Mathematical Programming
https://doi.org/10.1007/s10107-018-1342-9

FULL LENGTH PAPER

Series A

On the efficient computation of a generalized Jacobian of the projector over the Birkhoff polytope

Xudong Li[1] · Defeng Sun[2] · Kim-Chuan Toh[3]

Received: 5 August 2017 / Accepted: 15 October 2018

Abstract

We derive an explicit formula, as well as an efficient procedure, for constructing a generalized Jacobian for the projector of a given square matrix onto the Birkhoff polytope, i.e., the set of doubly stochastic matrices. To guarantee the high efficiency of our procedure, a semismooth Newton method for solving the dual of the projection problem is proposed and efficiently implemented. Extensive numerical experiments are presented to demonstrate the merits and effectiveness of our method by comparing its performance against other powerful solvers such as the commercial software Gurobi and the academic code PPROJ (Hager and Zhang in SIAM J Optim 26:1773–1798, 2016). In particular, our algorithm is able to solve the projection problem with over one billion variables and nonnegative constraints to a very high accuracy in less than 15 min on a modest desktop computer. More importantly, based on our efficient computation of the projections and their generalized Jacobians, we can design a highly efficient augmented Lagrangian method (ALM) for solving a class of convex quadratic programming (QP) problems constrained by the Birkhoff polytope. The resulted ALM is demonstrated to be much more efficient than Gurobi in solving a collection of QP problems arising from the relaxation of quadratic assignment problems.

Keywords Doubly stochastic matrix · Semismoothness · Newton's method · Generalized Jacobian

Mathematics Subject Classification 90C06 · 90C20 · 90C25 · 65F10

The research of Defeng Sun was supported in part by a start-up research grant from the Hong Kong Polytechnic University. The research of Kim-Chuan Toh was supported in part by the Ministry of Education, Singapore, Academic Research Fund under Grant R-146-000-257-112.

✉ Kim-Chuan Toh
mattohkc@nus.edu.sg

Extended author information available on the last page of the article

Published online: 27 October 2018

1 Introduction

The Birkhoff polytope is the set of $n \times n$ doubly stochastic matrices defined by

$$\mathfrak{B}_n := \{X \in \Re^{n\times n} \mid Xe = e,\ X^T e = e, X \geq 0\},$$

where $e \in \Re^n$ is the vector of all ones and $X \geq 0$ means that all the elements of X are nonnegative. In this paper, we focus on the problem of projecting a matrix $G \in \Re^{n\times n}$ onto the Birkhoff polytope $\mathfrak{B}_n$, i.e., solving the following special convex quadratic programming (QP) problem

$$\min \left\{ \frac{1}{2}\|X - G\|^2 \mid X \in \mathfrak{B}_n \right\}, \tag{1}$$

where $\|\cdot\|$ denotes the Frobenius norm. The optimal solution of (1), i.e., the Euclidean projection of G onto $\mathfrak{B}_n$, is denoted by $\Pi_{\mathfrak{B}_n}(G)$.

The Birkhoff polytope has long been an important object in statistics, combinatorics, physics and optimization. As the convex hull of the set of permutation matrices [3,42], the Birkhoff polytope has frequently been used to derive relaxations of nonconvex optimization problems involving permutations, such as the quadratic assignment problems [22] and the seriation problems [13,25]. Very often the algorithms that are designed to solve these relaxed problems need to compute the projection of matrices onto the polytope $\mathfrak{B}_n$ [13,22]. On the other hand, the availability of a fast solver for computing $\Pi_{\mathfrak{B}_n}(\cdot)$ can also influence how one would design an algorithm to solve the relaxed problems. As we shall demonstrate later, indeed one can design a highly efficient algorithm to solve QP problems involving Birkhoff polytope constraints if a fast solver for computing $\Pi_{\mathfrak{B}_n}(\cdot)$ and its generalized Jacobian is readily available.

Let D be a nonempty polyhedral convex set. Besides the computation of the Euclidean projector $\Pi_D(\cdot)$, the differential properties of the projector have long been recognized to be important in nonsmooth analysis and algorithmic design. In [20], Haraux showed that the projector onto a polyhedral convex set must be directionally differentiable. Pang [30], inspired by an unpublished report of Robinson [35], derived an explicit formula for the directional derivative and discussed the Fréchet differentiability of the projector. By using the piecewise linear structure of $\Pi_D(\cdot)$, one may further use the results of Pang and Ralph [31] to characterize the B-subdifferential and the corresponding Clarke generalized Jacobian [8] of the projector. However, for an arbitrary polyhedral convex set D, the calculations of these generalized Jacobians are generally very difficult to accomplish numerically, if feasible at all. In order to circumvent this difficulty, Han and Sun in [17] proposed a special multi-valued mapping as a more tractable replacement for the generalized Jacobian and used it in the design of the generalized Newton and quasi-Newton methods for solving a class of piecewise smooth equations. The idea of getting an element from the aforementioned multi-valued mapping in [17] is to find certain dual multipliers of the projection problem together with a corresponding set of linearly independent active constraints. Since the linear independence checking can be costly, in particular when the dimension of the underlying projection problem is large, in this paper, we aim at introducing a technique to avoid this checking and provide an efficient computation of a generalized Jacobian in the sense of [17] for the Euclidean projector over the polyhedral convex set with an emphasis on the Birkhoff polytope. We achieve this goal by deriving an explicit formula for constructing a special generalized Jacobian in the sense of [17]. In addition, based on the special structure of the Birkhoff polytope, we further simplify the formula and discuss efficient implementations for its calculation. We shall emphasize here that, in contrast to the previous work done in [17] and as a surprising result, our specially constructed Jacobian needs neither the knowledge of the dual multipliers associated with the projection problem nor the set of corresponding linearly independent active constraints.

