This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSIPN.2019.2901649, IEEE Transactions on Signal and Information Processing over Networks

On the Distributed Method of Multipliers for Separable Convex Optimization Problems

Thomas Sherson, Richard Heusdens, and W. Bastiaan Kleijn

***Abstract*—In this paper we present a novel method for convex optimization in distributed networks called the distributed method of multipliers (DMM). The proposed method is based on a combination of a particular dual lifting and classic monotone operator splitting approaches to produce an algorithm with guaranteed asymptotic convergence in undirected networks. The proposed method allows any separable convex problem with linear constraints to be solved in undirected networks. In contrast to typical distributed approaches, the structure of the network does not restrict the types of problems which can be solved. Furthermore, the solver can be applied to general separable problems, those with separable convex objectives and constraints, via the use of an additional primal lifting approach. Finally we demonstrate the use of DMM in solving a number of classic signal processing problems including beamforming, channel capacity maximization and portfolio optimization.**

***Index Terms*—Distributed signal processing, convex optimization, monotone operator, optimization over networks.**

I. Introduction

Large scale optimization has become a significant topic of interest in the fields of computer science and electrical engineering. Driven by applications from the likes of the "Internet of Things" paradigm [1], cloud computing [2] and large scale machine learning [3], there is a growing need for efficient methods to solve large scale problems.

A family of approaches which has seen significant interest to address this need are those based on distributed computation. Designed for use in networked systems, distributed methods are often characterized by only requiring local computations at nodes within the network and short range communication. This removes the need for data aggregation and centralized computations which can quickly become infeasible to implement for large scale networks.

In recent years a wide range of techniques have been proposed to perform distributed computation including the likes of distributed consensus/gossip [4]–[6], belief propagation/message passing approaches [7], [8], graph signal processing over networks [9]–[12] and more. In this work, we draw particular attention to the additional field of decentralized/distributed optimization as a method of achieving distributed computation. The motivation for this pursuit is the link between many signal processing applications and equivalent convex optimization problems [13]. By developing novel tools for distributed optimization we can facilitate a range of signal processing applications in a distributed manner.

While a number of distributed optimization algorithms exist within the literature, many were conceived for use in parallel computing rather than in-network applications [14]. The result is that while many algorithms allow for distribution over the structure of the target optimization problem they may require communication which does not respect the underlying network topology. As such, data aggregation approaches may be required which, as in the case of centralized computation, can be cumbersome or perhaps infeasible to implement. While the considered problem classes can be restricted to circumvent this point, this action in turn heavily limits their applicability to real world problems.

In this work we propose a novel method for distributed optimization which can be used to solve general *separable* convex optimization problems. The family of *separable* problems are characterized by having separable objectives functions and separable constraints. Unlike many existing approaches, for the proposed algorithm the underlying structure of the network need not affect the types of problems which can be addressed, allowing the solver to be readily applied to general undirected networked problems.

A. Related Work

The field of parallel and distributed optimization has an extensive history upon which this paper builds. Key figures within the literature include Rockafellar, whose fundamental work on network optimization [15] and the relation between convex optimization and monotone operator theory [16]–[18] remains central to many results to this day. Notably, Rockafellar showed how linearly constrained convex programs with separable objectives could be solved in parallel via Lagrangian duality. This fundamental notion was further developed by the likes of Bertsekas, Tsitsiklis and Eckstein [14], [19]–[23] where again separability was used as a mechanism to design a range of distributed algorithms.

More recently, the demand for large scale data processing, has seen a return to form of these approaches within the literature [24]–[28]. This period has also seen the development of new methods of operator splitting [29]–[31] which in turn have motivated the development of further distributed optimization algorithms [32]–[34] again leveraging the fundamental links Rockafellar forged back in the 1970's. Unfortunately, a key

Thomas Sherson is with the Department of Microelectronics, Circuits and Systems group, Delft University of Technology, The Netherlands. Email: t.sherson@tudelft.nl

Richard Heusdens is with the Department of Microelectronics, Circuits and Systems group, Delft University of Technology, The Netherlands. Email: r.heusdens@tudelft.nl

W. Bastiaan Kleijn is with the Department of Microelectronics, Circuits and Systems group, Delft University of Technology, The Netherlands. and with the School of Engineering and Computer Science, Victoria University of Wellington, New Zealand. Email: w.b.kleijn@tudelft.nl

limitations of many distributed algorithms is that they do not distribute in a way which respects the underlying network structure. This is highlighted in [35] where a distinction is drawn between the notions of distributability of an algorithm over the constraints and communication structure of a given networked optimization problem. The challenge is therefore to construct distributed algorithms that allow for simultaneous distribution over both the problem and network structure of a given application.

Within the literature, a number of approaches have been proposed to address the need for solving simultaneous distribution via varying means [35]–[42]. In [38], [39], the alternating direction method of multipliers (ADMM) was used as a means of ensuring dual consensus, in turn guaranteeing primal optimality. Similarly, the methods in [36], [42] exploit dual decomposition based approaches, in combination with a consensus step and a proximal minimization step respectively to achieve the same feat. The work of [35] continues this trend by utilizing a combination of Lagrangian duality and an internal consensus algorithm (a gossip variant in this instance) to perform approximate dual updates. In contrast, the works of [37], [40], [41] utilize primal-dual based techniques to tackle the presence of global constraints. Notably, the method in [37] aims to solve the more general problem of distributed saddle-point computation, a problem which includes distributed optimization as a special case. For distributed convex optimization, the proposed method reduces to a primal-dual sub-gradient algorithm incorporating a Laplacian averaging strategy.

B. Main Contributions

The main contribution of this work is the proposal of a convex optimization solver termed the distributed method of multipliers (DMM) that is simultaneously distributable in both the network and problem structure. The proposed method is deployable in any undirected network topology so long as the network forms a single connected component. The result is an algorithm that respects the connectivity of the physical network whilst being applicable to a wide range of optimization problems.

The DMM algorithm is derived from the perspective of monotone operator theory and as such incorporates classic operator splitting approaches. This leads to a straightforward derivation closely related with other traditional algorithms from within the literature including the alternating direction method of multipliers (ADMM) [24], forward backward splitting (FB) [43] and more. The convergence guarantees of DMM follow from its relation with Krasnosel'skiĭ-Mann iterations [44] and hold for all closed, convex and proper functions.

We demonstrate the use of the proposed method in practical signal processing problems including beamforming, channel capacity maximization and portfolio optimization. This numerically validates the performance claims of the algorithm while demonstrating how the approach can be deployed in practice.

C. Organization of Paper

The remainder of this paper is organized as follows. In Section II we introduce basic nomenclature to support the remainder of the article. In Section III we derive our proposed distributed method for separable problems with affine constraints from a basic prototype optimization problem. Section IV focuses on the computation of the iterates of our algorithm, demonstrating the efficiency and locality of the proposed method. Section IV-E highlights a primal lifting stage, allowing the proposed method to also be used for general separable problems and provides a means for further reducing computational complexity. Section V demonstrates the use of the proposed method for a range of distributed signal processing tasks including beamforming, channel capacity maximization and portfolio optimization before making our concluding remarks in Section VI.

II. Nomenclature

In this work we denote by $\mathbb{R}$ the set of real numbers, by $\mathbb{R}^N$ the set of real column vectors of length N and by $\mathbb{R}^{M\times N}$ the set of M by N real matrices. Let $\mathcal{X},\mathcal{Y}\subseteq\mathbb{R}^N$. A set valued operator $\mathbf{T}:\mathcal{X}\to\mathcal{Y}$ is defined by its graph, $\text{gra}\left(\mathbf{T}\right)=\{(\mathbf{x},\mathbf{y})\in\mathcal{X}\times\mathcal{Y}\mid\mathbf{y}\in\mathbf{T}\left(\mathbf{x}\right)\}$. Similarly, the notion of an inverse of an operator $\mathbf{T}^{-1}$ is defined via its graph so that $\text{gra}\left(\mathbf{T}^{-1}\right)=\{(\mathbf{y},\mathbf{x})\in\mathcal{Y}\times\mathcal{X}\mid\mathbf{y}\in\mathbf{T}\left(\mathbf{x}\right)\}$. $\mathbf{J}_{\mathbf{T},\rho}=\left(\mathbf{I}+\rho\mathbf{T}\right)^{-1}$ denotes the resolvent of an operator while $\mathbf{R}_{\mathbf{T},\rho}=2\mathbf{J}_{\mathbf{T},\rho}-\mathbf{I}$ denotes the reflected resolvent (Cayley operator). The fixed-point set of $\mathbf{T}$ is denoted by $\text{fix}\left(\mathbf{T}\right)=\{\mathbf{x}\in\mathcal{X}\mid\mathbf{T}\left(\mathbf{x}\right)=\mathbf{x}\}$.

III. Deriving a Distributed Solver For Separable Convex Problems With Affine Constraints

In this section we introduce the derivation of the DMM algorithm for separable convex problems with affine constraints. In particular we demonstrate how DMM can be constructed via monotone operator splitting while respecting the underlying structure of a given physical network. This derivation also directly leads to convergence guarantees by linking the method with Krasnosel'skiĭ-Mann iterative schemes. This algorithm is extended to general separable problems in Section IV-E.

A. Problem Statement and the Communication Graph

Consider an undirected network of compute nodes with which we want to perform distributed convex optimization. For now we make no assumptions on the structure of the network other than that it is simple, undirected and connected. In Section III-E we highlight the structural considerations of this network and how they can influence the implementation of our resulting distributed algorithm. The communication structure of such a network can be represented by an equivalent *communication graph* which we denote by $G(V,E)$ where V is the set of nodes, $|V|=N$ is the number of nodes, $|\bullet|$ denotes the cardinality of a set and E denotes the set of undirected edges. These edges represent the communication channels between nodes.

An example of such a graph is included in Fig. 1 for a simple eight node network. We denote by $\mathcal{N}(i)=\{j\in V\mid(i,j)\in E\}$ the neighborhood of node i, i.e. the set of nodes with which node i shares a physical connection. For

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSIPN.2019.2901649, IEEE Transactions on Signal and Information Processing over Networks

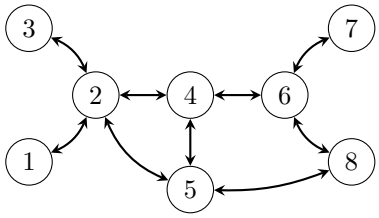


Fig. 1: The communication graph G of a simple eight node network. Numbered circles denote nodes and their identifiers while the double ended arrows denote the undirected edges.

instance, the neighborhood of node 4 in Fig. 1 is given by the set $\mathcal{N}(4) = \{2, 5, 6\}$.

In this work, we are interested in using networks to solve convex optimization problems in a distributed manner. For this section, we have restricted our attention to separable convex optimization problems with affine constraints. In such problems, each node is associated with a local objective function $f_i \in \Gamma_0\left(\mathbb{R}^{M_i}\right),\ \forall i \in V$, parameterized by $\mathbf{x}_i \in \mathbb{R}^{M_i}$, where Γ_0 denotes the set of closed, proper and convex functions. We define the scalar $M_V = \sum_{i\in V} M_i$, which denotes the total number of variables in the network. Specifically, we consider problems of the form

$$\min_{\mathbf{x}_i} \sum_{i\in V} f_i(\mathbf{x}_i) \quad \text{s.t } \mathbf{A}_k\mathbf{x} - \mathbf{b}_k \underset{k}{\blacktriangleright} \mathbf{0}\ \forall k \in \kappa \tag{1}$$

where for each k, $\underset{k}{\blacktriangleright}$ denotes either element-wise equality or inequality of the form $\geq$, $\kappa = \{1, 2, \cdots, K\}$, K denotes the total number of constraints and $\mathbf{x} = \left[\mathbf{x}_1^T, \cdots, \mathbf{x}_N^T\right]^T$. The matrices $\mathbf{A}_k \in \mathbb{R}^{\mathcal{M}_k \times M_V}$ and vectors $\mathbf{b}_k \in \mathbb{R}^{\mathcal{M}_k}$ impose linear constraints between the variables at each node where $\mathcal{M}_k$ denotes the dimensionality of the kth constraint set. Importantly, we assume that (1) is strictly feasible such that strong duality holds.

Due to the separability of linear constraints, we can rewrite (1) by defining the sets V_k which denote those nodes i whose variables $\mathbf{x}_i$ play an active role in the kth constraint. More formally, for each $k \in \kappa$, the set V_k is given by

$$V_k = \{i \in V \mid \mathbf{A}_{i,k} \neq \mathbf{0}\}.$$

Using this notation, (1) can be equivalently written as

$$\min_{\mathbf{x}_i} \sum_{i\in V} f_i(\mathbf{x}_i) \quad \text{s.t} \sum_{i\in V_k} \left(\mathbf{A}_{i,k}\mathbf{x}_i - \mathbf{b}_{i,k}\right) \underset{k}{\blacktriangleright} 0 \quad \forall k \in \kappa. \tag{2}$$

Here the matrices $\mathbf{A}_{i,k} \in \mathbb{R}^{\mathcal{M}_k \times M_i}$ are the ith set of columns of $\mathbf{A}_k$ such that $\mathbf{A}_k\mathbf{x} = \sum_{i\in V_k} \mathbf{A}_{i,k}\mathbf{x}_i$ while the vectors $\mathbf{b}_{i,k}$ are chosen such that $\sum_{i\in V_k} \mathbf{b}_{i,k} = \mathbf{b}_k$.

B. Implied Connectivity of the Constraint Graph

The constraints in (2) imply a secondary set of relationships between the local variables at each node which can be modeled via a *constraint graph* denoted by $G_C(V, E_C)$. Here, the edge set E_C captures the interdependence of node variables in the constraint functions. In particular, if two nodes $i,\ j$ are active in the same constraint k, then $(i, j) \in E_C$.

A fundamental challenge in distributed optimization follows from the differences between the edge sets of G_C and G as was highlighted in [35]. This challenge is best demonstrated with an example.

