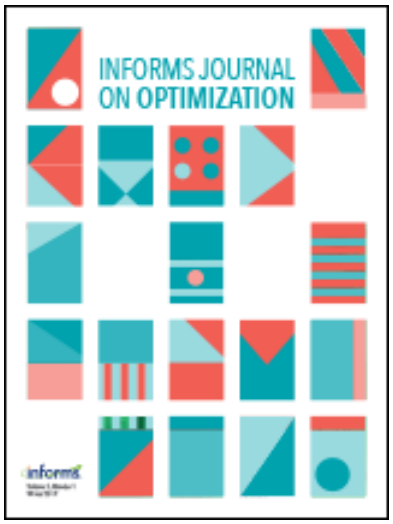

INFORMS Journal on Optimization

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Semi-proximal Augmented Lagrangian-Based Decomposition Methods for Primal Block-Angular Convex Composite Quadratic Conic Programming Problems

Xin-Yee Lam , Defeng Sun , Kim-Chuan Toh

To cite this article:
Xin-Yee Lam , Defeng Sun , Kim-Chuan Toh (2021) Semi-proximal Augmented Lagrangian-Based Decomposition Methods for Primal Block-Angular Convex Composite Quadratic Conic Programming Problems. INFORMS Journal on Optimization 3(3):254-277. https://doi.org/10.1287/ijoo.2019.0048

http://pubsonline.informs.org/journal/ijoo

INFORMS JOURNAL ON OPTIMIZATION
Vol. 3, No. 3, Summer 2021, pp. 254–277
ISSN 2575-1484 (print), ISSN 2575-1492 (online)

Semi-proximal Augmented Lagrangian-Based Decomposition Methods for Primal Block-Angular Convex Composite Quadratic Conic Programming Problems

Xin-Yee Lam,[a] Defeng Sun,[b] Kim-Chuan Toh[a,c]

[a] Department of Mathematics, National University of Singapore, Singapore 119076; [b] Department of Applied Mathematics, The Hong Kong Polytechnic University, Hung Hom, Hong Kong; [c] Institute of Operations Research and Analytics, National University of Singapore, Singapore 119076

Contact: mattohkc@math.nus.edu.sg, https://orcid.org/0000-0002-2348-6133 (X-YL); defeng.sun@polyu.edu.hk, https://orcid.org/0000-0003-0481-272X (DS); mattohkc@math.nus.edu.sg, https://orcid.org/0000-0001-7204-8933 (K-CT)

Received: December 12, 2018
Revised: December 30, 2019
Accepted: August 18, 2020
Published Online in Articles in Advance: March 11, 2021

https://doi.org/10.1287/ijoo.2019.0048

Abstract. We first propose a semi-proximal augmented Lagrangian-based decomposition method to directly solve the primal form of a convex composite quadratic conic-programming problem with a primal block-angular structure. Using our algorithmic framework, we are able to naturally derive several well-known augmented Lagrangian-based decomposition methods for stochastic programming, such as the diagonal quadratic approximation method of Mulvey and Ruszczyński. Although it is natural to develop an augmented Lagrangian decomposition algorithm based on the primal problem, here, we demonstrate that it is, in fact, numerically more economical to solve the dual problem by an appropriately designed decomposition algorithm. In particular, we propose a semi-proximal symmetric Gauss–Seidel-based alternating direction method of multipliers (sGS-ADMM) for solving the corresponding dual problem. Numerical results show that our dual-based sGS-ADMM algorithm can very efficiently solve some very large instances of primal block-angular convex quadratic-programming problems. For example, one instance with more than 300,000 linear constraints and 12.5 million nonnegative variables is solved to the accuracy of 10^{-5} in the relative KKT residual in less than a minute on a modest desktop computer.

Funding: D. Sun received financial support from The Hong Kong Polytechnic University [2017 Post-doctoral Fellowships Scheme]. K.-C. Toh received financial support from the Ministry of Education, Singapore, Academic Research Fund [Grant R-146-000-257-112].

Keywords: diagonal quadratic approximation method • symmetric Gauss–Seidel decomposition • ADMM • semi-proximal augmented Lagrangian method • primal block-angular structure

1. Introduction

In this paper, we focus on solving convex composite quadratic conic-programming problems with a primal block-angular structure. In particular, the convex objective function of the optimization problem is separable with respect to N blocks of variables, and there are N groups of constraints, each involving only a single distinct block of variables. At the same time, all the blocks of variables are connected by a group of linking linear constraints. Without specially designed strategies to exploit the underlying block-angular structure, a generic algorithm can be inefficient for solving such a problem because the constraints cannot be decomposed completely.

In practical applications, quadratic and linear problems with primal block-angular structure appear in many contexts, such as multicommodity flow problems (Assad 1978) and statistical disclosure control (Hundepool et al. 2012). These problems are often very large-scale in practice, and even highly practical interior-point methods, such as those implemented in Gurobi or Mosek, may not be efficient enough to solve them. In the literature, specialized algorithms designed to solve these problems have been studied extensively. Three of the most widely known algorithmic classes are (i) decomposition methods based on augmented Lagrangian and proximal-point algorithms (Ruszczyński 1986, 1989, Rockafellar and Wets 1991, Mulvey and Ruszczyński 1992, Ruszczyński 1995, 1999); (ii) interior-point log-barrier Lagrangian decomposition methods, such as those studied in Zhao (1999, 2001, 2005) and Mehrotra and Ozevin (2007, 2009); and (iii) standard interior-point methods that incorporate novel numerical linear algebraic techniques to exploit the underlying block-angular structure when solving the large linear systems arising at each iteration—for example, in Birge and Qi (1988), Choi and Goldfarb (1993), Gondzio et al. (1997), Schultz and Meyer (1991), Todd (1988), and Sun and Liu (2006), as well as Gondzio and Sarkissian (2003) and Gondzio and Grothey (2009).

Besides quadratic and linear problems, semidefinite programming (SDP) problems with primal block-angular structure are beginning to appear in the literature more frequently. Such problems are gaining more attention as researchers become more sophisticated in using SDP to model their application problems. For example, Hanasusanto and Kuhn (2018) reformulated a two-stage distributionally robust linear program as a completely positive cone program that has a block-angular structure and applied the reformulation to solve a multi-item newsvendor problem. Although linear-programming problems with primal block-angular structure have been studied extensively, the more complicated SDP version has yet to attract much attention. Apart from Mehrotra and Ozevin (2007), Sivaramakrishnan (2010), and Zhu and Ariyawansa (2011), we are not aware of other works.

By focusing on designing efficient algorithms for solving general conic-programming problems with primal block-angular structure, we can, in general, also use the same algorithmic framework to solve the primal block-angular linear-programming and quadratic-programming (QP) problems efficiently through designing novel numerical linear algebraic techniques to exploit the underlying structure. In this paper, our main objective is to design efficient, robust, and distributed algorithms for solving large-scale conic-programming problems with primal block-angular structure.

The existing approaches in the literature naturally inspired us to start by designing a decomposition-based algorithm to directly solve the primal problem. More specifically, we will design an inexact semi-proximal augmented Lagrangian method (spALM) for the primal problem, which attempts to exploit the block-angular structure to efficiently solve the spALM subproblem at each iteration in parallel. Our algorithm is motivated by the recent theoretical advances in the inexact semi-proximal augmented Lagrangian method that is developed in Chen et al. (2017), where one of the key advantages of the algorithm is that the subproblems can be solved approximately with progressive accuracy. Note that, following their terminology, "semi-proximal" means that the proximal term we use in the algorithm is a seminorm induced by a suitable positive semidefinite matrix. The primary motivation for using a positive semidefinite matrix rather than the usual positive definite one is to allow a more general choice of the proximal term. We will also elucidate the connection of our algorithm to the well-known diagonal quadratic approximation (DQA) algorithm of Mulvey and Ruszczyński (1992).

In the pioneering work of Kontogiorgis et al. (1996), an alternating direction method of multipliers (ADMM)-based framework was designed for a class of convex primal block-angular problems wherein the variables are duplicated, and auxiliary variables are introduced. This approach allowed the first ADMM subproblem at each iteration to be solvable in a distributed fashion. Additionally, the second ADMM subproblem is a simple quadratic program, which appears to be readily solvable at the first glance. However, the problem might still be difficult to solve if the size of the original problem is very large. To overcome the potential computational inefficiency induced by the extra variables and constraints, and also the relatively expensive step of having to solve a QP subproblem at each iteration in the primal approach, in this paper, we will solve (P) via its corresponding dual problem. Specifically, we will design and implement a semi-proximal symmetric Gauss–Seidel-based alternating direction method of multipliers (sGS-ADMM) to directly solve the dual problem, which will also solve the primal problem as a by-product. The advantage of tackling the dual problem directly is that no extra variables are introduced to decouple the constraints, and no coupled QP subproblems are needed to be solved at each iteration. We should mention that the sGS-ADMM we proposed in this paper is designed based on the abstract framework developed in Chen et al. (2017).

We consider the following primal block-angular optimization problem:

$$
\begin{aligned}
\text{(P)}\quad \min\ & \sum_{i=0}^{N} f_i(x_i) := \sum_{i=0}^{N} [\theta_i(x_i) + \tfrac{1}{2}\langle x_i, \mathcal{Q}_i x_i\rangle + \langle c_i, x_i\rangle] \\
\text{s.t.}\ & \underbrace{\begin{bmatrix} \mathcal{A}_0 & \mathcal{A}_1 & \cdots & \mathcal{A}_N \\ 0 & \mathcal{D}_1 & 0 & \vdots \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & 0 & \mathcal{D}_N \end{bmatrix}}_{\mathcal{B}} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_N \end{bmatrix}, \\
& x_i \in \mathcal{K}_i,\ i = 0, 1, \ldots, N,
\end{aligned}
$$

where for each $i = 0, 1, \ldots, N$; $\theta_i : \mathcal{X}_i \to (-\infty, \infty]$ is a proper closed convex function; $\mathcal{Q}_i : \mathcal{X}_i \to \mathcal{X}_i$ is a positive semidefinite linear operator; $\mathcal{A}_i : \mathcal{X}_i \to \mathcal{Y}_0$ and $\mathcal{D}_i : \mathcal{X}_i \to \mathcal{Y}_i$ are given linear maps; $c_i \in \mathcal{X}_i$ and $b_i \in \mathcal{Y}_i$ are given

data; $\mathcal{K}_i \subset \mathcal{X}_i$ is a closed convex set that is typically a cone, but not necessarily so; and $\mathcal{X}_i, \mathcal{Y}_i$ are real finite dimensional Euclidean spaces, each equipped with an inner product $\langle \cdot, \cdot \rangle$ and its induced norm $\|\cdot\|$. We should mention that the addition of the proper closed convex functions in the objective can give us the flexibility to handle nonlinear terms, or even nonsmooth terms, such as ℓ_1 regularization terms. In the setting of a doubly nonnegative SDP where x_i belongs to an SDP cone ($\mathbb{S}^{n_i}_+$), and at the same time also required to be nonnegative, one can handle the intersection of these conic constraints by setting $\theta_i(x_i) = \delta_{(\mathbb{R}^{n_i \times n_i}_+)}(x_i)$ and $\mathcal{K}_i = \mathbb{S}^{n_i}_+$. Here, for a given convex set K in an Euclidean space, δ_K is the indicator function defined by:

$$\delta_K(x) = \begin{cases} 0, & \text{if } x \in K, \\ +\infty, & \text{otherwise.} \end{cases}$$

We should also mention that a constraint of the form $b_i - \mathcal{D}_i x_i \in \mathcal{C}_i$, where $\mathcal{C}_i$ is a closed convex set can be put in the form in (P) by introducing a slack variable s_i, so that $[\mathcal{D}_i, I](x_i; s_i) = b_i$ and $(x_i; s_i) \in \mathcal{K}_i \times \mathcal{C}_i$.

We assume that the constraint matrix $\mathcal{B}$ in (P) has full row-rank. Let $n_i = \dim(\mathcal{X}_i)$ and $m_i = \dim(\mathcal{Y}_i)$, then (P) has $\sum_{i=0}^N m_i$ linear constraints and $\sum_{i=0}^N n_i$ decision variables. Thus, even if m_i and/or n_i are moderate numbers, the overall dimension of the problem can grow rapidly when N is large.

In the important special case of a block-angular linear-programming problem for which $\mathcal{Q}_i = 0$ and $\theta_i = 0$ for all $i = 0, \ldots, N$, the Dantzig–Wolfe decomposition method is a well-known classical approach for solving such a problem. The method may be viewed as a dual approach based on the partial Lagrangian function $\sum_{i=0}^N \langle c_i, x_i \rangle - \langle u, b_0 - \sum_{i=0}^N \mathcal{A}_i x_i \rangle$, and has the attractive property that at each iteration, x_i can be computed individually from a smaller linear program (LP), for $i = 0, \ldots, N$. However, it is generally acknowledged that an augmented Lagrangian approach has a number of important advantages over the Lagrangian dual method. For example, Ruszczyński (1995) stated that the dual approach based on the ordinary Lagrangian could suffer from the nonuniqueness of solutions of the subproblems. In addition, it is numerically easier to solve the subproblems of the augmented Lagrangian approach. In that paper, the well-known diagonal quadratic approximation method is introduced. Given that DQA is a very successful decomposition method, and that it has been a popular tool in stochastic programming, it is, thus, worthwhile for us to analyze its connection to the spALM algorithm we have proposed to see whether further enhancements are possible.

To summarize, our first contribution is in proposing several variants of augmented Lagrangian-based algorithms for directly solving the primal form (P) of the convex composite quadratic conic-programming problem with a primal block-angular structure. We also show that they can be considered as generalizations of the well-known DQA method. Although primal-based algorithms appear to be the most natural ones for solving the problem (P), we demonstrate in this paper that a suitably designed dual-based algorithm can be more efficient. In fact, we are not aware of a systematic comparison of the performance of primal-based algorithms versus dual-based algorithms, even for the case of linear primal block-angular problems. Our second contribution is in the design and implementation of a specialized algorithm (semi-proximal sGS-ADMM) for solving the dual problem of (P). We also develop essential numerical procedures for the efficient implementation of the algorithm. The algorithm is easy to implement and is highly amenable to parallelization. Hence, we expect it to be highly scalable for solving large-scale problems with millions of variables and constraints. Finally, we have conducted comprehensive numerical experiments to evaluate the performance of our algorithms for solving (P) and its dual problem against some other highly competitive state-of-the-art solvers, such as the BlockIP solver in Castro (2016).

This paper is organized as follows. We derive the dual of the primal block-angular problem (P) in Section 2. In Section 3, we present our inexact semi-proximal augmented Lagrangian methods for solving the primal problem (P). We also elucidate the connection of a certain variant of the spALM to the well-known DQA algorithm. In Section 4, we propose a semi-proximal sGS-ADMM for solving the dual problem of (P) and describe its efficient implementation. For all the algorithms we introduce, we conduct comprehensive numerical experiments to evaluate their performance, and the numerical results are reported in Sections 3.3 and 5. We conclude the paper in the final section.

1.1. Notation

We denote $[P; Q]$ or $(P; Q)$ as the matrix obtained by appending the matrix Q to the last row of the matrix P, whereas we denote $[P, Q]$ or (P, Q) as the matrix obtained by appending Q to the last column of matrix P, assuming that they have the same number of columns or rows, respectively. We also use the same notation symbolically for P and Q, which are linear maps with compatible domains and codomains.

For any linear map $\mathcal{T}:\mathcal{X}\to\mathcal{Y}$, we denote its adjoint as $\mathcal{T}^*$. If $\mathcal{X}=\mathcal{Y}$, and $\mathcal{T}$ is self-adjoint and positive semidefinite, then, for any $x\in\mathcal{X}$, we use the notation $\|x\|_{\mathcal{T}}:=\sqrt{\langle x,\mathcal{T}x\rangle}$. Note that $\|\cdot\|_{\mathcal{T}}$ is merely a seminorm if $\mathcal{T}$ is not positive definite.

Let $f:\mathcal{X}\to(-\infty,+\infty]$ be an arbitrary closed proper convex function. We denote $\operatorname{dom} f$ as its effective domain and ∂f as its subdifferential mapping. The Fenchel conjugate function of f is denoted as f^*.

The Moreau–Yosida proximal mapping of f is defined by $\operatorname{Prox}_f(y):=\operatorname{argmin}_x\{f(x)+\frac{1}{2}\|x-y\|^2\}$.

$\operatorname{diag}(M)$ denotes the vector extracted from the diagonal entries of a matrix M, whereas $\operatorname{diag}(M_1,M_2,\ldots,M_k)$ denotes the block diagonal matrix constructed from given square matrices $M_1,M_2,\ldots,M_k$.