As one can see later, the computation of the Euclidean projector $\Pi_D(\cdot)$ is one of the key steps in our construction of the aforementioned special generalized Jacobian. Hence, its efficiency is crucial to our construction. As a simple yet fundamental convex quadratic programming problem, various well developed algorithms have been used for computing the projection onto a polyhedral convex set such as the state-of-the-art interior-point based commercial solvers Gurobi [16] and CPLEX[1]. Recently, Hager and Zhang [18] proposed to compute the projector through the dual approach by combining the sparse reconstruction by separable approximation (SpaRSA) [44] and the dual active set algorithm. An efficient implementation called PPROJ is also provided in [18] and the comparisons between PPROJ and CPLEX indicate that PPROJ is robust, accurate and fast. In fact, the dual approach for solving Euclidean projection problems has been extensively studied in the literature. For example, both the dual quasi-Newton method [26] and the dual semismooth Newton method [32] have been developed to compute the Euclidean projector onto the intersection of an affine subspace and a closed convex cone. Another popular method for computing the projection over the intersection of an affine subspace and a closed convex cone is the alternating projections method with Dykstra's correction [11] that was proposed in [19]. It has been shown in [26, Theorem 5.1] that the alternating projections method with Dykstra's correction [11] is a dual gradient method with constant step size. As can be observed from the numerical comparison in [32], the semismooth Newton method outperformed the quasi-Newton and Dykstra's methods by a significant margin.

As already mentioned in the second paragraph above, the projection onto the Birkhoff polytope has important applications in different areas. It is also by itself a mathematically elegant problem to study. Thus in this paper, we shall focus on the case where the polyhedral convex set D is chosen to be the Birkhoff polytope $\mathfrak{B}_n$. Due to the elegant structure of $\mathfrak{B}_n$, we are able to derive a highly efficient procedure to compute a special generalized Jacobian of $\Pi_{\mathfrak{B}_n}$ by leveraging on its structure. As a crucial step in our procedure, we choose to use the semismooth Newton method for computing the projector $\Pi_{\mathfrak{B}_n}(\cdot)$ via solving the dual of the projection problem (1) and provide a highly efficient implementation. Extensive numerical experiments are presented to demonstrate the merits and effectiveness of our method by comparing its performance against other solvers such as Gurobi and PPROJ. In particular, our algorithm is able to solve a projection problem over the Birkhoff polytope with over one billion variables and nonnegative constraints to a very high accuracy in less than 15 min

[1] https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/index.html.

on a modest desktop computer. In order to further demonstrate the importance of our procedure, we also propose a highly efficient augmented Lagrangian method (ALM) for solving a class of convex QP problems with Birkhoff polytope constraints. Our ALM is demonstrated to be much more efficient than Gurobi in solving a collection of QP problems arising from the relaxation of quadratic assignment problems.

The remaining parts of this paper are organized as follows. The next section is devoted to studying the generalized Jacobians of the projector onto a general polyhedral convex set. In Sect. 3, a semismooth Newton method is designed for projecting a matrix onto the Birkhoff polytope. Then, a generalized Jacobian of the projector at the given matrix is computed. Efficient implementations of these steps are discussed. In Sect. 4, we show how the generalized Jacobian obtained in Sect. 3 can be employed in the design of a highly efficient augmented Lagrangian method for solving convex quadratic programming problems with Birkhoff polytope constraints. In Sect. 5, we conduct numerical experiments to evaluate the performance of our algorithms against Gurobi and PROJ for computing the projection onto the Birkhoff polytope. The advantage of incorporating the second order (generalized Hessian) information into the design of an algorithm for solving convex quadratic programming problems with Birkhoff polytope constraints is also demonstrated. We conclude our paper in the final section.

Before we move to the next section, here we list some notation to be used later in this paper. For any given $X \in \Re^{m\times n}$, we define its associated vector $\mathbf{vec}(X) \in \Re^{mn}$ by

$$\mathbf{vec}(X) := [x_{11}, \ldots, x_{m1}, x_{12}, \ldots, x_{m2}, \ldots .x_{1n}, \ldots, x_{mn}].$$

For any given vector $y \in \Re^n$, we denote by $\mathrm{Diag}(y)$ the diagonal matrix whose ith diagonal element is given by y_i. For any given matrix $A \in \Re^{m\times n}$, we use $\mathrm{Ran}(A)$ and $\mathrm{Null}(A)$ to denote the range space and the null space of A, respectively. Similar notation is used when A is replaced by a linear operator $\mathcal{A}$. We use I_n to denote the n by n identity matrix in $\Re^{n\times n}$ and $N^\dagger$ to denote the Moore–Penrose pseudo-inverse of a given matrix $N \in \Re^{m\times n}$.

2 Generalized Jacobians of the projector over polyhedral convex sets

In this section, we study the variational properties of the projection mapping $\Pi_D(\cdot)$ for a nonempty polyhedral convex set $D \subseteq \Re^n$ expressed in the following form

$$D := \{x \in \Re^n \mid Ax \geq b,\ Bx = d\}, \tag{2}$$

where $A \in \Re^{m\times n}$, $B \in \Re^{p\times n}$ and $b \in \Re^m$, $d \in \Re^p$ are given data. Without loss of generality, we assume that $\mathrm{rank}(B) = p$, $p \leq n$.