Consider again the network in Fig. 1 with which we want to solve the optimization problem

$$\begin{aligned} \min_{\mathbf{x}_i} \quad & \sum_{i=1}^{8} f_i(\mathbf{x}_i) \\ \text{s.t} \quad & \sum_{i=1,2,3} \left(\mathbf{A}_{i,1}^T\mathbf{x}_i - \mathbf{b}_{i,1}\right) = 0 \\ & \sum_{i=2,4,5,6} \left(\mathbf{A}_{i,2}^T\mathbf{x}_i - \mathbf{b}_{i,2}\right) = 0 \\ & \sum_{i=4,6,7,8} \left(\mathbf{A}_{i,3}^T\mathbf{x}_i - \mathbf{b}_{i,3}\right) = 0, \end{aligned} \tag{3}$$

where $\kappa = \{1, 2, 3\}$, $V_1 = \{1, 2, 3\}$, $V_2 = \{2, 4, 5, 6\}$ and $V_3 = \{4, 6, 7, 8\}$ in this instance. Note that for this example, we need not consider the dimensionality of the local variables, only the communication structure implied by the constraint set. Using the definition of E_C, we can form the constraint graph of (3) which is included in Fig. 2.

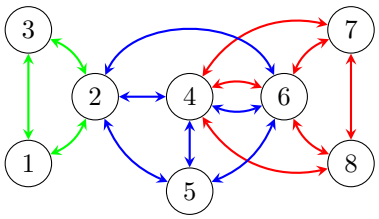


Fig. 2: The constraint graph G_C for the same eight node network as in Fig. 1. The green, blue and red colouration is used to denote the dependencies between nodes for first, second and third set of constraints.

Comparing Fig. 1 and 2, we can note that while the two graphs share the same node set V, the connectivity of the physical network (G) and that imposed by the constraint functions (G_C) may differ depending on the optimization problem we are trying to solve. In particular, note the discrepancy between the edges of E_C and E. The edge $(2, 6)$, for instance, is contained within E_C but not within E. Contrastingly, the edge $(5, 8)$ is contained within E but not within E_C as no constraints impose any relationship between the local variables at nodes 5 and 8 directly. This poses a challenge for many existing algorithms which aim to distribute over the constraint set [21], [24], [28] as while an algorithm may be distributable in G_C it may not be in G. Fig. 2 for instance suggests the need for communication between nodes 2 and 6 which cannot be achieved in the physical network without relaying messages via node 4. In the following we demonstrate a method to address this mismatch in a fully distributed manner via a dual lifting approach.

C. Exploiting Separability Via Lagrange Duality

Inspired by classic results from Rockafellar [15], we can exploit the separability of (2) to overcome the coupling of our

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSIPN.2019.2901649, IEEE Transactions on Signal and Information Processing over Networks

primal variables through the constraint functions and thus the discrepancies between G_C and G. Specifically, we can use Lagrangian duality[1] to rephrase (2) in an alternative form. For this purpose, at each node we define the set

$$\kappa_i = \{k \in \kappa | i \in V_k\},$$

to denote those k such that i is active in said constraints. Furthermore, the set of indices $k \in \kappa$ associated with inequality constraints in (2) is denoted by

$$\kappa_{\geq} = \left\{ k \in \kappa \mid \underset{k}{\blacktriangleright} \text{ is of the form } \geq \right\}.$$

The general form of the dual of (2) is therefore given by

$$\begin{aligned} \min_{\boldsymbol{\nu}_k} \quad & \sum_{i \in V} \left(f_i^* \left(\sum_{k \in \kappa_i} \mathbf{A}_{i,k}^T \boldsymbol{\nu}_k \right) - \sum_{k \in \kappa_i} \mathbf{b}_{i,k}^T \boldsymbol{\nu}_k \right) \\ \text{s.t.} \quad & \boldsymbol{\nu}_k \geq 0 \; \forall k \in \kappa_{\geq}, \end{aligned} \tag{4}$$

where $f_i^*(\mathbf{y}) = \sup_{\mathbf{x}} \left(\mathbf{y}^T \mathbf{x} - f(\mathbf{x}) \right)$ denotes the Fenchel conjugate of f_i and $\boldsymbol{\nu}_k \in \mathbb{R}^{\mathcal{M}_k}$ denotes the dual variable associated with the kth constraints. In the case of the graph considered in (2), a visualization of this point is provided in Fig. 3. Here, the dual problem is parameterized by three dual variables ($\boldsymbol{\nu}_1$, $\boldsymbol{\nu}_2$, $\boldsymbol{\nu}_3$) each associated with a constraint.

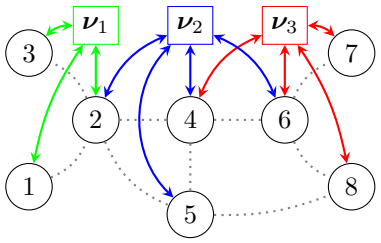


Fig. 3: Graph of the dual problem for the same eight node network as in Fig. 2. The green, blue and red colors denote the dependencies on the dual variables $\boldsymbol{\nu}_1$, $\boldsymbol{\nu}_2$ and $\boldsymbol{\nu}_3$.

At this point many classic algorithms begin to directly solve (4) by distributing over the set of constraints. However, as each dual variable $\boldsymbol{\nu}$ can parameterize the local objective functions of multiple nodes, their updating can be challenging from a distributer perspective. To demonstrate this challenge, consider a problem with a single constraint $k = 1$ in which every node plays an active role, i.e., $V_1 = V$. The resulting graphical model of the dual problem would exhibit a centralized topology in this instance. Such a topology implies the need for data aggregation to compute the dual variable which ultimately undermines the distributed intention of this work. For this reason, in the following we demonstrate a lifting approach which allows us to overcome dual variable coupling while naturally respecting the underlying topology of the network.

D. A Communication Graph Preserving Dual Lifting

Motivated by the dual lifting adopted in [45], we propose to address the coupling of the objectives functions by lifting the dimensionality of the dual problem in a specific fashion. In particular, our goal is to rephrase (4) into a set of node and edge based terms. The proposed lifting results in the *extended dual* of Eq. (1)

$$\begin{aligned} \min_{\boldsymbol{\lambda}_{i|j,k}} \quad & \sum_{i \in V} \left(f_i^* \left(\sum_{k \in \kappa_i} \sum_{j \in \mathcal{N}_k(i)} \frac{\mathbf{A}_{i,k}^T \boldsymbol{\lambda}_{(i,j),k}}{|\mathcal{N}_k(i)|} \right) \right. \\ & \left. - \sum_{k \in \kappa_i} \sum_{j \in \mathcal{N}_k(i)} \frac{\mathbf{b}_{i,k}^T}{|\mathcal{N}_k(i)|} \boldsymbol{\lambda}_{(i,j),k} \right) && \text{(5a)} \\ \text{s.t.} \quad & \boldsymbol{\lambda}_{(i,j),k} = \boldsymbol{\lambda}_{(j,i),k} \; \forall k \in \kappa, i \in V_k, j \in \mathcal{N}_k(i) && \text{(5b)} \\ & \boldsymbol{\lambda}_{(i,j),k} = \boldsymbol{\lambda}_{(i,l),k} \; \forall k \in \kappa, i \in V_k, j, l \in \mathcal{N}_k(i) && \text{(5c)} \\ & \boldsymbol{\lambda}_{(i,j),k} \geq 0 \; \forall k \in \kappa_{\geq}, i \in V_k, j \in \mathcal{N}_k(i). && \text{(5d)} \end{aligned}$$

Here, $\mathcal{N}_k(i)$ denotes the *constrained neighborhood* of each node $i \in V_k$ where

$$\mathcal{N}_k(i) = \{j \in V_k \mid j \in \mathcal{N}(i)\},$$

i.e. the subset of $\mathcal{N}(i)$ active in the kth set of constraints.

To perform this dual lifting, $\forall k \in \kappa$ new copies of each dual variable $\boldsymbol{\nu}_k$ have been introduced for each directed edge in the network. That is, $\forall k \in \kappa, i \in V_k, j \in \mathcal{N}_k(i)$, the variables $\boldsymbol{\lambda}_{(i,j),k}$, $\boldsymbol{\lambda}_{(j,i),k} \in \mathbb{R}^{M_k}$ are introduced. Making the substitution $\boldsymbol{\nu}_k = \sum_{j \in \mathcal{N}_k(i)} \boldsymbol{\lambda}_{(i,j),k} / |\mathcal{N}_k(i)|$ for each node i and constraint k results in the lifted objective of (5).

Equivalence with (4) is insured via the additional constraints introduced between dual variables corresponding to the same k. These can be divided into two sets: edge based constraints $\boldsymbol{\lambda}_{(i,j),k} = \boldsymbol{\lambda}_{(j,i),k} \; \forall k \in \kappa, (i,j) \in E$ and node based constraints $\boldsymbol{\lambda}_{(i,j),k} = \boldsymbol{\lambda}_{(i,l),k} \; \forall k \in \kappa, i \in V_k, j, l \in \mathcal{N}_k(i)$.

Performing the lifting in this way partitions the extended dual into four distinct sections: a fully node separable objective function (5a), a set of edge based consensus constraints (5b), an additional set of node based consensus constraints (5c) and finally a set of element-wise non-negativity constraints (5d). Such a problem structure is attractive in the context of alternating optimization methods as it partitions the problem into node and edge based terms.

For the example problem in (2), a visualization of the resulting lifted problem, indicating the relationship between the local copies of the dual variables, is included in Fig. 4.

Remark 1. *Typically lifting approaches such as the one adopted here result in an increase in computational complexity and memory load. However, in the case of the proposed method we will see this in not the case. Specifically, we mean that the computational cost of any local optimization problems at each node do not scale with the number of variables introduced. This is highlighted in Remark 3 and by the primal* $\mathbf{x}$ *updates in Lemma IV.1 and Algorithm 1 in Sections IV-A and IV-C respectively. This is also demonstrated for the specific simulation examples in Section V. Furthermore, when considering the affect on memory load, the total number of extended dual variables introduced at any node* i *is only dependent on the size of its constrained neighborhoods* $\mathcal{N}_k(i)$*. In this manner, while the memory load of the network is*

[1]This is sometimes referred to as Fenchel-Rockafellar duality.

increased, the local increase at any one node is bounded based on the physical connectivity of the network.

E. Network Topology Requirements

At this stage it is important to highlight how the topology of G affects the feasibility of the lifting in Eq. (5). In particular, the equivalence of Eq. (4) and (5) relies on the constraints enforcing consensus between dual variables $\forall k$. To this end, we demonstrate how this is guaranteed for a restricted set of network topologies which can then be generalized to the case of connected networks.

To begin, for the proposed lifting, a sufficient condition for the equivalence of Eq. (4) and (5) is given in Lemma III.1.

Lemma III.1. *If $\forall k \in \kappa$, the nodes $i \in V_k$ form a connected subgraph of G then* (4) *and* (5) *are equivalent problems.*

Proof. If $\forall k \in \kappa$, the set of nodes $i \in V_k$ form a connected subgraph of G then the constraints (5b) and (5c) ensure that $\exists\ \boldsymbol{\nu}_k$ such that at consensus, $\forall i \in V_k, j \in \mathcal{N}_k(i),\ \boldsymbol{\lambda}_{(i,j),k} = \boldsymbol{\nu}_k$. Hence the problems are equivalent. □

While sufficient to guarantee equivalence, Lemma III.1 seems restrictive. For instance, if a network forms a single connected component, data aggregation could be used to enforce dual consensus without satisfying this condition. As a demonstration, consider the communication graph and constraint graph given in Fig. 5a and 5b respectively. Clearly, the network has the physical connectivity to enforce any set of constraints between nodes but, due to the lack of activity of node 2 in the constraints, the set of active nodes forms a physically disjoint subgraph.

To generalize the class of applicable networks we can introduce a modification to the dual lifting in (5) to ensure its equivalence to (4). The basic notion is that $\forall k \in \kappa$ we can introduce additional nodes to the set V_k such that the resulting subnetworks form connected subgraphs. This action can always be performed due to our initial assumption that G forms a single connected component.

In the case of the networked problem in Fig. 5, the initial constraint set $V_k = \{1, 3, 4\}$ can be augmented to include node two such that $V_k = \{1, 2, 3, 4\}$ which in turn ensures that the constraint subgraph forms a single connected component. This introduces local copies of $\boldsymbol{\nu}_k$ at node 2 denoted by $\boldsymbol{\lambda}_{(2,j),k}\ \forall j \in \mathcal{N}_k(2)$. These additional variables in no way influence the objective cost of node 2 and exist only to enforce consensus between the lifted dual variables $\forall i \in V_k$. The additional matrix $\mathbf{A}_{2,k} = \mathbf{0}$ and vector $\mathbf{b}_{2,k} = \mathbf{0}$ are also introduced to complete the modification. For the remainder of the document, should a network require this modification to solve a particular problem we assume that this is performed.

F. Simplifying the Problem Notation

To assist with the remainder of this derivation, we introduce a compact notation to allow us to simplify Eq. (5). In particular, we show that (5) can be rewritten as

$$\begin{aligned} \min_{\boldsymbol{\lambda}} \quad & f^*(\mathbf{C}^T\boldsymbol{\lambda}) - \mathbf{d}^T\boldsymbol{\lambda} \\ \text{s.t.} \quad & (\mathbf{I} - \mathbf{P})\,\boldsymbol{\lambda} = \mathbf{0},\ \boldsymbol{\mathcal{L}}\boldsymbol{\lambda} = \mathbf{0},\ \mathbf{S}\boldsymbol{\lambda} \geq \mathbf{0}, \end{aligned} \tag{6}$$

where the three constraints correspond to (5b), (5c) and (5d) respectively. The additional matrices associated with this equivalent representation are defined below.

1) Forming a Single Dual Vector: We firstly define a vector notation for the extended dual variables. For each $k \in \kappa$, denote by $\boldsymbol{\lambda}_k$ the stacked vector of all $\boldsymbol{\lambda}_{(i,j),k}$. The ordering of this stacking is based on the directed edge index and is given by $(1,2) < (1,3) < \cdots < (1,N) < (2,1) < (2,3) < \cdots < (N, N-1)$. In this way, $\boldsymbol{\lambda}_k$ is given by

$$\boldsymbol{\lambda}_k = \left[\boldsymbol{\lambda}_{(1,2),k}, \cdots, \boldsymbol{\lambda}_{(1,N),k}, \boldsymbol{\lambda}_{(2,1),k}, \cdots, \boldsymbol{\lambda}_{(N,N-1),k}\right]^T.$$

By stacking the set of all $\boldsymbol{\lambda}_k$, we can then form a single dual vector $\boldsymbol{\lambda}$ as used in (6) such that

$$\boldsymbol{\lambda} = \left[\boldsymbol{\lambda}_1^T, \cdots, \boldsymbol{\lambda}_K^T\right]^T.$$

2) Compact Objective Notation: Given the compact notation of the dual variables, we now move to simplifying the objective function. Firstly, we define the global function

$$f:\ \mathbb{R}^{M_V} \mapsto \mathbb{R}, \mathbf{x} \mapsto \sum_{i \in V} f_i(\mathbf{x}_i)$$

where $\mathbb{R}^{M_V} = \mathbb{R}^{M_1} \times \ldots \times \mathbb{R}^{M_N}$. Similarly, the Fenchel conjugate of this function is denoted by f^*.