$\Pi_{\mathcal{K}}(\cdot)$ is the projection operator onto the convex set $\mathcal{K}$, which is defined by

$$\prod_{\mathcal{K}}(x):=\operatorname*{argmin}_y\left\{\frac{1}{2}\|x-y\|^2\mid y\in\mathcal{K}\right\}.$$

2. Derivation of the Dual of (P)

For notational convenience, we define

$$\boldsymbol{\mathcal{X}}=\mathcal{X}_0\times\mathcal{X}_1\times\cdots\times\mathcal{X}_N,\quad \boldsymbol{\mathcal{Y}}=\mathcal{Y}_0\times\mathcal{Y}_1\times\cdots\times\mathcal{Y}_N,\quad \boldsymbol{\mathcal{K}}=\mathcal{K}_0\times\mathcal{K}_1\times\cdots\times\mathcal{K}_N. \tag{1}$$

Every $x\in\boldsymbol{\mathcal{X}}$, $y\in\boldsymbol{\mathcal{Y}}$, $c\in\boldsymbol{\mathcal{X}}$, and $b\in\boldsymbol{\mathcal{Y}}$ can be expressed as

$$x=(x_0;x_1;\cdots;x_N),\quad y=(y_0;y_1;\cdots;y_N),\quad c=(c_0;c_1;\cdots;c_N),\quad b=(b_0;b_1;\cdots;b_N). \tag{2}$$

We also define $\mathcal{A}$, $\mathcal{Q}$, and $\boldsymbol{\theta}$ as follows:

$$\mathcal{A}=[\mathcal{A}_0,\mathcal{A}_1,\ldots,\mathcal{A}_N],\quad \mathcal{Q}=\operatorname{diag}(\mathcal{Q}_0,\mathcal{Q}_1,\ldots,\mathcal{Q}_N),\quad \boldsymbol{\theta}(x)=\sum_{i=0}^{N}\theta_i(x_i). \tag{3}$$

Using the notation in (1)–(3), we can write (P) compactly in the form of a general convex composite quadratic conic-programming problem:

$$\min\left\{\boldsymbol{\theta}(x)+\frac{1}{2}\langle x,\mathcal{Q}x\rangle+\langle c,x\rangle\mid \mathcal{B}x-b=0,\ x\in\boldsymbol{\mathcal{K}}\right\}. \tag{4}$$

In order to derive its dual problem, we introduce two auxiliary variables $u,v\in\boldsymbol{\mathcal{X}}$, to decouple the variables for the smooth part of the objective function (the quadratic-linear term $\frac{1}{2}\langle x,\mathcal{Q}x\rangle+\langle c,x\rangle$) from the possibly nonsmooth term $\boldsymbol{\theta}$, and the constraint "$x\in\boldsymbol{\mathcal{K}}$." By doing so, Problem (4) can equivalently be written as follows:

$$\begin{array}{ll}\min & \boldsymbol{\theta}(u)+\frac{1}{2}\langle x,\mathcal{Q}x\rangle+\langle c,x\rangle+\delta_{\boldsymbol{\mathcal{K}}}(v)\\ \text{s.t.} & \mathcal{B}x-b=0,\ u-x=0,\ v-x=0.\end{array} \tag{5}$$

Consider the following Lagrangian function for (5):

$$\begin{aligned}\mathcal{L}(x,u,v;y,s,z)&=\boldsymbol{\theta}(u)+\frac{1}{2}\langle x,\mathcal{Q}x\rangle+\langle c,x\rangle+\delta_{\boldsymbol{\mathcal{K}}}(v)-\langle y,\mathcal{B}x-b\rangle-\langle s,x-u\rangle-\langle z,x-v\rangle\\&=\frac{1}{2}\langle x,\mathcal{Q}x\rangle+\langle c-\mathcal{B}^*y-s-z,x\rangle+\boldsymbol{\theta}(u)+\langle s,u\rangle+\delta_{\boldsymbol{\mathcal{K}}}(v)+\langle z,v\rangle+\langle y,b\rangle,\end{aligned}$$

where $x,u,v,s,z\in\boldsymbol{\mathcal{X}},y\in\boldsymbol{\mathcal{Y}}$. Then, the dual of (5) is given by

$$\begin{aligned}&\max_{y,s,z}\inf_{x,u,v}\mathcal{L}(x,u,v;y,s,z)\\&\quad=\max_{y,s,z}\left\{\inf_x\left[\frac{1}{2}\langle x,\mathcal{Q}x\rangle+\langle c-\mathcal{B}^*y-s-z,x\rangle\right]+\inf_u[\boldsymbol{\theta}(u)+\langle s,u\rangle]+\inf_v[\delta_{\boldsymbol{\mathcal{K}}}(v)+\langle z,v\rangle]+\langle y,b\rangle\right\}.\end{aligned}$$

Note that in the above, we used "inf" because for some given dual variables y,s,z, the Lagrangian function $\mathcal{L}(x,u,v;y,s,z)$ may approach $-\infty$ when we take infimum over the primal variables x,u,v.

To obtain the dual problem more explicitly, we consider the inner infimum problems with respect to the variables x, u, and v below. From the definition of Fenchel conjugate functions, we have

$$\inf_u[\boldsymbol{\theta}(u)+\langle s,u\rangle]=-\boldsymbol{\theta}^*(-s);\quad \inf_v[\delta_{\boldsymbol{\mathcal{K}}}(v)+\langle z,v\rangle]=-\delta^*_{\boldsymbol{\mathcal{K}}}(-z).$$

Now, for a given subspace $\mathcal{W} \subseteq \mathcal{X}$ containing Range($\mathcal{Q}$), the range space of $\mathcal{Q}$, we have

$$\inf_x \left\{ \frac{1}{2}\langle x, \mathcal{Q}x\rangle + \langle c - \mathcal{B}^* y - s - z, x\rangle \right\} = \begin{cases} -\frac{1}{2}\langle w, \mathcal{Q}w\rangle, & \text{if } c - \mathcal{B}^* y - s - z = -\mathcal{Q}w \text{ for some } w \in \mathcal{W}, \\ -\infty, & \text{otherwise.} \end{cases}$$

We should emphasize that the result above is invariant to the choice of $\mathcal{W}$ as long as it contains Range($\mathcal{Q}$). Specifically, for any solution w' to the optimality condition $\mathcal{Q}x = -(c - \mathcal{B}^* y - s - z)$, one can express it as $w' = w + w^{\perp}$, with $w \in \mathcal{W}$ and $w^{\perp} \in \mathcal{W}^{\perp}$ (the orthogonal complement of $\mathcal{W}$). Now, as long as Range($\mathcal{Q}$) $\subset \mathcal{W}$, we have $\mathcal{W}^{\perp} \subset$ Null($\mathcal{Q}$), and hence $\langle w', \mathcal{Q}w'\rangle = \langle w, \mathcal{Q}w\rangle$. The purpose of having the flexibility of choosing $\mathcal{W}$ is for the convergence of the sGS-ADMM algorithm that we will design later for solving the dual problem. More specifically, because $\mathcal{Q}$ is only assumed to be positive semidefinite, a certain linear system involving $\mathcal{Q}$ in the sGS-ADMM algorithm might be singular, and the sequence of iterates generated by the algorithm may be unbounded. But by choosing the subspace $\mathcal{W} =$ Range($\mathcal{Q}$), we can remove the issues of singularity and unboundedness; see Section 4.2.2 for more details.

With the simplification in the last paragraph, the dual problem of (5) (and, hence, (P)) is given more explicitly as follows:

$$\begin{aligned} -\min \quad & \theta^*(-s) + \tfrac{1}{2}\langle w, \mathcal{Q}w\rangle - \langle b, y\rangle + \delta^*_{\mathcal{K}}(-z) \\ \text{s.t.} \quad & -\mathcal{Q}w + \mathcal{B}^* y + s + z = c, \\ & s \in \mathcal{X},\ y \in \mathcal{Y},\ w \in \mathcal{W}. \end{aligned} \tag{6}$$

Assuming that both the primal and dual problems satisfy the (generalized) Slater's condition, then the optimal solutions for both problems exist, and they satisfy the following Karush–Kuhn–Tucker (KKT) optimality conditions (Karush 1939, Kuhn and Tucker 1951):

$$\begin{cases} \mathcal{B}x - b = 0, \\ -\mathcal{Q}w + \mathcal{B}^* y + s + z - c = 0, \quad \mathcal{Q}w - \mathcal{Q}x = 0, \quad w \in \mathcal{W}, \\ -s \in \partial\theta(x) \ \Leftrightarrow\ x - \mathrm{Prox}_{\theta}(x - s) = 0, \\ x - \Pi_{\mathcal{K}}(x - z) = 0. \end{cases} \tag{7}$$

It is not difficult to check that for all $z = (z_0; z_1; \ldots; z_N)$, $s = (s_0; s_1; \ldots; s_N) \in \mathcal{X}$, we have

$$\delta^*_{\mathcal{K}}(-z) = \sum_{i=0}^{N} \delta^*_{\mathcal{K}_i}(-z_i), \quad \theta^*(-s) = \sum_{i=0}^{N} \theta^*_i(-s_i). \tag{8}$$

By applying the structure in (1)–(3) and (8) to (6), we obtain explicitly the dual of (P):

$$\begin{aligned} \text{(D)} \quad & -\min \sum_{i=0}^{N} \left[\theta^*_i(-s_i) + \delta^*_{\mathcal{K}_i}(-z_i) + \frac{1}{2}\langle w_i, \mathcal{Q}_i w_i\rangle - \langle b_i, y_i\rangle \right] \\ & \text{s.t.} \begin{bmatrix} \mathcal{A}_0^* \\ \mathcal{A}_1^* \\ \vdots \\ \mathcal{A}_N^* \end{bmatrix} y_0 + \begin{bmatrix} -\mathcal{Q}_0 w_0 + s_0 + z_0 \\ \mathcal{D}_1^* y_1 - \mathcal{Q}_1 w_1 + s_1 + z_1 \\ \vdots \\ \mathcal{D}_N^* y_N - \mathcal{Q}_N w_N + s_N + z_N \end{bmatrix} = c, \\ & w_i \in \mathcal{W}_i,\ i = 0, 1, \ldots, N, \end{aligned} \tag{9}$$

where for each $i = 0, 1, \ldots, N$, $\mathcal{W}_i \subset \mathcal{X}_i$ is a given subspace containing Range($\mathcal{Q}_i$).

3. Inexact Semi-proximal Augmented Lagrangian Methods for the Primal Problem (P)

Following the existing approaches in the literature, such as in Mulvey and Ruszczyński (1992), Rockafellar and Wets (1991), and Ruszczyński (1986, 1989, 1995, 1999), it is natural for us to first attempt to design augmented Lagrangian-based decomposition algorithms for the primal problem (P). To do so, first we rewrite (P) in the following form:

$$\min \left\{ \sum_{i=0}^{N} \left[f_i(x_i) + \delta_{F_i}(x_i) \right] \ \middle|\ \mathcal{A}x = b_0,\ x = (x_0; x_1; \ldots; x_N) \in \mathcal{X} \right\}, \tag{10}$$

where $F_0 = \mathcal{K}_0$, and $F_i = \{x_i \in \mathcal{X}_i \mid \mathcal{D}_i x_i = b_i, x_i \in \mathcal{K}_i\}$, $i = 1, \ldots, N$. For a given parameter $\sigma > 0$, we consider the following augmented Lagrangian function associated with (10):

$$L_\sigma(x; y_0) = \sum_{i=0}^{N} \big[f_i(x_i) + \delta_{F_i}(x_i) \big] + \frac{\sigma}{2} \big\| \mathcal{A}x - b_0 - \sigma^{-1} y_0 \big\|^2 - \frac{1}{2\sigma} \|y_0\|^2. \tag{11}$$

The augmented Lagrangian method for solving (10) has the following template.

ALM. Given $\sigma > 0$ and $y_0^0 \in \mathcal{Y}_0$. Perform the following steps at the k-th iteration for $k = 0, 1, \ldots$

Step 1. $x^{k+1} \approx \operatorname{argmin}\{L_\sigma(x; y_0^k) \mid x \in \mathcal{X}\}$

Step 2. $y_0^{k+1} = y_0^k + \tau\sigma(b_0 - \mathcal{A}x^{k+1})$, where $\tau \in (0, 2)$ is the step length (Bertsekas 1975).

Because of the presence of the term $\frac{\sigma}{2} \|\mathcal{A}x - b_0 - \sigma^{-1} y_0\|^2$ in L_σ, the subproblem in Step 1 of the ALM is not decomposable. This is an undesirable feature of the method, in which it destroys the computationally attractive separable structure in the Dantzig–Wolfe decomposition method.

Here, we propose to add a semi-proximal term to the augmented Lagrangian function to overcome the difficulty of nonseparability. In this case, the function $L_\sigma(x; y_0^k)$ in Step 1 of ALM is majorized by an additional semi-proximal term at the point x^k—that is, instead of the problem in Step 1 of ALM, we solve

$$x^{k+1} \approx \operatorname*{argmin}_x \Big\{ L_\sigma(x; y_0^k) + \frac{\sigma}{2} \|x - x^k\|_{\mathcal{T}}^2 \Big\},$$

where $\mathcal{T}$ is a given positive semidefinite self-adjoint linear operator, which should be chosen appropriately to decompose the computation of the x_i's in Step 1 of ALM, while at the same time, the added proximal term should be as small as possible in order not to perturb $L_\sigma(x; y_0^k)$ by too much. In this paper, we choose $\mathcal{T}$ to be the following positive semidefinite linear operator:

$$\mathcal{T} = \operatorname{diag}(\mathcal{J}_0, \ldots, \mathcal{J}_N) - \mathcal{A}^* \mathcal{A}, \tag{12}$$

where $\mathcal{J}_i = \beta_i I + \mathcal{A}_i^* \mathcal{A}_i$, with $\beta_i = \sum_{j=0, j \neq i}^{N} \|\mathcal{A}_i^* \mathcal{A}_j\|_2$, for each $i = 0, 1, \ldots, N$, and $\|\cdot\|_2$ denotes the spectral norm. The special choice of $\mathcal{T}$ in (12) is especially a good one when $\mathcal{A}_i$ and $\mathcal{A}_j$ are nearly orthogonal to one another for most of the index pairs (i, j)—in which case, $\|\mathcal{A}_i^* \mathcal{A}_j\|_2$ will take on a small value relative to $\|\mathcal{A}_i\|_2 + \|\mathcal{A}_j\|_2$, and, hence, β_i will also be a small constant.

With the choice in (12), we can decompose the computation of x^{k+1} into the computation of its individual blocks by using the following equality:

$$\begin{aligned}
L_\sigma(x; y_0^k) + \frac{\sigma}{2} \|x - x^k\|_{\mathcal{T}}^2 &= \sum_{i=0}^{N} \big(f_i(x_i) + \delta_{F_i}(x_i) \big) + \frac{\sigma}{2} \big\| \mathcal{A}x - b_0 - \sigma^{-1} y_0 \big\|^2 - \frac{1}{2\sigma} \|y_0\|^2 + \frac{\sigma}{2} \|x - x^k\|_{\mathcal{T}}^2 \\
&= \sum_{i=0}^{N} \big(f_i(x_i) + \delta_{F_i}(x_i) \big) - \frac{1}{2\sigma} \|y_0\|^2 \\
&\quad + \frac{\sigma}{2} \Big[\big\langle x, (\mathcal{A}^* \mathcal{A} + \mathcal{T}) x \big\rangle - 2 \big\langle x, \mathcal{A}^* (b_0 + \sigma^{-1} y_0) + \mathcal{T} x^k \big\rangle + \big\| b_0 + \sigma^{-1} y_0 \big\|^2 + \|x^k\|_{\mathcal{T}} \Big] \\
&= \sum_{i=0}^{N} \Big(f_i(x_i) + \delta_{F_i}(x_i) + \frac{\sigma}{2} \Big[\langle x_i, \mathcal{J}_i x_i \rangle - 2 \big\langle x_i, \mathcal{A}_i^* \big(b_0 + \sigma^{-1} y_0 - \mathcal{A} x^k \big) + \mathcal{J}_i x_i^k \big\rangle \Big] \Big) \\
&\quad + \frac{\sigma}{2} \Big[\big\| b_0 + \sigma^{-1} y_0 \big\|^2 + \|x^k\|_{\mathcal{T}} \Big] - \frac{1}{2\sigma} \|y_0\|^2.
\end{aligned} \tag{13}$$

Based on the above equality, the inexact semi-proximal ALM (spALM) for solving the primal block-angular problem (P) through the formulation in (10) is given as follows.

spALM. Given $\sigma > 0$ and $y_0^0 \in \mathcal{Y}_0$. Let $\{\varepsilon_k\}$ be a given summable sequence of nonnegative numbers. Perform the following steps at the kth iteration for $k = 0, 1, \ldots$

Step 1. Compute

$$x^{k+1} \approx \widehat{x}^{k+1} := \operatorname*{argmin}_{x \in \mathcal{X}} \Big\{ L_\sigma(x; y_0^k) + \frac{\sigma}{2} \|x - x^k\|_{\mathcal{T}}^2 \Big\}, \tag{14}$$

with residual

$$d^{k+1} \in \partial_x L_\sigma(x^{k+1}; y_0^k) + \sigma \mathcal{T}(x^{k+1} - x^k), \tag{15}$$

satisfying $\|d^{k+1}\| \le \varepsilon_k$. Let $\mathcal{G}_i := \mathcal{Q}_i + \sigma\mathcal{J}_i$, $\alpha_i^k := \mathcal{Q}_i x_i^k + c_i + \sigma\mathcal{A}_i^*(\mathcal{A}x^k - b_0 - \sigma^{-1}y_0^k) - \mathcal{G}_i x_i^k$. Because of the separability of the variables in (14) because of the specially chosen $\mathcal{T}$, one can compute **in parallel** for $i = 0, 1, \ldots, N$,

$$x_i^{k+1} \approx \widetilde{x}_i^{k+1} := \operatorname*{argmin}_{x_i}\left\{\theta_i(x_i) + \frac{1}{2}\langle x_i, \mathcal{G}_i x_i\rangle + \langle \alpha_i^k, x_i\rangle \mid x_i \in F_i\right\}, \tag{16}$$

with the residual $d_i^{k+1} := v_i^{k+1} + \mathcal{G}_i x_i^{k+1} + \alpha_i^k$ for some $v_i^{k+1} \in \partial(\theta_i + \delta_{F_i})(x_i^{k+1})$ and satisfying

$$\left\|d_i^{k+1}\right\| \le \frac{1}{\sqrt{N+1}}\varepsilon_k. \tag{17}$$

Step 2. $y_0^{k+1} = y_0^k + \tau\sigma(b_0 - \mathcal{A}x^{k+1})$, where $\tau \in (0, 2)$ is the steplength.