Given $x \in \Re^n$, from the representation of D in (2), we know that there exist multipliers $\lambda \in \Re^m$ and $\mu \in \Re^p$ such that

$$\begin{cases} \Pi_D(x) - x + A^T\lambda + B^T\mu = 0, \\ A\Pi_D(x) - b \geq 0, \quad B\Pi_D(x) - d = 0, \\ \lambda \leq 0, \quad \lambda^T(A\Pi_D(x) - b) = 0. \end{cases} \tag{3}$$

Let $M(x)$ be the set of multipliers associated with x, i.e.,

$$M(x) := \{(\lambda, \mu) \in \Re^m \times \Re^p \mid (x, \lambda, \mu) \text{ satisfies } (3)\}.$$

Since $M(x)$ is a nonempty polyhedral convex set containing no lines, it has at least one extreme point denoted as $(\bar{\lambda}, \bar{\mu})$ [36, Corollary 18.5.3]. Denote

$$I(x) := \{i \in \{1, \ldots, m\} \mid A_i \Pi_D(x) = b_i\}, \tag{4}$$

where A_i is the ith row of the matrix A. Define a collection of index sets:

$$\begin{aligned}\mathcal{D}(x) := \{\, K \subseteq \{1, \ldots, m\} \mid \exists\, (\lambda, \mu) \in M(x) \text{ s.t. } \operatorname{supp}(\lambda) \subseteq K \subseteq I(x),\\ [A_K^T\ B^T] \text{ is of full column rank}\},\end{aligned}$$

where $\operatorname{supp}(\lambda)$ denotes the support of λ, i.e., the set of indices i such that $\lambda_i \neq 0$ and A_K is the matrix consisting of the rows of A, indexed by K. As is already noted in [17], the set $\mathcal{D}(x)$ is nonempty due to the existence of the extreme point $(\bar{\lambda}, \bar{\mu})$ of $M(x)$. Since it is usually difficult to calculate the B-subdifferential $\partial_B \Pi_D(x)$ or the Clarke generalized Jacobian $\partial \Pi_D(x)$ for a general polyhedral convex set D at a given point x, Han and Sun in [17] introduced the following multi-valued mapping $\mathcal{P} : \Re^n \rightrightarrows \Re^{n \times n}$ as a computable replacement for $\partial_B \Pi_D(\cdot)$, namely,

$$\mathcal{P}(x) := \left\{ P \in \Re^{n \times n} \mid P = I_n - [A_K^T\ B^T] \left(\begin{bmatrix} A_K \\ B \end{bmatrix} [A_K^T\ B^T] \right)^{-1} \begin{bmatrix} A_K \\ B \end{bmatrix},\ K \in \mathcal{D}(x) \right\}. \tag{5}$$

The mapping $\mathcal{P}$ has a few important properties [17], which are summarized in the following proposition.

Proposition 1 *For any $x \in \Re^n$, there exists a neighborhood U of x such that*

$$\mathcal{D}(y) \subseteq \mathcal{D}(x), \quad \mathcal{P}(y) \subseteq \mathcal{P}(x), \quad \forall y \in U.$$

If $\mathcal{D}(y) \subseteq \mathcal{D}(x)$, it holds that

$$\Pi_D(y) = \Pi_D(x) + P(y - x), \quad \forall P \in \mathcal{P}(y).$$

Thus, $\partial_B \Pi_D(x) \subseteq \mathcal{P}(x)$.

Note that even with formula (5), for a given point $x \in \Re^n$, it is still not easy to find an element in $\mathcal{P}(x)$ as one needs to find a suitable index $K \in \mathcal{D}(x)$ corresponding to some multiplier $(\lambda, \mu) \in M(x)$. A key contribution made in this paper is that we are able to construct a matrix $P_0 \in \Re^{n \times n}$ such that $P_0 \in \mathcal{P}(x)$ without knowing the index set K and its corresponding multipliers. In addition, we show how to efficiently compute the matrix P_0 when the polyhedral set D possesses certain special structures. We shall emphasize here that the efficient computation of P_0 is crucial in the design of various second order algorithms for solving optimization problems involving the polyhedral constraint $x \in D$.

We present here a very useful lemma which will be used extensively in our later discussions.

Lemma 1 *Let $H \in \Re^{m\times n}$ be a given matrix and $\widehat{H} \in \Re^{m_1\times n}$ be a full row rank matrix satisfying* $\mathrm{Null}(\widehat{H}) = \mathrm{Null}(H)$. *Then it holds that*

$$H^T(HH^T)^\dagger H = \widehat{H}^T(\widehat{H}\widehat{H}^T)^{-1}\widehat{H}.$$

Proof By the singular value decomposition of H and the definition of the Moore–Penrose pseudo-inverse of HH^T, we can obtain through some simple calculations that

$$H^T(HH^T)^\dagger H\,d = \Pi_{\mathrm{Ran}(H^T)}(d), \quad \forall d \in \Re^n. \tag{6}$$

Meanwhile, since $\widehat{H}$ is of full row rank, we know (e.g., see [41, Page 46 (6.13)]) that

$$\widehat{H}^T(\widehat{H}\widehat{H}^T)^{-1}\widehat{H}\,d = \Pi_{\mathrm{Ran}(\widehat{H}^T)}(d), \quad \forall d \in \Re^n. \tag{7}$$

Equations (6) and (7), together with the fact that

$$\mathrm{Ran}(H^T) = \mathrm{Null}(H)^\perp = \mathrm{Null}(\widehat{H})^\perp = \mathrm{Ran}(\widehat{H}^T),$$

imply the desired result. □

Theorem 1 *For any given $x \in \Re^n$, let $I(x)$ be given in (4). Denote*

$$P_0 := I_n - [A_{I(x)}^T\ B^T]\left(\begin{bmatrix} A_{I(x)} \\ B \end{bmatrix}[A_{I(x)}^T\ B^T]\right)^\dagger \begin{bmatrix} A_{I(x)} \\ B \end{bmatrix}. \tag{8}$$

Then, $P_0 \in \mathcal{P}(x)$.