The next step is to define a matrix and vector to rewrite our objective using $\boldsymbol{\lambda}$ and f^*. This stage is broken into multiple steps. Firstly, $\forall k \in \kappa, i \in V_k$ we define $\mathbf{C}_{i,k}$ and $\mathbf{d}_{i,k}$ as

$$\begin{aligned} \mathbf{C}_{i,k} &= \mathbf{1}_{|\mathcal{N}_k(i)|} \otimes \frac{\mathbf{A}_{i,k}^T}{|\mathcal{N}_k(i)|}\ \forall i \in V_k, \\ \mathbf{d}_{i,k} &= \mathbf{1}_{|\mathcal{N}_k(i)|} \otimes \frac{\mathbf{b}_{i,k}}{|\mathcal{N}_k(i)|}\ \forall i \in V_k, \end{aligned}$$

where $\otimes$ denotes the Kronecker product of two matrices and the notation $\mathbf{1}_{|\mathcal{N}_k(i)|}$ is used to indicate a $|\mathcal{N}_k(i)|$-length column vector of ones. For each $k \in \kappa$ we therefore define the matrices $\mathbf{C}_k$ and $\mathbf{d}_k$ as

$$\mathbf{C}_k = \begin{bmatrix} \mathbf{C}_{1,k} & \ldots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \ldots & \mathbf{C}_{N,k} \end{bmatrix},\ \mathbf{d}_k = \left[\mathbf{d}_{1,k}^T, \cdots, \mathbf{d}_{N,k}^T\right]^T. \tag{7}$$

We can then form the final matrix $\mathbf{C}$ and vector $\mathbf{d}$ by stacking over the set of constraints so that

$$\begin{aligned} \mathbf{C} &= \left[\mathbf{C}_1^T, \cdots, \mathbf{C}_K^T\right]^T, \\ \mathbf{d} &= \left[\mathbf{d}_1^T, \cdots, \mathbf{d}_K^T\right]^T. \end{aligned}$$

Combining these two definitions the objective function of (5) can be compactly written as

$$f^*(\mathbf{C}^T\boldsymbol{\lambda}) - \mathbf{d}^T\boldsymbol{\lambda}.$$

3) Compact Constraint Notation: As with the objective, we can define a set of additional matrices to rewrite the constraints using our dual vector notation. To capture the edge based constraints (5b), we define for each $k \in \kappa$ the symmetric permutation matrix $\mathbf{P}_k$ which interchanges the edge variables

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSIPN.2019.2901649, IEEE Transactions on Signal and Information Processing over Networks

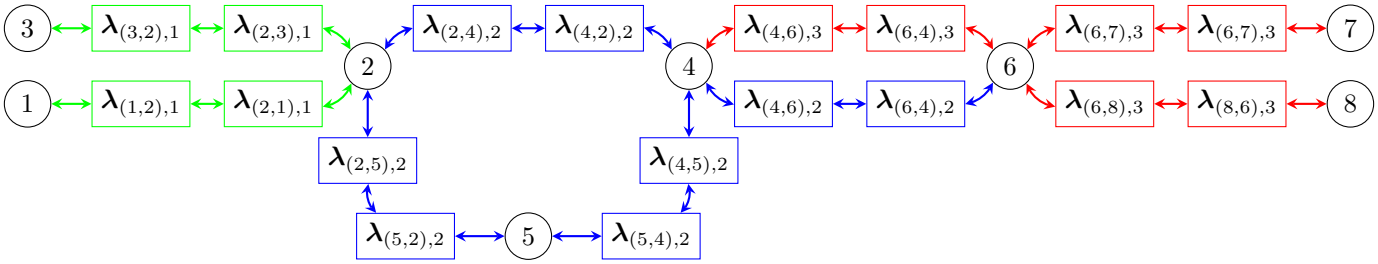
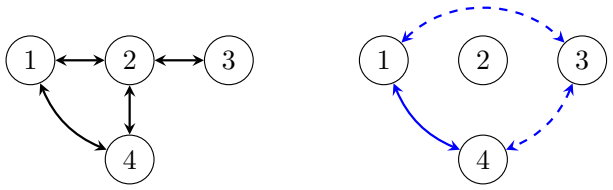


Fig. 4: A graph modeling the connectivity of extended dual problem for the same eight node network as in Fig. 2. The green, blue and red colors denote the dependencies on the three sets of constraints in (3).



(a) Communication Graph (b) Constraint Graph

Fig. 5: An example of a simple four node network. The target problem of interest imposes no constraints on the primal variables at node 2 but does affect the other three nodes.

$\boldsymbol{\lambda}_{(i,j),k}, \boldsymbol{\lambda}_{(j,i),k}$ $\forall i,j \in V_k$. By concatenating over all $k \in \kappa$, we can define the permutation matrix

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_1 & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathbf{P}_K \end{bmatrix}.$$

Combining with the definition of $\boldsymbol{\lambda}$ allows us to rewrite (5b) as $(\mathbf{I} - \mathbf{P})\,\boldsymbol{\lambda} = \mathbf{0}$ to enforce edge based consensus.

For the second set of constraints (5c), we define the matrices

$$\boldsymbol{\mathcal{L}}_k = \begin{bmatrix} \boldsymbol{\mathcal{L}}_{1,k} & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \boldsymbol{\mathcal{L}}_{N,k} \end{bmatrix} \otimes \mathbf{I}_{\mathcal{M}_k} \quad \forall k \in \kappa,$$

where the additional submatrices are given by

$$\begin{aligned} \boldsymbol{\mathcal{L}}_{i,k} &= \mathbf{D}_{i,k} - \mathbf{E}_{i,k}\ \forall i \in V_k \\ \mathbf{D}_{i,k} &= \left(|\mathcal{N}_k(i)| - 1\right)\mathbf{I}_{|\mathcal{N}_k(i)|}\ \forall i \in V_k \\ \mathbf{E}_{i,k} &= \mathbf{1}_{|\mathcal{N}_k(i)|}\mathbf{1}^T_{|\mathcal{N}_k(i)|} - \mathbf{I}_{|\mathcal{N}_k(i)|}\ \forall i \in V_k. \end{aligned}$$

Similar to $\mathbf{1}_M$, here the matrix $\mathbf{I}_M$ is used to denote an $M \times M$ identity matrix. It can be shown that $\forall k \in \kappa, i \in V_k$

$$\begin{aligned} \boldsymbol{\mathcal{L}}_{i,k}\mathbf{1}_{|\mathcal{N}_k(i)|} &= \left(\mathbf{D}_{i,k} - \mathbf{E}_{i,k}\right)\mathbf{1}_{|\mathcal{N}_k(i)|} \\ &= |\mathcal{N}_k(i)|\mathbf{1}_{|\mathcal{N}_k(i)|} - |\mathcal{N}_k(i)|\mathbf{1}_{|\mathcal{N}_k(i)|} \\ &= \mathbf{0}_{|\mathcal{N}_k(i)|}, \end{aligned}$$

where in the third line we have used the mixed-product property of Kronecker products. In this way, the kernel space of $\boldsymbol{\mathcal{L}}_{i,k}$ corresponds to the consensus vector and can therefore be used to impose the consensus constraints in (5c). Concatenating over the constraints, we can form the matrix

$$\boldsymbol{\mathcal{L}} = \begin{bmatrix} \boldsymbol{\mathcal{L}}_1 & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \boldsymbol{\mathcal{L}}_K \end{bmatrix},$$

such that (5c) can be rewritten as $\boldsymbol{\mathcal{L}}\boldsymbol{\lambda} = \mathbf{0}$. Furthermore, from the structure in (7), $\boldsymbol{\mathcal{L}}_{i,k}\mathbf{C}_{i,k} = \mathbf{0}$, $\boldsymbol{\mathcal{L}}_{i,k}\mathbf{d}_{i,k} = \mathbf{0}$, $\forall k \in \kappa, i \in V_k$ such that $\boldsymbol{\mathcal{L}}\mathbf{C} = \mathbf{0}$, $\boldsymbol{\mathcal{L}}\mathbf{d} = \mathbf{0}$.

For the final set of constraints (5d), $\forall k \in \kappa$ we define the selection matrices $\mathbf{S}_k \in \mathbb{R}^{\mathcal{M}_k \times \mathcal{M}_k}$ given by

$$\mathbf{S}_k = s_k \mathbf{I}_{\mathcal{M}_k \sum\limits_{i \in V_k} |\mathcal{N}_k(i)|},\ s_k = \begin{cases} 1 \text{ if } k \in \kappa_{\geq} \\ 0 \text{ otherwise,} \end{cases}$$

which preserves those dual variables associated with the inequality constraints. Concatenating over the constraints, we can form the final selection matrix given by

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}_1 & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathbf{S}_K \end{bmatrix},$$

such that (5d) can be rewritten as $\mathbf{S}\boldsymbol{\lambda} \geq \mathbf{0}$.

Using this compact notation, we are now ready to define our proposed distributed optimization algorithm.

G. From the Extended Dual Problem to a Monotonic Inclusion

Given the lifted dual problem (6), we now move to defining a distributed algorithm to compute an optimizer of (1). In particular, we want to construct an iterative algorithm

$$\mathbf{y}^{(t+1)} = \mathbf{U}_E \circ \mathbf{U}_V \left(\mathbf{y}^{(t)}\right), \tag{8}$$

which converges to a minimizer of (1) where t indicates the iteration number, $\mathbf{y} \in \mathbb{R}^p$ are the variables of interest and the operators $\mathbf{U}_E : \mathbb{R}^p \mapsto \mathbb{R}^p$ are $\mathbf{U}_V : \mathbb{R}^p \mapsto \mathbb{R}^p$ are parallelizable over the nodes and edges respectively. The additional notation $\circ$ is used to denote operator composition so that $\forall\,(\mathbf{x},\mathbf{z}) \in \mathrm{gra}\,(\mathbf{S}_1 \circ \mathbf{S}_2), \exists \mathbf{y} \mid (\mathbf{x},\mathbf{y}) \in \mathrm{gra}\,(\mathbf{S}_1), (\mathbf{y},\mathbf{z}) \in \mathrm{gra}\,(\mathbf{S}_2)$. We would like such operators to be at least nonexpansive so that classic iterative methods can be employed. The nonexpansiveness of an operator is defined as follows.

Definition III.1. *Nonexpansive Operators: An operator* $\mathbf{T} : \mathcal{X} \to \mathcal{Y}$ *is nonexpansive if*

$$\|\mathbf{u} - \mathbf{v}\| \leq \|\mathbf{x} - \mathbf{y}\| \quad (\mathbf{x},\mathbf{u}), (\mathbf{y},\mathbf{v}) \in \mathrm{gra}\,(\mathbf{T}),$$

where $\langle \mathbf{x},\mathbf{y} \rangle$ *denotes the inner product between* $\mathbf{x} \in \mathcal{X}$ *and* $\mathbf{y} \in \mathcal{Y}$ *and* $\|\mathbf{x}\|$ *denotes the associated induced norm.*

We can construct an iterative solver for (1) via operator splitting. In this case, we make use of the relationship between the subdifferentials of convex functions and monotone

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSIPN.2019.2901649, IEEE Transactions on Signal and Information Processing over Networks

operators. In particular, an operator is monotone if it satisfies the following definition:

Definition III.2. *Monotone Operators: An operator* $\mathbf{T}:\mathcal{X}\to\mathcal{Y}$ *is monotone iff*

$$\langle \mathbf{u}-\mathbf{v},\mathbf{x}-\mathbf{y}\rangle \geq 0 \quad \forall\,(\mathbf{x},\mathbf{u}),(\mathbf{y},\mathbf{v})\in \mathrm{gra}\,(\mathbf{T}),$$

Furthermore, $\mathbf{T}$ *is maximal monotone iff*

$$\nexists \text{ a monotone } \tilde{\mathbf{T}}:\mathcal{X}\to\mathcal{Y} \mid \mathrm{gra}(\mathbf{T})\subset \mathrm{gra}(\tilde{\mathbf{T}}).$$

To form our iterative approach, consider the equivalent unconstrained form of (6),

$$\min_{\boldsymbol{\lambda}} \quad f^*(\mathbf{C}^T\boldsymbol{\lambda})-\mathbf{d}^T\boldsymbol{\lambda}+\iota_{\mathcal{C}_1}(\boldsymbol{\lambda})+\iota_{\mathcal{C}_2}(\boldsymbol{\lambda})+\iota_{\mathcal{C}_3}(\boldsymbol{\lambda}), \quad (9)$$

where $\iota_{\mathcal{C}}$ denotes an indicator function of the set $\mathcal{C}$ such that

$$\iota_{\mathcal{C}}(\boldsymbol{\lambda})=\begin{cases}0 & \text{if } \boldsymbol{\lambda}\in\mathcal{C}\\ +\infty & \text{otherwise}\end{cases}.$$

Here the convex sets $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_3$ are given by

$$\mathcal{C}_1=\ker(\mathbf{I}-\mathbf{P}),\ \mathcal{C}_2=\ker(\boldsymbol{\mathcal{L}}),\ \mathcal{C}_3=\{\boldsymbol{\lambda}\mid \mathbf{S}\boldsymbol{\lambda}\geq\mathbf{0}\},$$

where $\ker(\mathcal{C})$ denotes the kernel of $\mathcal{C}$.

