

# A semi-proximal augmented Lagrangian based decomposition method for primal block angular convex composite quadratic conic programming problems

Xin-Yee Lam <sup>\*</sup>, Defeng Sun <sup>†</sup>, Kim-Chuan Toh <sup>‡</sup>

December 13, 2018

## Abstract

We propose a semi-proximal augmented Lagrangian based decomposition method for convex composite quadratic conic programming problems with primal block angular structures. 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. Moreover, we are able to derive novel enhancements and generalizations of these well known methods. We also propose a semi-proximal symmetric Gauss-Seidel based alternating direction method of multipliers for solving the corresponding dual problem. Numerical results show that our algorithms can perform well even for very large instances of primal block angular convex QP problems. For example, one instance with more than 300,000 linear constraints and 12,500,000 nonnegative variables is solved in less than a minute whereas Gurobi took more than 3 hours, and another instance `qp-gridgen1` with more than 331,000 linear constraints and 986,000 nonnegative variables is solved in about 5 minutes whereas Gurobi took more than 35 minutes.

## 1 Introduction

In this paper, we will focus on solving convex composite quadratic conic programming problems with a primal block angular structure, i.e. optimization problems with a separable

---

<sup>\*</sup>Department of Mathematics, National University of Singapore, 10 Lower Kent Ridge Road, Singapore 119076 ([matttohkc@math.nus.edu.sg](mailto:matttohkc@math.nus.edu.sg)).

<sup>†</sup>Department of Applied Mathematics, The Hong Kong Polytechnic University, Hung Hom, Hong Kong ([defeng.sun@polyu.edu.hk](mailto:defeng.sun@polyu.edu.hk)).

<sup>‡</sup>Department of Mathematics, and Institute of Operations Research and Analytics, National University of Singapore, 10 Lower Kent Ridge Road, Singapore 119076 ([matttohkc@math.nus.edu.sg](mailto:matttohkc@math.nus.edu.sg)). This author's research is supported in part by the Ministry of Education, Singapore, Academic Research Fund under Grant R-146-000-257-112.

convex objective function and conic constraints but the variables are coupled by linking linear constraints across different variables. Without specially designed strategies to exploit the underlying block angular structure, computational inefficiency of an algorithm will be severe 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 [1] and statistical disclosure control [23]. These problems are often very large scale in practice, and standard interior point methods such as those implemented in Gurobi or Mosek may not be efficient enough to solve such problems. 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, see for example [33, 39, 40, 41, 43, 44]; (ii) interior-point log-barrier Lagrangian decomposition methods such as those studied in [53, 54, 55, 31, 32]; and (iii) standard interior-point methods which incorporate novel numerical linear algebraic techniques to exploit the underlying block angular structures when solving the large linear systems arising in each iteration, for example in [9, 16, 19, 46, 50].

Besides quadratic and linear problems, semidefinite programming (SDP) problems with primal block angular structures are beginning to appear in the literature more frequently. It is gaining more attention as practitioners become more sophisticated in using SDP to model their application problems. For example, the authors in [21] reformulated a two-stage distributionally robust linear program as a completely positive cone program which bears a block angular structure and applied the reformulation to solve a multi-item newsvendor problem. Although linear programming problems with primal block angular structures have been studied extensively, the more complicated SDP version is still in its infancy stage. Apart from [31], [48] and [56], we are not aware of other works.

By focusing on designing efficient algorithms for solving general conic programming problems with primal block angular structures, we can in general also use the same algorithmic framework to solve the primal block angular linear and quadratic programming problems efficiently through designing novel numerical linear algebraic techniques to exploit the underlying structures. In this paper, our main objective is to design efficient and robust (distributed) algorithms for solving large scale conic programming problems with block angular structures. Specifically, we will design an inexact semi-proximal augmented Lagrangian method (ALM) for the primal problem which attempts to exploit the block angular structure to solve the problem in parallel. Our algorithm is motivated by the recent theoretical advances in inexact semi-proximal ALM that is embedded in [15]. In contrast to most existing augmented Lagrangian based decomposition algorithms where the solution for each subproblem must be computed exactly or to very high accuracy, our algorithm has the key advantage of allowing the subproblems to be solved approximately with progressive accuracy. We will also elucidate the connection of our algorithm to the well-known diagonal quadratic approximation (DQA) algorithm of Mulvey and Ruszczyński [33].

In the pioneering work in [25], an ADMM based framework was designed for the primal block angular problem (P) wherein the variables are duplicated and auxiliary variables are introduced to make the first ADMM subproblem in each iteration solvable in a distributed fashion and that the succeeding second ADMM subproblem is a sufficiently simple quadratic

program which is assumed to be easy to solve. However, the problem might still be difficult to solve if the scale of the original problem gets 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 in each iteration in the primal approach, in this paper we will adopt the dual approach of solving (P). 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 in each iteration. We note that the sGS-ADMM is an algorithm designed based on the recent advances in [15]; more details will be presented later.

We consider the following primal block-angular optimization problem:

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

where for each  $i = 0, 1, \dots, N$ ,  $\theta_i : \mathcal{X}_i \rightarrow (-\infty, \infty]$  is a proper closed convex function,  $\mathcal{Q}_i : \mathcal{X}_i \rightarrow \mathcal{X}_i$  is a positive semidefinite linear operator,  $\mathcal{A}_i : \mathcal{X}_i \rightarrow \mathcal{Y}_0$  and  $\mathcal{D}_i : \mathcal{X}_i \rightarrow \mathcal{Y}_i$  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  $\| \cdot \|$ . Note that the addition of the proper closed convex functions in the objective gives us the flexibility to add nonsmooth terms such as  $\ell_1$  regularization terms. 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  $(x_i; s_i) \in \mathcal{K}_i \times \mathcal{C}_i$ .

Without loss of generality, we assume that the constraint matrix  $\mathcal{B}$  in (P) has full row-rank. Let  $n_i = \dim(\mathcal{X}_i)$  and  $m_i = \dim(\mathcal{Y}_i)$ . Observe that the problem (P) has  $\sum_{i=0}^N m_i$  linear constraints and the dimension of the decision variable is  $\sum_{i=0}^N n_i$ . Thus even if  $m_i$  and/or  $n_i$  are moderate numbers, the overall dimension of the problem can easily get very large 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  $i = 0, \dots, N$ , the Dantzig-Wolfe decomposition method (which may be viewed as a dual method based on the Lagrangian function  $\sum_{i=0}^N \langle c_i, x_i \rangle - \langle u, b_0 - \sum_{i=0}^N \mathcal{A}_i x_i \rangle$ ) is a well known classical approach for solving such a problem. The Dantzig-Wolfe decomposition method has the attractive property that in each iteration,  $x_i$  can be computed individually from a smaller linear program (LP) for  $i = 1, \dots, N$ . However, it is generally acknowledged that an augmented Lagrangian approach has a number of important

advantages over the usual Lagrangian dual method. For example, Ruszczyński stated in [43] that the dual approach based on the ordinary Lagrangian can suffer from the nonuniqueness of the solutions of subproblems. In addition, solving the subproblem of the augmented Lagrangian approach would be more stable. In that paper, the well-known diagonal quadratic approximation (DQA) method is introduced. The DQA method is a very successful decomposition method and it has been a popular tool in stochastic programming. Thus it would be a worthwhile effort to analyse it again 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. Our second contribution is in the design and implementation of a specialized algorithm for solving the dual problem of (P). The algorithm is easy to implement and highly amenable to parallelization. Hence we expect it to be highly scalable for solving large scale problems with million of variables and constraints. Finally, we have proposed efficient implementations of the algorithms and conducted comprehensive numerical experiments to evaluate the performance of our algorithms against highly competitive state-of-the-art solvers in solving the problems (P) and (D).

This paper is organized as follows. We will derive the dual of the primal block angular problem (P) in section 2. In section 3, we will present our inexact semi-proximal augmented Lagrangian methods for the primal problem (P). In section 4, we will propose a semi-proximal symmetric Gauss-Seidel based ADMM for the dual problem of (P). For all algorithms we introduce, we conduct numerical experiments to evaluate their performance and the numerical results are reported in section 3.3 and section 5. We conclude the paper in the final section.

## 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 co-domains.
- For any linear map  $\mathcal{T} : \mathcal{X} \rightarrow \mathcal{Y}$ , 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 have the notation  $\|x\|_{\mathcal{T}} := \sqrt{\langle x, \mathcal{T}x \rangle}$ .
- Let  $f : \mathcal{X} \rightarrow (-\infty, +\infty]$  be an arbitrary closed proper convex function. We denote  $\text{dom} f$  as its effective domain and  $\partial f$  as its subdifferential mapping. The Fenchel conjugate function of  $f$  is denoted as  $f^*$ .
- The Moreau-Yosida proximal mapping of  $f$  is defined by  $\text{Prox}_f(y) := \arg \min_x \{f(x) + \frac{1}{2}\|x - y\|^2\}$ .

## 2 Derivation of the dual of (P)

For notational convenience, we define

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

For each  $x \in \mathcal{X}$ ,  $y \in \mathcal{Y}$ , and  $c \in \mathcal{X}$ ,  $b \in \mathcal{Y}$ , we can express them as

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

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

$$\mathcal{A} = [\mathcal{A}_0, \mathcal{A}_1, \dots, \mathcal{A}_N], \quad \mathcal{Q}(x) = (\mathcal{Q}_0(x_0); \mathcal{Q}_1(x_1); \dots; \mathcal{Q}_N(x_N)), \quad \theta(x) = \sum_{i=0}^N \theta_i(x_i). \quad (3)$$

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

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

By introducing auxiliary variables  $u, v \in \mathcal{X}$ , problem (4) can equivalently be written as the following model:

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

To derive the dual of (4), consider the following Lagrangian function for (5):

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

where  $x, u, v, s, z \in \mathcal{X}$ ,  $y \in \mathcal{Y}$ . Now for a given subspace  $\mathcal{W} \subset \mathcal{X}$  containing  $\text{Range}(\mathcal{Q})$ , the range space of  $\mathcal{Q}$ , we have

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

Also,

$$\begin{aligned} \inf_u \mathcal{L}(x, u, v; y, s, z) &= \inf_u [\theta(u) + \langle s, u \rangle] = -\theta^*(-s); \\ \inf_v \mathcal{L}(x, u, v; y, s, z) &= \inf_v [\delta_{\mathcal{K}}(v) + \langle z, v \rangle] = -\delta_{\mathcal{K}}^*(-z). \end{aligned}$$

Hence the dual of (5) is given by

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

or equivalently,

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

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

$$\delta_{\mathcal{K}}^*(-z) = \sum_{i=0}^N \delta_{\mathcal{K}_i}^*(-z_i), \quad \boldsymbol{\theta}^*(-s) = \sum_{i=0}^N \theta_i^*(-s_i). \tag{7}$$

Assume 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:

$$\begin{cases} \mathcal{B}x - b = 0, \\ -\mathcal{Q}w + \mathcal{B}^*y + s + z - c = 0, \quad \mathcal{Q}w - \mathcal{Q}x = 0, \quad w \in \mathcal{W}, \\ -s \in \partial\boldsymbol{\theta}(x) \Leftrightarrow x - \text{Prox}_{\boldsymbol{\theta}}(x - s) = 0, \\ x - \Pi_{\mathcal{K}}(x - z) = 0. \end{cases} \tag{8}$$

By applying the structures in (1)–(3) and (7) to (6), we get explicitly the dual of (P):

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

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

### 3 Inexact semi-proximal augmented Lagrangian methods for the primal problem (P)

First we rewrite (P) in the following form:

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

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

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

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 in each iteration.