Proof Let $(\bar{\lambda}, \bar{\mu})$ be an extreme point of $M(x)$. Denote $\overline{K} := \mathrm{supp}(\bar{\lambda})$. Then, $\overline{K} \subseteq I(x)$. From the definition of extreme points, we observe that $[A_{\overline{K}}^T\ B^T]$ has linearly independent columns. By adding indexes from $I(x)$ to $\overline{K}$ if necessary, one can obtain an index set K such that $\overline{K} \subseteq K \subseteq I(x)$, $[A_K^T\ B^T]$ has full column rank and

$$\mathrm{Ran}([A_K^T\ B^T]) = \mathrm{Ran}([A_{I(x)}^T\ B^T]). \tag{9}$$

That is, $K \in \mathcal{D}(x)$. Therefore,

$$P := I_n - [A_K^T\ B^T]\left(\begin{bmatrix} A_K \\ B \end{bmatrix}[A_K^T\ B^T]\right)^{-1}\begin{bmatrix} A_K \\ B \end{bmatrix} \in \mathcal{P}(x).$$

By (9) and Lemma 1, we know that

$$P_0 = P.$$

Thus, $P_0 \in \mathcal{P}(x)$ and this completes the proof for the theorem. □

Remark 1 From Lemma 1, we know that the matrix B in (8) can be replaced by any matrix $\widehat{B}$ satisfying $\text{Null}(B) = \text{Null}(\widehat{B})$. In fact, Lemma 1 and Theorem 1 together imply that P_0 is invariant with respect to the algebraic representation of the polyhedral convex set D, i.e., it is in fact a geometric quantity corresponding to D at x.

In general, it is not clear whether P_0 is a Clarke generalized Jacobian. As a computable replacement, the matrix P_0 will be referred to as the HS-Jacobian of Π_D at x in the sense of [17]. Apart from the calculation of $\Pi_D(x)$, one can observe from Theorem 1 that the key step involved in the computation of P_0 is the computation of the Moore–Penrose pseudo-inverse in (8). Next, we show that when the matrix A in the inequality constraints in (2) is the identity matrix, the procedure for computing P_0 can be further simplified.

Proposition 2 *Let $\theta \in \Re^n$ be a given vector with each entry θ_i being 0 or 1 for all $i = 1, \ldots, n$. Let $\Theta = \text{Diag}(\theta)$ and $\Sigma = I_n - \Theta$. For any given matrix $H \in \Re^{m\times n}$, it holds that*

$$P := I_n - [\Theta\ H^T]\left(\begin{bmatrix}\Theta \\ H\end{bmatrix}[\Theta\ H^T]\right)^{\dagger}\begin{bmatrix}\Theta \\ H\end{bmatrix} = \Sigma - \Sigma H^T(H\Sigma H^T)^{\dagger}H\Sigma. \quad (10)$$

Proof We only consider the case when $\Theta \neq 0$ as the conclusion holds trivially if $\Theta = 0$.

From Lemma 1, we observe that P is the orthorgonal projection onto $\text{Null}\begin{bmatrix}\Theta \\ H\end{bmatrix}$. Hence $\text{Ran}(P)\text{=Null}\begin{bmatrix}\Theta \\ H\end{bmatrix} \subset \text{Null}(\Theta) = \text{Ran}(\Sigma)$, where the last equality comes from the definitions of Θ and Σ. From here, it is easy to show that $P = \Sigma P$. Since P is a symmetric matrix, it further holds that

$$P = \Sigma P \Sigma. \quad (11)$$

Let $\widehat{\Theta}$ be the submatrix of Θ formed by deleting all the zero rows of Θ. Then, it is readily shown that

$$\text{Null}(\Theta) = \text{Null}(\widehat{\Theta}), \quad \widehat{\Theta}\widehat{\Theta}^T = I_r, \quad \widehat{\Theta}^T\widehat{\Theta} = \Theta,$$

where r is the number of rows of $\widehat{\Theta}$.

Let $\begin{bmatrix}\widehat{\Theta} \\ \widehat{H}\end{bmatrix}$ be a full row rank matrix such that

$\mathrm{Null}\begin{bmatrix}\widehat{\Theta}\\ \widehat{H}\end{bmatrix} = \mathrm{Null}\begin{bmatrix}\Theta\\ H\end{bmatrix}$. Then, by Lemma 1 and (11), we know that

$$P = I_n - [\widehat{\Theta}^T \ \widehat{H}^T]M^{-1}\begin{bmatrix}\widehat{\Theta}\\ \widehat{H}\end{bmatrix} = \Sigma - [0 \ \Sigma\widehat{H}^T]M^{-1}\begin{bmatrix}0\\ \widehat{H}\Sigma\end{bmatrix},$$

where

$$M := \begin{bmatrix}\widehat{\Theta}\\ \widehat{H}\end{bmatrix}[\widehat{\Theta}^T \ \widehat{H}^T] = \begin{bmatrix}I_r & \widehat{\Theta}\widehat{H}^T\\ \widehat{H}\widehat{\Theta}^T & \widehat{H}\widehat{H}^T\end{bmatrix}.$$

Therefore, we only need to focus on the (2, 2) block of the inverse of the partitioned matrix M. Simple calculations show that

$$(M^{-1})_{22} = (\widehat{H}\widehat{H}^T - \widehat{H}\widehat{\Theta}^T\widehat{\Theta}\widehat{H}^T)^{-1} = (\widehat{H}\Sigma\widehat{H}^T)^{-1}.$$

Therefore,

$$P = \Sigma - \Sigma\widehat{H}^T(\widehat{H}\Sigma\widehat{H}^T)^{-1}\widehat{H}\Sigma.$$

The desired result then follows directly from Lemma 1 since $\mathrm{Null}(\widehat{H}\Sigma) = \mathrm{Null}(H\Sigma)$. □

We should emphasize that the above proposition is particularly useful for calculating the HS-Jacobian of the projection over a polyhedral set defined by the intersection of hyperplanes and the nonnegative orthant. In particular, we will see how the proposition is applied to compute the HS-Jacobian of $\Pi_{\mathfrak{B}_n}$ in the next section. Here we provide a proposition on the projection over the general polyhedral set rather than on the Birkhoff polytope only as we believe that it can be useful in other situations.

3 Efficient procedures for computing $\Pi_{\mathfrak{B}_n}(\cdot)$ and its HS-Jacobian

In this section, we focus on the projection over the Birkhoff polytope $\mathfrak{B}_n$ and calculate the associated HS-Jacobian by employing the efficient procedure developed in Theorem 1 and Proposition 2. As a by-product, we also describe and implement a highly efficient algorithm for computing the projection $\Pi_{\mathfrak{B}_n}(G)$, i.e., the optimal solution for problem (1) with a given matrix $G \in \Re^{n\times n}$.