As the sets $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_3$ are closed subspaces such that $\iota_{\mathcal{C}_1}(\boldsymbol{\lambda})$, $\iota_{\mathcal{C}_2}(\boldsymbol{\lambda})$, $\iota_{\mathcal{C}_3}(\boldsymbol{\lambda})\in\Gamma_0$, as $f\in\Gamma_0$ and all functions contain a common feasible point, a minimizer of (9) can be found by finding a zero-point of its subdifferential.

$$\mathbf{0}\in\mathbf{C}\partial f^*\left(\mathbf{C}^T\boldsymbol{\lambda}\right)-\mathbf{d}+\partial\iota_{\mathcal{C}_1}(\boldsymbol{\lambda})+\partial\iota_{\mathcal{C}_2}(\boldsymbol{\lambda})+\partial\iota_{\mathcal{C}_3}(\boldsymbol{\lambda}). \quad (10)$$

Here ∂f^* is used to denote the subdifferential of f^*.

A zero-point of (10) can be found via a range of approaches including Forward Backward (FB) splitting, Douglas-Rachford (DR) Splitting, Chambolle-Pock (CP) and more (see [46] for an overview of such splitting methods). In the proposed distributed context, the choice of such a splitting method must be made to take advantage of the node and edge based structure we have introduced into the subdifferentials of (10).

In this work, we adopt a classic two operator splitting scheme to rephrase (10) as a more familiar fixed point inclusion. To do so, we define the two operators

$$\mathbf{T}_1(\boldsymbol{\lambda})=\mathbf{C}\partial f^*\left(\mathbf{C}^T\boldsymbol{\lambda}\right)-\mathbf{d}+\partial\iota_{\mathcal{C}_2}(\boldsymbol{\lambda}), \quad (11a)$$

$$\mathbf{T}_2(\boldsymbol{\lambda})=\partial\iota_{\mathcal{C}_1}(\boldsymbol{\lambda})+\partial\iota_{\mathcal{C}_3}(\boldsymbol{\lambda}), \quad (11b)$$

where, by design, $\mathbf{T}_1$ is node separable and $\mathbf{T}_2$ is edge separable. These operators are both maximal monotone by combining the results of [18] and [47].

H. Selecting an Operator Splitting Approach

To find a minimizer of (9), we can use averaged PR splitting, which includes DR splitting as a special case, to recast (10) as a fixed point problem of a nonexpansive operator. Our motivation for choosing this approach is that, unlike methods such as FB splitting, we need not impose additional functional restrictions beyond that $f\in\Gamma_0$. Additionally, averaged PR splitting allows us to take advantage of the node and edge separability of $\mathbf{T}_1$ and $\mathbf{T}_2$ respectively. Other methods, like CP, cannot take advantage of this point. In particular, CP, which aims to solve monotonic inclusions of the form

$$0\in\mathbf{C}\partial f\left(\mathbf{C}^T\boldsymbol{\lambda}\right)+\partial g(\boldsymbol{\lambda}),$$

would require that either $\mathbf{T}_1$ or $\mathbf{T}_2$ could be expressed as a composition of a subdifferential and a linear operator. While we could define an alternative operator $\hat{\mathbf{T}}_1=\mathbf{C}\partial f^*\left(\mathbf{C}^T\boldsymbol{\lambda}\right)-\mathbf{d}$ such that this method could be used, the second operator would be given by $\hat{\mathbf{T}}_2(\boldsymbol{\lambda})=\partial\iota_{\mathcal{C}_1}(\boldsymbol{\lambda})+\partial\iota_{\mathcal{C}_3}(\boldsymbol{\lambda})+\partial\iota_{\mathcal{C}_2}(\boldsymbol{\lambda})$ which is neither edge nor node separable, eliminating our ability to form a distributed solver. In this way, PR splitting was a natural choice for our particular application.

I. Forming the Distributed Method Of Multipliers

Given two maximal monotone operators $\mathcal{A}$ and $\mathcal{B}$ and a positive scalar $\rho>0$, PR splitting can be used to find a zero of $\mathcal{A}+\mathcal{B}$ by rephrasing it as a more familiar fixed point condition [46, Sec. 7.3] of the form

$$\mathbf{R}_{\mathcal{B},\rho}\circ\mathbf{R}_{\mathcal{A},\rho}(\mathbf{z})\in\mathbf{z},\ \boldsymbol{\lambda}=\mathbf{J}_{\mathcal{A},\rho}(\mathbf{z}), \quad (12)$$

where $\mathbf{J}_{\mathcal{A},\rho}$ and $\mathbf{R}_{\mathcal{A},\rho}$ denote the resolvent and reflected resolvent of $\mathcal{A}$ respectively. The newly introduced $\mathbf{z}$ variables will be referred to as *auxiliary* variables from here on out. We can therefore equivalently solve (10) by solving

$$\mathbf{R}_{\mathbf{T}_2,\rho}\circ\mathbf{R}_{\mathbf{T}_1,\rho}(\mathbf{z})\in\mathbf{z},\ \boldsymbol{\lambda}=\mathbf{J}_{\mathbf{T}_1,\rho}(\mathbf{z}). \quad (13)$$

As we will show in the coming section, the node and edge separable structure of $\mathbf{T}_1$ and $\mathbf{T}_2$, is inherited by the operators $\mathbf{J}_{\mathbf{T}_1,\rho}$ and $\mathbf{J}_{\mathbf{T}_2,\rho}$ respectively so that $\mathbf{R}_{\mathbf{T}_1,\rho}$ and $\mathbf{R}_{\mathbf{T}_2,\rho}$ form the operators $\mathbf{U}_V$ and $\mathbf{U}_E$ outlined in Eq. (8).

To form our distributed algorithm, we define the nonexpansive distributed method of multipliers (DMM) operator as $\mathbf{T}_{D,\rho}=\mathbf{R}_{\mathbf{T}_2,\rho}\circ\mathbf{R}_{\mathbf{T}_1,\rho}$. The nonexpansiveness here stems from the maximal monotonicity of $\mathbf{T}_1$ and $\mathbf{T}_2$ and thus the nonexpansiveness of $\mathbf{R}_{\mathbf{T}_1,\rho}$ and $\mathbf{R}_{\mathbf{T}_2,\rho}$.

Remark 2. *In the specific case that all the constraints are edge based (only two nodes are active in each constraint and they correspond to a physical edge of G) the DMM operator corresponds to the PDMM operator given in [45].*

Given the nonexpansiveness of $\mathbf{T}_{D,\rho}$, we can employ a Krasnosel'skiĭ-Mann iterative scheme to form our proposed averaged PR splitting method [44, Chapter 3]. Such a scheme is given by

$$\mathbf{z}^{(t+1)}=(1-\alpha^{(t)})\mathbf{z}^{(t)}+\alpha^{(t)}\mathbf{T}_{D,\rho}\left(\mathbf{z}^{(t)}\right). \quad (14)$$

where $\forall t\in\mathbb{N},\alpha^{(t)}\in(0,1)$ and the sequence of $\alpha^{(t)}$ is non-convergent i.e. $\sum_{t=0}^{+\infty}\alpha^{(t)}=+\infty$. As previously mentioned, when $\alpha^{(t)}=\frac{1}{2}\ \forall t\in\mathbb{N}$, we recover the DR variant of the DMM algorithm.

IV. Computation of the DMM updates and its Distributed Implementation

Given the basic iterative scheme for DMM, presented in (14), in this section we demonstrate how the structure of (2) can be used to simplify the computation of the iterates. This is comprised of two simplifications, one for each of the reflected resolvents, and is summarized in the following two Lemmas.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSIPN.2019.2901649, IEEE Transactions on Signal and Information Processing over Networks

A. Computing the Reflected Resolvent $\mathbf{R}_{\mathbf{T}_1,\rho}$

We begin with the first reflected resolvent operator $\mathbf{R}_{\mathbf{T}_1,\rho}$ and its method of computation.

Lemma IV.1.

$$\begin{aligned}\mathbf{R}_{\mathbf{T}_1,\rho}\left(\mathbf{z}^{(t)}\right) &= \mathbf{w}^{(t+1)} = 2\boldsymbol{\lambda}^{(t+1)} - \mathbf{z}^{(t)} \\ &= 2\boldsymbol{\gamma}^{(t+1)} - 2\rho\left(\mathbf{C}\mathbf{x}^{(t+1)} - \mathbf{d}\right) - \mathbf{z}^{(t)},\end{aligned}$$

where,

$$\boldsymbol{\lambda}^{(t+1)} = \mathbf{J}_{\mathbf{T}_1,\rho}\left(\mathbf{z}^{(t)}\right) \text{ by (13)}, \boldsymbol{\gamma}^{(t+1)} = \underset{\ker(\mathcal{L})}{\amalg}\left(\mathbf{z}^{(t)}\right),$$

$$\mathbf{x}^{(t+1)} = \arg\min_{\mathbf{x}}\left(f(\mathbf{x}) + \frac{\rho}{2}\|\mathbf{C}\mathbf{x}-\mathbf{d}\|^2 - \left\langle \mathbf{C}^T\boldsymbol{\gamma}^{(t+1)}, \mathbf{x}\right\rangle\right).$$

The proof for this Lemma can be found in Appendix A.

Remark 3. *The primal* $\mathbf{x}$ *update, which involves a convex optimization problem, has local variables of the same dimensionality as the original problem in* (1)*. If an alternative lifting were utilized, the increase in the number of local variables at each node may unnecessarily result in a more complex local optimization problem per iteration.*

B. Computing the Reflected Resolvent $\mathbf{R}_{\mathbf{T}_2,\rho}$

In the case of $\mathbf{T}_2$, it can be shown that the reflected resolvent $\mathbf{R}_{\mathbf{T}_2,\rho}$ also exhibits a naturally distributable solution.

Lemma IV.2.

$$\begin{aligned}\mathbf{R}_{\rho,\mathbf{T}_2}\left(\mathbf{w}^{(t+1)}\right) &= \mathbf{v}^{(t+1)} = 2\mathbf{y}^{(t+1)} - \mathbf{w}^{(t+1)} \\ &= \mathbf{P}\mathbf{w}^{(t+1)} - \min\{\mathbf{S}(\mathbf{I}+\mathbf{P})\mathbf{w}^{(t+1)}, \mathbf{0}\},\end{aligned}$$

where

$$\mathbf{y}^{(t+1)} = \frac{1}{2}(\mathbf{I}+\mathbf{P})\mathbf{w}^{(t+1)} - \min\{\mathbf{S}\frac{1}{2}(\mathbf{I}+\mathbf{P})\mathbf{w}^{(t+1)}, \mathbf{0}\}.$$

and $\min\{\bullet,\bullet\}$ *is used to denote elementwise minimization.*

The proof for this Lemma can be found in Appendix B. $\mathbf{R}_{\rho,\mathbf{T}_2}$ reduces to a local exchanging of information between neighboring nodes (indicated by the use of the permutation operation) followed by a localized post processing at each node comprised of linear operations and element-wise comparisons.

C. Implementation in a Distributed Network

By combining Lemmas IV.1 and IV.2, the DMM algorithm can be expressed as

$$\boldsymbol{\gamma}^{(t+1)} = \underset{\ker(\mathcal{L})}{\amalg}\mathbf{z}^{(t)} \tag{15a}$$

$$\mathbf{x}^{(t+1)} = \arg\min_{\mathbf{x}}\left(f(\mathbf{x}) - \left\langle \mathbf{C}^T\boldsymbol{\gamma}^{(t+1)}, \mathbf{x}\right\rangle + \frac{\rho}{2}\|\mathbf{C}\mathbf{x}-\mathbf{d}\|^2\right) \tag{15b}$$

$$\mathbf{w}^{(t+1)} = 2\boldsymbol{\gamma}^{(t+1)} - 2\rho\left(\mathbf{C}\mathbf{x}^{(t+1)} - \mathbf{d}\right) - \mathbf{z}^{(t)} \tag{15c}$$

$$\mathbf{v}^{(t+1)} = \mathbf{P}\mathbf{w}^{(t+1)} - \min\{\mathbf{S}(\mathbf{I}+\mathbf{P})\mathbf{w}^{(t+1)}, \mathbf{0}\} \tag{15d}$$

$$\mathbf{z}^{(t+1)} = \left(1-\alpha^{(t)}\right)\mathbf{z}^{(t)} + \alpha^{(t)}\mathbf{v}^{(t+1)} \tag{15e}$$

The computation of each iteration reduces to a local averaging step at each node (15a), a single optimization over the primal variables (15b), the sharing of data between neighboring nodes (15d) and a set of additional matrix vector multiplications and element-wise comparisons. Furthermore, all of these operations are inherently distributable within the original network with (15a), (15b) and (15c) corresponding to $\mathbf{R}_{\mathbf{T}_1,\rho}$ and (15d) to $\mathbf{R}_{\mathbf{T}_2,\rho}$. The final equation (15e) represents the averaging operation performed in (14). The distributed nature of the method is highlighted in Algorithm 1.

Algorithm 1 Distributed Method of Multipliers

Initialize: $\mathbf{z}^{(0)} \in \mathbb{R}^{M_E}$
for t=0,..., **do**
  for all $i \in V$ **do** ▷ Primal and Dual Updates
    $\boldsymbol{\gamma}_{i,k}^{(t+1)} = \frac{1}{|\mathcal{N}_k(i)|}\sum_{j\in\mathcal{N}_k(i)}\mathbf{z}_{(i,j),k}^{(t)}$
    $\mathbf{x}_i^{(t+1)} = \underset{\mathbf{x}_i}{\operatorname{argmin}}\left(f_i(\mathbf{x}_i) + \sum_{k\in\kappa}\left(-\left\langle \mathbf{A}_{i,k}^T\boldsymbol{\gamma}_{i,k}^{(t+1)}, \mathbf{x}_i\right\rangle + \frac{\rho}{2|\mathcal{N}_k(i)|}\|\mathbf{A}_{i,k}\mathbf{x}_i - \mathbf{b}_{i,k}\|^2\right)\right)$
    $\mathbf{w}_{(i,j),k}^{(t+1)} = 2\boldsymbol{\gamma}_{i,k}^{(t+1)} - \mathbf{z}_{(i,j),k}^{(t)} - 2\rho\left(\frac{\mathbf{A}_{i,k}\mathbf{x}_i}{|\mathcal{N}_k(i)|} - \frac{\mathbf{b}_{i,k}}{|\mathcal{N}_k(i)|}\right)$
  for all $k \in \kappa, i \in V_k, j \in \mathcal{N}_k(i)$ **do** ▷ Tx. Variables
    $\mathbf{Node}_j \leftarrow \mathbf{Node}_i(\mathbf{w}_{(i,j),k}^{(t+1)})$
  for all $k \in \kappa, i \in V_k, j \in \mathcal{N}_k(i)$ **do** ▷ Aux. Updates
    $\mathbf{v}_{(i,j),k}^{(t+1)} = \mathbf{w}_{(j,i),k}^{(t+1)} - s_k \min\left\{\frac{\mathbf{w}_{(i,j),k}^{(t+1)} + \mathbf{w}_{(j,i),k}^{(t+1)}}{2}, \mathbf{0}\right\}$
    $\mathbf{z}_{(i,j),k}^{(t+1)} = (1-\alpha^{(t)})\mathbf{z}_{(i,j),k}^{(t)} + \alpha^{(t)}\mathbf{v}_{(i,j),k}^{(t+1)}$.