Observe that with the addition of the semi-proximal term $\frac{\sigma}{2}\|x - x^k\|_{\mathcal{T}}^2$ to the augmented Lagrangian function in Step 1 of spALM, we have decomposed the large coupled problem involving x in ALM into $N+1$ smaller independent problems that can be solved in parallel. For the case of a quadratic or linear program, we can employ a powerful solver such as Gurobi or Mosek to efficiently solve these smaller problems.

In order to judge how accurately the decomposed subproblems in Step 1 must be solved, we need to analyze the stopping condition for (16) in detail—that is, we need to evaluate d_i^{k+1} by estimating v_i^{k+1} for $i = 0, 1, \ldots, N$. This can be done by considering the dual of the Subproblem (16), which can be written as:

$$\begin{aligned} -\min \quad & \theta_i^*(-s_i) + \tfrac{1}{2}\langle x_i, \mathcal{G}_i x_i\rangle - \langle b_i, y_i\rangle + \delta_{\mathcal{K}_i}^*(-z_i) \\ \text{s.t.} \quad & -\mathcal{G}_i x_i + \mathcal{D}_i^* y_i + s_i + z_i = \alpha_i^k, \\ & s_i \in \mathcal{X}_i,\ \ y_i \in \mathcal{Y}_i,\ \ x_i \in \mathcal{X}_i, \quad i = 1, \ldots, N. \end{aligned} \tag{18}$$

Note that for $i = 0$, we have a similar problem as the above, but the terms involving y_i are absent. For the discussion below, we will just focus on the case where $i = 1, \ldots, N$; the case for $i = 0$ can be derived similarly. We can estimate v_i^{k+1} by setting

$$v_i^{k+1} = -s_i^{k+1} - \mathcal{D}_i^* y_i^{k+1} - z_i^{k+1} \in \partial\big(\theta_i + \delta_{F_i}\big)\big(x_i^{k+1}\big),$$

for a computed dual solution $(x_i^{k+1}, y_i^{k+1}, s_i^{k+1}, z_i^{k+1})$ of (18) at the $(k+1)$-th iteration satisfying $-s_i^{k+1} \in \partial\theta_i(x_i^{k+1})$ and $-z_i^{k+1} \in \partial\delta_{\mathcal{K}_i}(x_i^{k+1})$. (Note that one can show that $-\mathcal{D}_i^* y_i^{k+1} - z_i^{k+1} \in \partial\delta_{F_i}(x_i^{k+1})$.) From here, it follows that $d_i^{k+1} = v_i^{k+1} + \mathcal{G}_i x_i^{k+1} + \alpha_i^k$ is the residual of the equality constraint in the Dual Problem (18).

Remark 1. In the spALM algorithm, some of the dual variables for (D) are not explicitly constructed. Here, we describe how they can be estimated. Recall that for (D), we want to obtain

$$-\mathcal{Q}_i x_i + \mathcal{A}_i^* y_0 + \mathcal{D}_i^* y_i + s_i + z_i - c_i = 0 \quad \forall\, i = 0, 1, \ldots, N.$$

Note that for convenience, we introduced $\mathcal{D}_0^* = 0$. From the KKT conditions for (16) and (18), we have the following relation for a computed solution $(x_i^{k+1}, y_i^{k+1}, s_i^{k+1}, z_i^{k+1})$ of (18):

$$-\mathcal{G}_i x_i^{k+1} + \mathcal{D}_i^* y_i^{k+1} + s_i^{k+1} + z_i^{k+1} - \alpha_i^k =: R_i^d \approx 0.$$

By using the expression for $\mathcal{G}_i$, α_i^k, and y_0^{k+1}, we obtain that

$$\begin{aligned} & -\mathcal{Q}_i x_i^{k+1} + \mathcal{A}_i^* y_0^{k+1} + \mathcal{D}_i^* y_i^{k+1} + s_i^{k+1} + z_i^{k+1} - c_i \\ & = R_i^d + \sigma\mathcal{J}_i\big(x_i^{k+1} - x_i^k\big) + \sigma\mathcal{A}_i^*\mathcal{A}\big(x^k - x^{k+1}\big) + (\tau - 1)\sigma\mathcal{A}_i^*\big(b_0 - \mathcal{A}x^{k+1}\big). \end{aligned}$$

Note that the right-hand-side quantity in the above equation will converge to zero based on the convergence of spALM and the KKT conditions for (16) and (18). Thus, by using the dual variables computed from solving (18), we can generate the dual variables for (D), where we have estimated w_i^{k+1} for (D) by setting it to be x_i^{k+1}.

3.1. Convergence of the Inexact spALM

The convergence of the inexact spALM for solving (10) can be established readily by using known results in Chen et al. (2019). To do that, we need to first reformulate (10) into the form required in Chen et al. (2019) as follows:

$$\min\{h(x) + \psi(x) \mid \mathcal{A}x = b_0,\ x = (x_0; x_1; \ldots; x_N) \in \boldsymbol{\mathcal{X}}\}, \tag{19}$$

where $h(x) = \sum_{i=0}^{N}[\frac{1}{2}\langle x_i, \mathcal{Q}_i x_i\rangle + \langle c_i, x_i\rangle]$ and $\psi(x) = \sum_{i=0}^{N}[\theta_i(x_i) + \delta_{F_i}(x_i)]$. Its corresponding KKT residual mapping is given by

$$\mathcal{R}(x, y_0) = \begin{pmatrix} b_0 - \mathcal{A}x \\ x - \mathrm{Prox}_{\psi}\big(x - \mathcal{Q}x - c - \mathcal{A}^* y_0\big) \end{pmatrix} \quad \forall\, x \in \boldsymbol{\mathcal{X}},\ y_0 \in \mathcal{Y}_0.$$

Then (x, y_0) is a solution of the KKT system of (19) if and only if $\mathcal{R}(x, y_0) = 0$.

Next we state the global convergence theorem of the inexact spALM for the convenience of the reader. Define the self-adjoint positive definite linear operator $\mathcal{V} : \boldsymbol{\mathcal{X}} \to \boldsymbol{\mathcal{X}}$ by

$$\mathcal{V} := \tau\sigma\left(\mathcal{Q} + \sigma\mathcal{T} + \frac{2-\tau}{6}\sigma\mathcal{A}\mathcal{A}^*\right),$$

where $\tau \in (0, 2)$ is the steplength in the spALM. The global convergence of the inexact spALM is given in the following theorem.

Theorem 1. *Assume that the solution set to the KKT system of* (10) *is nonempty and* $(\overline{x}, \overline{y}_0)$ *is a solution. Then, the sequence* $\{(x^k, y_0^k)\}$ *generated by spALM is well-defined such that for any* $k \geq 1$,

$$\big\|x^{k+1} - \widehat{x}^{k+1}\big\|^2_{\mathcal{Q}+\sigma\mathcal{T}+\sigma\mathcal{A}\mathcal{A}^*} \leq \big\langle d^{k+1}, x^{k+1} - \widehat{x}^{k+1}\big\rangle,$$

and for all $k = 0, 1, \ldots,$

$$\Big(\big\|x^{k+1} - \overline{x}\big\|^2_{\widehat{\mathcal{V}}} + \big\|y_0^{k+1} - \overline{y}_0\big\|^2\Big) - \Big(\big\|x^{k} - \overline{x}\big\|^2_{\widehat{\mathcal{V}}} + \big\|y_0^{k} - \overline{y}_0\big\|^2\Big) \leq -\left(\frac{2-\tau}{3\tau}\big\|y_0^k - y_0^{k+1}\big\|^2 + \big\|x^{k+1} - x^k\big\|^2_{\mathcal{V}} - 2\tau\sigma\langle d^k, x^{k+1} - \overline{x}\rangle\right),$$

where $\widehat{\mathcal{V}} = \mathcal{V} + \frac{2-\tau}{6}\sigma\mathcal{A}\mathcal{A}^*$. *Moreover, the sequence* $\{(x^k, y_0^k)\}$ *converges to a solution to the KKT system of* (10).

Proof. Our algorithm can be viewed as a special case of the inexact majorized indefinite-proximal ALM algorithm proposed in Chen et al. (2019). Thus, the result can be proved directly from the convergence result in Chen et al. (2019, theorem 1). □

The local linear convergence of spALM can also be established if the KKT residual mapping $\mathcal{R}$ satisfies the following error bound condition: There exist positive constants κ and r such that $\mathrm{dist}((x, y_0), \Omega) \leq \kappa\|\mathcal{R}(x, y_0)\|$ for all (x, y_0) satisfying $\|(x, y_0) - (x^*, y_0^*)\| \leq r$, where Ω is the solution set of (19) and (x^*, y_0^*) is a particular solution of (19). In order to save some space, we will not state the theorem here, but refer the reader to Chen et al. (2019, theorem 2).

3.2. Derivation of a New Diagonal Quadratic Approximation Method and Its Connection to the DQA Method of Ruszczyński

In Ruszczyński (1989) and Mulvey and Ruszczyński (1992), the diagonal quadratic approximation augmented Lagrangian method was proposed to solve a subclass of problems of the form (P). As already mentioned, DQA is a very successful decomposition method that is frequently used in stochastic programming.

Given the success of the DQA method, it is natural for us to look for an extension of the method to solve the more general problem (P). Here, our goal is to derive a new variant of the DQA method based on algorithm ALM. For the ease of reference, we call this new variant ALM-DQA-mod. The idea behind the derivation of the new method is that, instead of adding a proximal term to the augmented Lagrangian function $L_\sigma(x; y_0^k)$ in Step 1 of ALM to decompose the computation of x^{k+1}, we apply a proximal gradient method to solve the problem, $\min\{L_\sigma(x; y_0^k) \mid x \in \boldsymbol{\mathcal{X}}\}$, in Step 1 directly. In the proximal gradient method, we add a suitable proximal term to decompose the inner computation of the variable x. Specifically, starting from the initial point $\hat{x}^0 = x^k$, the sth step of the proximal gradient method solves the following subproblem to update $\hat{x}^s$—that is,

$$\hat{x}^{s+1} \approx \mathrm{argmin}\Big\{L_\sigma\big(x; y_0^k\big) + \frac{\sigma}{2}\|x - \hat{x}^s\|^2_{\widehat{\mathcal{T}}} \mid x \in \boldsymbol{\mathcal{X}}\Big\}, \quad s = 0, 1, \ldots,$$

where the proximal term $\frac{\sigma}{2}\|x - \hat{x}^s\|^2_{\widehat{\mathcal{T}}}$ is added to decompose the computation of $\hat{x}^{s+1}$. In the above, the linear operator $\widehat{\mathcal{T}}$ is given by

$$\widehat{\mathcal{T}} = \mathrm{diag}(\mathcal{E}_0, \ldots, \mathcal{E}_N) - \mathcal{A}^*\mathcal{A}, \tag{21}$$

where $\mathcal{E}_i = (N+1)\mathcal{A}_i^*\mathcal{A}_i$ for all $i = 0, 1, \ldots, N$. Note that it is not difficult to show that $\widehat{\mathcal{T}} \succeq 0$.

By expanding the function $L_\sigma(x;y_0^k)+\frac{\sigma}{2}\|x-\hat{x}^s\|^2_{\widehat{\mathcal{T}}}$ as a separable function of the blocks x_i for $i=0,1,\ldots,N$ as in (13), we see that the computation of $\hat{x}_i^{s+1}$ can be done in a separable manner. The details of our ALM-DQA-mod algorithm are given as follows.

ALM-DQA-mod. Given $\sigma>0$, $y_0^0\in\mathcal{Y}_0$ and $x^0\in\boldsymbol{\mathcal{X}}$. Perform the following steps at each iteration.

Step 1. Solve $x^{k+1}\approx\operatorname{argmin}\{L_\sigma(x;y_0^k)\mid x\in\boldsymbol{\mathcal{X}}\}$ by a proximal gradient method.

Let $\{\varepsilon_s\}$ be a given summable sequence of nonnegative numbers. Set $\hat{x}^0=x^k$.

For $s=0,1,\ldots$, iterate the following step

Step 1a. Compute $\hat{x}^{s+1}\approx\operatorname{argmin}\{L_\sigma(x;y_0^k)+\frac{\sigma}{2}\|x-\hat{x}^s\|^2_{\widehat{\mathcal{T}}}\mid x\in\boldsymbol{\mathcal{X}}\}$. As the problem is separable, one can compute in parallel for $i=0,1,\ldots,N$,

$$\begin{aligned}\hat{x}_i^{s+1}&\approx\operatorname{argmin}\Big\{f_i(x_i)+\frac{\sigma}{2}\big\langle x_i-\hat{x}_i^s,\mathcal{E}_i(x_i-\hat{x}_i^s)\big\rangle+\big\langle x_i-\hat{x}_i^s,\sigma\mathcal{A}_i^*\big(\mathcal{A}\hat{x}^s-b_0-\sigma^{-1}y_0^k\big)\big\rangle\mid x_i\in F_i\Big\}\\&=\operatorname{argmin}\Big\{\theta_i(x_i)+\frac{1}{2}\langle x_i,(\mathcal{Q}_i+\sigma\mathcal{E}_i)x_i\rangle+\big\langle x_i,\hat{\alpha}_i^s\big\rangle\mid x_i\in F_i\Big\},\end{aligned}\tag{22}$$

where $\hat{\alpha}_i^s=\mathcal{Q}_i\hat{x}_i^s+c_i+\sigma\mathcal{A}_i^*(\mathcal{A}\hat{x}^s-b_0-\sigma^{-1}y_0^k)-(\mathcal{Q}_i+\sigma\mathcal{E}_i)\hat{x}_i^s$.

Step 1b. If the residual d_i^{s+1} satisfies $\|d_i^{s+1}\|\le\varepsilon_s/\sqrt{N+1}$, where $d_i^{s+1}:=v_i^{s+1}+\mathcal{Q}_i\hat{x}_i^{s+1}+c_i+\sigma\mathcal{A}_i^*(\mathcal{A}\hat{x}^{s+1}-b_0-\sigma^{-1}y_0^k)$ for some $v_i^{s+1}\in\partial(\theta_i+\delta_{F_i})(\hat{x}_i^{s+1})$, then set $x^{k+1}=\hat{x}^{s+1}$ and proceed to Step 2; otherwise, proceed to Step 1(a).

Step 2. $y_0^{k+1}=y_0^k+\tau\sigma(b_0-\mathcal{A}x^{k+1})$, where $\tau\in(0,2)$ is the steplength.

Note that in ALM-DQA-mod, the reason we choose to use $\widehat{\mathcal{T}}$ instead of the operator $\mathcal{T}$ in (12) for the proximal term is to clarify the connection of our new method to the DQA method of Ruszczyński (1989). In order not to disrupt the flow of the presentation, the details on the connection between the two methods are presented in the appendix. In a nutshell, we may view the DQA method in Ruszczyński (1989) as an augmented Lagrangian method for which the subproblem in Step 1 of Algorithm ALM is solved by a *majorized* proximal gradient method with the proximal term chosen to be $\frac{\sigma}{2}\|x-\hat{x}^s\|^2_{\widehat{\mathcal{T}}}$ at each step. But for ALM-DQA-mod, it uses a proximal gradient method with the same proximal term to solve the subproblem in Step 1 of ALM.

Remark 2.

a. Here, we should point out the distinction between spALM and ALM-DQA-mod. At each iteration, the former solves the Proximal Subproblem (14) only once and follows by an update of the variable y_0^k, whereas the latter solves multiple proximal subproblems before updating the variable y_0^k.

b. With the derivation of ALM-DQA-mod as an augmented Lagrangian method with its subproblems solved by a specially chosen proximal gradient method, we can leverage on this viewpoint to design an accelerated variant of this method. Specifically, we can moderately improve the efficiency in solving the subproblems by using an inexact Nesterov's accelerated proximal gradient method. We refer the reader to the online version (arXiv:1812.04941) of this manuscript for such an extension that is named as ALM-iAPG.

3.3. Numerical Performance of spALM and ALM-DQA-mod

In the previous subsections, we have derived two augmented Lagrangian-based decomposition methods, spALM and ALM-DQA-mod, for directly solving (P). Naturally, one would be curious about their relative performance in practice. In this subsection, we compare their performance for solving several linear and quadratic test instances. The detailed description of the data sets is given in Section 5. For all the instances, we have $m_1=m_2=\cdots=m_N=:\bar{m}$ and $n_1=n_2=\cdots=n_N=:\bar{n}$. The values of $\bar{m}$ and $\bar{n}$ are reported in the comparison table.