**Step 1.**  $x^{k+1} \approx \operatorname{argmin}_x L_\sigma(x; y_0^k)$ .

**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.

As one may observe from Step 1 of the ALM, an undesirable feature in the method is that it destroys the separable structure in the Dantzig-Wolfe decomposition method. Although the feasible sets for the  $x_i$ 's are separable, the objective function has a quadratic term which couples all the  $x_i$ 's.

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

$$L_\sigma(x; y_0^k) + \frac{\sigma}{2} \|x - x^k\|_{\mathcal{T}}^2,$$

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 the ALM while at the same time the added proximal term should be as small as possible. In this paper, we choose  $\mathcal{T}$  to be the following positive semidefinite linear operator:

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

where  $\mathcal{J}_i \succeq \beta_i I + \mathcal{A}_i^* \mathcal{A}_i$ , with  $\beta_i = \sum_{j=0, j \neq i}^N \|\mathcal{A}_i^* \mathcal{A}_j\|_2$  for each  $i = 0, 1, \dots, N$ . Such a choice is generally less conservative than the usual choice of  $\widehat{\mathcal{T}}$  given in (20). It is especially a good choice when  $\mathcal{A}_i$  and  $\mathcal{A}_j$  are nearly orthogonal to one another for most of the index pairs  $(i, j)$ .

With the choice in (12), we get

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

The inexact semi-proximal ALM (sPALM) we consider for solving the primal block angular problem (P) through (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 in each iteration.

**Step 1.** Compute

$$x^{k+1} \approx \widehat{x}^{k+1} := \operatorname{argmin}\{L_\sigma(x; y_0^k) + \frac{\sigma}{2}\|x - x^k\|_{\mathcal{T}}^2\}, \quad (13)$$

with residual

$$d^{k+1} \in \partial_x L_\sigma(x^{k+1}; y_0^k) + \sigma\mathcal{T}(x^{k+1} - x^k), \quad (14)$$

satisfying  $\|d^{k+1}\| \leq \varepsilon_k$ . Let  $\mathcal{G}_i = \mathcal{Q}_i + \sigma\mathcal{J}_i$ ,  $g_i^k = \mathcal{Q}_i x_i^k + c_i + \sigma\mathcal{A}_i^*(Ax^k - b_0 - \sigma^{-1}y_0^k) - \mathcal{G}_i x_i^k$ . Due to the separability of the variables in (13) because of the specially chosen  $\mathcal{T}$ , one can compute **in parallel** for  $i = 0, 1, \dots, N$ ,

$$x_i^{k+1} \approx \widehat{x}_i^{k+1} := \operatorname{argmin}\left\{\theta_i(x_i) + \frac{1}{2}\langle x_i, \mathcal{G}_i x_i \rangle + \langle g_i^k, x_i \rangle \mid x_i \in F_i\right\}, \quad (15)$$

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

$$\|d_i^{k+1}\| \leq \frac{1}{\sqrt{N+1}}\varepsilon_k. \quad (16)$$

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

Observe that with the introduction of the semi-proximal term  $\frac{\sigma}{2}\|x - x^k\|_{\mathcal{T}}^2$  to the augmented Lagrangian function in Step 1 of the 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 analyse the stopping condition for (15) in detail. In particular, we need to find  $v_i^{k+1} \in \partial(\theta_i + \delta_{F_i})(x_i^{k+1})$  for  $i = 0, 1, \dots, N$ . This can be done by considering the dual of the subproblem (15), which could be written as:

$$\begin{aligned} & - \min \quad \theta_i^*(-s_i) + \frac{1}{2}\langle w_i, \mathcal{G}_i w_i \rangle - \langle b_i, y_i \rangle + \delta_{\mathcal{K}_i}^*(-z_i) \\ & \text{s.t.} \quad -\mathcal{G}_i w_i + \mathcal{D}_i^* y_i + s_i + z_i = g_i^k, \\ & \quad \quad s_i \in \mathcal{X}_i, \quad y_i \in \mathcal{Y}_i, \quad w_i \in \mathcal{W}_i, \quad i = 1, \dots, N. \end{aligned} \quad (17)$$

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  $i = 1, \dots, N$ , the case for  $i = 0$  can be derived similarly. One can estimate  $v_i^{k+1}$  to be  $-\mathcal{D}_i^* y_i^{k+1} - s_i^{k+1} - z_i^{k+1}$  for a



computed dual solution  $(y_i^{k+1}, s_i^{k+1}, z_i^{k+1})$  and the residual  $d_i^{k+1}$  is simply the residual in the dual feasibility constraint in the above problem.

**Remark 3.1** *In the sPALM, 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 get*

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

Note that for convenience, we introduced  $\mathcal{D}_0^* = 0$ . From the KKT conditions for (15) and (17), we have that

$$\begin{aligned} -\mathcal{G}_i w_i^{k+1} + \mathcal{D}_i^* y_i^{k+1} + s_i^{k+1} + z_i^{k+1} - g_i^k &=: R_i^d \approx 0, \\ \mathcal{G}_i w_i^{k+1} - \mathcal{G}_i x_i^{k+1} &\approx 0. \end{aligned}$$

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

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

Note that the right-hand-side quantity in the above equation will converge to 0 based on the convergence of sPALM and the KKT conditions for (15) and (17). Thus by using the dual variables computed from solving (17), we can generate the dual variables for (D).

### 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 [14]. To do that, we need to first reformulate (10) into the form required in [14] as follows:

$$\min \left\{ h(x) + \psi(x) \mid \mathcal{A}x = b_0, x = (x_0; x_1; \dots; x_N) \in \mathcal{X} \right\}, \quad (18)$$

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

$$\mathcal{R}(x, y_0) = \begin{pmatrix} b_0 - \mathcal{A}x \\ x - \text{Prox}_{\psi}(x - \mathcal{Q}x - c - \mathcal{A}^* y_0) \end{pmatrix} \quad \forall x \in \mathcal{X}, y_0 \in \mathcal{Y}_0. \quad (19)$$

Note that  $(x, y_0)$  is a solution of the KKT system of (18) if and only  $\mathcal{R}(x, y_0) = 0$ .

Now we state the global convergence theorem here for the convenience of the readers. Define the self-adjoint positive definite linear operator  $\mathcal{V} : \mathcal{X} \rightarrow \mathcal{X}$  by

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

We have the following convergence result for the inexact sPALM.

**Theorem 3.1** Assume that the solution set to the KKT system of (10) is nonempty and  $(\bar{x}, \bar{y}_0)$  is a solution. Then, the sequence  $\{(x^k, y_0^k)\}$  generated by sPALM is well-defined such that for any  $k \geq 1$ ,

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

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

$$\begin{aligned} & \left( \|x^{k+1} - \bar{x}\|_{\hat{\mathcal{V}}}^2 + \|y_0^{k+1} - \bar{y}_0\|^2 \right) - \left( \|x^k - \bar{x}\|_{\hat{\mathcal{V}}}^2 + \|y_0^k - \bar{y}_0\|^2 \right) \\ & \leq - \left( \frac{2-\tau}{3\tau} \|y_0^k - y_0^{k+1}\|^2 + \|x^{k+1} - x^k\|_{\hat{\mathcal{V}}}^2 - 2\tau\sigma \langle d^k, x^{k+1} - \bar{x} \rangle \right), \end{aligned}$$

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

*Proof.* The result can be proved directly from the convergence result in [14, Theorem 1].  $\square$

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  $\kappa$  and  $r$  such that  $\text{dist}((x, y_0), \Omega) \leq \kappa \|\mathcal{R}(x, y_0)\|$  for all  $(x, y_0)$  satisfying  $\|(x, y_0) - (x^*, y_0^*)\| \leq r$ , where  $\Omega$  is the solution set of (18) and  $(x^*, y_0^*)$  is a particular solution of (18). In order to save some space, we will not state the theorem here but refer the reader to [14, Theorem 2].

### 3.2 Comparison of sPALM with the diagonal quadratic approximation method and its recent variants

Let  $\rho := (N+1)^{-1}$ . Consider the following linear operator

$$\hat{\mathcal{T}} = \text{diag}(\mathcal{E}_0, \dots, \mathcal{E}_N) - \mathcal{A}^* \mathcal{A}, \quad (20)$$

where  $\mathcal{E}_i \succeq \rho^{-1} \mathcal{A}_i^* \mathcal{A}_i$  for all  $i = 0, 1, \dots, N$ . It is not difficult to show that  $\hat{\mathcal{T}} \succeq 0$ .

If instead of (12), we choose  $\mathcal{T}$  to be the linear operator given in (20), then instead of sPALM, we get the following variant of the inexact sPALM.

**sPALM-b.** 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 in each iteration.

**Step 1.** Let  $g_i^k = \mathcal{Q}_i x_i^k + c_i + \sigma \mathcal{A}_i^* (\mathcal{A} x^k - b_0 - \sigma^{-1} y_0^k) - (\mathcal{Q}_i + \sigma \mathcal{E}_i) x_i^k$ . Compute (in parallel) for  $i = 0, 1, \dots, N$ ,

$$x_i^{k+1} \approx \underset{x_i \in F_i}{\text{argmin}} \left\{ \theta_i(x_i) + \frac{1}{2} \langle x_i, (\mathcal{Q}_i + \sigma \mathcal{E}_i) x_i \rangle + \langle g_i^k, x_i \rangle \right\}, \quad (21)$$

with the residual  $d_i^{k+1} := v_i^{k+1} + (\mathcal{Q}_i + \sigma \mathcal{E}_i) x_i^{k+1} + g_i^k$  for some  $v_i^{k+1} \in \partial(\theta_i + \delta_{F_i})(x_i^{k+1})$  and satisfying  $\|d_i^{k+1}\| \leq \frac{1}{\sqrt{N+1}} \varepsilon_k$ .