Here, we have made use of the fact that $\boldsymbol{\gamma}^{(t+1)}, \mathbf{w}^{(t+1)}$, $\mathbf{y}^{(t+1)}$, $\mathbf{z}^{(t)}$ all share the same structure as $\boldsymbol{\lambda}^{(t+1)}$, i.e. $\mathbf{z}^{(t)} = \left[\mathbf{z}_1^T, \cdots, \mathbf{z}_K^T\right]^T$ where $\mathbf{z}_k = [\mathbf{z}_{(1,2),k}^T, \cdots, \mathbf{z}_{(1,N),k}^T, \mathbf{z}_{(2,1),k}^T, \cdots, \mathbf{z}_{(N,N-1),k}^T]^T$ and so on for the other terms. Furthermore we have exploited the fact that $\boldsymbol{\gamma}^{(t+1)}, \boldsymbol{\lambda}^{(t+1)} \in \ker(\mathcal{L})$ such that $\boldsymbol{\gamma}_{(i,j),k}^{(t+1)} = \boldsymbol{\gamma}_{i,k}^{(t+1)}$, $\boldsymbol{\lambda}_{(i,j),k}^{(t+1)} = \boldsymbol{\lambda}_{i,k}^{(t+1)}$ $\forall k \in \kappa,\ i \in V_k,\ j \in \mathcal{N}_k(i)$.

D. Convergence Guarantees

Having formed a solver for linearly separable convex optimization problems, we now turn our attention to guaranteeing its convergence to an optimal solution. Thankfully, due to the use of a classic operator splitting approach in its derivation, this convergence follows directly from known results. Specifically, we form the following proposition.

Proposition IV.3. *For a given vector* $\mathbf{z}^{(0)}$*,* $\rho > 0$ *and the set* $\left(\alpha^{(t)}\right)_{t\in\mathbb{N}}$ *where* $\sum_{t=0}^{+\infty}\alpha^{(t)} = +\infty$*, the iterates generated by*

$$\mathbf{z}^{(t+1)} = (1-\alpha^{(t)})\mathbf{z}^{(t)} + \alpha^{(t)}\mathbf{T}_{D,\rho}\left(\mathbf{z}^{(t)}\right)$$

converge to a fixed point of the operator $\mathbf{T}_{D,\rho}$*.*

The proof of this proposition follows directly from the convergence of the Krasnosel'skiĭ-Mann method (see [48, Theorem 5.15] or [44, Theorem 3.2]) and thus guarantees that the proposed DMM algorithm converges to a fixed point. In

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSIPN.2019.2901649, IEEE Transactions on Signal and Information Processing over Networks

the case that $\alpha = \alpha^{(t)}\ \forall t$ the auxiliary fixed point residual $\|\mathbf{z}^{(t+1)} - \mathbf{z}^{(t)}\|^2$ converges to zero at a rate of $\mathcal{O}\left(\frac{1}{t}\right)$. For finite dimensional problems, it follows that the auxiliary variables converge to a specific fixed point at the same rate.

E. Distributed Optimization of General Separable Problems

While the prototype problem given in (1) may seem initially restrictive, in general any problem which exhibits both a separable objective and separable constraints can be solved by combining DMM with a primal lifting stage. Separability of the local functions f_i at each node can also be exploited to reduce the computational complexity of the primal updates.

Consider a general separable optimization problem given by

$$\begin{aligned} \min_{\mathbf{x}_i} \ & \sum_{i\in V} \left(f_i(\mathbf{x}_i) + g_i(\mathbf{A}_{i,g}\mathbf{x}_i - \mathbf{b}_{i,g})\right) \\ \text{s.t} \ & \sum_{i\in V} h_i(\mathbf{x}_i) \leq \mathbf{0},\ \mathbf{A}_k\mathbf{x} - \mathbf{b}_k \underset{k}{\blacktriangleright} \mathbf{0}\ \forall k \in \kappa \end{aligned} \tag{16}$$

Here, the functions $g_i, h_i \in \Gamma_0\left(\mathbb{R}^{M_i}\right)\ \forall i \in V$.

The aim is to convert (16) to the form of (1), a point which can be achieved by introducing the additional primal variables $\mathbf{w}_i$, $\mathbf{z}_i$ at each node and the slack variables $\mathbf{y}_i$ such that (16) can be equivalently expressed as

$$\begin{aligned} \min_{\mathbf{w}_i,\mathbf{x}_i,\mathbf{y}_i,\mathbf{z}_i} \ & \sum_{i\in V} \left(f_i(\mathbf{x}_i) + g_i(\mathbf{z}_i)\right) \\ \text{s.t} \ & \sum_{i\in V} \mathbf{y}_i = \mathbf{0},\ \mathbf{A}_k\mathbf{x} - \mathbf{b}_k \underset{k}{\blacktriangleright} \mathbf{0}\ \forall k \in \kappa \\ & \mathbf{A}_{i,g}\mathbf{x}_i - \mathbf{b}_{i,g} = \mathbf{z}_i\ \forall i \in V \\ & h_i(\mathbf{w}_i) \leq \mathbf{y}_i,\ \mathbf{x}_i = \mathbf{w}_i\ \forall i \in V. \end{aligned} \tag{17}$$

The additional constraints enforce the equivalence of (16) and (17). Note that the only remaining constraints involving multiple nodes are affine with the convex constraints only acting locally at each node. By using indicator functions, we can shift these non-affine inequality constraints to the objective so that (17) can be rephrased as

$$\begin{aligned} \min_{\mathbf{w}_i,\mathbf{x}_i,\mathbf{y}_i,\mathbf{z}_i} \ & \sum_{i\in V} \left(f_i(\mathbf{x}_i) + g_i(\mathbf{z}_i) + \iota_{h_i(\mathbf{w}_i)\leq\mathbf{y}_i}\left(\mathbf{w}_i,\mathbf{y}_i\right)\right) \\ \text{s.t} \ & \sum_{i\in V} \mathbf{y}_i = \mathbf{0}\ \mathbf{A}_k\mathbf{x} - \mathbf{b}_k \underset{k}{\blacktriangleright} \mathbf{0}\ \forall k \in \kappa \\ & \mathbf{x}_i - \mathbf{w}_i = \mathbf{0},\ \mathbf{A}_{i,g}\mathbf{x}_i - \mathbf{z}_i - \mathbf{b}_{i,g} = \mathbf{0}\ \forall i \in V. \end{aligned}$$

This is exactly in the form of (1) and thus can be directly solved via the DMM algorithm. In essence here, we have introduced a set of *virtual nodes* into the network with each handling a subset of the newly introduced primal variables. By separating the roles of f_i, g_i and $\delta_{h_i(\mathbf{w}_i)\leq\mathbf{y}_i}$ across these virtual nodes we can potentially reduce the complexity of the primal updates of the DMM algorithm. Furthermore as these virtual nodes only need to communicate with a single physical node, this approach also introduces no additional overhead in terms of communication cost making it an attractive choice for practical distributed implementations.

V. Application to Distributed Signal Processing Tasks

The role of the following section is to demonstrate how the proposed DMM algorithm can be used to solve a range of practical distributed optimization problems. For this purpose we have chosen three signal processing examples including beamforming, channel capacity maximization and the optimization of a communal Markowitz portfolio.

A. Random Network Modeling

For all of the following examples, the networks we consider are generated via classic stochastic graph models. We consider three specific models: undirected Erdős-Rényi (ER) graphs [49], Watts-Strogatz (WS) small world graphs [50] and geometric random (GR) graphs [51]. Each different models, whilst being straightforward to implement, is constructed via different means and exhibit different network characteristics. In particular, WS and GR networks have been shown to be good candidates for modeling real world networks [50] [52]. Detailed explanations of the three topologies considered can be found in [49]–[51].

Each network was ensured to be sparsely connected and to form a single connected component as per our assumptions. This was achieved by configuring the associated parameters of each of the three network models. For the ER graphs, the probability of connection, which controls the set of constructed edges, was set to $\frac{\ln(N)}{N}$. This is referred to as the critical probability and generates networks with a low number of edges and a high probability the network being connected. For the WS graphs, the configuration process required two steps. Firstly, the initial K-hop lattice networks were configured so that $K = \lceil \ln(N) \rceil$. This choice generates networks with a similar number of edges to that of an equally sized ER graph. Secondly the probability of reconnection was configured to 5% to create only a limited number of random connections. Finally for the GR graphs, a three dimensional unit cube was used to bound the locations of the randomly placed nodes while the transmission distance of the nodes was set to $r = \sqrt[3]{\ln(N)/N}$ to again ensure a sparsely connected network with a high probability of connectivity.

B. A Reference Centralized Averaged PR-Splitting Method

In addition to demonstrating the performance of the DMM algorithm in different network topologies, the following simulations also draw a comparison to a centralized averaged PR-splitting based approach. The motivation for this comparison is to offer insight into the degradation in performance experienced through the use of the proposed lifting. As with the DMM algorithm, the centralized implementation stems from the prototype problem in (1) which we can equivalently write in the unconstrained form

$$\min_{\mathbf{x}} f(\mathbf{x}) + \sum_{k\in\kappa} \iota_{\mathbf{A}_k\mathbf{x}-\mathbf{b}_k \underset{k}{\blacktriangleright} \mathbf{0}}\left(\mathbf{x}\right). \tag{18}$$

As with the extended dual problem, we can solve (18) by equivalently finding a solution of the monotonic inclusion

$$\mathbf{0} \in \partial f(\mathbf{x}) + \sum_{k\in\kappa} \partial\iota_{\mathbf{A}_k\mathbf{x}-\mathbf{b}_k \underset{k}{\blacktriangleright} \mathbf{0}}\left(\mathbf{x}\right).$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSIPN.2019.2901649, IEEE Transactions on Signal and Information Processing over Networks

By setting $\mathbf{T}_1 = \partial f$ and $\mathbf{T}_2 = \sum_{k\in\kappa} \partial \iota_{\mathbf{A}_k\mathbf{x}-\mathbf{b}_k \underset{k}{\blacktriangleright} \mathbf{0}}$, we can therefore apply averaged PR-splitting as in Section III-I to produce the iterative centralized approach given in Algorithm 2. As with the distributed averaged PR splitting method, the convergence of this approach follows from its relationship with Krasnosel'skiĭ-Mann type iterations. In the following, references to a centralized implementation refer to this approach.

Algorithm 2 Centralized Averaged PR-Splitting

Initialize: $\mathbf{z}^{(0)} \in \mathbb{R}^{M_V}$
for t=0,..., **do**
  $\mathbf{x}^{(t+1)} = \underset{\mathbf{x}}{\operatorname{argmin}} \left(f(\mathbf{x}) + \frac{\rho}{2}\|\mathbf{x} - \mathbf{z}^{(t)}\|^2 \right)$
  $\mathbf{y}^{(t+1)} = \underset{\mathbf{y}}{\operatorname{argmin}} \left(\sum_{k\in\kappa} \iota_{\mathbf{A}_k\mathbf{y}-\mathbf{b}_k \underset{k}{\blacktriangleright} \mathbf{0}}(\mathbf{y}) + \frac{\rho}{2}\|\mathbf{y} - 2\mathbf{x}^{(t+1)} + \mathbf{z}^{(t)}\|^2 \right)$
  $z^{(t+1)} = (1-\alpha^{(t)})z^{(t)} + \alpha^{(t)} \left(2\mathbf{y}^{(t+1)} - 2\mathbf{x}^{(t+1)} + \mathbf{z}^{(t)}\right).$

C. Distributed Beamforming

For our first application, consider the use of an N node wireless sensor network (WSN) where each node is equipped with a single receiver used to measure an acoustic signal. Given a set of noisy measurements taken by the network, the aim is to recover an unknown target signal of interest. The noise is assumed to be spatially uncorrelated Gaussian noise at each node with variance $\sigma_i^2 \; \forall \; i \in V$.

Such signals are typically processed in the time-frequency domain such that any delay can be expressed as a phase shift and thus as a complex scaling. For each frequency bin our objective is to therefore to design a linear filter which preserves the signal in a target subspace $\mathbf{\Lambda}$ while reducing the power of received noise. Such a filter is a minimum variance distortionless response (MVDR) beamfomer for the specific case of uncorrelated noise [53] and can be computed as a solution to the following optimization problem:

$$\min_{\mathbf{w}} \sum_{i\in V} \frac{1}{2} x_i^H \sigma_i^2 x_i \quad \text{s.t.} \sum_{i\in V} \left(\Lambda_i x_i - \frac{1}{N} \right) = 0,$$

Here, $\mathbf{x}$ denotes the vector of filter weights. Assuming that the elements of the vector Λ_i and σ_i are known locally at each node i, this problem is exactly in the form of (1).

Directly applying DMM, we can define the distributed MVDR beamformer given in Algorithm 3. Note that as there are no inequality constraints, the auxiliary updates have been simplified to remove the dependences on $\mathbf{S}$.