Table 1 compares the performance of the primal-based solvers spALM and ALM-DQA-mod for solving the Primal Problem (P) through (10). We also compare their performance to that of the sGS-ADMM that is designed to directly solve the Dual Problem (9). Although the details of the dual approach will only be presented in the next section, we give the comparison here to save some space and, at the same time, to contrast directly the performance of the primal-based algorithms against the dual-based algorithm. As we are not aware of a systematic comparison of the performance of primal-based algorithms versus a dual-based algorithm, we believe such a comparison is a valuable one, even though we shall see later that the dual-based algorithm is much more efficient.

Table 1. Comparison of Computational Results Between sGS-ADMM and Two Variants of ALM for Primal Block-Angular Problems

						sGS-ADMM			spALM			DQA-mod		
Data	m_0	$\bar{m}$	n_0	$\bar{n}$	N	Iter	Time	Acc	Iter	Time	Acc	Iter	Time	Acc
qp-rand0-t1	1	1	20	20	10	321	0.16	2.1-6	153	3.48	4.1-9	9	5.55	1.4-6
qp-rand1-t1	50	50	80	80	10	421	0.19	8.8-6	262	13.47	9.7-6	25	75.39	9.6-6
qp-rand7-t2	10	10	20	20	10	1,501	0.57	2.5-6	2,759	63.36	2.5-6	23	71.55	7.4-6
qp-rand8-t2	50	50	80	80	10	141	0.14	3.9-6	96	4.84	8.9-6	26	50.00	7.7-6
tripart1	2,096	192	2,096	2,096	16	1,981	1.87	9.1-6	3,641	173.33	9.5-6	620	238.69	9.2-6
tripart2	8,432	768	8,432	8,432	16	6,771	32.80	9.3-6	5,000	861.58	8.1-6	1,465	1,019.56	1.0-5
qp-tripart1	2,096	192	2,096	2,096	16	653	0.84	9.9-6	314	15.05	7.0-6	28	27.97	8.2-6
qp-tripart2	8,432	768	8,432	8,432	16	971	5.84	9.4-6	336	56.98	1.0-5	49	123.65	8.9-6
qp-pds1	87	126	372	372	11	971	0.39	1.0-5	207	6.05	9.6-6	243	38.84	9.2-6
qp-SDC7-t1	10,000	200	0	10,000	100	31	1.31	7.8-6	10	14.82	3.3-6	9	22.25	4.8-6
qp-SDC8-t2	10,000	200	0	10,000	100	31	1.33	3.2-6	9	13.94	3.7-6	2	10.02	4.9-6
qp-SDC9-t1	10,000	200	0	10,000	100	31	1.31	9.8-6	10	15.09	3.4-6	10	24.99	4.8-6
qp-SDC10-t2	10,000	200	0	10,000	100	31	1.32	3.1-6	9	14.31	3.9-6	3	11.16	5.0-6
qp-SDC11-t1	10,000	200	0	10,000	100	32	1.35	7.2-6	10	15.46	3.2-6	10	26.86	7.2-6
qp-SDC12-t2	10,000	200	0	10,000	100	31	1.32	2.9-6	9	14.55	4.2-6	3	11.08	7.3-6
qp-SDC13-t1	10,000	200	0	10,000	200	41	3.22	9.0-6	10	47.77	2.5-6	10	61.41	5.2-6
qp-SDC14-t1	20,000	300	0	20,000	200	34	6.52	9.2-6	10	119.61	2.4-6	9	125.61	7.4-6
qp-SDC15-t1	40,000	400	0	40,000	200	31	10.93	4.7-6	10	238.47	2.1-6	9	282.97	4.6-6
M64-64	405	64	511	511	64	1,991	2.23	1.0-5	5,000	318.05	1.7-4	437	853.24	9.9-6

Notes. Here, "Iter" is the number of outer iterations performed, "Time" is the total runtime in seconds, and "Acc" is the accuracy of the computed solution that is defined in Section 5. Under the column "Acc," we record the accuracy "x.ye-0z" using the format "x.y-z" to save some space.

For the two primal approaches, we use eight parallel workers to solve the subproblem in Step 1. Although our dual-based algorithm (sGS-ADMM) can also be parallelized to solve the subproblems, their computations are already so efficient that it becomes counterproductive to parallelize the computations due to the overhead incurred by MATLAB. Thus, we have implemented our dual approach serially. More details on this issue are given in a later section on sGS-ADMM. We set the maximum number of iterations to be 5,000 for the primal approaches and 10,000 for the dual approach, respectively. Here, we observe that, although the primal variants generally take a smaller number of outer iterations than sGS-ADMM, they always require much longer runtime to achieve the same accuracy level in terms of the relative KKT residual. In addition, ALM-DQA-mod is slightly slower than spALM, though the difference is not too significant.

Although the performance of primal-based algorithms is not as good as sGS-ADMM, we believe that it is still worthwhile to present them in this paper because existing decomposition algorithms are primarily based on the primal approach. A key observation we can make from the comparison is that the inferior performance of the primal-based algorithms is due to their need to solve complicated constrained subproblems in Step 1 of the algorithms, which can incur heavy computational costs that make their overall runtimes longer than that of the sGS-ADMM. As we shall see in the next section, the sGS-ADMM is designed in such a way that only simple subproblems are solved at each iteration. As the simple subproblems generally can be solved analytically, the computational costs incurred are relatively low. The inferior performance of the two primal approaches has, thus, motivated us to instead focus on the dual approach of designing an efficient algorithm for the Dual Problem (9).

4. A Semi-proximal Symmetric Gauss–Seidel-Based ADMM for the Dual Problem (D)

In the previous section, we have designed the spALM algorithm to solve the Primal Problem (P) directly. One can also attempt to solve (P) via its Dual Problem (D) given in (9). Based on the structure in (D), we find that it is highly conducive for us to employ a symmetric Gauss–Seidel-based ADMM (sGS-ADMM) to solve the problem, as we shall see later when the details are presented.

To derive sGS-ADMM for solving (D), it is more convenient for us to express (D) in a more compact form as follows:

$$\min \{p(s) + f(y_{1:N}, w, s) + q(z) + g(y_0, z) \mid \mathcal{F}^*[y_{1:N}; w; s] + \mathcal{G}^*[y_0; z] = c\}, \tag{23}$$

where $y_{1:N} = [y_1; \ldots; y_N]$, and

$$\mathcal{F}^* := [\mathcal{D}^*, -\mathcal{Q}, I], \quad \mathcal{G}^* := [\mathcal{A}^*, I],$$
$$p(s) := \theta^*(-s), \quad f(y_{1:N}, w, s) := -\langle b_{1:N}, y_{1:N}\rangle + \frac{1}{2}\langle w, \mathcal{Q}w\rangle + \delta_{\mathcal{W}}(w),$$
$$q(z) := \delta^*_{\mathcal{K}}(-z), \quad g(y_0, z) := -\langle b_0, y_0\rangle.$$

Here, we take $\mathcal{W} = \mathrm{Range}(\mathcal{Q})$. This is a multiblock linearly constrained convex programming problem for which the direct application of the multiblock extension of the classical two-block ADMM is not guaranteed to converge. Thus, we adapt the recently developed inexact sGS-ADMM (Li et al. 2016, Chen et al. 2017) whose convergence is guaranteed to solve the Dual Problem (D).

Given a positive parameter σ, the augmented Lagrangian function for (D) is given by

$$\begin{aligned}\mathcal{L}_\sigma(y,w,s,z;x) &= p(s) + f(y_{1:N},w,s) + q(z) + g(y_0,z) + \frac{\sigma}{2}\left\|\mathcal{F}^*[y_{1:N};w;s] + \mathcal{G}^*[y_0;z] - c + \frac{1}{\sigma}x\right\|^2 - \frac{1}{2\sigma}\|x\|^2 \\ &= \sum_{i=0}^{N}\left[\theta_i^*(-s_i) + \delta^*_{\mathcal{K}_i}(-z_i) + \frac{1}{2}\langle w_i, \mathcal{Q}_i w_i\rangle - \langle b_i, y_i\rangle\right] + \frac{\sigma}{2}\left\|-\mathcal{Q}_0 w_0 + \mathcal{A}_0^* y_0 + s_0 + z_0 - c_0 + \sigma^{-1}x_0\right\|^2 - \frac{1}{2\sigma}\|x_0\|^2. \\ &\quad + \sum_{i=1}^{N}\left[\frac{\sigma}{2}\left\|-\mathcal{Q}_i w_i + \mathcal{A}_i^* y_0 + \mathcal{D}_i^* y_i + s_i + z_i - c_i + \sigma^{-1}x_i\right\|^2 - \frac{1}{2\sigma}\|x_i\|^2\right]. \end{aligned} \tag{24}$$

In order to develop sGS-ADMM, we need to analyze the block structure of the quadratic terms in $\mathcal{L}_\sigma(y,w,s,z;x)$ corresponding the blocks $[y_{1:N};w;s]$ and $[y_0;z]$. They are, respectively, given as follows:

$$\mathcal{F}\mathcal{F}^* := \begin{bmatrix} \mathcal{D}\mathcal{D}^* & -\mathcal{D}\mathcal{Q} & \mathcal{D} \\ -\mathcal{Q}\mathcal{D}^* & \mathcal{Q}^2 & -\mathcal{Q} \\ \mathcal{D}^* & -\mathcal{Q} & I \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & -\mathcal{D}\mathcal{Q} & \mathcal{D} \\ 0 & 0 & -\mathcal{Q} \\ 0 & 0 & 0 \end{bmatrix}}_{\mathcal{U}_{\mathcal{F}}} + \underbrace{\begin{bmatrix} \mathcal{D}\mathcal{D}^* & 0 & 0 \\ 0 & \mathcal{Q}^2 & 0 \\ 0 & 0 & I \end{bmatrix}}_{\mathcal{D}_{\mathcal{F}}} + \mathcal{U}_{\mathcal{F}}^*,$$

$$\mathcal{G}\mathcal{G}^* := \begin{bmatrix} \mathcal{A}\mathcal{A}^* & \mathcal{A} \\ \mathcal{A}^* & I \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & \mathcal{A} \\ 0 & 0 \end{bmatrix}}_{\mathcal{U}_{\mathcal{G}}} + \underbrace{\begin{bmatrix} \mathcal{A}\mathcal{A}^* & 0 \\ 0 & I \end{bmatrix}}_{\mathcal{D}_{\mathcal{G}}} + \mathcal{U}_{\mathcal{G}}^*.$$

Based on the above (symmetric Gauss–Seidel) decompositions, we define the following positive semidefinite linear operators associated with the decompositions:

$$\mathrm{sGS}(\mathcal{F}\mathcal{F}^*) = \mathcal{U}_{\mathcal{F}}^*\mathcal{D}_{\mathcal{F}}^{-1}\mathcal{U}_{\mathcal{F}}, \quad \mathrm{sGS}(\mathcal{G}\mathcal{G}^*) = \mathcal{U}_{\mathcal{G}}^*\mathcal{D}_{\mathcal{G}}^{-1}\mathcal{U}_{\mathcal{G}}. \tag{25}$$

Note that here we view $\mathcal{Q}$ as a linear operator defined on $\mathcal{W}$, and because we take $\mathcal{W} = \mathrm{Range}(\mathcal{Q})$, $\mathcal{Q}^2$ is positive definite on $\mathcal{W}$, and, hence, $\mathcal{D}_{\mathcal{F}}$ is invertible. Because $\mathcal{A}$ is assumed to have full row-rank, $\mathcal{D}_{\mathcal{G}}$ is also invertible.

The basic template of the sGS-ADMM for (23) is given as follows.

sGS-ADMM (Basic Template)**.** Given $(y^0, w^0, s^0, z^0, x^0) \in \mathcal{Y}\times\mathcal{W}\times\mathcal{X}\times\mathcal{X}\times\mathcal{X}$, perform the following steps at the k-th iteration for $k = 0, 1, \ldots$

Step 1. Compute

$$(y_{1:N}^{k+1}, w^{k+1}, s^{k+1}) = \underset{y_{1:N}\in\mathcal{Y}_1\times\cdots\times\mathcal{Y}_N, w\in\mathcal{W}, s\in\mathcal{X}}{\mathrm{argmin}} \left\{ \begin{array}{l} p(s) + f(y_{1:N}, w, s) \\ + \frac{\sigma}{2}\left\|\mathcal{F}^*[y_{1:N};w;s] + \mathcal{G}^*[y_0^k;z^k] - c + \frac{1}{\sigma}x^k\right\|^2 \\ + \frac{\sigma}{2}\left\|[y_{1:N};w;s] - [y_{1:N}^k;w^k;s^k]\right\|^2_{\mathrm{sGS}(\mathcal{F}\mathcal{F}^*)} \end{array} \right\}.$$

Step 2. Compute

$$(y_0^{k+1}, z^{k+1}) = \underset{y_0\in\mathcal{Y}_0, z\in\mathcal{X}}{\mathrm{argmin}} \left\{ \begin{array}{l} q(z) + g(y_0, z) + \frac{\sigma}{2}\left\|\mathcal{F}^*[y_{1:N}^{k+1};w^{k+1};s^{k+1}] + \mathcal{G}^*[y_0;z] - c + \frac{1}{\sigma}x^k\right\|^2 \\ + \frac{\sigma}{2}\left\|[y_0;z] - [y_0^k;z^k]\right\|^2_{\mathrm{sGS}(\mathcal{G}\mathcal{G}^*)} \end{array} \right\}.$$

Step 3. Compute $x^{k+1} = x^k + \tau\sigma(\mathcal{F}^*[y_{1:N}^{k+1};w^{k+1};s^{k+1}] + \mathcal{G}^*[y_0^{k+1};z^{k+1}] - c)$, where $\tau \in (0, \frac{1+\sqrt{5}}{2})$ is the steplength determined by Glowinski (1984).

Now, we explain the motivation for adding the proximal terms in Steps 1 and 2 involving the special sGS operators in (25). For simplicity, we focus the discussion on Step 2 and the case where $\mathcal{K}$ is a closed convex cone. Observe that the minimization problem involved in Step 2 is a convex quadratic-programming problem in the variable (y_0, z), and z is constrained to be in $\mathcal{K}^*$. If we were to solve this problem directly for (y_0^{k+1}, z^{k+1}) by using an appropriate solver, such as Gurobi, it would make each sGS-ADMM iteration very expensive. Fortunately, with the specially designed proximal term involving the sGS operator $\mathrm{sGS}(\mathcal{G}\mathcal{G}^*)$, one can make use of the sGS-decomposition theorem in Li et al. (2019) to decompose the computation of y_0^{k+1} and z^{k+1} into a cyclic manner that is outlined in Step 2(a) to Step 2(c) of the computational version of the sGS-ADMM described below. In this way, the computation of y_0^{k+1} and z^{k+1} can be done by solving much simpler minimization subproblems, and the overall cost is expected to be much lower than solving the problem in Step 2 directly. In a similar way, the computation of $(y_{1:N}^{k+1}, w^{k+1}, s^{k+1})$ in Step 1 can also be decomposed into a cyclic manner by employing the special proximal term involving the sGS operator $\mathrm{sGS}(\mathcal{F}\mathcal{F}^*)$.

With the above preparations, we can now give the detailed description of sGS-ADMM algorithm for solving (9).

sGS-ADMM (Computational Version) on (9)**.** Given $(y^0, w^0, s^0, z^0, x^0) \in \mathcal{Y} \times \mathcal{W} \times \mathcal{X} \times \mathcal{X} \times \mathcal{X}$. For $k = 0, 1, \ldots$, compute $(y^{k+1}, w^{k+1}, s^{k+1}, z^{k+1}, x^{k+1})$ in the following order.

$$
\begin{aligned}
&\textbf{Step 1a.} && (\bar{y}_1^k, \ldots, \bar{y}_N^k) = \operatorname*{argmin}_{y_1 \in \mathcal{Y}_1, \ldots, y_N \in \mathcal{Y}_N} \{\mathcal{L}_\sigma((y_0^k, y_1, \ldots, y_N), w^k, s^k, z^k; x^k)\}\\
&\textbf{Step 1b.} && (\bar{w}_0^k, \ldots, \bar{w}_N^k) = \operatorname*{argmin}_{w_0 \in \mathcal{W}_0, \ldots, w_N \in \mathcal{W}_N} \{\mathcal{L}_\sigma((y_0^k, \bar{y}_1^k, \ldots, \bar{y}_N^k), (w_0, w_1, \ldots, w_N), s^k, z^k; x^k)\}\\
&\textbf{Step 1c.} && (s_0^{k+1}, \ldots, s_N^{k+1}) = \operatorname*{argmin}_{s_0 \in \mathcal{X}_0, \ldots, s_N \in \mathcal{X}_N} \{\mathcal{L}_\sigma((y_0^k, \bar{y}_1^k, \ldots, \bar{y}_N^k), \bar{w}^k, (s_0, s_1, \ldots, s_N), z^k; x^k)\}\\
&\textbf{Step 1d.} && (w_0^{k+1}, \ldots, w_N^{k+1}) = \operatorname*{argmin}_{w_0 \in \mathcal{W}_0, \ldots, w_N \in \mathcal{W}_N} \{\mathcal{L}_\sigma((y_0^k, \bar{y}_1^k, \ldots, \bar{y}_N^k), (w_0, w_1, \ldots, w_N), s^{k+1}, z^k; x^k)\}\\
&\textbf{Step 1e.} && (y_1^{k+1}, \ldots, y_N^{k+1}) = \operatorname*{argmin}_{y_1 \in \mathcal{Y}_1, \ldots, y_N \in \mathcal{Y}_N} \{\mathcal{L}_\sigma((y_0^k, y_1, \ldots, y_N), w^{k+1}, s^{k+1}, z^k; x^k)\}\\
&\textbf{Step 2a.} && \bar{y}_0^k = \operatorname*{argmin}_{y_0 \in \mathcal{Y}_0} \{\mathcal{L}_\sigma((y_0, y_1^{k+1}, \ldots, y_N^{k+1}), w^{k+1}, s^{k+1}, z^k; x^k)\}\\
&\textbf{Step 2b.} && (z_0^{k+1}, \ldots, z_N^{k+1}) = \operatorname*{argmin}_{z_0 \in \mathcal{X}_0, \ldots, z_N \in \mathcal{X}_N} \{\mathcal{L}_\sigma((\bar{y}_0^k, y_1^{k+1}, \ldots, y_N^{k+1}), w^{k+1}, s^{k+1}, (z_0, \ldots, z_N); x^k)\}\\
&\textbf{Step 2c.} && y_0^{k+1} = \operatorname*{argmin}_{y_0 \in \mathcal{Y}_0} \{\mathcal{L}_\sigma((y_0, y_1^{k+1}, \ldots, y_N^{k+1}), w^{k+1}, s^{k+1}, z^{k+1}; x^k)\}\\
&\textbf{Step 3.} && x^{k+1} = x^k + \tau\sigma(-\mathcal{Q}w^{k+1} + \mathcal{B}^* y^{k+1} + s^{k+1} + z^{k+1} - c),
\end{aligned}
$$

where $\tau \in (0, \frac{1+\sqrt{5}}{2})$ is the steplength.