**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.

In [41], Ruszczyński proposed the diagonal quadratic approximation (DQA) augmented Lagrangian method that aims to solve a problem of the form (P). As already mentioned, the DQA method is a very successful decomposition method that is frequently used in stochastic programming. Although it was not derived in our way in [41], we shall see later that the DQA method can roughly be derived as the augmented Lagrangian method described in **ALM** where the minimization problem in Step 1 is solved approximately by a proximal gradient method, with the proximal term chosen specially using the linear operator  $\widehat{\mathcal{T}}$  in (20) to make the resulting subproblem separable.

**ALM-DQA-mod.** Given  $\sigma > 0$ ,  $y_0^0 \in \mathcal{Y}_0$  and  $x^0 \in \mathcal{X}$ . Let  $\{\varepsilon_k\}$  be a given summable sequence of nonnegative numbers. Perform the following steps in each iteration.

**Step 1.** Starting with  $\hat{x}^0 = x^k$ , iterate the following step for  $s = 0, 1, \dots$  until convergence:

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

$$\begin{aligned} \hat{x}_i^{s+1} &\approx \operatorname{argmin}\{f_i(x_i) + \frac{\sigma}{2}\langle x_i - \hat{x}_i^s, \mathcal{E}_i(x_i - \hat{x}_i^s) \rangle + \langle x_i - \hat{x}_i^s, \hat{g}_i^s \rangle \mid x_i \in F_i\} \\ &= \operatorname{argmin}\{\theta_i(x_i) + \frac{1}{2}\langle x_i, (\mathcal{Q}_i + \sigma\mathcal{E}_i)x_i \rangle + \langle x_i, \bar{g}_i^s \rangle \mid x_i \in F_i\}, \end{aligned} \quad (22)$$

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

At termination, set  $x^{k+1} = \hat{x}^{s+1}$ .

**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 the subproblem (21) in Step 1 of sPALM-b is exactly one step of the proximal gradient method (22) in Step 1 of the ALM-DQA-mod. As solving the problem of the form in (22) multiple times for each iteration of the ALM-DQA-mod may be expensive, it is highly conceivable that the overall efficiency of sPALM-b could be better than that of the ALM-DQA-mod.

Next, we elucidate the connection between **ALM-DQA-mod** and the DQA method described in [41]. Given  $\hat{x}_i^s \in F_i$ , we can parameterize a given  $x_i$  as

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

with  $\rho = (N + 1)^{-1} \in (0, 1]$ . Then by convexity,  $f_i(x_i) \leq (1 - \rho)f_i(\hat{x}_i^s) + \rho f_i(\hat{x}_i^s + d_i)$ . Also, if

$\hat{x}_i^s + d_i \in F_i$ , then  $x_i \in F_i$  since  $\hat{x}_i^s \in F_i$ . From here, we have that for all  $x \in F_0 \times F_1 \times \cdots \times F_N$ ,

$$\begin{aligned}
& L_\sigma(x; y_0^k) + \frac{\sigma}{2} \|x - \hat{x}^s\|_{\mathcal{F}}^2 + \frac{1}{2\sigma} \|y_0^k\|^2 \\
& \leq (1 - \rho) \sum_{i=0}^N f_i(\hat{x}_i^s) + \rho \sum_{i=0}^N f_i(\hat{x}_i^s + d_i) + \frac{\sigma}{2} \|\mathcal{A}(\hat{x}^s + \rho d) - b_0 - \sigma^{-1} y_0^k\|^2 + \frac{\sigma \rho^2}{2} \|d\|_{\mathcal{F}}^2 \\
& = \sum_{i=0}^N \rho f_i(\hat{x}_i^s + d_i) + \rho \langle d_i, \hat{g}_i^s \rangle + \frac{\sigma \rho^2}{2} \langle d_i, \mathcal{E}_i d_i \rangle + (1 - \rho) \sum_{i=0}^N f_i(\hat{x}_i^s) + \frac{\sigma}{2} \|\mathcal{A} \hat{x}^s - b_0 - \sigma^{-1} y_0^k\|^2. \quad (23)
\end{aligned}$$

Hence instead of (22), we may consider to minimize the majorization of  $L_\sigma(x; y_0^k) + \frac{\sigma}{2} \|x - \hat{x}^s\|_{\mathcal{F}}^2$  in (23), and compute for  $i = 0, 1, \dots, N$ ,

$$d_i^{s+1} = \operatorname{argmin} \rho \left\{ f_i(\hat{x}_i^s + d_i) + \frac{\sigma \rho}{2} \langle d_i, \rho \mathcal{E}_i d_i \rangle + \langle d_i, \hat{g}_i^s \rangle \mid \hat{x}_i^s + d_i \in F_i, d_i \in \mathcal{X}_i \right\}. \quad (24)$$

We get the DQA method of [41] if we take  $\mathcal{E}_i = \rho^{-1} \mathcal{A}_i^* \mathcal{A}_i$ , compute  $d^{s+1}$  exactly in the above subproblem (24), and set

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

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

**Remark 3.2** When the  $\mathcal{A}_i$ 's are matrices, the majorization  $\mathcal{A}^* \mathcal{A} \preceq \operatorname{diag}(\mathcal{E}_0, \dots, \mathcal{E}_N)$  can be improved as follows, as has been done in [13]. Let

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

Then

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

That is,  $\mathcal{A}^* \mathcal{A} \preceq \operatorname{diag}(\chi \mathcal{A}_0^* \mathcal{A}_0, \dots, \chi \mathcal{A}_N^* \mathcal{A}_N)$ . Such an improvement has been considered in [13]. 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 (20) by  $\chi \mathcal{A}_i^* \mathcal{A}_i$  for each  $i = 0, 1, \dots, N$ .

With the derivation of the **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 improve the efficiency in solving the subproblems by using an inexact accelerated proximal gradient (iAPG) method, and we will also use a proximal term based on the linear operator (12),

which is typically less conservative than the term  $\frac{\sigma}{2}\|x - x^k\|_{\hat{\gamma}}$  used in the DQA method.

**ALM-iAPG.** Given  $\sigma > 0$ ,  $y_0^0 \in \mathcal{Y}_0$  and  $x^0 \in \mathcal{X}$ . Let  $\{\varepsilon_k\}$  be a given summable sequence of nonnegative numbers. Perform the following steps in each iteration.

**Step 1.** Starting with  $\hat{x}^0 = \bar{x}^0 = x^k$ ,  $t_0 = 1$ , iterate the following step for  $s = 0, 1, \dots$  until convergence:

- Compute  $\hat{x}^{s+1} \approx \operatorname{argmin}\{L_\sigma(x; y_0^k) + \frac{\sigma}{2}\|x - \bar{x}^s\|_{\hat{\gamma}}^2 \mid x_i \in F_i, i = 0, 1, \dots, N\}$ . As the problem is separable, one can compute in parallel for  $i = 0, 1, \dots, N$ ,

$$\hat{x}_i^{s+1} \approx \operatorname{argmin}\left\{\theta_i(x_i) + \frac{1}{2}\langle x_i, \mathcal{G}_i x_i \rangle + \langle x_i, \bar{g}_i^s \rangle \mid x_i \in F_i\right\}, \quad (25)$$

where  $\mathcal{G}_i = \mathcal{Q}_i + \sigma \mathcal{J}_i$ ,  $\bar{g}_i^s = \mathcal{Q}_i \bar{x}_i^s + c_i + \sigma \mathcal{A}_i^*(\mathcal{A} \bar{x}_i^s - b_0 - \sigma^{-1} y_0^k) - \mathcal{G}_i \bar{x}_i^s$ .

- Compute  $t_{s+1} = (1 + \sqrt{1 + 4t_s^2})/2$ ,  $\beta_{s+1} = (t_s - 1)/t_{s+1}$ .
- Compute  $\bar{x}^{s+1} = (1 + \beta_{s+1})\hat{x}^{s+1} - \beta_{s+1}\hat{x}^s$ .

At termination, set  $x^{k+1} = \hat{x}^{s+1}$ .

**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.

### 3.3 Numerical performance of sPALM and ALM-DQA-mod

In this subsection, we compare the performance of the sPALM and ALM-DQA-mod algorithms for solving several linear and quadratic test instances. The detailed description of the datasets is given in Section 5. We also report the number of constraints and variables of the instances in the table. For all the instances, we have  $m_1 = m_2 = \dots = m_N$  and  $n_1 = n_2 = \dots = n_N$ . Hence we denote them as  $m_i$  and  $n_i$  respectively.

Table 1 compares the performance of the two solvers sPALM and ALM-DQA-mod for the primal problem (P) through (10) against that of the solver sGS-ADMM for the dual problem (9). The details of the dual approach will be presented in the next section. Here, we could observe that sPALM and ALM-DQA-mod always require much longer runtime to achieve the same accuracy level in the relative KKT residual when compare to sGS-ADMM, although the former algorithms generally take a smaller number of outer iterations. In addition, the ALM-DQA-mod algorithm is slightly slower than sPALM on the whole though the difference is not too significant. Note that our preliminary implementation of the algorithms is in MATLAB which does not have a good support for parallel computing. In a full scale implementation, one may try to implement these algorithms on an appropriate parallel computing platform with a good parallelization support. Nevertheless, the inferior performance of the two primal approaches has motivated us to instead consider the dual approach of designing an efficient algorithm for the dual problem (9).

Table 1: Comparison of computational results between sGS-ADMM and two variants of ALM for primal block angular problem. All the run result are obtained using **single thread**. Here, “Iter” is the number of outer iterations performed, and “Time” is the total runtime in seconds.