To demonstrate the performance of the proposed method, three 1000 node networks were generated as per the different methods outlined in Section V-A. The noise variances σ_i and target subspace $\mathbf{\Lambda}_i$ which were used for all three networks were generated randomly. The step size of DMM was empirically chosen for each network to optimize convergence rate. Additionally, $\forall t \in \mathbb{N}, \alpha^{(t(t))} = \frac{1}{2}$ was selected, resulting in a DR splitting variant of the DMM algorithm. The primal convergence of the algorithm is given in Fig. 6 in addition to

Algorithm 3 Distributed Beamforming for Uncorrelated Noise

Initialize: $\mathbf{z}^{(0)} \in \mathbb{R}^{M_E}$
for t=0,..., **do**
  for all $i \in V$ **do** ▷ Primal and Dual Updates
    $\gamma_i^{(t+1)} = \frac{1}{|\mathcal{N}(i)|} \sum_{j\in\mathcal{N}(i)} z_{(i,j)}^{(t)}$
    $x_i^{(t+1)} = \left(\sigma_i^2 + \frac{\rho\Lambda_i^H\Lambda_i}{|\mathcal{N}(i)|}\right)^{-1} \left(\Lambda_i^H\gamma_i^{(t+1)} + \frac{\rho\Lambda_i^H}{N|\mathcal{N}(i)|}\right)$
    $w_{(i,j)}^{(t+1)} = 2\gamma_i^{(t+1)} - z_{(i,j)}^{(t)} - 2\rho\left(\frac{x_i^{(t+1)}}{|\mathcal{N}(i)|} - \frac{1}{N|\mathcal{N}(i)|}\right)$
  for all $i \in V, j \in \mathcal{N}(i)$ **do** ▷ Transmit Variables
    $\mathbf{Node}_j \leftarrow \mathbf{Node}_i(w_{(i,j)}^{(t+1)})$
  for all $i \in V, j \in \mathcal{N}(i)$ **do** ▷ Auxiliary Variables
    $z_{(i,j)}^{(t+1)} = (1-\alpha^{(t)})z_{(i,j)}^{(t)} + \alpha^{(t)} w_{(j,i)}^{(t+1)}.$

the convergence of the objective function and relative objective error $\|f(\mathbf{x}) - f(\mathbf{x}^*)\|^2$. In addition to the three types of networks considered, we also include results for the reference centralized approach where again the step size parameter ρ was empirically optimized. The final precision of both algorithms is due to the physical hardware utilized.

For the given number of nodes, the ability of the algorithm to achieve machine precision in less than N iterations is a satisfying result. Furthermore, due to the quadratic nature of the local optimization problems at each node, the primal updates are analytic and inexpensive to compute. The convergence rate is also far better than the asymptotic bound of Krasnosel'skiĭ-Mann type schemes [48, Theorem 5.15] which most likely stems from the strong convexity and smoothness of each local objective function. Of additional interest is the fact that the proposed DMM offers comparable performance to that of the centralized PR-splitting method. This is most clearly demonstrated in the primal variables with the primal error only converging twice as fast in the centralized case. In this way, the additional dual lifting has not come at a considerable reduction in convergence rate while allowing for a fully distributable implementation.

We can also compare the proposed DMM algorithm and the equivalent centralized averaged PR splitting method in terms of computational complexity and memory load. In the case of computational complexity, the most expensive operation for both methods is the primal update of $\mathbf{x}^{(t+1)}$. Due to the separable and quadratic nature of $f(\mathbf{x})$ in this case, this has a complexity of $\mathbf{O}(N)$ for the centralized approach. For the proposed DMM algorithm, each node must solve a scalar quadratic problem with a complexity of $\mathbf{O}(1)$. With regards to memory load, the centralized algorithm must store a total of $3N$ variables in memory. In contrast, for DMM each node must store a maximum of $2 + 3\mathcal{N}(i)$ variables resulting in a total memory load of $2N + 6|E|$ across the entire network. These costs are summarized in Table I.

From these results we can make a number of comments. Firstly, in terms of computational complexity, the proposed DMM algorithm does not result in an increase in complexity, a point previously alluded to in Section III-C. This is despite the fact that the lifted problem has a larger number of variables.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSIPN.2019.2901649, IEEE Transactions on Signal and Information Processing over Networks

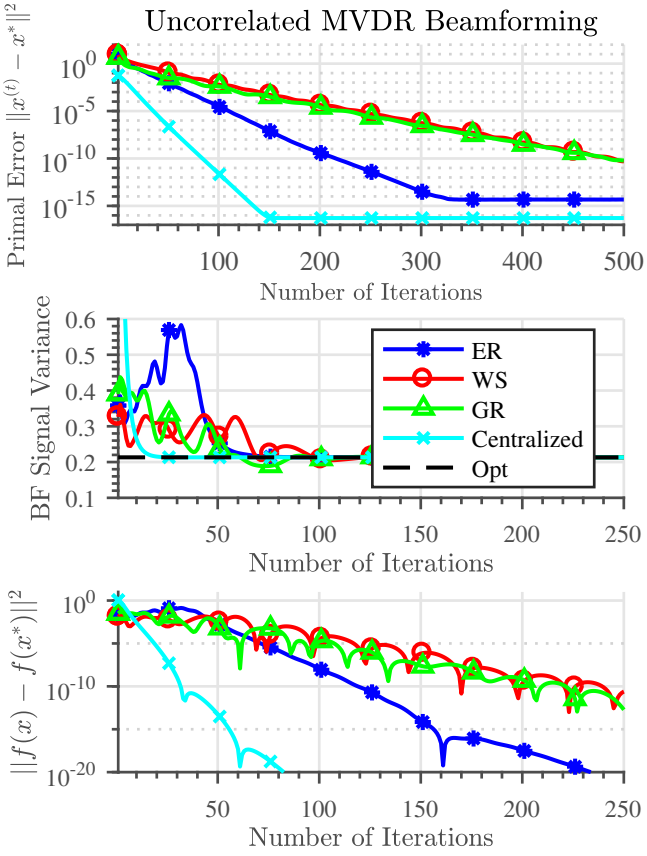


Fig. 6: An example of uncorrelated MVDR beamforming for a 1000 node network via the DMM algorithm. We compare the primal mean squared error, the variance of the resulting beamformed signal and the relative objective error for the three networks in question and the centralized approach.

Type	Alg. 2	DMM (network)	DMM (per node)
Complexity	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(1)$
Memory Load	$3N$	$2N+6\|E\|$	$2+3\mathcal{N}(i)$

TABLE I: Computational complexity and memory load for the uncorrelated MVDR problem.

Secondly, in terms of memory load, while there is an increase in the total memory requirement, this is distributed over the set of nodes themselves with the total memory required by any one node being parameterized by the size of its neighborhood. For sparsely connected networks, such a scaling is a tolerable price to pay for a fully distributed implementation.

D. Gaussian Channel Capacity Maximization

As a second example, consider a WSN of N independent antennas trying to communicate a signal back to a target location over a set of N additive white Gaussian channels (AWGNs). Given a local bandwidth B_i for each channel, the objective of this problem is to optimally configure the transmission power of the antennas ($\mathbf{x}$) to maximize channel capacity under a total power constraint. From the Shannon-Hartley theorem [54], the capacity of each channel (C_i) is given by

$$C_i = B_i \log_2\left(1+\frac{x_i}{\sigma_i}\right) = \frac{B_i(\ln(\sigma_i+x_i)-\ln(\sigma_i))}{\ln(2)},$$

where σ_i^2 is the noise variance of the ith channel.

The channel capacity maximization problem under a maximum power constraint can be rephrased as a convex optimization problem of the form

$$\min_{\mathbf{x}} \; -\sum_{i\in V} B_i \ln(x_i+\sigma_i) \;\text{ s.t }\; \sum_{i\in V} x_i = 1,\; \mathbf{x}\geq \mathbf{0},$$

where the non-negativity constraints stem from the fact that power is non-negative. If assuming each node i has an additional local maximum transmission power constraint $x_i \leq \beta_i$, the final optimization problem is given by

$$\min_{\mathbf{x}} \; -\sum_{i\in V} B_i \ln(x_i+\sigma_i) \;\text{ s.t }\; \sum_{i\in V} x_i = 1,\; \boldsymbol{\beta}\geq \mathbf{x}\geq \mathbf{0}, \quad (19)$$

where the vector $\boldsymbol{\beta}$ is the stacked vector of all β_i.

By using indicator functions to move the local constraints to the objective, (19) can be converted into the form of (1). The resulting problem is given by

$$\begin{aligned}\min_{\mathbf{x}} \quad & \sum_{i\in V}\Big(-B_i\ln(x_i+\sigma_i)+\iota_{\geq 0}(x_i)+\iota_{\leq \beta_i}(x_i)\Big)\\ \text{s.t} \quad & \sum_{i\in V}\left(x_i-\frac{1}{N}\right)=0,\end{aligned}$$

after which we can directly apply the proposed DMM algorithm. This is summarized in Algorithm 4.

Algorithm 4 Distributed Channel Capacity Maximization

Initialize: $\mathbf{z}^{(0)} \in \mathbb{R}^{M_E}$
for t=0,..., **do**
  for all $i \in V$ **do** ▷ Primal and Dual Updates
    $\gamma_i^{(t+1)} = \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)} z_{(i,j)}^{(t)}$
    $x_i^{(t+1)} = \arg\min_{0\leq x_i\leq\beta_i}\Big(-B_i\ln(x_i+\sigma_i) - \Big\langle\gamma_i^{(t+1)}, x_i\Big\rangle + \frac{\rho}{2|\mathcal{N}(i)|}||x_i-\frac{1}{N}||^2\Big)$
    $w_{(i,j)}^{(t+1)} = 2\gamma_i^{(t+1)} - z_{(i,j)}^{(t)} - 2\rho\left(\frac{x_i^{(t+1)}}{|\mathcal{N}(i)|}-\frac{1}{N|\mathcal{N}(i)|}\right)$
  for all $i \in V, j\in\mathcal{N}(i)$ **do** ▷ Transmit Variables
    $\mathbf{Node}_j \leftarrow \mathbf{Node}_i(w_{(i,j)}^{(t+1)})$
  for all $i \in V, j\in\mathcal{N}(i)$ **do** ▷ Auxiliary Variables
    $z_{(i,j)}^{(t+1)} = (1-\alpha^{(t)})z_{(i,j)}^{(t)} + \alpha^{(t)} w_{(j,i)}^{(t+1)}$.

Again, we consider the performance of this algorithm in three random networks, each comprised of 100 nodes as well as the centralized approach for comparison. For each, the same randomly generated σ_i and β_i were used. The resulting convergence characteristics are given in Fig. 7 where the step size ρ was chosen to optimize the convergence rate for each network. It was found that for this particular problem large step sizes (on the order of 10^3) provided much faster convergence

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSIPN.2019.2901649, IEEE Transactions on Signal and Information Processing over Networks

rates. As with the DS beamformer, $\forall t \in \mathbb{N}, \alpha^{(t)} = \frac{1}{2}$ was selected. Again, the finite noise floor observed is due to the finite numerical precision of the simulations.

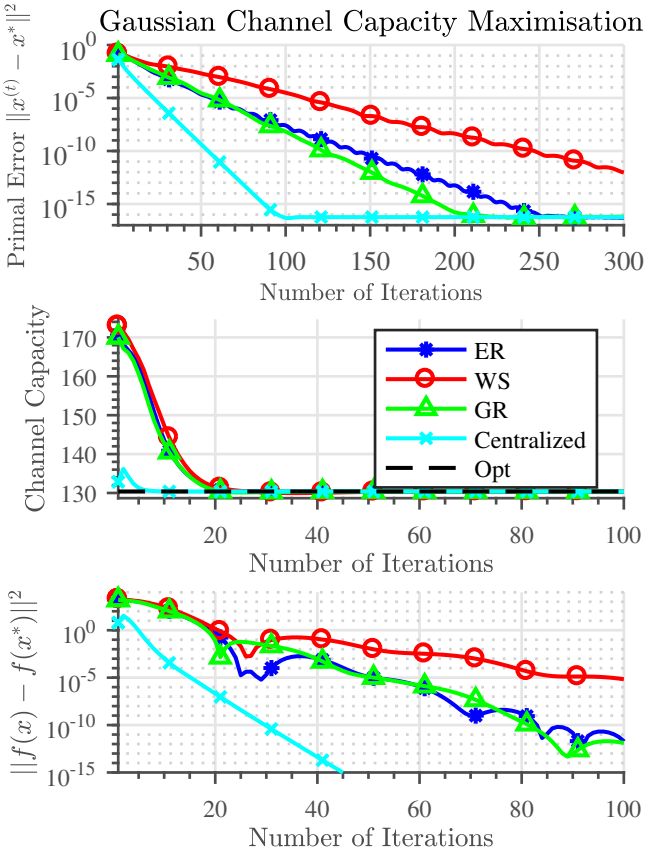


Fig. 7: An example of channel capacity maximization for a 100 node network via the DMM algorithm. Note that the initial overshoot in terms of channel capacity stems from the violation of the constraint functions for these iterations.

We can observe that the algorithm quickly converges, reaching a primal precision of 10^{-15} in 200-350 iterations. As with the distributed DS beam-former, the proposed algorithm exhibits a linear convergence rate. Again, this most likely stems from the strong convexity and smoothness of each local problem over the allowed domain $\beta_i \geq x_i \geq 0 \ \forall i \in V$ although no proof of this is offered at this time. Finally, we can also observe that, when compared with the centralized approach, there is no significant degradation in convergence rate through the use of the DMM algorithm.

As with our pervious beamforming example, we can also compare the two algorithms in terms of computational complexity and memory load. In the case of the latter, as the local objective functions and the number of constraints are equivalent to those in Section V-C, the total memory loads for this problem are identical to those as given in Table I. In the case of computational complexity, again the primal updates are the most expensive operations for both algorithms as they require the minimization of the sum of a negative logarithm and a quadratic term. However, due to the separable nature of the objective, the centralized $\mathbf{x}$ updates can be performed in parallel. The scalar nature and bounded feasible range of each subproblem means that they can be solved to machine precision in $\mathcal{O}(1)$ operations. For the centralized approach, this results in a total complexity of $\mathcal{O}(N)$. The $\mathbf{x}$ updates of the DMM algorithm have the same per node complexity as the separated subproblems in the centralized case and thus incur a cost of $\mathcal{O}(1)$. These values are summarized in Table II.

Type	Alg. 2	DMM (network)	DMM (per node)
Complexity	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(1)$
Memory Load	$3N$	$2N+6\|E\|$	$2+3\mathcal{N}(i)$

TABLE II: Computational complexity and memory load for Gaussian channel capacity maximization problem.

Again we can note that the computational complexity of this approach does not scale due to the use of the proposed lifting while the increase in maximum memory load per node is bounded based on the connectivity of the network.