Let the linear operator $\mathcal{B} : \Re^{n\times n} \to \Re^{2n}$ be defined by

$$\mathcal{B}(X) := [e^T X^T \ e^T X]^T, \quad X \in \Re^{n\times n}.$$

Then, problem (1) can be represented as

$$\min\left\{\frac{1}{2}\|X - G\|^2 \ \middle|\ \mathcal{B}X = b,\ X \in C\right\}, \tag{12}$$

where $b := [e^T \; e^T]^T \in \Re^{2n}$ and $C := \{X \in \Re^{n\times n} \mid X \geq 0\}$. Note that $b \in \mathrm{Ran}(\mathcal{B})$ and $\dim(\mathrm{Ran}(\mathcal{B})) = 2n - 1$.

Suppose that $\overline{G} := \Pi_{\mathfrak{B}_n}(G)$ has been computed. We then aim to find the HS-Jacobian of $\Pi_{\mathfrak{B}_n}$ at the given point G. Define the linear operator $\Xi : \Re^{n\times n} \to \Re^{n\times n}$ by

$$\Xi(H) := H - \Theta^G \circ H, \quad H \in \Re^{n\times n}, \tag{13}$$

where "$\circ$" denotes the Hadamard product of two matrices and $\Theta^G \in \Re^{n\times n}$ is given as follows: for all $1 \leq i, j \leq n$,

$$\Theta^G_{ij} = \begin{cases} 1, & \text{if } \overline{G}_{ij} = 0, \\ 0, & \text{otherwise.} \end{cases}$$

Proposition 3 *Given $G \in \Re^{n\times n}$, let Ξ be the linear operator defined in (13). Then the linear operator $\mathcal{P} : \Re^{n\times n} \to \Re^{n\times n}$ given by*

$$\mathcal{P}(H) := \Xi(H) - \Xi\mathcal{B}^*(\mathcal{B}\Xi\mathcal{B}^*)^\dagger \mathcal{B}\Xi(H), \quad \forall\, H \in \Re^{n\times n}, \tag{14}$$

is the HS-Jacobian of $\Pi_{\mathfrak{B}_n}$ at G. Moreover, $\mathcal{P}$ is self-adjoint and positive semidefinite.

Proof The desired result follows directly from Theorem 1, Proposition 2 and Remark 1. □

Next, we focus on designing an efficient algorithm for computing the optimal solution of problem (12), i.e, the projection $\Pi_{\mathcal{B}_n}(G)$. By some simple calculations, we can derive the dual of (12) in the minimization form as follows:

$$\min \left\{ \varphi(y) := \frac{1}{2}\|\Pi_C(\mathcal{B}^* y + G)\|^2 - \langle b, \; y\rangle - \frac{1}{2}\|G\|^2 \;\mid\; y \in \mathrm{Ran}(\mathcal{B}) \right\}. \tag{15}$$

With no difficulty, we can write down the KKT conditions associated with problems (12) and (15) as follows:

$$X = \Pi_C(\mathcal{B}^* y + G), \quad \mathcal{B}X = b, \quad y \in \mathrm{Ran}(\mathcal{B}). \tag{16}$$

Note that the subspace constraint $y \in \mathrm{Ran}(\mathcal{B})$ is imposed to ensure the boundedness of the solution set of (15). Indeed, since $\mathrm{int}(C) \neq \emptyset$ and $\mathcal{B} : \Re^{n\times n} \to \mathrm{Ran}(\mathcal{B})$ is surjective, we have from [4, Theorem 2.165] that the solution set to the KKT system (16) is nonempty and for any $\tau \in \Re$ the level set $\{y \in \mathrm{Ran}(\mathcal{B}) \mid \varphi(y) \leq \tau\}$ is convex, closed and bounded.

Note that $\varphi(\cdot)$ is convex and continuously differentiable on $\text{Ran}(\mathcal{B})$ with

$$\nabla\varphi(y) = \mathcal{B}\Pi_C(\mathcal{B}^* y + G) - b, \quad \forall\, y \in \text{Ran}(\mathcal{B}).$$

Let $\bar{y}$ be a solution to the following nonsmooth equation

$$\nabla\varphi(y) = 0, \quad y \in \text{Ran}(\mathcal{B}) \tag{17}$$

and denote $\overline{X} := \Pi_C(\mathcal{B}^*\bar{y}+G)$. Then, $(\overline{X}, \bar{y})$ solves the KKT system (16), i.e., $\overline{X}$ is the unique optimal solution to problem (12) and $\bar{y}$ solves problem (15). Let $y \in \text{Ran}(\mathcal{B})$ be any given point. Define the following operator

$$\hat{\partial}^2\varphi(y) := \mathcal{B}\partial\Pi_C(\mathcal{B}^* y + G)\mathcal{B}^*,$$

where $\partial\Pi_C(\mathcal{B}^* y + G)$ is the Clarke subdifferential [8] of the Lipschitz continuous mapping $\Pi_C(\cdot)$ at $\mathcal{B}^* y + G$. From [21], we have that

$$\partial^2\varphi(y)h = \hat{\partial}^2\varphi(y)h, \quad \forall\, h \in \Re^{2n},$$

where $\partial^2\varphi(y)$ denotes the generalized Hessian of φ at y, i.e., the Clarke subdifferential of $\nabla\varphi$ at y. Given $X \in \Re^{n\times n}$, define the linear operator $\mathcal{U} : \Re^{n\times n} \to \Re^{n\times n}$ as follows:

$$\mathcal{U}(H) := \Omega^X \circ H, \quad \forall\, H \in \Re^{n\times n}, \tag{18}$$

where "$\circ''$ denotes the Hadamard product of two matrices and for $1 \le i, j \le n$,

$$\Omega^X_{ij} = \begin{cases} 1, & \text{if } X_{ij} \ge 0, \\ 0, & \text{otherwise.} \end{cases} \tag{19}$$

From the definition of the simple polyhedral convex set C, it is easy to see that $\mathcal{U} \in \partial\Pi_C(X)$.