Data				sGS-ADMM		sPALM		ALM-DQA-mod	
	$m_0 m_i$	$n_0 n_i$	$N$	Iter	Time(s)	Iter	Time (s)	Iter	Time (s)
qp-rand-m1-n20-N10-t1	1   1	20   20	10	321	0.50	153	8.43	12	15.62
qp-rand-m50-n80-N10-t1	50   50	80   80	10	421	0.64	268	59.88	44	192.17
qp-rand-m10-n20-N10-t2	10   10	20   20	10	1501	1.20	2971	208.56	54	137.83
qp-rand-m50-n80-N10-t2	50   50	80   80	10	141	0.20	92	19.20	32	150.32
tripart1	2096   192	2096   2096	16	1981	3.01	3880	1212.86	1422	1113.26
tripart2	8432   768	8432   8432	16	6771	51.65	5000	6369.20	1610	5723.31
qp-tripart1	2096   192	2096   2096	16	653	1.44	308	94.60	114	202.45
qp-tripart2	8432   768	8432   8432	16	971	9.79	347	419.16	124	1043.54
qp-pds1	87   126	372   372	11	971	0.99	538	49.18	535	79.24
qp-SDC-r100-c50-l100-p1000-t1	5000   150	0   5000	100	32	1.52	10	37.49	7	61.51
qp-SDC-r100-c50-l100-p1000-t2	5000   150	0   5000	100	31	1.34	9	34.22	2	33.40
qp-SDC-r100-c50-l100-p5000-t1	5000   150	0   5000	100	32	1.40	10	37.75	8	75.85
qp-SDC-r100-c50-l100-p5000-t2	5000   150	0   5000	100	31	1.37	9	34.50	3	36.63
qp-SDC-r100-c50-l100-p10000-t1	5000   150	0   5000	100	32	1.37	10	37.93	9	86.82
qp-SDC-r100-c50-l100-p10000-t2	5000   150	0   5000	100	31	1.35	9	34.78	3	37.30
qp-SDC-r100-c100-l100-p1000-t1	10000   200	0   10000	100	31	2.67	10	73.50	7	116.72
qp-SDC-r100-c100-l100-p1000-t2	10000   200	0   10000	100	31	2.67	9	68.08	2	63.91
qp-SDC-r100-c100-l100-p5000-t1	10000   200	0   10000	100	31	2.70	10	74.02	8	137.19
qp-SDC-r100-c100-l100-p5000-t2	10000   200	0   10000	100	31	2.63	9	68.04	2	64.26
qp-SDC-r100-c100-l100-p10000-t1	10000   200	0   10000	100	32	2.65	10	74.65	8	147.16
qp-SDC-r100-c100-l100-p10000-t2	10000   200	0   10000	100	31	2.63	9	67.66	3	71.18
qp-SDC-r100-c100-l200-p20000-t1	10000   200	0   10000	200	41	6.68	10	183.21	8	302.29
qp-SDC-r200-c100-l200-p20000-t1	20000   300	0   20000	200	34	11.96	10	360.48	7	513.61
qp-SDC-r200-c200-l200-p20000-t1	40000   400	0   40000	200	31	22.33	10	783.49	7	1068.04
M64-64	405   64	511   511	64	1991	3.16	5000	2874.88	625	1241.12

## 4 A semi-proximal symmetric Gauss-Seidel based ADMM for the dual problem (D)

In the last 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 (D) to solve the problem, as we shall see later when the details are presented.

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

$$\min \{p(s) + f(y_{1:N}, w, s) + q(z) + g(y_0, z) \mid \mathcal{F}^*[y_{1:N}; w; s] + \mathcal{G}^*[y_0; z] = c\}, \quad (26)$$

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

$$\mathcal{F}^* := [ \mathcal{D}^*, -\mathcal{Q}, I ], \quad \mathcal{G}^* := [ \mathcal{A}^*, I ],$$

$$p(s) := \boldsymbol{\theta}^*(-s), \quad f(y_{1:N}, w, s) := -\langle b_{1:N}, y_{1:N} \rangle + \frac{1}{2} \langle w, \mathcal{Q}w \rangle + \delta_{\mathcal{W}}(w),$$

$$q(z) := \delta_{\mathcal{K}}^*(-z), \quad g(y_0, z) := -\langle b_0, y_0 \rangle.$$

Here we take  $\mathcal{W} = \text{Range}(\mathcal{Q})$ . This is a multi-block linearly constrained convex programming problem for which the direct application of the classical ADMM is not guaranteed to converge. Thus we adapt the recently developed sGS-ADMM [15, 27] whose convergence is guaranteed to solve the dual problem (D).

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

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

Now to develop the 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  $[y_0; z]$ , which are respectively given as follows:

$$\begin{aligned} \mathcal{F}\mathcal{F}^* &= \begin{bmatrix} \mathcal{D}\mathcal{D}^* & -\mathcal{D}\mathcal{Q} & \mathcal{D} \\ -\mathcal{Q}\mathcal{D}^* & \mathcal{Q}^2 & -\mathcal{Q} \\ \mathcal{D}^* & -\mathcal{Q} & I \end{bmatrix} \\ &= \underbrace{\begin{bmatrix} 0 & -\mathcal{D}\mathcal{Q} & \mathcal{D} \\ 0 & 0 & -\mathcal{Q} \\ 0 & 0 & 0 \end{bmatrix}}_{\mathbf{U}_{\mathcal{F}}} + \underbrace{\begin{bmatrix} \mathcal{D}\mathcal{D}^* & 0 & 0 \\ 0 & \mathcal{Q}^2 & 0 \\ 0 & 0 & I \end{bmatrix}}_{\mathbf{D}_{\mathcal{F}}} + \mathbf{U}_{\mathcal{F}}^* \\ \mathcal{G}\mathcal{G}^* &= \begin{bmatrix} \mathcal{A}\mathcal{A}^* & \mathcal{A} \\ \mathcal{A}^* & I \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & \mathcal{A} \\ 0 & 0 \end{bmatrix}}_{\mathbf{U}_{\mathcal{G}}} + \underbrace{\begin{bmatrix} \mathcal{A}\mathcal{A}^* & 0 \\ 0 & I \end{bmatrix}}_{\mathbf{D}_{\mathcal{G}}} + \mathbf{U}_{\mathcal{G}}^*. \end{aligned}$$

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

$$\text{sGS}(\mathcal{F}\mathcal{F}^*) = \mathbf{U}_{\mathcal{F}}^* \mathbf{D}_{\mathcal{F}}^{-1} \mathbf{U}_{\mathcal{F}}, \quad \text{sGS}(\mathcal{G}\mathcal{G}^*) = \mathbf{U}_{\mathcal{G}}^* \mathbf{D}_{\mathcal{G}}^{-1} \mathbf{U}_{\mathcal{G}}. \quad (27)$$

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

Given the current iterate  $(y^k, s^k, w^k, z^k, x^k)$ , the basic template of the sGS-ADMM for (26) at the  $k$ -th iteration is given as follows.

**Step 1.** Compute

$$(y_{1:N}^{k+1}, w^{k+1}, s^{k+1}) = \operatorname{argmin}_{y_{1:N}, w, s} \left\{ \begin{array}{l} p(s) + f(y_{1:N}, w, s) \\ + \frac{\sigma}{2} \|\mathcal{F}^*[y_{1:N}; w; s] + \mathcal{G}^*[y_0^k; z^k] - c + \frac{1}{\sigma} x^k\|^2 \\ + \frac{\sigma}{2} \|[y_{1:N}; w; s] - [y_{1:N}^k; w^k; s^k]\|_{\text{sGS}(\mathcal{F}\mathcal{F}^*)}^2 \end{array} \right\}.$$

**Step 2.** Compute

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

**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.

By using the sGS-decomposition theorem in [28], we can show that the computation in Step 1 can be done by updating the blocks  $(y_{1:N}, w, s)$  in a symmetric Gauss-Seidel fashion. Similarly, the computation in Step 2 can be done by updating the blocks  $(y_0, z)$  in a symmetric Gauss-Seidel fashion. With the above preparations, we can now give the detailed description of the sGS-ADMM algorithm for solving (9).

**sGS-ADMM on (9).** Given  $(y^0, w^0, s^0, z^0, x^0) \in \mathbf{Y} \times \mathbf{W} \times \mathbf{X} \times \mathbf{X} \times \mathbf{X}$ , perform the following steps in each iteration. Note that for notational convenience, we define  $\mathcal{D}_0 = 0$  in the algorithm.

**Step 1a.** Let  $g^k = \mathcal{A}^* y_0^k + z^k - c + \sigma^{-1} x^k$ . Compute

$$(\bar{y}_1^k, \dots, \bar{y}_N^k) = \operatorname{argmin}_{y_1, \dots, y_N} \left\{ \mathcal{L}_\sigma((y_0^k, y_1, \dots, y_N), w^k, s^k, z^k; x^k) \right\},$$

which can be done in parallel by computing for  $i = 1, \dots, N$ ,

$$\bar{y}_i^k = \operatorname{argmin}_{y_i} \left\{ -\langle b_i, y_i \rangle + \frac{\sigma}{2} \left\| -\mathcal{Q}_i w_i^k + \mathcal{D}_i^* y_i + s_i^k + g_i^k \right\|^2 \right\}.$$

Specifically, for  $i = 1, \dots, N$ ,  $\bar{y}_i^k$  is the solution of the following linear system:

$$\mathcal{D}_i \mathcal{D}_i^* y_i = \sigma^{-1} b_i - \mathcal{D}_i(-\mathcal{Q}_i w_i^k + s_i^k + g_i^k). \quad (28)$$

**Step 1b** Compute  $\bar{w}^k = \operatorname{argmin}\{\mathcal{L}_\sigma((y_0^k, \bar{y}_1^k, \dots, \bar{y}_N^k), w, s^k, z^k; x^k)\}$  by computing in parallel for  $i = 0, 1, \dots, N$ ,

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



It is important to note that  $\bar{w}_i^k$  is only needed theoretically but not needed explicitly in practice. This is because in practical computation, only  $\mathcal{Q}_i \bar{w}_i^k$  is needed. To compute  $\mathcal{Q}_i \bar{w}_i^k$ , we first compute the solution  $\tilde{w}_i^k$  of the linear system below:

$$(I + \sigma \mathcal{Q}_i) \tilde{w}_i = \sigma (\mathcal{D}_i^* \bar{y}_i^k + s_i^k + g_i^k). \quad (29)$$

Then we can compute  $\mathcal{Q}_i \bar{w}_i^k = \mathcal{Q}_i \tilde{w}_i^k$ . The precise mechanism as to why the latter equality is valid will be given in the remark after the presentation of this algorithm.

