$

where $G_{\star\bullet} = \operatorname{diag}(G^1_{\star\bullet}, \ldots, G^N_{\star\bullet})$ and $\Delta = \operatorname{diag}(\Delta^1, \ldots, \Delta^N)$. Let us assume that the interconnected system is nominally or internally stable, i.e., $(I - \Gamma G_{zw})^{-1} \in \mathcal{RH}_\infty^{\bar{m}\times\bar{m}}$ with $\bar{m} = \sum_{i=1}^N m_i$. It was then shown in [2] that the system is robustly stable if there exist

$$
\bar{\Pi} = \begin{bmatrix} \bar{\Pi}_{11} & \bar{\Pi}_{12} \\ \bar{\Pi}_{21} & \bar{\Pi}_{22} \end{bmatrix},
$$

with $\bar{\Pi}_{\star\bullet} = \operatorname{diag}(\Pi^1_{\star\bullet}, \ldots, \Pi^N_{\star\bullet})$ and $\Delta^i \in \operatorname{IQC}\left(\begin{bmatrix} \Pi^i_{11} & \Pi^i_{12} \\ \Pi^i_{21} & \Pi^i_{22} \end{bmatrix}\right)$, and a diagonal matrix $X \succ 0$ such that

$$
\begin{bmatrix} G_{pq} & G_{pw} \\ I & 0 \end{bmatrix}^* \begin{bmatrix} \bar{\Pi}_{11} & \bar{\Pi}_{12} \\ \bar{\Pi}_{21} & \bar{\Pi}_{22} \end{bmatrix} \begin{bmatrix} G_{pq} & G_{pw} \\ I & 0 \end{bmatrix} - \\
\begin{bmatrix} -G_{zq}^*\Gamma^T \\ I - G_{zw}^*\Gamma^T \end{bmatrix} X \begin{bmatrix} -\Gamma G_{zq} & I - \Gamma G_{zw} \end{bmatrix} \preceq -\epsilon I. \tag{57}
$$

It is possible to rewrite this problem in the following standard form

$$
\text{find} \quad y \tag{58a}
$$
$$
\text{subject to} \quad \sum_{i=1}^m y_i \bar{Q}^i + W \preceq 0 \tag{58b}
$$

where $\bar{Q}^i \in \mathbb{H}^{\bar{m}+\bar{d}}$ for all $i = 1, \ldots, m$ and $W \in \mathbb{S}^{\bar{m}+\bar{d}}$ with $\bar{d} = \sum_{i=1}^N d_i$. We can equivalently rewrite the problem in (58) as below

$$
\text{find} \quad y \tag{59a}
$$
$$
\text{subject to} \quad \sum_{i=1}^m y_i \begin{bmatrix} \operatorname{Re}(\bar{Q}^i) & -\operatorname{Im}(\bar{Q}^i) \\ \operatorname{Im}(\bar{Q}^i) & \operatorname{Re}(\bar{Q}^i) \end{bmatrix} + \begin{bmatrix} W & 0 \\ 0 & W \end{bmatrix} \preceq 0 \tag{59b}
$$

where all the data matrices are real, [11]. In case Γ is sparse, then this SDP is also sparse and can be written in the same format as in (46) with $c = 0$. As a result we can use the approach presented in Section VI for reformulating this problem as a coupled problem, and employ the algorithm presented in Section V for solving it.

Remark 4: As was discussed in Remark 3, the decomposition can be conducted directly on (57) or (58). However, we here choose to reformulate the problem in (59), with real data matrices, for ease of notation and ease of use of the algorithm described in Section V.