4.1. Convergence Theorems of sGS-ADMM

The convergence theorem of sGS-ADMM can be established directly by using known results from Chen et al. (2017) and Zhang et al. (2018). Here, we present the global convergence result and the linear rate of convergence for the convenience of the reader. In order to state the convergence theorems, we need some definitions.

Denote $u := (y, w, s, z, x) \in \mathcal{U} := \mathcal{Y} \times \mathcal{W} \times \mathcal{X} \times \mathcal{X} \times \mathcal{X}$. We consider the KKT mapping $\mathcal{R} : \mathcal{U} \to \mathcal{U}$ of (4) defined by

$$
\mathcal{R}(u) := \begin{pmatrix} \mathcal{B}x - b \\ -\mathcal{Q}w + \mathcal{B}^* y + s + z - c \\ \mathcal{Q}w - \mathcal{Q}x \\ x - \mathrm{Prox}_\theta(x - s) \\ x - \Pi_{\mathcal{K}}(x - z) \end{pmatrix}. \tag{26}
$$

Note that the set of KKT points of (P) and (D) is precisely the set $\bar{\Omega} := \{u \in \mathcal{U} \mid \mathcal{R}(u) = 0\}$. The KKT mapping $\mathcal{R}$ is said to satisfy the error bound condition at $0 \in \mathcal{U}$ with modulus $\kappa > 0$ if there exist $\epsilon > 0$ such that for all u satisfying $\|\mathcal{R}(u)\| \le \epsilon$, we have

$$
\mathrm{dist}(u, \bar{\Omega}) \le \kappa \|\mathcal{R}(u)\|.
$$

Now, we are ready to present the convergence theorem of sGS-ADMM.

Theorem 2. *Let $\{u^k := (y^k, w^k, s^k, z^k; x^k)\}$ be the sequence generated by sGS-ADMM. Then, we have the following results.*

a. *The sequence $\{(y^k, w^k, s^k, z^k)\}$ converges to an optimal solution of the compact form (6) of the Dual Problem (D), and the sequence $\{x^k\}$ converges to an optimal solution of the compact form (4) of the Primal Problem (P).*

b. *Suppose that the sequence $\{u^k\}$ converges to a KKT point $\bar{u} := (\bar{y}^k, \bar{w}^k, \bar{s}^k, \bar{z}^k, \bar{x}^k)$, and the KKT mapping $\mathcal{R}$ satisfies the error bound condition at $0 \in \mathcal{U}$. Then, the sequence $\{u^k\}$ is linearly convergent to $\bar{u}$.*

Proof.

a. We can regroup the variables (y, w, s, z, x) in the order $(((s_0, \ldots, s_N), (w_0, \ldots, w_N), (y_1, \ldots, y_N)), ((z_0, \ldots, z_N), y_0), (x_0, \ldots, x_N))$. Then, the sequence generated by sGS-ADMM is exactly the same as the sequence generated by sGS-imsPADMM given in Chen et al. (2017); thus, the global convergence result follows directly.

b. By regrouping (y, w, s, z, x) in the same order as in (a), we can recover the sequence generated by algorithm 2 in Zhang et al. (2018). Because the mapping $\mathcal{R}$ satisfies the error bound condition at $0 \in \mathcal{U}$, it is also metrically subregular at $\bar{u}$ for $0 \in \mathcal{U}$. From here, the result follows directly by applying the convergence result in Zhang et al. (2018, proposition 4.1). □

Remark 3. By theorem 1 and remark 1 in Li et al. (2018), we know that when (P) is a convex programming problem such that for each $i = 0, \ldots, N$, θ_i is piecewise linear-quadratic or strongly convex, and $\mathcal{K}_i$ is polyhedral, then $\mathcal{R}$ satisfies the error bound condition at $0 \in \mathcal{U}$. Thus, sGS-ADMM converges locally at a linear rate to an optimal solution of (P) and (D) under the previous conditions on θ_i and $\mathcal{K}_i$. In particular, for the special case of a primal block-angular quadratic-programming problem where $\theta_i \equiv 0$ and $\mathcal{K}_i = \mathbb{R}^{n_i}_+$ for all i, we know that sGS-ADMM is not only locally linearly convergent, but can even be proven to converge globally linearly.

4.2. Computational Details for Implementing sGS-ADMM

In this subsection, we discuss in detail how we can solve each of the subproblem in sGS-ADMM efficiently. These issues are critical for the overall computational efficiency of the algorithm, and one of our contributions in this paper is in proposing efficient numerical techniques to solve the subproblems.

4.2.1. Steps 1(a) and 1(e). Given the augmented Lagrangian function in (24), the subproblem we need to solve in Step 1(a) is:

$$(\bar{y}_1^k, \ldots, \bar{y}_N^k) = \underset{y_1 \in \mathcal{Y}_1, \ldots, y_N \in \mathcal{Y}_N}{\operatorname{argmin}} \sum_{i=1}^{N} \Big[-\langle b_i, y_i \rangle + \frac{\sigma}{2} \big\| -\mathcal{Q}_i w_i^k + \mathcal{A}_i^* y_0^k + \mathcal{D}_i^* y_i + s_i^k + z_i^k - c_i + \sigma^{-1} x_i^k \big\|^2 \Big].$$

This minimization problem is separable in y_i for $i = 1, \ldots, N$. Hence, it can be solved in parallel—that is,

$$\bar{y}_i^k = \underset{y_i \in \mathcal{Y}_i}{\operatorname{argmin}} \Big\{ -\langle b_i, y_i \rangle + \frac{\sigma}{2} \big\| -\mathcal{Q}_i w_i^k + \mathcal{D}_i^* y_i + s_i^k + g_i^k \big\|^2 \Big\} \quad \text{for } i = 1, \ldots, N,$$

where $g^k := \mathcal{A}^* y_0^k + z^k - c + \sigma^{-1} x^k$. Specifically, $\bar{y}_i^k$ is the solution of the following linear system:

$$\mathcal{D}_i \mathcal{D}_i^* \, y_i = \sigma^{-1} b_i - \mathcal{D}_i \big(-\mathcal{Q}_i w_i^k + s_i^k + g_i^k \big) \quad \text{for } i = 1, \ldots, N. \tag{27}$$

Similarly, in Step 1(e), y_i^{k+1} is the solution of the following linear system:

$$\mathcal{D}_i \mathcal{D}_i^* \, y_i = \sigma^{-1} b_i - \mathcal{D}_i \big(-\mathcal{Q}_i w_i^{k+1} + s_i^{k+1} + g_i^k \big) \quad \text{for } i = 1, \ldots, N. \tag{28}$$

We can observe that the only difference between System (27) and (28) is that w_i^k and s_i^k are replaced by w_i^{k+1} and s_i^{k+1} respectively. Several remarks on how to solve (27) and (28) efficiently must now be mentioned.

Remark 4. Note that the $m_i \times m_i$ symmetric positive definite matrices $\mathcal{D}_i \mathcal{D}_i^*$, $\forall i = 1, \ldots, N$ are constant throughout the algorithm. Therefore, one can precompute the Cholesky factorization of $\mathcal{D}_i \mathcal{D}_i^*$, $\forall i = 1, \ldots, N$, if they can be stored in the memory and computed at a reasonable cost. Then, at each sGS-ADMM iteration, $\bar{y}_i^k$ and y_i^{k+1}, $\forall i = 1, \ldots, N$ can be computed cheaply by solving triangular linear systems. However, when computing the coefficient matrix or its Cholesky factorization is too expensive, one can use a preconditioned conjugate gradient (PCG) method to solve the linear system iteratively.

We should emphasize again that sGS-ADMM has the flexibility of allowing for inexact computations, as already mentioned in Chen et al. (2017). Although the computation in Steps 1(a) and 1(e) are assumed to be

done exactly (up to machine precision), the computation can, in fact, be done inexactly, subject to a certain prespecified accuracy requirements on the computed approximate solution. Thus, iterative methods such as the PCG method can be used to solve the linear systems when their dimensions are large. We omit the details here for the sake of brevity.

Remark 5. The computation in Step 1(e) can be omitted if the quantity $\bar{y}_i^k$ computed in Step 1(a) is already a sufficiently good approximate solution to the current subproblem. More precisely, if the approximation $\bar{y}_i^k$ for y_i^{k+1} satisfies the admissible accuracy condition required in the inexact sGS-ADMM designed by Chen et al. (2017), then we can just set $y_i^{k+1} = \bar{y}_i^k$ instead of solving the subproblem in Step 1(e).

4.2.2. Steps 1(b) and 1(d). In Step 1(b) of sGS-ADMM, we can also solve the minimization problem involving $w_0, w_1, \ldots, w_N$ independently by solving in parallel $\forall i = 0, 1, \ldots, N$,

$$\begin{aligned}\bar{w}_i^k &= \operatorname*{argmin}_{w_i}\left\{\frac{1}{2}\langle w_i, \mathcal{Q}_i w_i\rangle + \frac{\sigma}{2}\left\| - \mathcal{Q}_i w_i + \mathcal{D}_i^* \bar{y}_i^k + s_i^k + g_i^k\right\|^2 \mid w_i \in \mathrm{Range}(\mathcal{Q}_i)\right\},\\ &= \left\{w_i \in \mathrm{Range}(\mathcal{Q}_i) \mid \mathcal{Q}_i(I + \sigma\mathcal{Q}_i)w_i = \sigma\mathcal{Q}_i\left(\mathcal{D}_i^* \bar{y}_i^k + s_i^k + g_i^k\right)\right\},\end{aligned} \tag{29}$$

where we define $\mathcal{D}_0 = 0$ for notational convenience.

In fact, $\bar{w}_i^k$ is only required theoretically. In practical implementations, only $\mathcal{Q}_i\bar{w}_i^k$ and $\langle \bar{w}_i^k, \mathcal{Q}_i\bar{w}_i^k\rangle$ are needed. In the next proposition, we show that we can compute $\mathcal{Q}_i\bar{w}_i^k = \mathcal{Q}_i\tilde{w}_i^k$ and $\langle \bar{w}_i^k, \mathcal{Q}_i\bar{w}_i^k\rangle = \langle \tilde{w}_i^k, \mathcal{Q}_i\tilde{w}_i^k\rangle$, where $\tilde{w}_i^k$ is the solution of the simpler linear system below:

$$(I + \sigma\mathcal{Q}_i)\tilde{w}_i = \sigma\left(\mathcal{D}_i^* \bar{y}_i^k + s_i^k + g_i^k\right) \quad \forall\, i = 0, 1, \ldots, N. \tag{30}$$

Proposition 1. *For $i = 0, 1, \ldots, N$, $\mathcal{Q}_i\bar{w}_i^k = \mathcal{Q}_i\tilde{w}_i^k$ and $\langle \bar{w}_i^k, \mathcal{Q}_i\bar{w}_i^k\rangle = \langle \tilde{w}_i^k, \mathcal{Q}_i\tilde{w}_i^k\rangle$, where $\bar{w}_i^k$ is the solution of (29) and $\tilde{w}_i^k$ is the solution of (30).*

Proof. It is sufficient for us to prove the result for a particular $i \in \{0, 1, \ldots, N\}$. By our assumption, $\mathcal{Q}_i$ is symmetric positive semidefinite. Consider the spectral decomposition $\mathcal{Q}_i = U_i S_i U_i^T$, where $S_i \in \mathbb{R}^{r\times r}$ is a diagonal matrix whose diagonal elements are the positive eigenvalues of $\mathcal{Q}_i$, and the columns of $U_i \in \mathbb{R}^{n_i\times r}$ are their corresponding orthonormal set of eigenvectors. We let $V_i \in \mathbb{R}^{n_i\times(n_i-r)}$ be the matrix whose columns form an orthonormal set of eigenvectors of $\mathcal{Q}_i$ corresponding to the zero eigenvalues— that is, $[U_i,\ V_i]$ is an orthogonal matrix.

Because $w_i \in \mathrm{Range}(\mathcal{Q}_i)$, we can parametrize it by $w_i = U_i\xi_i$ for some vector $\xi_i \in \mathbb{R}^r$. Then,

$$\begin{aligned}\frac{1}{2}\langle w_i, \mathcal{Q}_i w_i\rangle + \frac{\sigma}{2}\left\|-\mathcal{Q}_i w_i + \mathcal{D}_i^* \bar{y}_i^k + s_i^k + g_i^k\right\|^2 &= \frac{1}{2}\langle U_i\xi_i, U_i S_i U_i^T U_i\xi_i\rangle + \frac{\sigma}{2}\left\| -U_i S_i U_i^T U_i\xi_i + \zeta_i^k\right\|^2\\ &= \frac{1}{2}\langle \xi_i, S_i\xi_i\rangle + \frac{\sigma}{2}\left\|\left[\,U_i, V_i\,\right]^T\left(-U_i S_i\xi_i + \zeta_i^k\right)\right\|^2\\ &= \frac{1}{2}\langle \xi_i, S_i\xi_i\rangle + \frac{\sigma}{2}\left\|S_i\xi_i - U_i^T\zeta_i^k\right\|^2 + \frac{\sigma}{2}\left\|V_i^T\zeta_i^k\right\|^2,\end{aligned}$$

where we let $\zeta_i^k := \mathcal{D}_i^* \bar{y}_i^k + s_i^k + g_i^k$ for convenience. Hence, the minimization problem in (29) is equivalent to the following:

$$\min_{\xi_i}\left\{\frac{1}{2}\langle \xi_i, S_i\xi_i\rangle + \frac{\sigma}{2}\left\|S_i\xi_i - U_i^T\zeta_i^k\right\|^2 \mid \xi_i \in \mathbb{R}^r\right\}, \tag{31}$$

whose minimizer is the solution of the following linear system:

$$(I + \sigma S_i)\xi_i = \sigma U_i^T\zeta_i^k.$$

Now, from solving (30), we obtain that

$$(I + \sigma S_i)U_i^T\tilde{w}_i^k = \sigma U_i^T\zeta_i^k, \quad V_i^T\tilde{w}_i^k = \sigma V_i^T\zeta_i^k.$$

This shows that $U_i^T\tilde{w}_i^k$ is the unique solution to the Problem (31). Hence, $\bar{w}_i^k = U_i(U_i^T\tilde{w}_i^k)$ is the unique solution to (30), and we have $U_i^T\bar{w}_i^k = U_i^T\tilde{w}_i^k$. From here, we get $\mathcal{Q}_i\bar{w}_i^k = U_i S_i(U_i^T\bar{w}_i^k) = U_i S_i(U_i^T\tilde{w}_i^k) = \mathcal{Q}_i\tilde{w}_i^k$. In addition, $\langle \bar{w}_i^k, \mathcal{Q}_i\bar{w}_i^k\rangle = \langle U_i^T\bar{w}_i^k, S_i U_i^T\bar{w}_i^k\rangle = \langle U_i^T\tilde{w}_i^k, S_i U_i^T\tilde{w}_i^k\rangle = \langle \tilde{w}_i^k, \mathcal{Q}_i\tilde{w}_i^k\rangle$. □

By the same reasoning, we can reduce the computation in Step 1(d) to finding the solution of the following linear system:

$$(I + \sigma\mathcal{Q}_i)\tilde{w}_i = \sigma\big(\mathcal{D}_i^*\bar{y}_i^k + s_i^{k+1} + g_i^k\big) \quad \forall i = 0, 1, \ldots, N. \tag{33}$$

We can again observe that the only difference between System (30) and (33) is that s_i^k has been replaced by s_i^{k+1}. Hence, Remarks 4 and 5 in Subsection 4.2.1 are applicable here. Additionally, if $\mathcal{Q} \equiv 0$, then Steps 1(b) and 1(d) are vacuous, and Step 1 only consists of Steps 1(a), 1(c), and 1(e).