**Step 1c.** Compute

$$(s_0^{k+1}, \dots, s_N^{k+1}) = \operatorname{argmin}_{s_0, \dots, s_N} \left\{ \mathcal{L}_\sigma((y_0^k, \bar{y}_1^k, \dots, \bar{y}_N^k), \bar{w}^k, (s_0, s_1, \dots, s_N), z^k; x^k) \right\},$$

which can be done in parallel by computing for  $i = 0, 1, \dots, N$ ,

$$\begin{aligned} s_i^{k+1} &= \operatorname{argmin}_{y_i} \left\{ \theta_i^*(-s_i) + \frac{\sigma}{2} \left\| -\mathcal{Q}_i \bar{w}_i^k + \mathcal{D}_i^* \bar{y}_i^k + s_i + g_i^k \right\|^2 \right\} \\ &= -\operatorname{Prox}_{\theta_i^*/\sigma}(-\mathcal{Q}_i \bar{w}_i^k + \mathcal{D}_i^* \bar{y}_i^k + g_i^k) \\ &= \frac{1}{\sigma} \operatorname{Prox}_{\sigma \theta_i}(\sigma(-\mathcal{Q}_i \bar{w}_i^k + \mathcal{D}_i^* \bar{y}_i^k + g_i^k)) - (-\mathcal{Q}_i \bar{w}_i^k + \mathcal{D}_i^* \bar{y}_i^k + g_i^k). \end{aligned}$$

**Step 1d** Compute  $w^{k+1} = \operatorname{argmin} \left\{ \mathcal{L}_\sigma((y_0^k, \bar{y}_1^k, \dots, \bar{y}_N^k), w, s^{k+1}, z^k; x^k) \right\}$  by computing in parallel for  $i = 0, 1, \dots, N$ ,

$$w_i^{k+1} = \operatorname{argmin}_{w_i} \left\{ \frac{1}{2} \langle w_i, \mathcal{Q}_i w_i \rangle + \frac{\sigma}{2} \left\| -\mathcal{Q}_i w_i + \mathcal{D}_i^* \bar{y}_i^k + s_i^{k+1} + g_i^k \right\|^2 \mid w_i \in \operatorname{Range}(\mathcal{Q}_i) \right\}.$$

Note that the same remark in Step 1b is applicable here.

**Step 1e** Compute

$$(y_1^{k+1}, \dots, y_N^{k+1}) = \operatorname{argmin}_{y_1, \dots, y_N} \left\{ \mathcal{L}_\sigma((y_0^k, y_1, \dots, y_N), w^{k+1}, s^{k+1}, z^k; x^k) \right\},$$

which can be done in parallel by computing for  $i = 1, \dots, N$ ,

$$y_i^{k+1} = \operatorname{argmin}_{y_i} \left\{ -\langle b_i, y_i \rangle + \frac{\sigma}{2} \left\| -\mathcal{Q}_i w_i^{k+1} + \mathcal{D}_i^* y_i + s_i^{k+1} + g_i^k \right\|^2 \right\}.$$

**Step 2a.** Let  $h^k = -\mathcal{Q} w^{k+1} + \mathcal{D}^* y^{k+1} + s^{k+1} - c + \sigma^{-1} x^k$ . Compute

$$\begin{aligned} \bar{y}_0^k &= \operatorname{argmin}_{y_0} \left\{ \mathcal{L}_\sigma((y_0, y_1^{k+1}, \dots, y_N^{k+1}), w^{k+1}, s^{k+1}, z^k; x^k) \right\} \\ &= \operatorname{argmin}_{y_0} \left\{ -\langle b_0, y_0 \rangle + \frac{\sigma}{2} \left\| \mathcal{A}_0^* y_0 + z_0^k + h_0^k \right\|^2 + \sum_{i=1}^N \frac{\sigma}{2} \left\| \mathcal{A}_i^* y_0 + z_i^k + h_i^k \right\|^2 \right\}. \end{aligned}$$

Specifically,  $\bar{y}_0^k$  is the solution to the following linear system of equations:

$$\left( \sum_{i=0}^N \mathcal{A}_i \mathcal{A}_i^* \right) y_0 = \sigma^{-1} b_0 - \sum_{i=0}^N \mathcal{A}_i (z_i^k + h_i^k). \quad (30)$$

**Step 2b** Compute  $z^{k+1} = \operatorname{argmin}\{\mathcal{L}_\sigma((\bar{y}_0^k, y_1^{k+1}, \dots, y_N^{k+1}), w^{k+1}, s^{k+1}, z; x^k)\}$  by computing in parallel for  $i = 0, 1, \dots, N$ ,

$$\begin{aligned} z_i^{k+1} &= \operatorname{argmin}_{z_i} \left\{ \delta_{\mathcal{K}_i}^*(-z_i) + \frac{\sigma}{2} \|\mathcal{A}_i^* \bar{y}_0^k + z_i + h_i^k\|^2 \right\} = -\operatorname{Prox}_{\sigma^{-1} \delta_{\mathcal{K}_i}^*}(\mathcal{A}_i^* \bar{y}_0^k + h_i^k) \\ &= \frac{1}{\sigma} \Pi_{\mathcal{K}_i}(\sigma(\mathcal{A}_i^* \bar{y}_0^k + h_i^k)) - (\mathcal{A}_i^* \bar{y}_0^k + h_i^k). \end{aligned}$$

**Step 2c** Compute

$$\begin{aligned} y_0^{k+1} &= \operatorname{argmin}_{y_0} \left\{ \mathcal{L}_\sigma((y_0, y_1^{k+1}, \dots, y_N^{k+1}), w^{k+1}, s^{k+1}, z^{k+1}; x^k) \right\} \\ &= \operatorname{argmin}_{y_0} \left\{ -\langle b_0, y_0 \rangle + \frac{\sigma}{2} \|\mathcal{A}_0^* y_0 + z_0^{k+1} + h_0^k\|^2 + \sum_{i=1}^N \frac{\sigma}{2} \|\mathcal{A}_i^* y_0 + z_i^{k+1} + h_i^k\|^2 \right\}. \end{aligned}$$

Note that the computation in Step 2a is applicable here.

**Step 3** Compute

$$x^{k+1} = x^k + \tau \sigma (-\mathcal{Q}w^{k+1} + \mathcal{B}^*y^{k+1} + s^{k+1} + z^{k+1} - c),$$

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

Now we make some important remarks concerning the computations in sGS-ADMM.

1. If the term  $\boldsymbol{\theta} \equiv 0$  in Step 1c, then this step is vacuous, and Step 1b and Step 1d are identical. Hence the computation needs only to be done for Step 1d. Hence Step 1 only consists of Step 1a, 1d, and 1e.
2. If  $\mathcal{Q} \equiv 0$ , then Step 1b and 1d are vacuous. Therefore Step 1 only consists of Step 1a, 1c, and 1e.
3. The computation in Step 1d can be omitted if the quantity  $\bar{w}_i^k$  computed in Step 1b is already a sufficiently good approximate solution to the current subproblem. More precisely, if the approximation  $\bar{w}_i^k$  for  $w_i^{k+1}$  satisfies the admissible accuracy condition required in the inexact sGS-ADMM designed in [15], then we can just set  $w_i^{k+1} = \bar{w}_i^k$  instead of using the exact solution to the current subproblem. Similar remark is also applicable to the computation in Step 1e and Step 2c.
4. The sGS-ADMM in fact has the flexibility of allowing for inexact computations as already shown in [15]. While the computation in Step 1a and 1e (similarly for Step 1b and 1d, Step 2a and 2c) are assumed to be done exactly (up to machine precision), the computation can in fact be done inexactly subject to a certain predefined accuracy requirement on the computed approximate solution. Thus iterative methods such as the preconditioned conjugate gradient (PCG) method can be used to solve the linear systems when their dimensions are too large. We omit the details here for the sake of brevity.

5. In solving the linear system (30), the  $m_0 \times m_0$  symmetric positive definite matrix  $\sum_{i=0}^N \mathcal{A}_i \mathcal{A}_i^*$  is fixed, and one can pre-compute the matrix if it can be stored in the memory and its Cholesky factorization can be computed at a reasonable cost. Then in each sGS-ADMM iteration,  $\bar{y}_0^k$  and  $y_0^{k+1}$  can be computed cheaply by solving triangular linear systems. In the event when computing the coefficient matrix or its Cholesky factorization is out of reach, one can use a PCG method to solve the linear system. In that case, one can implement the computation of the matrix-vector product in parallel by computing  $\mathcal{A}_i \mathcal{A}_i^* y_0$  in parallel for  $i = 0, 1, \dots, N$ , given any  $y_0$ . Note that when the PCG method is employed, the use of the inexact sGS-ADMM framework just mentioned above will become necessary.

The same remark above also applies to the linear system (28) for each  $i = 1, \dots, N$ .

For the multi-commodity flow problem which we will consider later in the numerical experiments, we note that the linear system in (30) has 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 (28) is equal to the Laplacian matrix of the network graph for all  $i = 1, \dots, N$ . Thus both (30) and (28) can be solved efficiently by a direct solver.

6. In Step 1b, we claimed that  $\mathcal{Q}_i \bar{w}_i^k = \mathcal{Q}_i \tilde{w}_i^k$ . Here we show why the result holds. For simplicity, we assume that  $\mathcal{Q}_i$  is a symmetric positive semidefinite matrix rather than a linear operator. Consider the spectral decomposition  $\mathcal{Q}_i = U D U^T$ , where  $D \in \mathbb{R}^{r \times r}$  is a diagonal matrix whose diagonal elements are the positive eigenvalues of  $\mathcal{Q}_i$  and the columns of  $U \in \mathbb{R}^{m_i \times r}$  are their corresponding orthonormal set of eigenvectors. We let  $V \in \mathbb{R}^{m_i \times (m_i - r)}$  be the matrix whose columns form an orthonormal set of eigenvectors of  $\mathcal{Q}_i$  correspond to the zero eigenvalues. With this decomposition and the parameterization  $w_i = U \xi$  (because  $w_i \in \text{Range}(\mathcal{Q}_i)$ ), the minimization for  $\bar{w}_i^k$  is equivalent to the following:

$$\operatorname{argmin} \left\{ \frac{1}{2} \langle \xi, D \xi \rangle + \frac{\sigma}{2} \|D \xi - U^T g\|^2 + \frac{\sigma}{2} \|V^T g\|^2 \mid \xi \in \mathbb{R}^r \right\}, \quad (31)$$

where we have set  $g = z_i^k + h_i^k$  for convenience. Now from solving (29), we get that

$$(I + \sigma D) U^T \tilde{w}_i^k = \sigma U^T g, \quad V^T \tilde{w}_i^k = \sigma V^T g.$$

This show that  $U^T \tilde{w}_i^k$  is the unique solution to the problem (31). Hence  $\bar{w}_i^k = U(U^T \tilde{w}_i^k)$  is the unique solution to (29). From here, we have that  $\mathcal{Q}_i \bar{w}_i^k = U D U^T (U U^T \tilde{w}_i^k) = U D U^T \tilde{w}_i^k = \mathcal{Q}_i \tilde{w}_i^k$ .