Next we illustrate this approach and study the performance of the algorithm using numerical experiments.
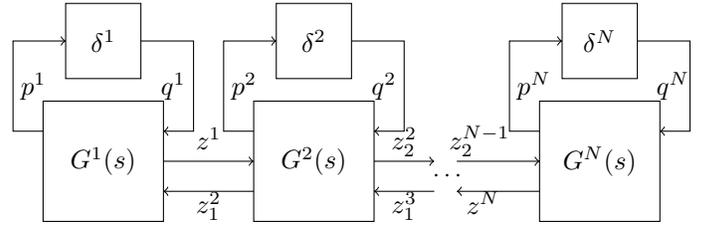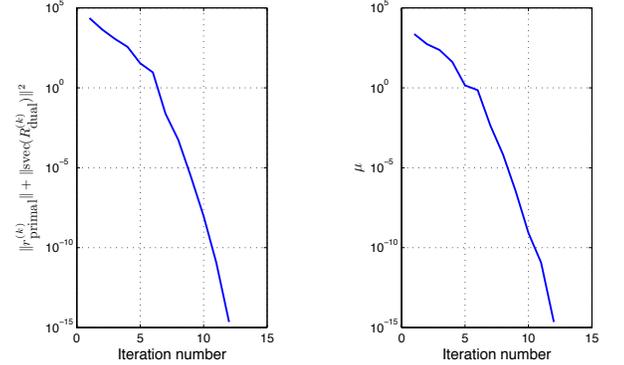

Fig. 4. A chain of N uncertain subsystem.

Fig. 5. Convergence behavior of the algorithm for analysis of a chain of uncertain systems. The figure on the left shows the sum of primal and dual residuals and the figure on the right depicts the surrogate duality gap.

VIII. NUMERICAL EXPERIMENTS

In this section we consider two examples, namely a chain of uncertain systems and an interconnected uncertain system over a so-called scale-free network. These examples are taken from [2]. Let us start with the analysis of a chain of uncertain systems, as illustrated in Figure 4. As can be seen from the figure, for subsystems $1 < i < N$, $z^i, w^i \in \mathbb{R}^2$ and for subsystems $i = 1, N$, $z^i, w^i \in \mathbb{R}$. The uncertainty in each subsystem i is represented using δ^i, which is assumed to be an unknown gain in the normalized interval $[-1, 1]$. We can hence describe the uncertainties as $\delta^i \in \operatorname{IQC}(\Pi^i)$ with $\Pi^i = \begin{bmatrix} r_i(j\omega) & 0 \\ 0 & -r_i(j\omega) \end{bmatrix}$, and $r_i(j\omega) \geq 0$, [31]. The interconnection matrix for this interconnected system is described by the nonzero blocks $\Gamma_{i,i-1} = \Gamma_{i-1,i}^T$ for $i = 2, \ldots, N$, where $\Gamma_{i,i-1} = \Gamma_{i-1,i}^T = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$, $i = 3, \ldots, N-1$, and $\Gamma_{21} = \Gamma_{12}^T = (1,0)$, $\Gamma_{N-1,N} = \Gamma_{N,N-1}^T = (0,1)$. We considered the analysis problem for this system with $N = 100$ subsystems in the chain, at a single frequency $\omega = 1\ rad/s$. We solved 10 instances of this problem with different transfer function matrices for the subsystems. The transfer function matrices for each instance were randomly generated using the approach presented in [2]. This guarantees that the interconnected system is robustly stable for all instances. Furthermore, for this problem the multiplier X

was chosen as $X = \mathrm{diag}(x_1, \ldots, x_{2N-2})$. This resulted in a problem in the same format as in (58), with $m = 298$ and $W \in \mathbb{S}^{298}$.