4.2.3. Step 1(c). In Step 1(c) of sGS-ADMM, the subproblem can be solved in parallel—that is, $\forall i = 0, 1, \ldots, N$,

$$\begin{aligned} s_i^{k+1} &= \operatorname*{argmin}_{s_i \in \mathcal{X}_i}\Big\{\theta_i^*(-s_i) + \frac{\sigma}{2}\big\|-\mathcal{Q}_i\bar{w}_i^k + \mathcal{D}_i^*\bar{y}_i^k + s_i + g_i^k\big\|^2\Big\} \\ &= -\mathrm{Prox}_{\theta_i^*/\sigma}\big(-\mathcal{Q}_i\bar{w}_i^k + \mathcal{D}_i^*\bar{y}_i^k + g_i^k\big) = \frac{1}{\sigma}\mathrm{Prox}_{\sigma\theta_i}\big(\sigma\big(-\mathcal{Q}_i\bar{w}_i^k + \mathcal{D}_i^*\bar{y}_i^k + g_i^k\big)\big) - \big(-\mathcal{Q}_i\bar{w}_i^k + \mathcal{D}_i^*\bar{y}_i^k + g_i^k\big), \end{aligned}$$

where the last equality is given by the Moreau identity.

In most cases, the computation for the proximal mapping of $\sigma\theta_i$ should be easy, such as for the case when θ_i is the ℓ_1-norm function. In a more complicated case, one can still compute the proximal mapping efficiently. For example, in the nonlinear model that we will consider in Section 5.4, θ_i is a twice continuously differentiable function. Thus, we can compute

$$\mathrm{Prox}_{\sigma\theta_i}(\beta) = \operatorname*{argmin}_{t \in \mathcal{X}_i}\Big\{\phi(t) := \sigma\theta_i(t) + \frac{1}{2}\|t - \beta\|^2\Big\},$$

via the Newton's method for solving the equation $\nabla\phi(t) = 0$. At each sGS-ADMM iteration, we can warm-start the Newton's method by using the quantity that has already been computed in the previous iteration. Although s_i^{k+1} may not be computed exactly, the convergence of sGS-ADMM is not affected as long as s_i^{k+1} satisfies the admissible accuracy condition required in the inexact sGS-ADMM method developed in Chen et al. (2017).

Moreover, when $\boldsymbol{\theta} \equiv 0$, this step is vacuous. Then, Steps 1(b) and 1(d) are identical. Hence, the computation needs only to be done for Step 1(d). In the other words, Step 1 only consists of Steps 1(a), 1(d), and 1(e).

4.2.4. Steps 2(a) and 2(c). In Step 2(a), we solve the following minimization problem:

$$\bar{y}_0^k = \operatorname*{argmin}_{y_0 \in \mathcal{Y}_0}\left\{-\langle b_0, y_0\rangle + \frac{\sigma}{2}\big\|\mathcal{A}_0^*y_0 + z_0^k + h_0^k\big\|^2 + \sum_{i=1}^N \frac{\sigma}{2}\big\|\mathcal{A}_i^*y_0 + z_i^k + h_i^k\big\|^2\right\},$$

where $h^k := -\mathcal{Q}w^{k+1} + \mathcal{D}^*y^{k+1} + s^{k+1} - c + \sigma^{-1}x^k$. Specifically, $\bar{y}_0^k$ is the solution to the following linear system of equations:

$$\left(\sum_{i=0}^N \mathcal{A}_i\mathcal{A}_i^*\right)y_0 = \sigma^{-1}b_0 - \sum_{i=0}^N \mathcal{A}_i\big(z_i^k + h_i^k\big). \tag{34}$$

Similarly, in Step 2(c), y_0^{k+1} is the solution of the following linear system:

$$\left(\sum_{i=0}^N \mathcal{A}_i\mathcal{A}_i^*\right)y_0 = \sigma^{-1}b_0 - \sum_{i=0}^N \mathcal{A}_i\big(z_i^{k+1} + h_i^k\big). \tag{35}$$

Observe that the only difference between (34) and (35) is in the terms z_i^k and z_i^{k+1} for $i = 0, 1, \ldots, N$. Furthermore, the matrix $\sum_{i=0}^N \mathcal{A}_i\mathcal{A}_i^*$ is constant throughout the algorithm. Hence, Remarks 4 and 5 in Subsection 4.2.1 are applicable here. In the case when PCG method is employed to solve the linear system iteratively, one can also implement the computation of the matrix-vector product in parallel by computing $\mathcal{A}_i\mathcal{A}_i^*y_0$ in parallel for $i = 0, 1, \ldots, N$, given any y_0.

For the multicommodity flow problem, which we will consider later in the numerical experiments, we note that the linear systems in (34) and (35) have a very simple coefficient matrix given by $\sum_{i=0}^N \mathcal{A}_i\mathcal{A}_i^* = (N+1)I_m$,

and the coefficient matrix $\mathcal{D}_i\mathcal{D}_i^*$ in (27) and (28) is equal to the Laplacian matrix of the network graph for all $i = 1, \ldots, N$. Thus, (34) and (35), as well as (27) and (28), can be solved efficiently by a direct solver.

4.2.5. Step 2(b). In Step 2(b), we can solve the subproblem in parallel for $i = 0, 1, \ldots, N$,

$$z_i^{k+1} = \underset{z_i \in \mathcal{X}_i}{\operatorname{argmin}} \left\{ \delta^*_{\mathcal{K}_i}(-z_i) + \frac{\sigma}{2} \left\| \mathcal{A}_i^* \bar{y}_0^k + z_i + h_i^k \right\|^2 \right\} = -\operatorname{Prox}_{\sigma^{-1}\delta^*_{\mathcal{K}_i}} \left(\mathcal{A}_i^* \bar{y}_0^k + h_i^k \right)$$
$$= \frac{1}{\sigma} \Pi_{\mathcal{K}_i} \left(\sigma \left(\mathcal{A}_i^* \bar{y}_0^k + h_i^k \right) \right) - \left(\mathcal{A}_i^* \bar{y}_0^k + h_i^k \right).$$

Note that this step can be computed at a very low cost when $\mathcal{K}_i$ is a simple polyhedral set, such as a nonnegative orthant. In particular, when $\mathcal{K}_i$ is the entire Euclidean space for $i = 0, 1, \ldots, N$, then this step can be omitted, and Step 2 only consists of Step 2(c).

4.3. Computational Cost

Now, we discuss the main computational cost of sGS-ADMM. We observe that the most time-consuming computations are in solving large linear system of Equations (27), (28), (30), (33), (34), and (35) in Steps 1(a), 1(e), 1(b), 1(d), 2(a), and 2(c), respectively.

In general, suppose that for every iteration we need to solve a $d \times d$ linear system of equations:

$$Mx = r. \tag{36}$$

Assuming that M is stored, then we can compute its (sparse) Cholesky factorization at the cost of $O(d^3)$ operations, which only needs to be done once at the very beginning of the algorithm. After that, whenever we need to solve the equation, we just need to compute the right-hand-side vector r and solve two $d \times d$ triangular systems of linear equations at the cost of $O(d^2)$ operations.

We can roughly summarize the costs incurred in solving $Mx = r$ as follows:

(C_1) Cost for computing the coefficient matrix M (only once);
(C_2) Cost for computing the Cholesky factorization of M (only once);
(C_3) Cost for computing right-hand-side vector r;
(C_4) Cost for solving two triangular systems of linear equations.

The computational cost C_1, C_2, C_3, C_4 above for each of the linear systems of equations in Steps 1(a), 1(b), 1(d), 1(e), 2(a), and 2(c) are tabulated in Table 2.

5. Numerical Experiments

In this section, we evaluate the performance of the algorithm we have designed for solving the problem (P) via (D). We conduct numerical experiments on three major types of primal block-angular models, including linear, quadratic, and nonlinear problems. Apart from randomly generated data sets, we also demonstrate that our algorithms can be quite efficient in solving realistic problems encountered in the literature.

Our implementation of sGS-ADMM is in MATLAB. We should emphasize that, although the computation for solving the subproblems can be parallelized, we do not implement it here. This is because the calculations involve only solving linear systems or computing simple proximal mappings. These operations can already be done very efficiently in MATLAB. If we create a parallel pool and use a parallel-for loop to perform these computation in parallel, the overhead cost is much greater than the benefit of doing those computations in parallel. Nevertheless, when the computation of the subproblems is extremely expensive, then a proper parallel implementation of our algorithm should be beneficial for solving primal block-angular problems efficiently. We illustrate this phenomenon in Table 3. One can observe that the runtime can differ by a factor of more than 10. Thus, we choose to implement our algorithm here serially.

Table 2. Computational Cost for Solving the Linear Systems of Equations in sGS-ADMM

Step	C_1 (once)	C_2 (once)	C_3 (each iteration)	C_4 (each iteration)
1a and 1e($i = 1, \ldots, N$)	$O(m_i^2 n_i)$	$O(m_i^3)$	$O(n_i^2 + m_i n_i)$	$O(m_i^2)$
1b and 1d($i = 1, \ldots, N$)	$O(n_i^2)$	$O(n_i^3)$	$O(m_i n_i)$	$O(n_i^2)$
2a and 2c	$O(m_0^2 n_0)$	$O(m_0^3)$	$O(m_0 n_0)$	$O(m_0^2)$

Table 3. Runtime of sGS-ADMM When Implemented Serially (One Worker) vs. Parallel (Eight Workers) in MATLAB

Data	m_0	$\bar{m}$	n_0	$\bar{n}$	N	Time (1 worker)	Time (8 workers)
pds15	1,812	2,125	7,756	7,756	11	16.41	169.60
pds90	8,777	12,186	46,161	46,161	11	271.90	4,022.86

5.1. Stopping Condition

Based on the optimality conditions in (7), we measure the accuracy of a computed solution by the following relative residual:

$$\eta = \max\{\eta_P, \eta_D, \eta_Q, \eta_K, \eta_S\},$$

where

$$\eta_P = \frac{\|\mathcal{B}x - b\|}{1 + \|b\|}, \quad \eta_D = \frac{\|-\mathcal{Q}w + \mathcal{B}^* y + s + z - c\|}{1 + \|c\|}, \quad \eta_Q = \frac{\|\mathcal{Q}w - \mathcal{Q}x\|}{1 + \|\mathcal{Q}\|},$$
$$\eta_K = \frac{\|x - \Pi_{\mathcal{K}}(x - z)\|}{1 + \|x\| + \|z\|}, \quad \eta_S = \frac{\|x - \mathrm{Prox}_{\theta}(x - s)\|}{1 + \|x\| + \|s\|}.$$

We terminate our algorithm when $\eta \le 10^{-5}$.

5.2. Block-Angular Problems with Linear Objective Functions

In this subsection, we perform numerical experiments on minimization problems having linear objective functions and primal block-angular constraints. Multicommodity flow (MCF) problems are one of the main representatives in this class of problems. It is a model to solve the routing problem of multiple commodities throughout a network from a set of supply nodes to a set of demand nodes. These problems usually exhibit primal block-angular structure due to the network nature in the constraints. In Ouorou et al. (2000), the authors investigated several algorithms for solving convex MCFs. Besides solving MCFs as a linear programming with special structure, there are some other methods that tend to focus on exploiting the network structure of the problem—for example, the flow-deviation method (Fratta et al. 1973, LeBlanc 1973) and the projection method (Bertsekas and Gallager 1987).

Consider a connected network graph $(\mathcal{N}, \mathcal{E})$ with m nodes and $n = |\mathcal{E}|$ arcs for which N commodities must be transported through the network. We assume that each commodity has a single source-sink pair (s_k, t_k), and we are given the flow r_k that must be transported from s_k to t_k, for $k = 1, \ldots, N$. Let $M \in \mathbb{R}^{m \times n}$ be node-arc incidence matrix of the graph. Then, the MCF problem can be expressed in the form given in (P) with the following data:

$$\mathcal{K}_0 = \{x_0 \in \mathbb{R}^n \mid 0 \le x_0 \le u\}, \quad \mathcal{K}_i = \mathbb{R}^n_+, \quad \mathcal{A}_0 = I_n, \quad \mathcal{A}_i = -I_n, \quad i = 1, \ldots, N,$$
$$\mathcal{Q}_i = 0, \quad \theta_i(\cdot) = 0, \quad \forall i = 0, 1, \ldots, N,$$
$$\mathcal{D}_1 = \mathcal{D}_2 = \cdots = \mathcal{D}_N = M \text{ is the node-arc incidence matrix.}$$

For this problem, x_i denotes the flow of the i-th commodity ($i = 1, \ldots, N$) through the network, x_0 is the total flow, and u is a given upper bound vector on the total flow.

5.2.1. Description of Data Sets. Following Castro and Cuesta (2011), the data sets we used are as follows.

a. **Tripart** and **gridgen**: These are five-commodity instances obtained with the Tripart and Gridgen generators. They are available at http://www-eio.upc.es/~jcastro/mmcnf_data.html.

b. **pds**: The PDS problems come from a model of transporting patients away from a place of military conflict. They are available at http://www.di.unipi.it/optimize/Data/MMCF.html#Pds.

c. **M**{m}**-**{k}: These are the problems generated by the Mnetgen generator, which is a well-known generator of random multicommodity flow instances. Here, m is the number of nodes in the network, and k is the number of commodities. They are available at http://www.di.unipi.it/optimize/Data/MMCF.html#MNetGen.

5.2.2. Numerical Results. In Table 4, we compare our sGS-ADMM algorithm against the solvers Gurobi and BlockIP. We should emphasize that, although Gurobi is not specially designed for solving primal block-angular problems, it is an extremely powerful commercial software package for solving sparse general linear

Table 4. Comparison of Computational Results Between sGS-ADMM, Gurobi, and BlockIP for Linear Primal Block-Angular Problems

					sGS-ADMM			Gurobi			BlockIP		
Data	m_0	$\bar{m}$	$n_0 = \bar{n}$	N	Iter	Time	Acc	Iter	Time	Acc	Iter	Time	Acc
tripart1	2,096	192	2,096	16	1,981	1.9	9.1-6	10,130	0.5	2.5e-17	48	0.6	2.0-4
tripart2	8,432	768	8,432	16	6,771	32.8	9.3-6	6,691	4.4	3.1e-16	67	5.3	2.2-4
tripart3	16,380	1,200	16,380	20	5,561	52.9	1.0-5	13,563	11.6	4.9e-16	81	26.2	1.5-4
tripart4	24,815	1,050	24,815	35	8,581	182.7	1.0-5	23,786	49.2	4.3e-16	115	77.4	3.9-5
gridgen1	3,072	1,025	3,072	320	7,541	260.8	8.5-6	486,386	6,429.0	2.8e-16	203	890.2	4.9-7
pds15	1,812	2,125	7,756	11	2,893	16.4	1.0-5	9,300	0.9	8.5e-17	81	6.4	1.8-5
pds30	3,491	4,223	16,148	11	4,471	73.2	9.7-6	15,093	4.4	9.1e-17	110	28.8	4.4-2
pds60	6,778	8,423	33,388	11	7,719	276.5	9.8-6	43,623	13.9	4.1e-17	145	263.9	1.4-2
pds90	8,777	12,186	46,161	11	5,315	271.9	1.0-5	98,695	23.2	8.9e-18	162	521.9	8.8-3
M64-64	405	64	511	64	1,991	2.2	1.0-5	7,805	0.5	9.8e-17	51	0.4	2.5-4
M128-64	936	128	1,171	64	2,601	5.1	1.0-5	870	1.8	1.1e-16	52	1.6	1.6-3
M128-128	979	128	1,204	128	3,801	23.4	9.3-6	2,328	3.5	1.1e-16	127	6.0	7.0-3
M256-256	1,802	256	2,204	256	6,821	138.3	7.7-6	103,896	13.2	1.4e-16	97	49.4	6.5-4
M512-64	3,853	512	4,768	64	2,631	29.5	1.0-5	48,120	6.0	1.2e-16	72	26.6	1.0-3
M512-128	3,882	512	4,786	128	3,581	87.3	9.5-6	86,754	12.8	1.3e-16	97	81.0	1.5-3
M512-512	707	512	1,797	512	7,021	267.6	8.2-6	199,535	11.5	1.0e-16	146	171.2	1.7-2

Notes. Here, "Iter" is the number of outer iterations performed, "Time" is the total runtime in seconds, and "Acc" is the accuracy of the computed solution that is defined in Section 5. Under the column "Acc," we record the accuracy "x.ye-0z" using the format "x.y-z" to save some space.

and convex quadratic-programming problems. Hence, we use Gurobi as one of our benchmarks, because a specially designed algorithm should at least be competitive to an efficient general-purpose solver. On the other hand, BlockIP (Castro 2016) is an efficient interior-point algorithm specially designed for solving primal block-angular problems, especially those arising from MCF problems. As reported in Castro (2016), it has been very successful in solving many large-scale instances of primal block-angular LP and QP problems.

In the following numerical experiments, we employ Gurobi directly on the compact Formulation (4). To be more specific, we input $\mathcal{B}$ as a general sparse matrix. The feasibility and objective gap tolerance is set to be 10^{-5}. All the other parameters are set to their default values. Similarly, for BlockIP, all three tolerances (primal and dual feasibility and relative objective gap) are set to be 10^{-5} for consistency. Its maximum number of iterations is set to be 500.

From Table 4, we observe that Gurobi is the fastest to solve 13 out of 16 instances. Gurobi is extremely fast in solving the pdsxx and Mxxx-xx problems, but has difficulty in solving gridgen1 efficiently. On the other hand, sGS-ADMM and BlockIP are highly efficient in solving the latter instance, for which sGS-ADMM is in fact the fastest solver—it is 24.6 times faster than Gurobi and 3.4 times faster than BlockIP. The numerical results here show that, although Gurobi is not specially designed to exploit the block-angular structure of the tested LP instances, it can exploit any data sparsity extremely well, and, together with the highly optimized implementations of its solvers, it can even outperform the specialized solvers sGS-ADMM and BlockIP by a big margin on some of the instances—for example, pds90 and M512-512.