## 4.1 Convergence theorems of sGS-ADMM

The convergence theorem of sGS-ADMM can be established directly by using known results from [15] and [52]. Here we present the global convergence result and the linear rate of convergence for the convenience of reader.

In order to state the convergence theorems, we need some definitions.

**Definition 4.1** Let  $\mathbb{F} : \mathcal{X} \rightrightarrows \mathcal{Y}$  be a multivalued mapping and denote its inverse by  $\mathbb{F}^{-1}$ . The graph of multivalued function  $\mathbb{F}$  is defined by  $\operatorname{gph} \mathbb{F} := \{(x, y) \in \mathcal{X} \times \mathcal{Y} \mid y \in \mathbb{F}(x)\}$ .

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

$$\mathcal{R}(u) := \begin{pmatrix} \mathcal{B}x - b \\ -\mathcal{Q}w + \mathcal{B}^*y + s + z - c \\ \mathcal{Q}w - \mathcal{Q}x \\ x - \text{Prox}_{\theta}(x - s) \\ x - \Pi_{\mathcal{K}}(x - z) \end{pmatrix}. \quad (32)$$

Denote the set of KKT points by  $\bar{\Omega}$ . The KKT mapping  $\mathcal{R}$  is said to be metrically subregular at  $(\bar{u}, 0) \in \text{gph}\mathcal{R}$  with modulus  $\eta > 0$  if there exists a scalar  $\rho > 0$  such that

$$\text{dist}(u, \bar{\Omega}) \leq \eta \|\mathcal{R}(u)\| \quad \forall u \in \{u \in \mathcal{U} : \|u - \bar{u}\| \leq \rho\}.$$

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

**Theorem 4.1** *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  $\mathcal{R}$  is metrically subregular at  $(\bar{u}, 0) \in \text{gph}\mathcal{R}$ . Then the sequence  $\{u^k\}$  is linearly convergent to  $\bar{u}$ .*

*Proof.* (a) The global convergence result follows from that in [15]. (b) The result follows directly by applying the convergence result in [52, Proposition 4.1] (which slightly improves an earlier result in [20]) to the compact formulation (6) of (D).  $\square$

**Remark 4.1** *By Theorem 1 and Remark 1 in [29], we know that when (P) is a convex programming problem where for each  $i = 0, \dots, N$ ,  $\theta_i$  is piecewise linear-quadratic or strongly convex, and  $\mathcal{K}_i$  is polyhedral, then  $\mathcal{R}$  is metrically subregular at  $(\bar{u}, 0) \in \text{gph}\mathcal{R}$  for any KKT point  $\bar{u}$ . Thus sGS-ADMM converges locally at a linear rate to an optimal solution of (P) and (D) under the previous conditions on  $\theta_i$  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 locally linearly convergent, which can even be proven to converge globally linearly.*

## 4.2 Computational cost

Now we would discuss the main computational cost of sGS-ADMM. We could observe that the most time-consuming computations are in solving large linear system of equations in Step 1a, 1b, 1d, 1e, 2a, and 2c.

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

$$Mx = r. \quad (33)$$

Assuming that  $M$  is stored, then we can compute its Cholesky factorization at the cost of  $O(d^3)$  operations, which needs only 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 at the beginning of algorithm);
- ( $C_2$ ) Cost for computing Cholesky factorization of  $M$  (only once at the beginning of algorithm);
- ( $C_3$ ) 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 equations in Step 1a, 1b, 1d, 1e, 2a, and 2c are tabulated in Table 2.

Table 2: Computational cost for solving the linear systems of equations in each of the steps.

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

## 5 Numerical experiments

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

### 5.1 Stopping condition

Based on the optimality conditions in (8), we measure the accuracy of a computed solution by the following relative residuals:

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

where

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

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

## 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 representative 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 structures due to the network nature in the constraints.

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  $k = 1, \dots, N$ . Let  $M \in \mathbb{R}^{m \times |\mathcal{E}|}$  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:

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

For this problem,  $x_i$  denotes the flow of the  $i$ -th commodity ( $i = 1, \dots, N$ ) 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 datasets

Following [11], the datasets we used are as follows.

**tripart and gridgen:** These are five multicommodity instances obtained with the Tripart and Gridgen generators. They could be downloaded from [http://www-eio.upc.es/~jcastro/mmcnf\\_data.html](http://www-eio.upc.es/~jcastro/mmcnf_data.html).

**pds:** The PDS problems come from a model of transporting patients away from a place of military conflict. It could be downloaded from <http://www.di.unipi.it/optimize/Data/MMCF.html#Pds>.

**M{n}-{k}:** These are the problems generated by the Mnetgen generator, which is one of the most famous random generator of Multicommodity Min Cost Flow instances. Here  $n$  is the number of nodes in the network and  $k$  is the number of commodity. It could be downloaded from <http://www.di.unipi.it/optimize/Data/MMCF.html#MNetGen>.

### 5.2.2 Numerical results

In Table 3, we compare our sGS-ADMM algorithm against the solvers Gurobi and BlockIP. We should emphasize that Gurobi is a state-of-the-art solver for solving general linear and quadratic programming problems. Although it is not a specialized algorithm for primal block angular problems, it has been so powerful in solving sparse general linear and convex quadratic programming problems that it should be used as the benchmark for any newly developed algorithm. On the other hand, BlockIP [12] is an efficient interior-point algorithm specially designed for solving primal block angular problems, especially those arising from MCF problems. As reported in [12], it has been 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  $1e-5$ , and the number of threads is set to be 1. All the other parameters remain as default setting. Similarly for BlockIP, all the three tolerances (primal and dual feasibility, and relative objective gap) are set to be  $1e-5$  for consistency. Its maximum number of iteration is set to be 500.

Table 3: Comparison of computational results between sGS-ADMM, Gurobi, and BlockIP for **linear** primal block angular problems. All the results are obtained using a **single thread**. ‘Iter’ under the column for Gurobi means the total number of simplex iterations.

Data	$m_0 m_i$		$n_0 n_i$		$N$	sGS-ADMM		Gurobi		BlockIP	
						Iter	Time(s)	Iter	Time (s)	Iter	Time (s)
tripart1	2096	192	2096	2096	16	1981	3.01	5155	0.78	48	1.23
tripart2	8432	768	8432	8432	16	6771	51.65	42070	42.81	67	10.32
tripart3	16380	1200	16380	16380	20	5561	104.96	85390	189.37	81	48.70
tripart4	24815	1050	24815	24815	35	8581	343.32	246340	1685.50	115	139.36
gridgen1	3072	1025	3072	3072	320	7541	409.75	497709	8039.40	203	1589.04
pds15	1812	2125	7756	7756	11	2893	22.60	8545	1.01	81	12.19
pds30	3491	4223	16148	16148	11	4471	111.49	27645	4.79	110	51.66
pds60	6778	8423	33388	33388	11	7719	465.06	70168	17.57	145	403.23
pds90	8777	12186	46161	46161	11	5315	479.59	100858	25.18	162	822.45
M64-64	405	64	511	511	64	1991	3.16	7601	0.77	51	0.85
M128-64	936	128	1171	1171	64	2601	7.32	18108	3.93	52	3.15
M128-128	979	128	1204	1204	128	3801	28.89	32736	7.24	127	11.75
M256-256	1802	256	2204	2204	256	6821	225.31	103561	18.53	97	89.92
M512-64	3853	512	4768	4768	64	2631	45.77	48235	8.76	72	48.44
M512-128	3882	512	4786	4786	128	3581	137.23	87659	17.96	97	144.77
M512-512	707	512	1797	1797	512	7021	373.58	199260	16.79	146	308.95

From Table 3, we observe that Gurobi is the fastest to solve 11 out of 16 instances. Gurobi is extremely fast in solving the **pdsxx** and **Mxxx-xx** problems but have difficulty in solving **tripart4** and **gridgen1** efficiently. On the other hand, sGS-ADMM and BlockIP are highly efficient in solving the latter instances. On the other hand, BlockIP is the fastest

when solving the `tripart2,3,4` instances while sGS-ADMM is the fastest in solving the `gridgen1` and `M512-128` instances.

Our sGS-ADMM solver outperforms Gurobi when the instance is both hard and huge, for example, `tripart4` and `gridgen1`. For the latter instance, it is in fact the fastest solver. We also noticed that BlockIP is quite sensitive to the practical setting of the upper bound on the unbounded variables. For example, setting “9e6” and “9e8” as the upper bounds for the unbounded variables can lead to a significant difference in the number of iterations.

### 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.

One of the main class of this type of problem is again from the multicommodity flow problem. Following [12], we add in the quadratic objective term,  $\mathcal{Q}_i = 0.1I$ ,  $\forall i = 0, \dots, N$ . The corresponding datasets start with a prefix “qp-”, including `tripart`, `gridgen` and `pds`.

Another main class of quadratic primal block angular problems arises in the field of statistical disclosure control. Castro [10] studied the controlled tabular adjustment (CTA) to find a closest, perturbed, yet safe table given a three-dimensional table for which the content need to be protected. In particular, we have

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

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

#### 5.3.1 Description of datasets

The datasets we used are as follows.

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

- Type 1 problem (with suffix `-t1`) has diagonal quadratic objective cost, i.e.  $\mathcal{Q}_i$  is a random diagonal matrix given by `spdiags(rand(n_i, 1), 0, n_i, n_i)`.
- Type 2 problem (with suffix `-t2`) does not necessarily have diagonal quadratic objective cost. In this case  $\mathcal{Q}_i$  is still very sparse but remained to be positive semidefinite. We use the following routine to generate  $\mathcal{Q}_i$  for every  $i = 0, 1, \dots, N$ :

```
tmp=sprandn(n_i,n_i,0.1); Q_i = tmp*tmp'
```

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



**L2CTA3D:** This is an extra large instance (with a total of 10M variables and 210K constraints) provided in [http://www-eio.upc.es/~jcastro/huge\\_sdc\\_3D.html](http://www-eio.upc.es/~jcastro/huge_sdc_3D.html).

**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](http://www-eio.upc.es/~jcastro/CTA_3Dtables.html).

### 5.3.2 Numerical results

As in the last subsection, we compare our sGS-ADMM algorithm against Gurobi and BlockIP solver in Table 4.