Next, we present an inexact semismooth Newton method for solving problem (15) and study its global and local convergence. Since $\Pi_C(\cdot)$ is strongly semismooth as it is a Lipschitz continuous piecewise affine function [28,33], we can design a superlinearly or even quadratically convergent semismooth Newton method to solve the nonsmooth equation (17).

The template of the semismooth Newton conjugate gradient (CG) method for solving (15) is presented as follows.

Algorithm Ssncg1: **A semismooth Newton-CG algorithm for solving (15).**

Given $\mu \in (0, 1/2)$, $\bar{\eta} \in (0, 1)$, $\tau_1, \tau_2 \in (0, 1)$, $\tau \in (0, 1]$, and $\delta \in (0, 1)$, choose $y^0 \in \mathrm{Ran}(\mathcal{B})$. Iterate the following steps for $j = 0, 1, \ldots$:

Step 1. Choose $\mathcal{U}_j \in \partial \Pi_C(\mathcal{B}^* y^j + G)$ as given in (18). Let $\mathcal{V}_j := \mathcal{B}\mathcal{U}_j\mathcal{B}^*$ and $\varepsilon_j = \tau_1 \min\{\tau_2, \|\nabla\varphi(y^j)\|\}$. Apply the CG algorithm with the zero vector as the starting point to find an approximate solution d^j to the following linear system

$$(\mathcal{V}_j + \varepsilon_j I_{2n})d + \nabla\varphi(y^j) = 0, \quad d \in \mathrm{Ran}(\mathcal{B}) \tag{20}$$

such that

$$\|(\mathcal{V}_j + \varepsilon_j I_{2n})d^j + \nabla\varphi(y^j)\| \leq \min(\bar{\eta}, \|\nabla\varphi(y^j)\|^{1+\tau}).$$

Step 2. (Line search) Set $\alpha_j = \delta^{m_j}$, where m_j is the first nonnegative integer m for which

$$\varphi(y^j + \delta^m d^j) \leq \varphi(y^j) + \mu\delta^m \langle \nabla\varphi(y^j), d^j \rangle.$$

Step 3. Set $y^{j+1} = y^j + \alpha_j d^j$.

We note that at each iteration of Algorithm Ssncg1, $\mathcal{V}_j$ is self-adjoint positive semidefinite. Indeed, for $j = 0, 1, \ldots$, the self-adjointness of $\mathcal{V}_j$ follows from the self-adjointness of $\mathcal{U}_j$ and it further holds that

$$\langle d, \mathcal{V}_j d \rangle = \langle d, \mathcal{B}\mathcal{U}_j\mathcal{B}^* d \rangle = \sum_{(k,l)\in\Gamma_j} (\mathcal{B}^* d)_{kl}^2 \geq 0, \quad \forall\, d \in \Re^{2n},$$

where the last equation follows from the definition of $\mathcal{U}_j$ given in (18) and the index set Γ_j is defined by $\Gamma_j := \{(k, l) \mid (\mathcal{B}^* y^j + G)_{kl} \geq 0,\ 1 \leq k, l \leq n\}$. The convergence results for the above Ssncg1 algorithm are stated in the next theorem.

Theorem 2 *Let $\{y^j\}$ be the infinite sequence generated by Algorithm* Ssncg1 *for solving problem* (15). *Then, $\{y^j\} \subseteq Ran(\mathcal{B})$ is a bounded sequence and any accumulation point $\hat{y}$ $(\in Ran(\mathcal{B}))$ of $\{y^j\}$ is an optimal solution to problem* (15).

Proof Since $\nabla\varphi(y) \in \mathrm{Ran}(\mathcal{B})$ for any given $y \in \Re^{2n}$, from the properties of the CG algorithm [41, Theorem 38.1], we know that for all $j \geq 0$, $d^j \in \mathrm{Ran}(\mathcal{B})$. Thus, $\{y^j\} \subseteq \mathrm{Ran}(\mathcal{B})$. All the other results follow directly from [45, Theorem 3.4]. □

Next, we state a theorem on the convergence rate of Algorithm Ssncg1. We shall omit the proof here as it can be proved in the same fashion as [45, Theorem 3.5].

Theorem 3 *Let $\bar{y}$ be an accumulation point of the infinite sequence $\{y^j\}$ generated by Algorithm* Ssncg1 *for solving problem* (15). *Assume that the following constraint nondegeneracy condition*

$$\mathcal{B}\,\mathrm{lin}(\mathcal{T}_C(\widehat{G})) = Ran(\mathcal{B}) \tag{21}$$

holds at $\widehat{G} := \Pi_C(\mathcal{B}^*\bar{y} + G)$*, where* $\mathrm{lin}(\mathcal{T}_C(\widehat{G}))$ *denotes the lineality space of the tangent cone of* C *at* $\widehat{G}$*. Then, the whole sequence* $\{y^j\}$ *converges to* $\bar{y}$ *and*

$$\|y^{j+1} - \bar{y}\| = O(\|y^j - \bar{y}\|^{1+\tau}).$$

Remark 2 In fact, given the piecewise linear-quadratic structure in problem (15), the results given in [12] and [39] further imply that our Algorithm SSNCG1 with the Newton linear systems (20) solved exactly can enjoy a finite termination property. Therefore, we can expect to obtain an approximate solution to (12) through Algorithm SSNCG1 with the error on the order of the machine precision (provided the rounding errors introduced by the intermediate computations are not amplified significantly).

3.1 Efficient implementations

In our implementation of Algorithm SSNCG1, the key part is to solve the linear system (20) efficiently. Note that a similar linear system is also involved in the calculation of the HS-Jacobian in (14). Here, we propose to use the conjugate gradient method to solve (20). In this subsection, we shall discuss the efficient implementation of the corresponding matrix vector multiplications.