We also noticed that BlockIP is quite sensitive to the practical setting of the upper bound on the unbounded variables. For example, setting "9×10^6" and "9×10^8" as the upper bounds for the unbounded variables can lead to a significant difference in the number of iterations. Also, BlockIP often terminates prematurely without reaching the required accuracy level. For example, BlockIP only attains the accuracy level of about 10^{-2} for pds60 and M512-512. On the other hand, sGS-ADMM is able to reach the required accuracy level of 10^{-5} in the relative KKT residual for all the instances. We note that Gurobi automatically decides to use a simplex method to solve the tested LP problems, and, hence, the reported iteration counts are generally large. It often reaches an accuracy level that is much higher than required because of the finite termination property of the simplex method. For BlockIP, the reported numbers of interior-point iterations are generally less than two hundreds, but the computational cost per iteration is high. For sGS-ADMM, the number of iterations it takes to solve a problem is usually an order of magnitude higher than that of the BlockIP solver, but the cost it takes to solve the subproblems at each iteration is much lower. Overall, sGS-ADMM outperforms the specialized interior-point solver BlockIP because it can strike a good balance between the number of iterations needed to attain the desired accuracy and the computational cost taken per iteration.

5.3. Block-Angular Problems with Convex Quadratic Objective Functions

In this subsection, we perform numerical experiments on optimization problems having convex quadratic objective functions and primal block-angular constraints.

Multicommodity flow problem is, again, an important class for this type of problem. Following Castro (2016), we add the quadratic objective term, $\mathcal{Q}_i = 0.1I$, $\forall i = 0, \ldots, N$. The corresponding data sets are named with the prefix "qp-", including tripart, gridgen, and pds.

Another class of quadratic primal block-angular problems arises in the field of statistical disclosure control. Castro (2005) studied the controlled tabular adjustment (CTA) to find a perturbed, but safe, table that is closest to a given three-dimensional table for which the content must be protected. In particular, we have

$$\begin{aligned}
&\mathcal{Q}_i = I,\ \theta_i(\cdot) = 0,\ i = 0, \ldots, N,\\
&\mathcal{A}_0 = I,\ \mathcal{A}_i = -I,\ \forall i = 1, \ldots, N,\\
&\mathcal{D}_1 = \mathcal{D}_2 = \cdots = \mathcal{D}_N \text{ is a node-arc incidence matrix,}
\end{aligned}$$

and $\mathcal{K}_i$ $(i = 0, 1, \ldots, N)$ is the same as in Section 5.2.

5.3.1. Description of Data Sets. The data sets we used are as follows.

a. **rand:** These instances are randomly generated sparse problems. Here, we generated two types of problems.

b. Type 1 problem (with suffix -t1) has diagonal quadratic objective cost. We construct $\mathcal{Q}_i$ as a sparse diagonal matrix where the diagonal entries are randomly generated—that is, Qi = spdiags(rand(ni,1),0,ni,ni) in MATLAB syntax.

c. Type 2 problem (with suffix -t2) has nondiagonal quadratic objective cost. In this case, $\mathcal{Q}_i$ is still very sparse, but remain positive semidefinite. Practically, we can construct $\mathcal{Q}_i$ by first generating a random $n_i \times n_i$ matrix $\bar{\mathcal{Q}}_i$ and then set $\mathcal{Q}_i := \bar{\mathcal{Q}}_i \bar{\mathcal{Q}}_i^T$—that is, tmp=sprandn(ni,ni,0.1); Qi = tmp*tmp' in MATLAB syntax.

d. For both types of problems, we generate $\mathcal{A}_i$ and $\mathcal{D}_i$ similarly for $i = 0, \ldots, N$ using the MATLAB command sprandn with density 0.5 and 0.3, respectively. Note that by convention, we have $\mathcal{D}_0 = 0$.

e. **L2CTA3D:** This is a very large instance (with a total of 10 million variables and 210,000 constraints) provided at http://www-eio.upc.es/~jcastro/huge_sdc_3D.html.

f. **SDC:** These are some of the CTA instances we generated using the generator provided by J. Castro at http://www-eio.upc.es/~jcastro/CTA_3Dtables.html.

5.3.2. Numerical Results. As in Section 5.2, we compare our sGS-ADMM algorithm against Gurobi and BlockIP in Table 5.

Table 5 shows that Gurobi is frequently the slowest to solve the test instances, except for the qp-pdsxx instances, whereas our sGS-ADMM performs almost as efficiently as BlockIP in solving these quadratic primal block-angular problems. Here, we note that Gurobi used a barrier method to solve these problems and hence the reported numbers of iterations are generally within a hundred. For these randomly generated problems, we observe that the presolve routine in Gurobi can take a substantial amount of time to analyze the problems but does not manage to reduce the number of variables or constraints. Hence we turn off the presolve feature in Gurobi when collecting the numerical results in Table 5 because this can greatly reduce its runtime for solving the problems.

It is worth noting that our sGS-ADMM method works very well on the large-scale, randomly generated problems compared with BlockIP. A plausible explanation for BlockIP's inferior performance is that, for these instances, the matrices $\mathcal{A}_i$ and $\mathcal{Q}_i$, $i = 0, \ldots, N$ are no longer simple identity matrices, for which it can take special advantage in its implementation. Also, BlockIP runs out of memory for two of the largest instances, qp-rand4-t1 and qp-rand6-t1. Moreover, it often cannot solve the problems to the desired accuracy of 10^{-5} in the relative KKT residual.

It is also observed that BlockIP could not solve for the qp-randx-t2 problems because it is not designed for problems with nondiagonal quadratic objective cost. For these problems, our sGS-ADMM can substantially outperform Gurobi, sometimes by a factor of more than nine. For the qp-SDCx-x problems, both BlockIP and sGS-ADMM are extremely efficient, and they are often more then 10 times faster than Gurobi. For example, our sGS-ADMM is more than 67 times faster than Gurobi in solving the problem qp-SDC17-t1. It is not surprising that the Gurobi is less competitive than sGS-ADMM in solving the convex quadratic problems tested in Table 5 because each iteration of the barrier method it uses is expensive, especially when the number of linear constraints $\sum_{i=0}^{N} m_i$ in the problem is very large. On the other hand, our sGS-ADMM is designed to fully exploit the block-angular structure in the tested QP problems to keep its computational cost per

Table 5. Comparison of Computational Results Between sGS-ADMM, Gurobi, and BlockIP for Quadratic Primal Block-Angular Problems

				sGS-ADMM			Gurobi			BlockIP		
Data	m_0; $\bar{m}$	n_0; $\bar{n}$	N	Iter	Time	Acc	Iter	Time	Acc	Iter	Time	Acc
qp-rand1-t1	50; 50	80; 80	10	421	0.19	8.8-6	15	0.15	9.6-5	29	0.06	1.6-4
qp-rand2-t1	1,000; 1,000	1,500; 1,500	10	748	39.6	1.0-5	16	52.5	5.2-6	39	305.1	6.8-4
qp-rand3-t1	100; 100	200; 200	100	331	2.7	7.4-6	21	1.8	1.4-5	54	5.3	1.3-2
qp-rand4-t1	1,000; 1,000	1,500; 1,500	100	361	218.7	9.6-6	19	410.2	6.0-7	/	/	/
qp-rand5-t1	100; 100	200; 200	150	341	4.2	8.5-6	22	2.7	2.6-5	58	26.5	1.6-1
qp-rand6-t1	1,000; 1,000	1,500; 1,500	150	448	363.5	9.9-6	19	608.8	2.3-6	/	/	/
qp-rand7-t2	10; 10	20; 20	10	1,501	0.57	2.5-6	14	0.12	5.5-5	*	*	*
qp-rand8-t2	50; 50	80; 80	10	141	0.14	3.9-6	16	0.18	2.4-5	*	*	*
qp-rand9-t2	1,000; 1,000	1,500; 1,500	10	131	36.6	1.0-6	12	329.9	1.7-6	*	*	*
qp-rand10-t2	100; 100	200; 200	100	81	2.4	5.7-6	16	7.3	1.8-5	*	*	*
qp-rand11-t2	1,000; 1,000	1,500; 1,500	100	220	394.0	4.6-7	13	1,352.2	1.4-6	*	*	*
qp-rand12-t2	100; 100	200; 200	150	74	3.5	8.8-6	17	11.5	1.8-5	*	*	*
qp-rand13-t2	1,000; 1,000	1,500; 1,500	150	252	618.6	4.9-7	13	2,090.2	1.6-6	*	*	*
qp-tripart1	2,096; 192	2,096; 2,096	16	653	0.84	9.9-6	16	0.48	3.6-7	24	0.1	2.4-5
qp-tripart2	8,432; 768	8,432; 8,432	16	971	5.8	9.4-6	21	2.2	2.2-7	38	0.7	1.4-5
qp-tripart3	16,380; 1,200	16,380; 16,380	20	1,034	13.1	9.9-6	27	6.6	1.7-7	55	3.7	1.9-5
qp-tripart4	24,815; 1,050	24,815; 24,815	35	5,871	220.0	8.6-6	27	25.9	1.5-7	67	9.6	1.3-4
qp-gridgen1	3,072; 1,025	3,072; 3,072	320	4,081	183.7	6.9-6	58	200.7	1.4-5	208	680.4	5.7-7
qp-pds15	1,812; 2,125	7,756; 7,756	11	1,110	7.4	9.9-6	53	2.2	1.9-5	90	5.8	5.5-2
qp-pds30	3,491; 4,223	16,148; 16,148	11	1,941	37.1	9.4-6	87	7.8	1.4-5	113	24.4	4.9-2
qp-pds60	6,778; 8,423	33,388; 33,388	11	4,685	188.3	9.8-6	78	20.4	1.2-8	134	113.8	2.4-2
qp-pds90	8,777; 12,186	46,161; 46,161	11	3,021	172.6	5.9-6	71	31.1	5.9-9	165	320.6	1.1-2
qp-L2CTA3D	110,000; 1,000	0; 100,000	100	21	16.7	1.7-6	8	577.6	2.8-7	7	14.1	6.4-4
qp-SDC13-t1	10,000; 200	0; 10,000	200	41	3.2	9.0-6	8	87.1	6.4-6	8	2.7	6.5-4
qp-SDC14-t1	20,000; 300	0; 20,000	200	34	6.5	9.2-6	8	298.1	5.6-6	8	5.4	5.5-4
qp-SDC15-t1	40,000; 400	0; 40,000	200	31	10.9	4.7-6	8	2,974.1	4.2-6	7	12.3	9.8-4
qp-SDC16-t1	25,000; 550	0; 25,000	500	41	25.3	4.7-6	8	627.3	5.3-6	8	14.2	7.6-4
qp-SDC17-t1	250,000; 1000	0; 250,000	50	20	15.0	9.9-7	8	1,015.2	2.4-6	7	31.1	8.6-4

Notes. The entry "/" means that the solver runs out of memory, and "*" means the solver is not compatible to solve the problem. Here, "Iter" is the number of outer iterations performed, "Time" is the total runtime in seconds, and "Acc" is the accuracy of the computed solution that is defined in Section 5. Under the column "Acc," we record the accuracy "x.ye-0z" using the format "x.y-z" to save some space.

iteration at a much lower level. We should note that, despite the inferior performance of Gurobi compared with sGS-ADMM, it is still impressive for a general-purpose solver, as it is able to solve the very-large-scale QP problems tested here. For example, the problem qp-SDC17-t1 has 12.5 million variables and 300,000 constraints. Overall, our sGS-ADMM is the most efficient and robust in solving very-large-scale convex quadratic primal block-angular instances tested in our experiments.

5.4. Block-Angular Problems with Nonlinear Convex Objective Functions

In this subsection, we perform numerical experiments on problems having nonlinear convex objective functions and primal block-angular constraints. In particular, on nonlinear multicommodity flow problems that usually arise in transportation and telecommunication, where two commonly used nonlinear convex objective functions are:

$$h(t) = \begin{cases} \sum_{i=1}^{m} f_{\mathrm{Kr}}(t_i; \mathrm{cap}_i), & \text{known as Kleinrock function;} \\ \sum_{i=1}^{m} f_{\mathrm{BPR}}(t_i; \mathrm{cap}_i, r_i), & \text{known as BPR (Bureau of Public Roads) function,} \end{cases}$$

such that

$$f_{\mathrm{Kr}}(\alpha; c) = \begin{cases} \frac{\alpha}{c-\alpha} & \text{if } 0 \le \alpha < c, \\ +\infty & \text{otherwise,} \end{cases} \quad f_{\mathrm{BPR}}(\alpha; c, r) = \begin{cases} r\alpha\left[1 + B\left(\frac{\alpha}{c}\right)^{\beta}\right] & \text{if } \alpha \ge 0\ , \\ +\infty & \text{otherwise.} \end{cases}$$

The Kleinrock function is used to model delay in a telecommunication problem, whereas the BPR function is used to model congestion in a transportation problem. Here, cap_i is the capacity of arc i, r_i is the free flow time of arc i, and β, B are two positive parameters. In general, β is chosen to be four.

In our problem setting, we have

$$\begin{aligned}
&\theta_0(x_0) = h(x_0),\ \theta_i(x_i) = 0,\ \mathcal{A}_0 = I,\ \mathcal{A}_i = -I,\ \forall\, i = 1,\ldots,N,\\
&\mathcal{Q}_i = 0,\ c_i = 0,\ \forall\, i = 0,\ldots,N,;\\
&\mathcal{D}_1 = \mathcal{D}_2 = \cdots = \mathcal{D}_N \text{ is a node-arc incidence matrix,}\\
&b_0 = 0,\ b_i = d_i,\ \forall\, i = 1,\ldots,N \text{ for some demand } d_i \text{ on each commodity } i,\\
&\mathcal{K}_i = \begin{cases} [0,\mathrm{cap}_1]\times[0,\mathrm{cap}_2]\times\cdots\times[0,\mathrm{cap}_{\bar{n}}], & \text{for Kleinrock function;}\\ \mathbb{R}^{\bar{n}}_+, & \text{for BPR function.}\end{cases}
\end{aligned}$$

Following Babonneau and Vial (2009), the data sets we used are the planar and grid problems, which can be downloaded from http://www.di.unipi.it/optimize/Data/MMCF.html#Plnr.

Remark 6. In Step 1(c) of the sGS-ADMM for solving the current class of problems, we need to compute the proximal mapping of $\sigma\theta_0$:

$$\mathrm{Prox}_{\sigma\theta_0}(v) = \mathrm{argmin}\left\{\sigma\theta_0(t) + \frac{1}{2}\|t-v\|^2 \mid t\in\mathbb{R}^{n_0}\right\},\quad v\in\mathbb{R}^{n_0}.$$

Because $\theta_0(t) = \sum_{i=1}^{n_0} f_{\mathrm{Kr}}(t_i;c)$ is separable in the case when we use the Kleinrock function (similarly for the BPR function), the computation of $\mathrm{Prox}_{\sigma\theta_0}(v)$ then amounts to separately solving n_0 strongly convex scalar minimization problems. In our implementation, we solve these scalar problems by Newton's method with Armijo linesearch and warm-start (by using the iterate computed in the previous sGS-ADMM iteration as the starting point). In our experiments, we observe that the computation of the above proximal mapping contributes only to a small percentage of the total runtime.

5.4.1. Numerical Results. In this subsection, we compare our sGS-ADMM algorithm against BlockIP, IPOPT (Wächter and Bielger 2006), and PDCO (https://web.stanford.edu/group/SOL/software/pdco/). PDCO is a primal-dual interior-point method designed for linearly constrained problems with separable convex objective functions, whereas IPOPT is one of the state-of-the-art solvers for solving general nonlinear programming problems. It is an implementation of a primal-dual interior-point algorithm with a filter line-search method for nonlinear programming. At each iteration, it solves a symmetric linear system that is analogous to the case of an interior-point method for solving a convex quadratic-programming problem. The only difference is that the constraint matrices may change from iteration to iteration if the original equality constraints are nonlinear, and the Hessian of the nonlinear objective function may be indefinite if the problem is nonconvex. When the solver is applied to a linearly constrained nonlinear convex problem with bound constraints, the main ingredient at each iteration of IPOPT is, in fact, similar to that of the interior-point method PDCO designed especially for such a class of problems. The reason we include IPOPT and PDCO as benchmarks is the same as why we include Gurobi as a benchmark before. They could be used to judge how much gain a structure-specialized algorithm can obtain compared with a general solver. We use the Kleinrock function as our objective function here.

Table 6 shows that IPOPT is almost always the slowest to solve the test instances. On the other hand, although PDCO is more robust, its performance still lags behind that of sGS-ADMM. For example, PDCO is 8.7 times slower than our algorithm on the problem grid10. It is not surprising for IPOPT and PDCO to perform less efficiently because they are general solvers for nonlinear or convex programs. Lastly, we observe that, even though BlockIP is specially designed for convex block-angular problems, it is not as efficient as PDCO. Moreover, it runs into memory issues when solving almost half of the instances. This is due to the fact that BlockIP uses a preconditioned conjugate gradient method and Cholesky factorization to solve the linear system arising at each iteration of the interior-point method. In the middle of the run, when the PCG method did not converge, the algorithm switches to use Cholesky factorization to solve the linear system, and that often leads to out-of-memory errors. Even when the PCG method works well, BlockIP can still be 10 times slower than our algorithm.