E. Portfolio Optimization

As a final example we consider the task of Markowitz portfolio optimization [55]. While this problem in its standard form is inherently non-distributed, here we consider a variant for a collaborative network of investors. In the non-collaborative case [56, Sec. 4.4.1], the basic premise of this problem is that each node within the network has a local portfolio of stocks or assets into which they want to invest a given local wealth w_i while minimizing the risk of the investment for a certain return r_i. The return on the set of such stocks is modeled by the random vector $\mathbf{p}_i \in \mathbb{R}^{M_i}$ such that $\mathbb{E}[\mathbf{p}_i] = \bar{\mathbf{p}}_i \in \mathbb{R}^{M_i}$ and $\mathbb{E}\left[(\mathbf{p}_i - \bar{\mathbf{p}}_i)^2\right] = \mathbf{Q}_i \in \mathbb{R}^{M_i \times M_i}$. The risk of investment can therefore be modeled as a quadratic cost. Ultimately, each node $i \in V$ wants to solve

$$\min_{\mathbf{x}_i} \ \frac{1}{2}\mathbf{x}_i^T \mathbf{Q}_i \mathbf{x}_i \text{ s.t. } \bar{\mathbf{p}}_i^T \mathbf{x}_i \geq r_i, \mathbf{1}^T \mathbf{x}_i = w_i, \ \mathbf{x}_i \geq 0,$$

where $\mathbf{x}_i$ denotes the vector of investments made in the stocks considered and the final set of constraints indicate that we do not want to considering short positions in our investments.

We consider a variant of this problem where the set of nodes work together to lower the total risk of the network by investing in each others portfolios while maintaining their own individual return ambitions. Additionally we incorporate local investment constraints which require a certain amount of each nodes wealth to be invested locally. reflecting a prior favoritism by an investor. The collaborative variant of the portfolio optimization problem is given by

$$\begin{aligned} \min_{\mathbf{x}} \quad & \sum_{i \in V} \frac{1}{2}\mathbf{x}_i^T \mathbf{Q}_i \mathbf{x}_i \\ \text{s.t.} \quad & \sum_{i \in V} \left(\bar{\mathbf{p}}_i^T \mathbf{x}_i - r_i\right) \geq 0, \ \sum_{i \in V} \left(\mathbf{1}^T \mathbf{x}_i - w_i\right) = 0 \\ & \mathbf{x}_i \geq 0, \ \mathbf{1}^T \mathbf{x}_i \geq \eta_i w_i \ \forall \ i \in V. \end{aligned} \tag{20}$$

Here, the first two constraints ensure that the total return and total wealth requirements of the network are satisfied in a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSIPN.2019.2901649, IEEE Transactions on Signal and Information Processing over Networks

collaborative rather than node based case. The third constraint is the non-shorting constraint while the final constraint captures the required local portfolio investment at each node. The variables $\eta_i \in [0,1]$ capture the required ratio of local portfolio investment to local wealth.

Defining the matrices $\mathbf{A}_i$ and vector $\mathbf{b}_i$ as

$$\mathbf{A}_i = \begin{bmatrix} \bar{\mathbf{p}}_i^T & \mathbf{0} \\ \mathbf{0} & \mathbf{1}^T \end{bmatrix}, \ \mathbf{b}_i = \begin{bmatrix} r_i \\ w_i \end{bmatrix} \ \forall i \in V,$$

and by utilizing indicator functions to shift the local constraints to the objective function, (20) can be rewritten as

$$\begin{aligned} \min_{\mathbf{x}} \quad & \sum_{i \in V} \left(\frac{1}{2} \mathbf{x}_i^T \mathbf{Q}_i \mathbf{x}_i + \iota_{\mathbf{x}_i \geq \mathbf{0}} (\mathbf{x}_i) + \iota_{\mathbf{1}^T \mathbf{x}_i \geq \eta_i w_i} (\mathbf{x}_i) \right) \\ \text{s.t} \quad & \sum_{i \in V} (\mathbf{A}_i \mathbf{x}_i - \mathbf{b}_i) \blacktriangleright 0, \end{aligned} \tag{21}$$

It follows that (21) is exactly in the form given by (1). Therefore, applying the DMM algorithm, (20) can be solved distributedly via Algorithm 5. In Algorithm 5, $\odot$ is used to denote the element-wise or Hadamard product of two vectors.

Algorithm 5 Collaborative Markowitz Portfolio Optimization

Initialize: $\mathbf{z}^{(0)} \in \mathbb{R}^{M_E}$
for t=0,..., **do**
  for all $i \in V$ **do** ▷ Primal and Dual Updates
    $\boldsymbol{\gamma}_i^{(t+1)} = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \mathbf{z}_{(i,j)}^{(t)}$
    $\mathbf{x}_i^{(t+1)} = \underset{\mathbf{x}_i \geq 0, \mathbf{1}^T \mathbf{x}_i \geq \eta_i w_i}{\arg\min} \left(\frac{1}{2} \mathbf{x}_i^T \mathbf{Q}_i \mathbf{x}_i - \left\langle \mathbf{A}_i \boldsymbol{\gamma}_i^{(t+1)}, \mathbf{x}_i \right\rangle + \frac{\rho}{2|\mathcal{N}(i)|} \|\mathbf{A}_i \mathbf{x}_i - \mathbf{b}_i\|^2 \right)$
    $\mathbf{w}_{(i,j)}^{(t+1)} = 2\boldsymbol{\gamma}_i^{(t+1)} - \mathbf{z}_{(i,j)}^{(t)} - 2\rho \left(\frac{\mathbf{A}_i \mathbf{x}_i^{(t+1)}}{|\mathcal{N}(i)|} - \frac{\mathbf{b}_i}{|\mathcal{N}(i)|} \right)$
  for all $i \in V, j \in \mathcal{N}(i)$ **do** ▷ Transmit Variables
    $\mathbf{Node}_j \leftarrow \mathbf{Node}_i(\mathbf{w}_{(i,j)}^{(t+1)})$
  for all $i \in V, j \in \mathcal{N}(i)$ **do** ▷ Auxiliary Variables
    $\mathbf{v}_{(i,j)}^{(t+1)} = \mathbf{w}_{(j,i)}^{(t+1)} - \begin{bmatrix} 1 \\ 0 \end{bmatrix} \odot \min \left\{ \mathbf{w}_{(i,j)}^{(t+1)} + \mathbf{w}_{(j,i)}^{(t+1)}, \mathbf{0} \right\}$
    $\mathbf{z}_{(i,j)}^{(t+1)} = (1 - \alpha^{(t)}) \mathbf{z}_{(i,j)}^{(t)} + \alpha^{(t)} \mathbf{v}_{(i,j)}^{(t+1)}.$

For demonstration purposes we consider an Erdős-Rényi network comprised of 100 nodes, each with a local portfolio size of 20 elements. The system parameters $(\mathbf{Q}_i, \bar{\mathbf{p}}_i, r_i, w_i, \eta_i)$ were generated randomly and ρ was empirically selected to optimize convergence rate. The centralized approach was also applied for comparisons sake. The resulting convergence characteristics are included in Fig. 8.

As with our other two examples, we can observe the familiar linear convergence rate of the primal error in addition to the rapid optimality of the configuration. In particular, within around 200 iterations the total portfolio risk is within 0.1% of the optimal configuration. In contrast to the other two examples however, here we can note a significant degradation in convergence between the DMM implementation and the centralized approach. Additionally, the DMM algorithm required far more iterations to converge in this instance than

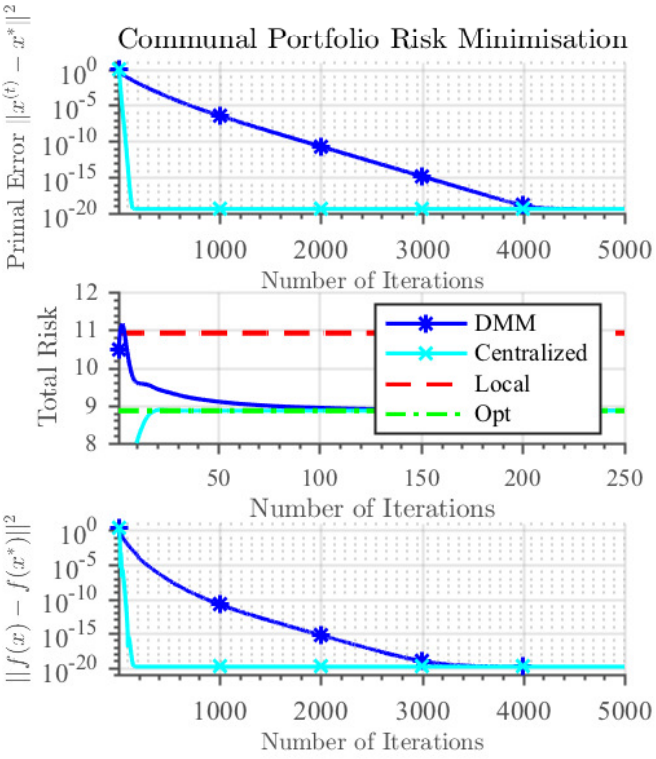


Fig. 8: Distributed Markowitz portfolio optimization for a 100 node network with 20 stocks per node solved via the DMM algorithm. The red line denotes the total risk of the network without collaboration, the green is the optimal configuration.

for the other examples despite having a comparable network size. While not as encouraging for the usability of DMM as the other two examples, this result is more inline with our expectation for a distributed method, i.e., that shifting to a distributed context should reduce convergence rate compared with a centralized approach.

Finally, we can again compare DMM and our centralized approach in terms of computational complexity and memory load. In the case of the prior, the most expensive operation for both algorithms is the primal $\mathbf{x}$ update. Due to the separability of the objective function and the quadratic nature of the problem, this comes at a cost of $\mathcal{O}(\sum_{i=1}^N M_i^3)$ for the centralized algorithm while for DMM the cost is $\mathcal{O}\left(M_i^3\right)$ per node. Both of these costs stem from the need to compute matrix inverses during these updates. In the case of memory load, the centralized approach stores $3\sum_{i=1}^N M_i$ variables in memory while for DMM each node stores $2 + M_i + 8\mathcal{N}(i)$ variables resulting in a total of $2N + \sum_{i=1}^N M_i + 16|E|$ variables across the network. These costs are summarized in Table III.

We can again note that the computational complexity of the algorithm does not scale when shifting to the distributed implementation while, in this case, the memory load may be lower in the distributed case. In particular, for the considered case of $M_i = 20 \ \forall i \in V$, if $|\mathcal{N}(i)| \in \{1, \cdots, 4\} \ \forall i \in V$ then the total memory requirement of the entire network is lower for the DMM method than the centralized approach.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSIPN.2019.2901649, IEEE Transactions on Signal and Information Processing over Networks

Type	Alg. 2	DMM (network)	DMM (per node)
Complexity	$\mathcal{O}\left(\sum_{i=1}^{N} M_i^3\right)$	$\mathcal{O}\left(\sum_{i=1}^{N} M_i^3\right)$	$\mathcal{O}\left(M_i^3\right)$
Memory Load	$3\sum_{i=1}^{N} M_i$	$2N+\sum_{i=1}^{N} M_i+16\lvert E\rvert$	$2+M_i+8\mathcal{N}(i)$

TABLE III: Computational complexity and memory load for the communal portfolio maximization problem.

VI. Conclusions

In this paper we have presented a novel method for distributed optimization of separable convex problems. In contrast to other existing methods within the literature, the prototype problem of the proposed DMM algorithm is not restricted based on the topology of the underlying network. This allows DMM to be used in a much broader range of applications whilst preserving its distributed operation. Furthermore, the derivation for this method is based on classical monotone operator theory with the DMM algorithm itself being based on a combination of Peaceman-Rachford splitting and Krasnosel'skiĭ-Mann iterations, providing an intuitive interpretation of the approach. The convergence of DMM follows from the existing results for these methods. The use of DMM was demonstrated for a range of practical signal processing problems including beamforming, channel capacity maximization and portfolio optimization for a range of network types. Overall, DMM demonstrates that any separable problem can be solved in a distributed manner in undirected networks and thus provides a novel tool for distributed convex optimization.

Appendix

A. Proof of Lemma IV.1

From (13), the resolvent of $\mathbf{z}^{(t)}$ is defined as

$$\boldsymbol{\lambda}^{(t+1)} = \mathbf{J}_{\mathbf{T}_1,\rho}\left(\mathbf{z}^{(t)}\right).$$

From the definition of $\mathbf{T}_1$ (11a), it follows that

$$\boldsymbol{\lambda}^{(t+1)} \in \mathbf{z}^{(t)} - \rho\left(\mathbf{C}\partial f^*\left(\mathbf{C}^T\boldsymbol{\lambda}^{(t+1)}\right) - \mathbf{d} + \partial\iota_{\mathcal{C}_2}\left(\boldsymbol{\lambda}^{(t+1)}\right)\right).$$

Defining, $\mathbf{x} \in \partial f^*\left(\mathbf{C}^T\boldsymbol{\lambda}\right)$, for a given $\mathbf{x}^{(t+1)}$, $\boldsymbol{\lambda}^{(t+1)}$ can be computed as

$$\boldsymbol{\lambda}^{(t+1)} = \underset{\boldsymbol{\mathcal{L}}\boldsymbol{\lambda}=\mathbf{0}}{\arg\min}\left(\frac{1}{2}\|\boldsymbol{\lambda} - \mathbf{z}^{(t)} + \rho\left(\mathbf{C}\mathbf{x}^{(t+1)} - \mathbf{d}\right)\|^2\right) \quad (22)$$

Given the structure of $\mathbf{C}$ and $\mathbf{d}$, $\boldsymbol{\mathcal{L}}\left(\mathbf{C}\mathbf{x}^{(t+1)} - \mathbf{d}\right) = \mathbf{0}$ by design. We can therefore perform a change of variables in the dual update of (22) by defining the additional variable

$$\boldsymbol{\gamma}^{(t+1)} = \boldsymbol{\lambda}^{(t+1)} + \rho\left(\mathbf{C}\mathbf{x}^{(t+1)} - \mathbf{d}\right).$$

From (22), $\boldsymbol{\gamma}^{(t+1)}$ can be computed as

$$\boldsymbol{\gamma}^{(t+1)} = \underset{\mathcal{L}\boldsymbol{\gamma}=\mathbf{0}}{\arg\min}\left(\frac{1}{2}\|\boldsymbol{\gamma} - \mathbf{z}^{(t)}\|^2\right) = \underset{\ker(\mathcal{L})}{\Pi}\mathbf{z}^{(t)}.$$

Using the definitions of $\mathbf{x}^{(t+1)}$ and $\boldsymbol{\gamma}^{(t+1)}$, it follows that

$$\mathbf{x}^{(t+1)} \in \partial f^*\left(\mathbf{C}^T\left(\boldsymbol{\gamma}^{(t+1)} - \rho\left(\mathbf{C}\mathbf{x}^{(t+1)} - \mathbf{d}\right)\right)\right),$$

and thus that the primal updates satisfy the inclusion

$$\partial f\left(\mathbf{x}^{(t+1)}\right) \ni \mathbf{C}^T\left(\boldsymbol{\gamma}^{(t+1)} - \rho\left(\mathbf{C}\mathbf{x}^{(t+1)} - \mathbf{d}\right)\right).$$

The primal updates $\mathbf{x}^{(t+1)}$ can therefore be computed as

$$\mathbf{x}^{(t+1)} = \arg\min_{\mathbf{x}}\left(f(\mathbf{x}) + \frac{\rho}{2}\|\mathbf{C}\mathbf{x} - \mathbf{d}\|^2 - \left\langle \mathbf{C}^T\boldsymbol{\gamma}^{(t+1)}, \mathbf{x}\right\rangle\right).$$

The definition of the reflected resolvent completes the proof.