Forming (59) for this analysis problem, resulted in an LMI with a chordal sparsity pattern, with 198 cliques where the largest clique was of size 8. The clique tree over these cliques had a height of 99. In order to establish chordality of the sparsity pattern graph and generate its cliques a greedy search algorithm with *min degree* criterion was used, [13]. If we now form the problem in (49), this problem will comprise 198 subproblems and can be solved distributedly over the clique tree. The parameters within the primal-dual method were chosen to be the same for all instances and are chosen as $a = 1$, $\tau = 0.98$, $\epsilon = \epsilon_{\mathrm{feas}} = 10^{-12}$, $\bar{v}^{i,(0)} = 0$ and $v^{i,(0)} = 0$ for all $i = 1, \ldots, N$, and $X^{(0)}$ and $S^{i,(0)}$ for $i = 1, \ldots, N$ were chosen to be diagonal matrices with positive diagonal entries generated randomly with a uniform distribution in the interval $(0.1, 2)$. In the worst case the primal-dual method converged after 12 iterations. The convergence behavior of this instance is illustrated in Figure 5, and as can be seen mimics that of a standard primal-dual method, i.e., convergence within 10 to 50 iterations with a quadratic convergence phase, [11]. Considering the height of the tree, this algorithm then, in the worst case, converged after $6 \times 2 \times 99 \times 12 = 14256$ steps. During the run of the algorithm, each agent was required to compute a factorization 12 times and needed to communicate with its neighbors 144 times. The computations in the remaining steps were trivial.

We further tested the performance of the algorithm on a larger example with a more complicated interconnection description. Particularly we used the same scale-free network as in [2, Sec. 5.2] for describing the interconnections among the subsystems. This resulted in an extremely sparse interconnection matrix. The transfer function matrices for the subsystems were also generated using the approach presented in [2]. Forming (59) for this analysis problem resulted in an LMI that is sparse with $m = 1498$ and $W \in \mathbb{S}^{1498}$. The chordal embedding for the sparsity pattern graph of this LMI was generated by introducing 2.4% fill-in, also using a greedy search algorithm, with 579 cliques. The largest of these cliques had a size of 261. The corresponding clique tree for this problem was of height 35. This means that the corresponding problem in (49) will comprise of 579 subproblems and can be solved distributedly over this clique tree. We tested the performance of the proposed algorithm over 10 instances of this problem. The parameters of the primal-dual method were chosen to be the same as above. In the worst case the algorithm converged after 14 iterations. The convergence behavior of this instance is illustrated in Figure 6. As a result, in the worst case, the algorithm converged after $6 \times 2 \times 35 \times 14 = 5880$ steps. During the run of the algorithm, each agent needed to compute a factorization only 14 times and were required to communicate with its neighbors 168 times.

IX. Conclusions

In this paper we put forth a distributed algorithm for solving coupled SDPs with a tree structure. The proposed algorithm, unlike the existing ones, does not use first-order splitting
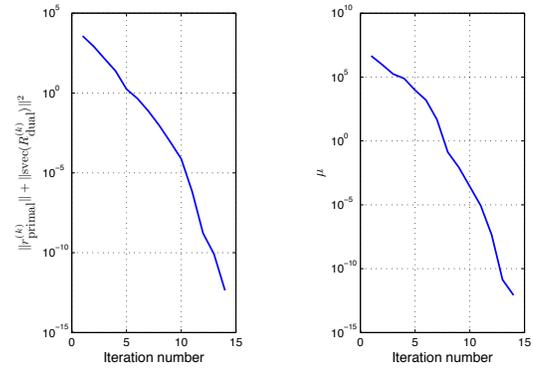

Fig. 6. Convergence behavior of the algorithm for analysis of an interconnected system over a scale-free network. The figure on the left shows the sum of primal and dual residuals and the figure on the right depicts the surrogate duality gap.

methods but instead uses primal-dual interior-point methods. Particularly, this algorithm utilizes the inherent tree structure in the problem as its computational graph, and distributes the computations at each iteration of the primal-dual method among the computational agents. In order to compute the search directions at every iteration, we employ a message-passing algorithm. This enables us to compute the exact search directions in a finite number of iterations. Furthermore, we showed that this number can be computed a priori and only depends on the height of the tree. We applied the proposed algorithm for solving robustness analysis of large-scale interconnected uncertain systems, and illustrated the performance of the algorithm using numerical experiments.

As was discussed in the introduction, designing distributed algorithms are commonly conducted in two phases. Namely, a decomposition or reformulation phase and a splitting phase. In this paper, we mainly focused on the second phase of this procedure, that is design of efficient methods to distribute the computations of solving a *given* coupled SDP. However, it is possible to further improve the computational and/or implementation properties of the devised algorithm, by using the available flexibilities in decomposition or reformulation phase. We will explore such possibilities as future line of research. This will mainly concern devising heuristics for clique or cluster merging to reduce the overall computational cost of the algorithm and/or to better represent the intuitive properties of the problem, such as physical structure in the problem.