The sGS-ADMM algorithm is able to efficiently solve all the tested nonlinear convex block-angular problems to the required accuracy of 10^{-5} in the relative KKT residual. In particular, it takes only 461 seconds to solve the very-large-scale instance grid10, which has 4.8 million variables and 1.25 million linear constraints.

Table 6. Comparison of Computational Results Between sGS-ADMM, BlockIP, IPOPT, and PDCO for Nonlinear Primal Block-Angular Problem

						sGS-ADMM			BlockIP			IPOPT			PDCO		
Data	m_0	$\bar{m}$	n_0	$\bar{n}$	N	Iter	Time	Acc	Iter	Time	Acc	Iter	Time	Acc	Iter	Time	Acc
grid1	80	24	80	80	50	591	0.23	4.8-6	28	0.06	1.1-3	81	2.08	4.8-5	35	0.37	2.6-6
grid3	360	99	360	360	50	381	0.34	9.2-6	41	0.75	3.4-3	84	10.19	4.7-5	37	1.67	2.7-6
grid5	840	224	840	840	100	581	1.43	9.1-6	—	—	—	85	60.15	4.9-5	42	14.59	2.6-6
grid8	2,400	624	2,400	2,400	500	4,171	145.44	9.9-6	215	1,937.77	2.8-5	/	/	/	60	774.13	9.3-6
grid10	2,400	624	2,400	2,400	2,000	3,432	460.97	9.9-6	221	16,628.50	3.8-5	/	/	/	73	40,27.51	9.4-6
planar30	150	29	150	150	92	431	0.25	1.0-5	93	0.73	1.3-3	91	4.49	6.7-5	27	0.57	5.1-6
planar80	440	79	440	440	543	1,875	12.95	1.0-5	—	—	—	402	673.86	7.3-5	36	30.12	8.3-6
planar100	532	99	532	532	1,085	2,614	42.55	1.0-5	—	—	—	114	585.51	3.1-4	41	91.64	7.3-6

Notes. The entry "—" means that the solver encounters memory issue, while the entry"/" means that the solver could not solve the problem within the time limit of five hours. Here, "Iter" is the number of outer iterations performed, "Time" is the total runtime in seconds, and "Acc" is the accuracy of the computed solution that is defined in Section 5. Under the column "Acc," we record the accuracy "x.ye-0z" using the format "x.y-z" to save some space.

6. Conclusion

We have designed a semi-proximal augmented Lagrangian-based decomposition method for directly solving the primal form of a convex composite quadratic conic-programming problem with a primal block-angular structure. One of the variants of the method is shown to be closely related to the well-known DQA method of Mulvey and Ruszczyński. We have also proposed an inexact symmetric Gauss–Seidel ADMM for solving the corresponding dual problem. A systematic comparison between the primal-based semi-proximal augmented Lagrangian decomposition algorithms and the dual-based sGS-ADMM has shown that the latter is numerically much more efficient. Numerical experiments on evaluating the performance of our sGS-ADMM against state-of-the-art solvers have shown that our sGS-ADMM algorithm is especially efficient for large instances with convex quadratic objective functions. As a future project, we plan to implement our algorithm for solving semidefinite programming problems with a primal block-angular structure. Also, it would be natural to explore the implementation of the sGS-ADMM on a better parallel computing and programming platform to fully realize its potential.

Acknowledgments

The authors thank Professor Jordi Castro for sharing his solver BlockIP, so that they are able to evaluate the performance of their algorithm more comprehensively, and for providing the test instances he has taken great effort to generate over the years when developing BlockIP. Thanks also go to the Optimization Group at the Department of Computer Science of the University of Pisa for collecting/generating several suites of test data and making them publicly available. The authors also thank the referees and the Associate Editor for their constructive suggestions to improve the presentation of the paper.

Appendix: Relation Between ALM-DQA-Mod and the DQA Method of Ruszczyński

Here, we elucidate the connection between ALM-DQA-mod in Section 3.2 and the DQA method proposed in Ruszczyński (1989) and Mulvey and Ruszczyński (1992). Given $\hat{x}_i^s \in F_i$, we can parameterize a given x_i as

$$x_i = \hat{x}_i^s + \rho d_i = (1-\rho)\hat{x}_i^s + \rho(\hat{x}_i^s + d_i),\ i = 0, 1, \ldots, N,$$

with $\rho = (N+1)^{-1} \in (0,1]$. Then, by convexity, $f_i(x_i) \le (1-\rho)f_i(\hat{x}_i^s) + \rho f_i(\hat{x}_i^s + d_i)$. Also, if $\hat{x}_i^s + d_i \in F_i$, then, by the convexity of the set F_i and the fact that $\hat{x}_i^s \in F_i$, we know that $x_i \in F_i$. From here, we have that for all $x \in F_0 \times F_1 \times \cdots \times F_N$,

$$\begin{aligned}
&L_\sigma(x; y_0^k) + \frac{\sigma}{2}\|x - \hat{x}^s\|_{\widehat{\mathcal{T}}}^2 + \frac{1}{2\sigma}\|y_0^k\|^2 \\
&= \sum_{i=0}^N f_i(x_i) + \frac{\sigma}{2}\|\mathcal{A}x - b_0 - \sigma^{-1}y_0^k\|^2 - \frac{1}{2\sigma}\|y_0^k\|^2 + \frac{\sigma}{2}\|x - \hat{x}^s\|_{\widehat{\mathcal{T}}}^2 + \frac{1}{2\sigma}\|y_0^k\|^2 \\
&\le (1-\rho)\sum_{i=0}^N f_i(\hat{x}_i^s) + \rho\sum_{i=0}^N f_i(\hat{x}_i^s + d_i) + \frac{\sigma}{2}\|\mathcal{A}(\hat{x}^s + \rho d) - b_0 - \sigma^{-1}y_0^k\|^2 + \frac{\sigma\rho^2}{2}\|d\|_{\widehat{\mathcal{T}}}^2 \\
&= \sum_{i=0}^N \left[\rho f_i(\hat{x}_i^s + d_i) + \rho\langle d_i, \hat{\alpha}_i^s\rangle + \frac{\sigma\rho^2}{2}\langle d_i, \mathcal{E}_i d_i\rangle\right] + (1-\rho)\sum_{i=0}^N f_i(\hat{x}_i^s) + \frac{\sigma}{2}\|\mathcal{A}\hat{x}^s - b_0 - \sigma^{-1}y_0^k\|^2. \qquad \text{(A.1)}
\end{aligned}$$

Hence, instead of (22) in ALM-DQA-mod, we may consider to minimize the majorization of $L_\sigma(x; y_0^k) + \frac{\sigma}{2}\|x - \hat{x}^s\|_{\widehat{\mathcal{T}}}^2$ in (A.1), and compute for $i = 0, 1, \ldots, N$,

$$d_i^{s+1} = \underset{d_i}{\operatorname{argmin}} \, \rho\Big\{ f_i(\hat{x}_i^s + d_i) + \frac{\sigma\rho}{2}\langle d_i, \mathcal{E}_i d_i\rangle + \langle d_i, \hat{\alpha}_i^s\rangle \mid \hat{x}_i^s + d_i \in F_i,\ d_i \in \mathcal{X}_i \Big\}. \tag{A.2}$$

We obtain the DQA method of Ruszczyński (1989) if we compute d^{s+1} exactly in the above Subproblem (A.2), and set

$$\hat{x}_i^{s+1} = \hat{x}_i^s + \rho d_i^{s+1}, \quad i = 0, 1, \ldots, N,$$

instead of the solution in (22). Thus, we may view the DQA method in Ruszczyński (1989) as an augmented Lagrangian method for which the subproblem in Step 1 of Algorithm ALM is solved by a majorized proximal gradient method with the proximal term chosen to be $\frac{\sigma}{2}\|x - \hat{x}^s\|_{\widehat{\mathcal{T}}}^2$ at each step.

Remark A.1. When the $\mathcal{A}_i$'s are matrices, the majorization $\mathcal{A}^*\mathcal{A} \preceq \operatorname{diag}(\mathcal{E}_0, \ldots, \mathcal{E}_N)$ can be improved as follows, as has been done in Chatzipanagiotis et al. (2015). Define the vector e_j to be the j^{th} column of the identity matrix. Let

$$I_j = \Big\{ i \in \{0, 1, \ldots, N\} \mid e_j^T \mathcal{A}_i \neq 0 \Big\}, \quad \chi := \max\big\{ |I_j| \mid j = 1, \ldots, m \big\} \ \le\ N + 1 = \rho^{-1}.$$

Then,

$$\begin{aligned}
\|\mathcal{A}x\|^2 = \Big\| \sum_{i=0}^N \mathcal{A}_i x_i \Big\|^2 &= \sum_{j=1}^m \Big| \sum_{i=0}^N e_j^T \mathcal{A}_i x_i \Big|^2 = \sum_{j=1}^m \Big| \sum_{i \in I_j} e_j^T \mathcal{A}_i x_i \Big|^2 \\
&\le \sum_{j=1}^m \Big(|I_j| \sum_{i \in I_j} \big| e_j^T \mathcal{A}_i x_i \big|^2 \Big) \le \chi \sum_{j=1}^m \sum_{i \in I_j} \big| e_j^T \mathcal{A}_i x_i \big|^2 \\
&= \chi \sum_{j=1}^m \sum_{i=0}^N \big| e_j^T \mathcal{A}_i x_i \big|^2 = \chi \sum_{i=0}^N \|\mathcal{A}_i x_i\|^2.
\end{aligned}$$

That is, $\mathcal{A}^*\mathcal{A} \preceq \operatorname{diag}(\chi)\mathcal{A}_0^*\mathcal{A}_0, \cdots, \chi\mathcal{A}_N^*\mathcal{A}_N)$. It is straightforward to incorporate the improvement into ALM-DQA-mod by simply replacing $\mathcal{E}_i = \rho^{-1}\mathcal{A}_i^*\mathcal{A}_i$ in (21) by $\chi\mathcal{A}_i^*\mathcal{A}_i$ for each $i = 0, 1, \ldots, N$.

References

Assad A (1978) Multicommodity network flows—A survey. *Networks* 8(1):37–91.

Babonneau F, Vial JP (2009) ACCPM with a nonlinear constraint and an active set strategy to solve nonlinear multicommodity flow problems. *Math. Programming* 120:179–210.

Bertsekas DP (1975) On the method of multipliers for convex programming. *IEEE Trans. Automatic Control* 20(3):385–388.

Bertsekas DP, Gallager RG (1987) *Data Networks* (Prentice-Hall, Upper Saddle River, NJ).

Birge JR, Qi L (1988) Computing block-angular Karmarkar projections with applications to stochastic programming. *Management Sci.* 34(12):1472–1479.

Castro J (2005) Quadratic interior-point methods in statistical disclosure control. *Comput. Management Sci.* 2:107–121.

Castro J (2016) Interior-point solver for convex separable block-angular problems. *Optim. Methods Software* 31(1):88–109.

Castro J, Cuesta J (2011) Quadratic regularizations in an interior-point method for primal block-angular problems. *Math. Programming* 130:415–445.

Chatzipanagiotis N, Dentcheva D, Zavlanos MM (2015) An augmented Lagrangian method for distributed optimization. *Math. Programming* 152:405–434.

Chen L, Sun DF, Toh KC (2017) An efficient inexact symmetric Gauss-Seidel based majorized ADMM for high-dimensional convex composite conic programming. *Math. Programming* 161:237–270.

Chen L, Li XD, Sun DF, Toh KC (2019) On the equivalence of inexact proximal ALM and ADMM for a class of convex composite programming. *Math. Programming*, ePub ahead of print August 26, https://doi.org/10.1007/s10107-019-01423-x.

Choi IC, Goldfarb D (1993) Exploiting special structure in a primal-dual path following algorithm. *Math. Programming* 58:33–52.

Fratta L, Gerla M, Kleinrock L (1973) The flow deviation method: An approach to store-and-forward communication network design. *Networks* 3(2):97–133.

Glowinski R (1984) *Numerical Methods for Nonlinear Variational Problems* (Springer-Verlag, New York).

Gondzio J, Grothey A (2009) Exploiting structure in parallel implementation of interior point methods for optimization. *Comput. Management Sci.* 6:135–160.

Gondzio J, Sarkissian R (2003) Parallel interior point solver for structured linear programs. *Math. Programming* 96(3):561–584.

Gondzio J, Sarkissian R, Vial JP (1997) Using an interior point method for the master problem in a decomposition approach. *Eur. J. Oper. Res.* 101(3):577–587.

Hanasusanto GA, Kuhn D (2018) Conic programming reformulations of two-stage distributionally robust linear programs over Wasserstein balls. *Oper. Res.* 66(3):849–869.

Hundepool A, Domingo-Ferrer J, Franconi L, Giessing S, Nordholt ES, Spicer K, de Wolf PP (2012) *Statistical Disclosure Control* (Wiley, New York).

Karush W (1939) Minima of functions of several variables with inequalities as side constraints. Unpublished M.Sc. dissertation, Department of Mathematics, University of Chicago, Chicago.

Kontogiorgis S, De Leone R, Meyer RR (1996) Alternating direction splitting for block angular parallel optimization. *J. Optim. Theory Appl.* 99:1–29.

Kuhn HW, Tucker AW (1951) Nonlinear programming. Neyman J, ed. *Proc. 2nd Berkeley Sympos. Math. Statist. Probab.* (University of California Press, Berkeley), 481–492.

Lam XY, Sun DF, Toh KC (2018) A semi-proximal augmented Lagrangian based decomposition method for primal block angular convex composite quadratic conic programming problems. Preprint, submitted December 12, https://arxiv.org/abs/1812.04941.

LeBlanc L (1973) Mathematical programming algorithms for large scale network equilibrium and network design problems. Unpublished Ph.D. thesis, IE/MS Department, Northwestern University, Evanston, IL.

Li XD, Sun DF, Toh KC (2016) A Schur complement based semi-proximal ADMM for convex quadratic conic programming and extensions. *Math. Programming* 155:333–373.

Li XD, Sun DF, Toh KC (2018) A highly efficient semismooth Newton augmented Lagrangian method for solving Lasso problems. *SIAM J. Optim.* 28(1):433–458.

Li XD, Sun DF, Toh KC (2019) A block symmetric Gauss-Seidel decomposition theorem for convex composite quadratic programming and its applications. *Math. Programming* 175(1-2):395–418.

Mehrotra S, Ozevin MG (2007) Decomposition-based interior point methods for two-stage stochastic semidefinite programming. *SIAM J. Optim.* 18(1):206–222.

Mehrotra S, Ozevin MG (2009) Decomposition based interior point methods for two-stage stochastic convex quadratic programs with recourse. *Oper. Res.* 57(4):964–974.

Mulvey JM, Ruszczyński A (1992) A diagonal quadratic approximation method for large scale linear programs. *Oper. Res. Lett.* 12(4):205–215.

Ouorou A, Mahey P, Vial JP (2000) A survey of algorithms for convex multicommodity flow problems. *Management Sci.* 46(1):126–147.

Rockafellar RT, Wets RJB (1991) Scenarios and policy aggregation in optimization under uncertainty. *Math. Oper. Res.* 16(1):119–147.

Ruszczyński A (1986) A regularized decomposition method for minimizing a sum of polyhedral functions. *Math. Programming* 35:309–333.

Ruszczyński A (1989) An augmented Lagrangian decomposition method for block diagonal linear programming problems. *Oper. Res. Lett.* 8(5):287–294.

Ruszczyński A (1995) On convergence of an augmented Lagrangian decomposition method for sparse convex optimization. *Math. Oper. Res.* 20(3):634–656.

Ruszczyński A (1999) Some advances in decomposition methods for stochastic linear programming. *Ann. Oper. Res.* 85:153–172.

Schultz G, Meyer RR (1991) An interior point method for block angular optimization. *SIAM J. Optim.* 1(4):583–602.

Sivaramakrishnan K (2010) A parallel interior point decomposition algorithm for block angular semidefinite programs. *Comput. Optim. Appl.* 46:1–29.

Sun J, Liu XW (2006) Scenario formulation of stochastic linear programs and the homogeneous self-dual interior-point method. *INFORMS J. Comput.* 18(4):444–454.

Todd MJ (1988) Exploiting special structure in Karmarkar's linear programming algorithm. *Math. Programming* 41:81–103.

Wächter A, Bielger LT (2006) On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Math. Programming* 106(1):25–57.

Zhang N, Wu J, Zhang L (2018) A linearly convergent majorized ADMM with indefinite proximal terms for convex composite programming and its applications. Preprint, submitted June 6, https://arxiv.org/abs/1706.01698.

Zhao G (1999) Interior-point methods with decomposition for solving large-scale linear programs. *Optim. Theor. Appl.* 102:169–192.

Zhao G (2001) A Log-Barrier method with Benders decomposition for solving two-stage stochastic programs. *Math. Programming* 90:507–536.

Zhao G (2005) A Lagrangian dual method with self-concordant barrier for multi-stage stochastic convex programming. *Math. Programming* 102:1–24.

Zhu Y, Ariyawansa KA (2011) A preliminary set of applications leading to stochastic semidefinite programs and chance-constrained semidefinite programs. *Appl. Math. Model.* 35(5):2425–2442.