B. Proof of Lemma (IV.2)

Define the output of the resolvent operator by the vector

$$\mathbf{y}^{(t+1)} = \mathbf{J}_{\rho,\mathbf{T}_2}\left(\mathbf{w}^{(t+1)}\right).$$

By the definition of the operator $\mathbf{T}_2$ (11b), $\mathbf{y}^{(t+1)}$ can be computed as the minimizer of

$$\min_{\mathbf{y}} \frac{1}{2}\|\mathbf{y} - \mathbf{w}^{(t+1)}\|^2 \quad \text{s.t. } (\mathbf{I}-\mathbf{P})\,\mathbf{y} = \mathbf{0},\ \mathbf{S}\mathbf{y}^* \geq \mathbf{0}. \quad (23)$$

Equivalently, it can be computed by solving the KKT system of equations [56] given by

$$\mathbf{y}^* - \mathbf{w}^{(t+1)} - (\mathbf{I}-\mathbf{P})\,\boldsymbol{\mu}^* - \mathbf{S}\boldsymbol{\nu}^* = \mathbf{0},\ (\mathbf{I}-\mathbf{P})\,\mathbf{y}^* = \mathbf{0},$$
$$\mathbf{S}\mathbf{y}^* \geq \mathbf{0},\ \boldsymbol{\nu}^* \geq \mathbf{0},\ (\mathbf{S}\mathbf{y}^*) \odot \boldsymbol{\nu}^* = \mathbf{0},$$

where $\odot$ denotes the Hadamard product, the element-wise product of two vectors. The variables $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ denote the dual variables associated with the constraints in (23).

Combining the first two equalities and using the self-inverse property of $\mathbf{P}$, $\mathbf{P}^2 = \mathbf{I}$, it follows that

$$-(\mathbf{I}-\mathbf{P})\,\boldsymbol{\mu}^* = \frac{1}{2}(\mathbf{I}-\mathbf{P})\left(\mathbf{w}^{(t+1)} + \mathbf{S}\boldsymbol{\nu}^*\right),$$

and thus, that

$$\mathbf{y}^* = \frac{1}{2}(\mathbf{I}+\mathbf{P})\left(\mathbf{w}^{(t+1)} + \mathbf{S}\boldsymbol{\nu}^*\right).$$

For those dual variables corresponding to equality constraints, $\boldsymbol{\nu}$ plays no role. For the remaining variables, combining the non-negativity of $\mathbf{y}^*$ and $\boldsymbol{\nu}^*$ with the complementary slackness condition $(\mathbf{S}\mathbf{y}^*) \odot \boldsymbol{\nu}^* = \mathbf{0}$, it follows that $\mathbf{y}^* = \max\{\frac{1}{2}(\mathbf{I}+\mathbf{P})\,\mathbf{w}^{(t+1)}, \mathbf{0}\}$ where $\max\{\bullet,\bullet\}$ is an abuse of notation used to denote element-wise maximization between two vectors. The vector $\mathbf{y}^{(t+1)}$ is therefore given by

$$\mathbf{y}^{(t+1)} = \frac{1}{2}(\mathbf{I}+\mathbf{P})\,\mathbf{w}^{(t+1)} - \min\left\{\frac{1}{2}\mathbf{S}(\mathbf{I}+\mathbf{P})\,\mathbf{w}^{(t+1)}, \mathbf{0}\right\},$$

where, similar to $\max\{\bullet,\bullet\}$, $\min\{\bullet,\bullet\}$ denotes element-wise minimization. The definition of the reflected resolvent completes the proof.

References

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] G. Lu and W. H. Zeng, "Cloud computing survey," in *Applied Mechanics and Materials*, vol. 530. Trans. Tech. Publ, 2014, pp. 650–661.

[3] R. Bekkerman, M. Bilenko, and J. Langford, *Scaling up machine learning: Parallel and distributed approaches*. Cambridge University Press, 2011.

[4] A. Nedic, A. Olshevsky, A. Ozdaglar, and J. Tsitsiklis, "On distributed averaging algorithms and quantization effects," *IEEE Trans. on Automatic Control*, vol. 54, no. 11, pp. 2506–2517, 2009.

[5] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE/ACM Trans. on Networking (TON)*, vol. 14, no. SI, pp. 2508–2530, 2006.

[6] F. Bénézit, V. Blondel, P. Thiran, J. Tsitsiklis, and M. Vetterli, "Weighted gossip: Distributed averaging using non-doubly stochastic matrices," in *Information theory proceedings (isit), 2010 IEEE Int. symposium on*. IEEE, 2010, pp. 1753–1757.

[7] K. Murphy, Y. Weiss, and M. Jordan, "Loopy belief propagation for approximate inference: An empirical study," in *Proceedings of the Fifteenth Conf. on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1999, pp. 467–475.

[8] A. Schwing, T. Hazan, M. Pollefeys, and R. Urtasun, "Distributed message passing for large scale graphical models," in *Computer vision and pattern recognition (CVPR), 2011 IEEE Conf. on*. IEEE, 2011, pp. 1833–1840.

[9] D. Shuman, S. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Mag.*, vol. 30, no. 3, pp. 83–98, 2013.

[10] S. Chen, A. Sandryhaila, J. Moura, and J. Kovacevic, "Signal denoising on graphs via graph filtering," in *Signal and Information Processing (GlobalSIP), 2014 IEEE Global Conf. on*. IEEE, 2014, pp. 872–876.

[11] A. Loukas, A. Simonetto, and G. Leus, "Distributed autoregressive moving average graph filters," *IEEE Signal Processing Letters*, vol. 22, no. 11, pp. 1931–1935, 2015.

[12] E. Isufi, A. Simonetto, A. Loukas, and G. Leus, "Stochastic graph filtering on time-varying graphs," in *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2015 IEEE 6th Int. Workshop on*. IEEE, 2015, pp. 89–92.

[13] Z.-Q. Luo and W. Yu, "An introduction to convex optimization for communications and signal processing," *IEEE Journal on selected areas in communications*, vol. 24, no. 8, pp. 1426–1438, 2006.

[14] D. Bertsekas and J. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Prentice hall Englewood Cliffs, NJ, 1989, vol. 23.

[15] R. Rockafellar, "Network flows and monotropic optimization," 1984.

[16] ——, *Convex analysis*. Princeton, NJ: Princeton University Press, 1970.

[17] ——, "Monotone operators and the proximal point algorithm," *SIAM journal on control and Opt.*, vol. 14, no. 5, pp. 877–898, 1976.

[18] ——, "On the maximal monotonicity of subdifferential mappings," *Pacific Journal of Mathematics*, vol. 33, no. 1, pp. 209–216, 1970.

[19] J. Tsitsiklis, "Problems in decentralized decision making and computation." Massachusetts Inst. of Tech. Cambridge lab for information and decision systems, Tech. Rep., 1984.

[20] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Trans. on automatic control*, vol. 31, no. 9, pp. 803–812, 1986.

[21] J. Eckstein, "Splitting methods for monotone operators with applications to parallel optimization," Ph.D. dissertation, Massachusetts Institute of Technology, 1989.

[22] J. Eckstein and D. Bertsekas, "On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators," *Math. Programming*, vol. 55, no. 1-3, pp. 293–318, 1992.

[23] J. Eckstein, "Parallel alternating direction multiplier decomposition of convex programs," *Journal of Opt. Theory and Applications*, vol. 80, no. 1, pp. 39–62, 1994.

[24] 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*, vol. 3, no. 1, pp. 1–122, 2011.

[25] M. Zhu and S. Martínez, "On distributed convex optimization under inequality and equality constraints," *IEEE Trans. on Automatic Control*, vol. 57, no. 1, pp. 151–164, 2012.

[26] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[27] G. Scutari, F. Facchinei, and L. Lampariello, "Parallel and distributed methods for constrained nonconvex optimization-part i: Theory." *IEEE Trans. Signal Processing*, vol. 65, no. 8, pp. 1929–1944, 2017.

[28] G. Zhang and R. Heusdens, "Distributed optimization using the primal-dual method of multipliers," *IEEE Trans. on Signal and Information Processing over Networks*, 2017.

[29] B. Vũ, "A splitting algorithm for dual monotone inclusions involving cocoercive operators," *Advances in Computational Mathematics*, vol. 38, no. 3, pp. 667–681, 2013.

[30] L. Condat, "A primal-dual splitting method for convex optimization involving lipschitzian, proximable and linear composite terms," *Journal of Opt. Theory and Applications*, vol. 158, no. 2, pp. 460–479, 2013.

[31] D. Davis and W. Yin, "A three-operator splitting scheme and its optimization applications," *Set-valued and variational analysis*, vol. 25, no. 4, pp. 829–858, 2017.

[32] P. Bianchi, W. Hachem, and I. Franck, "A stochastic coordinate descent primal-dual algorithm and applications," in *Machine Learning for Signal Processing (MLSP), 2014 IEEE Int. Workshop on*. IEEE, 2014, pp. 1–6.

[33] P. Latafat and P. Patrinos, "Asymmetric forward-backward-adjoint splitting for solving monotone inclusions involving three operators," *Computational Opt. and Applications*, pp. 1–37, 2016.

[34] P. Latafat, L. Stella, and P. Patrinos, "New primal-dual proximal algorithm for distributed optimization," in *Decision and Control (CDC), 2016 IEEE 55th Conf. on*. IEEE, 2016, pp. 1959–1964.

[35] D. Mosk-Aoyama, T. Roughgarden, and D. Shah, "Fully distributed algorithms for convex optimization problems," *SIAM Journal on Opt.*, vol. 20, no. 6, pp. 3260–3279, 2010.

[36] A. Simonetto and H. Jamali-Rad, "Primal recovery from consensus-based dual decomposition for distributed convex optimization," *Journal of Opt. Theory and Applications*, vol. 168, no. 1, pp. 172–197, 2016.

[37] D. Mateos-Núnez and J. Cortés, "Distributed subgradient methods for saddle-point problems," in *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on*. IEEE, 2015, pp. 5462–5467.

[38] T.-H. Chang, A. Nedić, and A. Scaglione, "Distributed constrained optimization by consensus-based primal-dual perturbation method," *IEEE Trans. on Automatic Control*, vol. 59, no. 6, pp. 1524–1538, 2014.

[39] T.-H. Chang, "A proximal dual consensus admm method for multi-agent constrained optimization," *IEEE Trans. on Signal Processing*, vol. 64, no. 14, pp. 3719–3734, 2016.

[40] S. Lee and M. M. Zavlanos, "Distributed primal-dual methods for online constrained optimization," in *American Control Conference (ACC), 2016*. IEEE, 2016, pp. 7171–7176.

[41] I. Notarnicola and G. Notarstefano, "Constraint coupled distributed optimization: Relaxation and duality approach," *arXiv preprint arXiv:1711.09221*, 2017.

[42] A. Falsone, K. Margellos, S. Garatti, and M. Prandini, "Dual decomposition for multi-agent distributed optimization with coupling constraints," *Automatica*, vol. 84, pp. 149–158, 2017.

[43] P. L. Combettes and V. R. Wajs, "Signal recovery by proximal forward-backward splitting," *Multiscale Modeling & Simulation*, vol. 4, no. 4, pp. 1168–1200, 2005.

[44] V. Berinde, *Iterative approximation of fixed points*. Springer, 2007, vol. 1912.

[45] T. Sherson, R. Heusdens, and W. B. Kleijn, "Derivation and analysis of the primal-dual method of multipliers based on monotone operator theory," *arXiv preprint arXiv:1706.02654*, 2018.

[46] E. K. Ryu and S. Boyd, "Primer on monotone operator methods," *Appl. Comput. Math*, vol. 15, no. 1, pp. 3–43, 2016.

[47] R. Rockafellar, "On the maximality of sums of nonlinear monotone operators," *Trans. of the American Mathematical Society*, vol. 149, no. 1, pp. 75–88, 1970.

[48] H. H. Bauschke and P. L. Combettes, *Convex analysis and monotone operator theory in Hilbert spaces*. New York, NY.: Springer New York,, 2017, vol. 408.

[49] P. Erdös and A. Rényi, "On random graphs, i," *Publicationes Mathematicae (Debrecen)*, vol. 6, pp. 290–297, 1959.

[50] D. J. Watts and S. H. Strogatz, "Collective dynamics of "Small-World" networks," *nature*, vol. 393, no. 6684, p. 440, 1998.

[51] M. Penrose, *Random geometric graphs*. Oxford university press, 2003, no. 5.

[52] P. Gupta and P. Kumar, "The capacity of wireless networks," *IEEE Trans. on Information Theory*, vol. 46, no. 2, pp. 388–404, 2000.

[53] H. L. Van Trees, *Optimum array processing: Part IV of detection, estimation, and modulation theory*. John Wiley & Sons, 2004.

[54] C. Shannon, "Communication in the presence of noise," *Proceedings of the IRE*, vol. 37, no. 1, pp. 10–21, 1949.

[55] H. Markowitz, "Portfolio selection," *The journal of finance*, vol. 7, no. 1, pp. 77–91, 1952.

[56] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University press, 2004.