References

[1] M. S. Andersen, A. Hansson, S. Khoshfetrat Pakazad, and A. Rantzer. Distributed robust stability analysis of interconnected uncertain systems. In *Proceedings of the 51st IEEE Conference on Decision and Control*, 2012.

[2] M. S. Andersen, S. Khoshfetrat Pakazad, A. Hansson, and A. Rantzer. Robust stability analysis of sparsely interconnected uncertain systems. *IEEE Transactions on Automatic Control*, 19(1):2594–2599, 2014.

[3] M. Annergren, S. Khoshfetrat Pakazad, A. Hansson, and B. Wahlberg. A distributed primal-dual interior-point method for loosely coupled problems using admm. *Submitted to Optimization Methods and Software*, 2015.

[4] X. Bai, H. Wei, K. Fujisawa, and Y. Wang. Semidefinite programming for optimal power flow problems. *International Journal of Electrical Power and Energy Systems*, 30(6-7):383–392, 2008.

[5] U. Bertel and F. Brioschi. On non-serial dynamic programming. *Journal of Combinatorial Theory, Series A*, 14(2):137–148, 1973.

[6] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.

[7] P. Biswas, T. C. Lian, T. C. Wang, and Y. Ye. Semidefinite programming based algorithms for sensor network localization. *ACM Transactions on Sensor Networks*, 2(2):188–220, May 2006.

[8] P. Biswas, K. C. Toh, and Y. Ye. A distributed SDP approach for large-scale noisy anchor-free graph realization with applications to molecular conformation. *SIAM Journal on Scientific Computing*, 30(3):1251–1277, March 2008.

[9] J. R. S. Blair and B. W. Peyton. An introduction to chordal graphs and clique trees. In J. A. George, J. R. Gilbert, and J. W-H. Liu, editors, *Graph Theory and Sparse Matrix Computations*, volume 56, pages 1–27. Springer-Verlag, 1994.

[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(1):1–122, 2011.

[11] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[12] Z. Cai and K. Toh. Solving second order cone programming via a reduced augmented system approach. *SIAM Journal on Optimization*, 17(3):711–737, 2006.

[13] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction To Algorithms*. MIT Press, 2001.

[14] E. Dall'Anese, H. Zhu, and G.B. Giannakis. Distributed optimal power flow for smart microgrids. *IEEE Transactions on Smart Grid*, 4(3):1464–1475, September 2013.

[15] E. de Klerk, T. Terlaky, and K. Roos. Self-dual embeddings. In H. Wolkowicz, R. Saigal, and L. Vandenberghe, editors, *Handbook of semidefinite programming: Theory, algorithms, and applications*, volume 27, pages 111–138. Springer Science & Business Media, 2000.

[16] M. Fukuda, M. Kojima, , K. Murota, and K. Nakata. Exploiting sparsity in semidefinite programming via matrix completion I: General framework. *SIAM Journal on Optimization*, 11:647–674, 2000.

[17] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Elsevier, 2nd edition, 2004.

[18] R. Grone, C. R. Johnson, E. M. Sá, and H. Wolkowicz. Positive definite completions of partial hermitian matrices. *Linear Algebra and its Applications*, 58:109–124, 1984.

[19] A. Hansson and L. Vandenberghe. Efficient solution of linear matrix inequalities for integral quadratic constraints. In *Proceedings of the 39th IEEE Conference on Decision and Control*, volume 5, pages 5033–5034, 2000.

[20] U. Jönsson. Lecture notes on integral quadratic constraints, May 2001.

[21] S. Khoshfetrat Pakazad, A. Hansson, and M. S. Andersen. Distributed interior-point method for loosely coupled problems. In *Proceedings of the 19th IFAC World Congress*, Cape Town, South Africa, August 2014.