Table 4: Comparison of computational results between sGS-ADMM, Gurobi, and BlockIP for **quadratic** primal block angular problems. All the results are obtained using **single thread**. ‘Iter’ under the column for Gurobi means the total number of barrier iterations. A ‘/’ under the column for BlockIP means that the solver runs out of memory, and a ‘\*’ means the solver is not compatible to solve the problem.

Data	$m_0 m_i$	$n_0 n_i$	$N$	sGS-ADMM		Gurobi		BlockIP	
				Iter	Time(s)	Iter	Time (s)	Iter	Time (s)
qp-rand-m50-n80-N10-t1	50   50	80   80	10	421	0.64	14	0.28	29	0.13
qp-rand-m1000-n1500-N10-t1	1000   1000	1500   1500	10	748	57.67	15	1641.91	39	360.12
qp-rand-m100-n200-N100-t1	100   100	200   200	100	331	3.81	18	14.09	54	8.51
qp-rand-m1000-n1500-N100-t1	1000   1000	1500   1500	100	361	312.61	18	17175.81	/	/
qp-rand-m100-n200-N150-t1	100   100	200   200	150	341	6.56	19	20.94	58	60.17
qp-rand-m1000-n1500-N150-t1	1000   1000	1500   1500	150	448	559.20	17	36591.57	/	/
qp-rand-m10-n20-N10-t2	10   10	20   20	10	1501	1.20	14	0.25	*	*
qp-rand-m50-n80-N10-t2	50   50	80   80	10	141	0.20	14	0.44	*	*
qp-rand-m1000-n1500-N10-t2	1000   1000	1500   1500	10	131	50.81	12	6916.43	*	*
qp-rand-m100-n200-N100-t2	100   100	200   200	100	81	3.61	14	28.40	*	*
qp-rand-m1000-n1500-N100-t2	1000   1000	1500   1500	100	220	576.62	13	8823.43	*	*
qp-rand-m100-n200-N150-t2	100   100	200   200	150	74	5.36	15	45.91	*	*
qp-rand-m1000-n1500-N150-t2	1000   1000	1500   1500	150	252	930.81	13	15299.33	*	*
qp-tripart1	2096   192	2096   2096	16	653	1.44	15	1.17	24	0.22
qp-tripart2	8432   768	8432   8432	16	971	9.79	19	6.66	38	1.36
qp-tripart3	16380   1200	16380   16380	20	1034	27.09	22	30.08	55	6.78
qp-tripart4	24815   1050	24815   24815	35	5871	413.35	22	238.92	67	17.46
qp-gridgen1	3072   1025	3072   3072	320	4081	308.05	40	2143.19	208	1197.24
qp-pds15	1812   2125	7756   7756	11	1110	10.40	48	14.49	90	11.56
qp-pds30	3491   4223	16148   16148	11	1941	57.58	53	59.95	113	44.32
qp-pds60	6778   8423	33388   33388	11	4685	337.13	58	226.56	134	192.85
qp-pds90	8777   12186	46161   46161	11	3021	318.43	58	402.51	165	547.32
qp-L2CTA3D_100x100x1000_5000	110000   1000	0   100000	100	21	31.24	8	6696.47	7	22.72
qp-SDC-r100-c50-l100-p1000-t1	5000   150	0   5000	100	32	1.52	8	101.91	7	0.84
qp-SDC-r100-c50-l100-p1000-t2	5000   150	0   5000	100	31	1.34	6	96.47	6	0.80
qp-SDC-r100-c50-l100-p5000-t1	5000   150	0   5000	100	32	1.40	8	100.84	8	0.93
qp-SDC-r100-c50-l100-p5000-t2	5000   150	0   5000	100	31	1.37	6	106.82	6	0.79
qp-SDC-r100-c50-l100-p10000-t1	5000   150	0   5000	100	32	1.37	8	97.74	8	0.94
qp-SDC-r100-c50-l100-p10000-t2	5000   150	0   5000	100	31	1.35	6	102.74	6	0.77
qp-SDC-r100-c100-l100-p1000-t1	10000   200	0   10000	100	31	2.67	8	810.58	7	2.16
qp-SDC-r100-c100-l100-p1000-t2	10000   200	0   10000	100	31	2.67	6	1107.95	6	2.10
qp-SDC-r100-c100-l100-p5000-t1	10000   200	0   10000	100	31	2.70	8	1266.75	7	2.12

Table 4: Comparison of computational results between sGS-ADMM, Gurobi, and BlockIP for **quadratic** primal block angular problems. All the results are obtained using **single thread**. ‘Iter’ under the column for Gurobi means the total number of barrier iterations. A ‘/’ under the column for BlockIP means that the solver runs out of memory, and a ‘\*’ means the solver is not compatible to solve the problem.

Data	$m_0 m_i$	$n_0 n_i$	$N$	sGS-ADMM		Gurobi		BlockIP	
				Iter	Time(s)	Iter	Time (s)	Iter	Time (s)
qp-SDC-r100-c100-l100-p5000-t2	10000   200	0   10000	100	31	2.63	6	751.24	6	2.02
qp-SDC-r100-c100-l100-p10000-t1	10000   200	0   10000	100	32	2.65	8	779.24	8	2.42
qp-SDC-r100-c100-l100-p10000-t2	10000   200	0   10000	100	31	2.63	6	810.03	6	1.98
qp-SDC-r100-c100-l200-p20000-t1	10000   200	0   10000	200	41	6.68	8	1418.31	8	4.87
qp-SDC-r200-c100-l200-p20000-t1	20000   300	0   20000	200	34	11.96	8	5194.45	8	9.47
qp-SDC-r200-c200-l200-p20000-t1	40000   400	0   40000	200	31	22.33	8	53964.31	7	23.34
qp-SDC-r500-c50-l500-p50000-t1	25000   550	0   25000	500	41	43.36	8	11025.98	8	24.56
qp-SDC-r500-c500-l50-p5000-t1	250000   1000	0   250000	50	20	27.04	8	11360.16	/	/

Table 4 shows that Gurobi is almost always slowest to solve the test instances in this case, whereas our sGS-ADMM performs almost as efficiently as BlockIP in solving these quadratic primal block angular problems. It is worth noting that our sGS-ADMM method works very well on the large scale randomly generated problems compared to BlockIP, because for these instances the matrices  $\mathcal{A}_i$  and  $\mathcal{Q}_i$  are no longer simple identity matrices for which the BlockIP solver can take special advantage of. Also, BlockIP runs out of memory for three of the huge instances `qp-rand-m1000-n1500-N100-t1`, `qp-rand-m1000-n1500-N150-t1` and `qp-SDC-r500-c500-l50-p5000-t1`.

It is also observed that BlockIP solver could not solve for the `qp-rand-xxx-t2` problem because it is not designed to cater for solving problems with nondiagonal quadratic objective cost. For these types of problem, our sGS-ADMM algorithm can substantially outperform Gurobi, sometimes by a factor of more than 10.

## 5.4 Block angular problems with nonlinear convex objective functions

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

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

where

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

The Kleinrock function is normally used to model delay in a telecommunication problem; whereas the BPR function is mainly used to model congestion in a transportation problem. Here  $\text{cap}_i$  is the capacity of arc  $i$ ,  $r_i$  is the free flow time of arc  $i$ , and  $\beta, B$  are two positive parameters.

Thus in our problem setting, we have

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

Following [2], the datasets we used are the **planar** and **grid** problems, which could be downloaded from <http://www.di.unipi.it/optimize/Data/MMCF.html#Plnr>.

**Remark 5.1** *In Step 1c of the sGS-ADMM algorithm, we need to update  $s_i^{k+1}$  by*

$$s_i^{k+1} = \frac{1}{\sigma} \text{Prox}_{\sigma\theta_i}(\sigma(-\mathcal{Q}_i\bar{w}_i^k + \mathcal{D}_i^*\bar{y}_i^k + g_i^k)) - (-\mathcal{Q}_i\bar{w}_i^k + \mathcal{D}_i^*\bar{y}_i^k + g_i^k) \quad i = 0, 1, \dots, N.$$

*To compute the proximal mapping for a given  $s$ :*

$$\text{Prox}_{\sigma\theta_i}(s) = \arg \min \left\{ g(t) := \sigma\theta_i(t) + \frac{1}{2}\|t - s\|^2 \right\},$$

*we can use Newton's method to solve the equation  $\nabla g(t) = 0$ . In each sGS-ADMM iteration, we warm-start Newton's method by using the quantity already computed in the previous iteration to generate  $s_i^k$ .*

*Another point to note is that although  $s_i^{k+1}$  is not computed exactly, the convergence of the sGS-ADMM algorithm is not affected as long as  $s_i^{k+1}$  is computed to satisfy the admissible accuracy condition required in each iteration of the inexact sGS-ADMM method developed in [15].*

### 5.4.1 Numerical results

In this subsection, we compare our sGS-ADMM algorithm against BlockIP and IPOPT. IPOPT is one of the state-of-the-art solvers for solving general nonlinear programs. We use the Kleinrock function as our objective function here.

Table 5: Comparison of computational results between sGS-ADMM, BlockIP and IPOPT for nonlinear primal block angular problem. A ‘-’ under the column for BlockIP means that the solver encounters memory issue.

Data	$m_0 m_i$	$n_0 n_i$	$N$	sGS-ADMM		BlockIP		IPOPT	
				Iter	Time(s)	Iter	Time(s)	Iter	Time (s)
grid1	80   24	80   80	50	591	0.66	28	0.13	76	3.10
grid3	360   99	360   360	50	381	0.53	41	1.60	86	21.30
grid5	840   224	840   840	100	581	1.95	-	-	90	127.50
grid8	2400   624	2400   2400	500	4171	261.73	215	4568.28	51	5027.50
grid10	2400   624	2400   2400	2000	3432	893.98	221	36035.85	14	5340.39
planar30	150   29	150   150	92	431	0.44	93	1.59	90	7.55
planar80	440   79	440   440	543	1875	20.07	-	-	430	1400.91
planar100	532   99	532   532	1085	2614	70.99	-	-	117	1184.46

Table 5 shows that IPOPT is almost always the slowest to solve the test instances but it is very robust in the sense that it is able to solve all the test instances to the required accuracy. It is not surprising for it to perform less efficiently since it is a general solver for nonlinear programs.

On the other hand, we observed that BlockIP runs into memory issue when solving almost half of the instances. This may be due to the fact that BlockIP uses a preconditioned conjugate gradient (PCG) method and Cholesky factorization to solve the linear systems arising in each iteration of the interior-point method. At some point of the iteration, the PCG method did not converge and the algorithm switches to use a Cholesky factorization to solve the linear system, which causes the out-of-memory error. Even when the PCG method works well, it might still converge in almost 10 times slower than our algorithm.