Let

$$V := B\,\mathrm{Diag}(\mathbf{vec}(\Omega))B^T \in \Re^{2n\times 2n},$$

where $B \in \Re^{2n\times n^2}$ denotes the matrix representation of $\mathcal{B}$ with respect to the standard basis of $\Re^{n\times n}$ and $\Re^{2n}$ and Ω is given in (19). Given $\varepsilon \geq 0$, we shall focus on the following linear system

$$(V + \varepsilon I_{2n})d = r. \tag{22}$$

Here, $r \in \Re^{2n}$ is a given vector. At the first glance, the cost of computing the matrix-vector multiplication Vd for a given vector $d \in \Re^{2n}$ would be very expensive when the dimension n is large. Fortunately, the matrix B has a special structure which we can exploit to derive a closed form formula for V. Indeed, we have that

$$B = \begin{bmatrix} e^T \otimes I_n \\ I_n \otimes e^T \end{bmatrix},$$

where "$\otimes$" denotes the Kronecker product. Therefore, we can derive the closed form representation of V as follows:

$$V = \begin{bmatrix} \mathrm{Diag}(\Omega e) & \Omega \\ (\Omega)^T & \mathrm{Diag}((\Omega)^T e) \end{bmatrix}.$$

Now it is clear that the computational cost of Vd is only of the order $\mathcal{O}(n^2)$. Furthermore, from the 0-1 structure of Ω and a close examination of the sparsity of Ω, it is not difficult to show that the computational cost of Vd can further be reduced to $\min\left\{\mathcal{O}(\gamma + n), \mathcal{O}(n^2 - \gamma + n)\right\}$, where γ is the number of nonzero elements in Ω. Following the terminology used in [24], this sparsity will be referred to as the second order sparsity of the underlying projection problem. Similar as is shown in [24], this second order sparsity is the key ingredient for our efficient implementation of Algorithm SSNCG1. Meanwhile, from the above representation of V, we can construct a simple preconditioner for the coefficient matrix in (22) as follows

$$\widehat{V} := \mathrm{Diag}([e^T(\Omega)^T \ e^T\Omega]^T) + \varepsilon I_{2n}.$$

Clearly, $\widehat{V}$ will be a good approximation for $V + \varepsilon I_{2n}$ when Ω is a sparse matrix.

4 Quadratic programming problems with Birkhoff polytope constraints

As a demonstration on how one can take advantage of the efficient computation of the projection $\Pi_{\mathfrak{B}_n}$ and its HS-Jacobian presented in the last section, here we show how such an efficient computation can be employed in the design of efficient algorithms for solving the following convex quadratic programming problem:

$$(\mathbf{P}) \quad \min\left\{ f(X) := \frac{1}{2}\langle X,\ \mathcal{Q}X\rangle + \langle G,\ X\rangle + \delta_{\mathfrak{B}_n}(X) \right\},$$

where $\mathcal{Q} : \Re^{n\times n} \to \Re^{n\times n}$ is a self-adjoint positive semidefinite linear operator, $G \in \Re^{n\times n}$ is a given matrix, $\delta_{\mathfrak{B}_n}$ is the indicator function of $\mathfrak{B}_n$. Its dual problem in the minimization form is given by

$$(\mathbf{D}) \quad \min\left\{ \delta^*_{\mathfrak{B}_n}(Z) + \frac{1}{2}\langle W,\ \mathcal{Q}W\rangle \mid Z + \mathcal{Q}W + G = 0,\ W \in \mathrm{Ran}(\mathcal{Q}) \right\},$$

where $\mathrm{Ran}(\mathcal{Q})$ denotes the range space of $\mathcal{Q}$ and $\delta^*_{\mathfrak{B}_n}$ is the conjugate of the indicator function $\delta_{\mathfrak{B}_n}$. Similar to the subspace constraint $y \in \mathrm{Ran}(\mathcal{B})$ in problem (15), the constraint $W \in \mathrm{Ran}(\mathcal{Q})$ ensures the boundedness of the solution set of (**D**). Specifically, under this subspace constraint, since $\mathfrak{B}_n$ is a compact set with a nonempty interior, we know that both the primal and dual optimal solution sets are nonempty and compact. In addition, the fact that $\mathcal{Q}$ is positive definite on $\mathrm{Ran}(\mathcal{Q})$ further implies that problem (**D**) has a unique optimal solution $(Z^*, W^*) \in \Re^{n\times n} \times \mathrm{Ran}(\mathcal{Q})$.

Equipped with the efficient solver (SSNCG1 developed in the last section) for computing $\Pi_{\mathfrak{B}_n}(\cdot)$, it is reasonable for us to use a simple first order method to solve (**P**) and (**D**). For example, one can adapt the accelerated proximal gradient (APG) [2,29] method to solve (**P**) and the classic two block alternating direction method of multipliers [14,15] method with the step-length of 1.618 to solve (**D**). However, these first order methods may encounter stagnation difficulties or suffer from extremely slow local convergence, especially when one is searching for high accuracy solutions for (**P**) and (**D**). In order to be competitive against state-of-the-art interior point method based QP solvers such as those implemented in Gurobi, here we propose a semismooth Newton based augmented Lagrangian method for solving (**D**), wherein we show how one can take full advantage of the efficient computation of $\Pi_{\mathfrak{B}_n}(\cdot)$ and its HS-Jacobian to design a fast algorithm.