[22] S. Khoshfetrat Pakazad, A. Hansson, and M. S. Andersen. Distributed primal-dual interior-point methods for solving loosely coupled problems using message passing. *ArXiv e-prints*, February 2015.

[23] S. Khoshfetrat Pakazad, A. Hansson, M. S. Andersen, and A. Rantzer. Distributed robustness analysis of interconnected uncertain systems using chordal decomposition. In *Proceedings of the 19th IFAC World Congress*, volume 19, pages 2594–2599, 2014.

[24] S. Kim, M. Kojima, M. Mevissen, and M. Yamashita. Exploiting sparsity in linear and nonlinear matrix inequalities via positive semidefinite matrix completion. *Mathematical Programming*, 129(1):33–68, 2011.

[25] S. Kim, M. Kojima, and H. Waki. Exploiting sparsity in SDP relaxation for sensor network localization. *SIAM Journal on Optimization*, 20(1):192–215, 2009.

[26] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT press, 2009.

[27] Z. Lu, A. Nemirovski, and R. C. Monteiro. Large-scale semidefinite programming via a saddle point mirror-prox algorithm. *Mathematical Programming*, 109(2):211–237, January 2007.

[28] Z.-Q. Luo, J. F. Sturm, and S. Zhang. Duality and self-duality for conic convex programming. Technical Report technical report 9719/A, Erasmus University Rotterdam, 1996.

[29] R. Madani, S. Sojoudi, and J. Lavaei. Convex relaxation for optimal power flow problem: Mesh networks. *IEEE Transactions on Power Systems*, 30(1):199–211, January 2015.

[30] J. Mattingley and S. Boyd. CVXGEN: A code generator for embedded convex optimization. *Optimization and Engineering*, 13(1):1–27, 2012.

[31] A. Megretski and A. Rantzer. System analysis via integral quadratic constraints. *IEEE Transactions on Automatic Control*, 42(6):819–830, June 1997.

[32] Y. Nesterov and M. J. Todd. Primal-dual interior-point methods for self-scaled cones. *SIAM Journal on Optimization*, 8:324–364, 1995.

[33] Y. Nesterov and M. J. Todd. Self-scaled barriers and interior-point methods for convex programming. *Mathematics of Operations Research*, 22(1):1–42, February 1997.

[34] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239, 2014.

[35] A. Rantzer. On the Kalman-Yakubovich-Popov lemma. *Systems and Control Letters*, 28(1):7–10, 1996.

[36] A. Simonetto and G. Leus. Distributed maximum likelihood sensor network localization. *IEEE Transactions on Signal Processing*, 62(6):1424–1437, March 2014.

[37] Y. Sun, M. S. Andersen, and L. Vandenberghe. Decomposition in conic optimization with partially separable structure. *SIAM Journal on Optimization*, 24(2):873–897, 2014.

[38] M. J. Todd, K. C. Toh, and R. H. Tütüncü. On the nesterov–todd direction in semidefinite programming. *SIAM J. on Optimization*, 8(3):769–796, March 1998.

[39] R. H. Tütüncü, K. C. Toh, and M. J. Todd. Solving semidefinite-quadratic-linear programs using SDPT3. *Mathematical Programming*, 95:189–217, 2003.

[40] L. Vandenberghe, V. R. Balakrishnan, R. Wallin, A. Hansson, and T. Roh. Interior-point algorithms for semidefinite programming problems derived from the KYP lemma. In D. Henrion and A. Garulli, editors, *Positive polynomials in control*, volume 312, pages 195–238. Springer, February 2005.

[41] R. Wallin, A. Hansson, and J. H. Johansson. A structure exploiting preprocessor for semidefinite programs derived from the Kalman-Yakubovich-Popov lemma. *IEEE Transactions on Automatic Control*, 54(4):697–704, April 2009.

[42] S. Wright. Stability of augmented system factorizations in interior-point methods. *SIAM Journal on Matrix Analysis and Applications*, 18(1):191–222, 1997.