## 6 Conclusion

In conclusion, we have designed efficient methods for solving convex composite quadratic conic programming problems with a primal block angular structure. Numerical experiments show that our 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 primal block angular structures. Also, it would be ideal to utilize a good parallel computing and programming platform to implement the algorithm to realize its full potential.

## 7 Acknowledgements

We would like to thank Professor Jordi Castro for sharing with us his solver BlockIP so that we are able to evaluate the performance of our algorithm more comprehensively. We are also grateful to him for providing us with 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.

## References

- [1] A. Assad, *Multicommodity network flows – A survey*, Networks, 8 (1978), pp. 37–91.
- [2] F. Babonneau and J.-P. Vial, *ACCPM with a nonlinear constraint and an active set strategy to solve nonlinear multicommodity flow problems*, Mathematical Programming, 120 (2009), 179–210.
- [3] A. Berkelaar, C. Dert, B. Oldenkamp and S. Zhang, *A primal-dual decomposition-based interior point approach to two-stage stochastic linear programming*, Operations Research, 50 (2002), 904–915.
- [4] N.J. Berland and K.K. Haugen, *Mixing stochastic dynamic programming and scenario aggregation*, Stochastic programming, algorithms and models (Lillehammer, 1994). Ann. Oper. Res., 64 (1996), 1–19.
- [5] D.P. Bertsekas, J.N. Tsitsiklis, *Parallel and Distributed Computation Numerical Methods*, Athena Scientific (1997).
- [6] J.R. Birge, *Current trends in stochastic programming computation and applications*, Technique Report 95-15, Department of Industrial and Operations Engineering, University of Michigan, 1995.
- [7] J.R. Birge, M.A.H. Dempster, H.I. Gassmann, E.A. Gunn, A.J. King and S.W. Wallace, *A standard input format for multistage stochastic linear programs*, COAL Newsletter, 17 (1987), 1–19.
- [8] J.R. Birge and F. Louveaux, *Introduction to Stochastic programming*, Springer-Verlag, 1997.
- [9] J.R. Birge and L. Qi, *Computing block-angular Karmarkar projections with applications to stochastic programming*, Management Science, 34 (1988), 1472–1479.
- [10] J. Castro, *Quadratic interior-point methods in statistical disclosure control*, Computational Management Science, 2 (2005), 107–121.
- [11] J. Castro and J. Cuesta, *Quadratic regularizations in an interior-point method for primal block-angular problems*, Mathematical Programming, 130 (2011), 415–445.
- [12] J. Castro, *Interior-point solver for convex separable block-angular problems*, Optimization Methods and Software, 31:1 (2016), 88–109.
- [13] N. Chatzipanagiotis, D. Dentcheva, M. M. Zavlanos. *An augmented Lagrangian method for distributed optimization*, Mathematical Programming, 152 (2015), 405–434.
- [14] L. Chen, X.D. Li, D.F. Sun and K.-C. Toh, *On the equivalence of inexact proximal ALM and ADMM for a class of convex composite programming*, arXiv:1803.10803, 2018.

- [15] L. Chen, D. F. Sun and K.-C. Toh, *An efficient inexact symmetric Gauss-Seidel based majorized ADMM for high-dimensional convex composite conic programming*, Math. Program., 161 (2017), 237–270.
- [16] I.C. Choi and D. Goldfarb, *Exploiting special structure in a primal-dual path following algorithm*, Mathematical Programming, 58 (1993), 33–52.
- [17] B.J. Chun and S.M. Robinson, *Scenario analysis via bundle decomposition*, Ann. Oper. Res., 56 (1995), 39–63.
- [18] H.I. Gassmann, *MSLiP: A computer code for the multistage stochastic linear programming problem*, Math. Prog., 47 (1990), 407–423.
- [19] J. Gondzio, R. Sarkissian, and J.-P. Vial, *Using an interior point method for the master problem in a decomposition approach*, European Journal of Operational Research, 101 (1997), 577 – 587.
- [20] D.R. Han, D.F. Sun, and L.W. Zhang, *Linear rate convergence of the alternating direction method of multipliers for convex composite programming*, Mathematics of Operations Research, 42 (2017), pp. 622–637.
- [21] G. A. Hanasusanto and D. Kuhn, *Conic programming reformulation of two-stage distributionally robust linear programs over Wasserstein balls*, arXiv:1609.07505 (2017).
- [22] T. Helgason and S.W. Wallace, *Approximate scenario solutions in the progressive hedging algorithm*, Ann. Oper. Res., 31 (1991), 425–444.
- [23] A. Hundepool, J. Domingo-Ferrer, L. Franconi, S. Giessing, E. S. Nordholt, K. Spicer, and P.-P. de Wolf, *Statistical Disclosure Control*, Wiley, 2012.
- [24] P. Kall and S.W. Wallace, *Stochastic Programming*, John Wiley & Sons, NY, 1994.
- [25] S. Kontogiorgis, R. De Leone, and R.R. Meyer, *Alternating direction splitting for block angular parallel optimization*, J. Optimization Theory and Applications, 99 (1996), 1–29.
- [26] A.J. King, *Stochastic programming problems: examples from the literature*, Numerical Techniques for Stochastic Optimization, Y. Ermoliev and R. J-B Wets eds., Springer-Verlag, 1988, 543–567.
- [27] X.D. Li, D.F. Sun and K.C. Toh, *A Schur complement based semi-proximal ADMM for convex quadratic conic programming and extensions*, Mathematical Programming, 155 (2016), pp. 333–373.
- [28] X.D. Li, D.F. Sun and K.C. Toh, *A block symmetric Gauss-Seidel decomposition theorem for convex composite quadratic programming and its applications*, Mathematical Programming, in print.

- [29] X.D. Li, D.F. Sun, K.C. Toh, *A highly efficient semismooth Newton augmented Lagrangian method for solving Lasso problems*, SIAM J. Optimization, 28 (2018), pp. 433–458.
- [30] J. Mayer, *Stochastic Linear Programming Algorithms: A Comparison based on a Model Management System*, Gordon and Breach Science Publishers, 1998.
- [31] S. Mehrotra, and M. G. Ozevin, *Decomposition-based interior point methods for two-stage stochastic semidefinite programming*, SIAM J. Optimization, 18 (2007), 206–222.
- [32] S. Mehrotra, and M.G. Ozevin, *Decomposition based interior point methods for two-stage stochastic convex quadratic programs with recourse*, Operations Research, 57 (2009), 964–974.
- [33] J. Mulvey and A. Ruszczyński, *A diagonal quadratic approximation method for large scale linear programs*, Operations Research Letters, 12 (1992), 205–215.
- [34] J.M. Mulvey and H. Vladimirov, *Applying the progressive hedging algorithm to stochastic generalized networks*, Ann. Oper. Res., 31 (1991), 399–424.
- [35] W. Oliveira, C. Sagastizabal, and S. Scheimberg, *Inexact bundle methods for two-stage stochastic programming*, SIAM J. Optimization, 21 (2011), 517–544.
- [36] R.J. Peters, K. Boskma and H.A.E. Kuper, *Stochastic programming in production planning: a case with non-simple recourse*, Statistica Neerlandica, 31 (1977), 113–126.
- [37] A. Prékopa, *Stochastic Programming*, Kluwer Academic Publishers, 1995.
- [38] S.M. Robinson, *Extended scenario analysis*, Ann. Oper. Res., 31 (1991), 385–398.
- [39] R.T. Rockafellar and R.J-B. Wets, *Scenarios and policy aggregation in optimization under uncertainty*, Math. Oper. Res., 16 (1991), 119–147.
- [40] A. Ruszczyński, *A regularized decomposition method for minimizing a sum of polyhedral functions*, Mathematical Programming, 35 (1986), 309–333.
- [41] A. Ruszczyński, *An augmented Lagrangian decomposition method for block diagonal linear programming problems*, Operations Research Letters, 8 (1989), 287–294.
- [42] A. Ruszczyński, *Parallel decomposition of multistage stochastic programming problems*, Math. Prog., 58 (1993), 201–228.
- [43] A. Ruszczyński, *On convergence of an augmented Lagrangian decomposition method for sparse convex optimization*, Math. Oper. Res., 20 (1995), 634–656.
- [44] A. Ruszczyński, *Some advances in decomposition methods for stochastic linear programming*, Stochastic programming, State of the art, 1998 (Vancouver, BC). Ann. Oper. Res., 85 (1999), 153–172.

- [45] A. Ruszczyński, *Optimality and duality in stochastic programming*, in A. Ruszczyński (ed.) *Stochastic Programming, Handbooks in Operations Research and Management Science*, pp. 65–139. Elsevier, Amsterdam (2003).
- [46] G. Schultz and R.R. Meyer, *An interior point method for block angular optimization*, *SIAM Journal on Optimization*, 1 (1991), 583–602.
- [47] S. Shakkottai, R. Srikant, *Network optimization and control*, *Foundations and Trends in Networking*, 2 (2007), 271–379.
- [48] K. Sivaramakrishnan, *A parallel interior point decomposition algorithm for block angular semidefinite programs*, *Computational Optimization and Applications*, 46 (2010), 1–29.
- [49] H.C. Suarez, *Computation of upper and lower bounds in limit analysis using second-order cone programming and mesh adaptivity*, Master Thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, 2004.
- [50] M. J. Todd, *Exploiting special structure in Karmarkar’s linear programming algorithm*, *Mathematical Programming*, 41 (1988), 81–103.
- [51] A. Wächter and L. T. Biegler, *On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming*, *Mathematical Programming*, 106(1), pp. 25–57, 2006.
- [52] N. Zhang, J. Wu, and L. Zhang, *A linearly convergent majorized ADMM with indefinite proximal terms for convex composite programming and its applications*, arXiv:1706.01698, 2018.
- [53] G. Zhao, *Interior-point methods with decomposition for solving large-scale linear programs*, *J. Optimization Theory and Applications*, 102 (1999), 169–192.
- [54] G. Zhao, *A log-barrier method with Benders decomposition for solving two-stage stochastic programs*, *Mathematical Programming*, 90 (2001), 507–536.
- [55] G. Zhao, *A Lagrangian dual method with self-concordant barrier for multi-stage stochastic convex programming*, *Mathematical Programming*, 102 (2005), 1–24.
- [56] Y. Zhu, and K.A. Ariyawansa, *A preliminary set of applications leading to stochastic semidefinite programs and chance-constrained semidefinite programs*, *Applied Mathematical Modelling*, 35 (2011), 2425–2442.