Here, the main reason for using the dual ALM approach is that the subproblem in each iteration of the dual ALM is a strongly convex minimization problem. Armed with this critical property, as will be shown in Theorem 5 and Remark 4, one can naturally apply the inexact semismooth Newton-CG method to solve a reduced problem in the variable $W \in \mathrm{Ran}(\mathcal{Q}) \subset \mathcal{S}^n$ and the SSNCG method is guaranteed to converge superlinearly (or even quadratically if the inexact direction is computed with high accuracy). In contrast, if one were to apply the ALM to the primal problem, $\min\{\frac{1}{2}\langle X,\ \mathcal{Q}X\rangle + \langle G,\ X\rangle + \delta_{\mathcal{B}_n}(X)\}$, one would first introduce the constraint $X - Y = 0$ to make the terms in the objective function separable, i.e., $\min\{\frac{1}{2}\langle X,\ \mathcal{Q}X\rangle + \langle G,\ X\rangle + \delta_{\mathcal{B}_n}(Y) \mid X - Y = 0\}$. Then the corresponding reduced subproblem at the kth iteration of the primal ALM approach would take the following form:

$$\min\Big\{\phi_k(X) := \frac{1}{2}\langle X,\ \mathcal{Q}X\rangle + \langle G,\ X\rangle + \frac{\sigma}{2}\|(X + \sigma^{-1}\Lambda^k) \\ -\Pi_{\mathcal{B}_n}(X + \sigma^{-1}\Lambda^k)\|^2 \mid X \in \mathcal{S}^n\Big\},$$

where Λ^k denotes the multiplier corresponding to the constraint $X - Y = 0$. However, for this reduced subproblem in the variable $X \in \mathcal{S}^n$, the objective function is not necessarily strongly convex when $\mathcal{Q}$ is singular (especially for the extreme case when $\mathcal{Q} = 0$). Therefore, the SSNCG method applied to this reduced subproblem in X may not have superlinear linear convergence.

Given $\sigma > 0$, the augmented Lagrangian function associated with (**D**) is given as follows:

$$\mathcal{L}_\sigma(Z, W; X) = \delta^*_{\mathfrak{B}_n}(Z) + \frac{1}{2}\langle W,\ \mathcal{Q}W\rangle - \langle X,\ Z + \mathcal{Q}W + G\rangle + \frac{\sigma}{2}\|Z + \mathcal{Q}W + G\|^2,$$

where $(Z, W, X) \in \Re^{n\times n} \times \mathrm{Ran}(\mathcal{Q}) \times \Re^{n\times n}$. The augmented Lagrangian method for solving (**D**) has the following template. In the algorithm, the notation $\sigma_{k+1} \uparrow \sigma_\infty \le \infty$ means that $\sigma_{k+1} \ge \sigma_k$ and the limit of $\{\sigma_k\}$, denoted as σ_∞, can be some constant finite number or ∞. As one can observed later in Theorem 4, the global convergence of Algorithm ALM can be obtained without requiring $\sigma_\infty = \infty$.

Algorithm ALM: **An augmented Lagrangian method for solving (D).**

Let $\sigma_0 > 0$ be a given parameter. Choose $(W^0, X^0) \in \mathrm{Ran}(\mathcal{Q}) \times \Re^{n\times n}$ and $Z^0 \in \mathrm{dom}(\delta^*_{\mathfrak{B}_n})$. For $k = 0, 1, \ldots,$ perform the following steps in each iteration:

Step 1. Compute

$$(Z^{k+1}, W^{k+1}) \\ \approx \mathrm{argmin}\Big\{\Psi_k(Z, W) := \mathcal{L}_{\sigma_k}(Z, W; X^k) \mid (Z, W) \in \Re^{n\times n} \times \mathrm{Ran}(\mathcal{Q})\Big\}. \tag{23}$$

Step 2. Compute

$$X^{k+1} = X^k - \sigma_k(Z^{k+1} + \mathcal{Q}W^{k+1} + G). \tag{24}$$

Update $\sigma_{k+1} \uparrow \sigma_\infty \le \infty$.

We shall discuss first the stopping criteria for approximately solving subproblem (23). For any $k \geq 0$, define

$$f_k(X) = -\frac{1}{2}\langle X,\ \mathcal{Q}X\rangle - \langle X,\ G\rangle - \frac{1}{2\sigma_k}\|X - X^k\|^2, \quad \forall X \in \Re^{n\times n}.$$

Note that $f_k(\cdot)$ is in fact the objective function in the dual problem of (23). Let $\{\varepsilon_k\}$ and $\{\delta_k\}$ be two given positive summable sequences. Given $X^k \in \Re^{n\times n}$, we propose to terminate solving the subproblem (23) with either one of the following two easy-to-check stopping criteria:

$$\text{(A)} \quad \begin{cases} \Psi_k(Z^{k+1}, W^{k+1}) - f_k(X^{k+1}) \leq \varepsilon_k^2/2\sigma_k, \\ \gamma(X^{k+1}) \leq \alpha_k \varepsilon_k/\sqrt{2\sigma_k}, \end{cases}$$
$$\text{(B)} \quad \begin{cases} \Psi_k(Z^{k+1}, W^{k+1}) - f_k(X^{k+1}) \leq \delta_k^2\|X^{k+1} - X^k\|^2/2\sigma_k, \\ \gamma(X^{k+1}) \leq \beta_k \delta_k \|X^{k+1} - X^k\|/\sqrt{2\sigma_k}, \end{cases}$$

where $\gamma(X^{k+1}) := \|X^{k+1} - \Pi_{\mathfrak{B}_n}(X^{k+1})\|$,

$$\alpha_k = \min\left\{1, \sqrt{\sigma_k}, \frac{\varepsilon_k}{\sqrt{2\sigma_k}\|\nabla f_k(X^{k+1})\|}\right\}$$
$$\text{and} \quad \beta_k = \min\left\{1, \sqrt{\sigma_k}, \frac{\delta_k\|X^{k+1} - X^k\|}{\sqrt{2\sigma_k}\|\nabla f_k(X^{k+1})\|}\right\}.$$

From [9, Proposition 4.3] and [23, Lemma 2.2], criteria (A) and (B) can be used in ALM to guarantee the global and local convergence of ALM. Indeed, from J. Sun's thesis [40] on the investigation of the subdifferentials of convex piecewise linear-quadratic functions, we know that ∂f is a polyhedral multifunction (see also [38, Proposition 12.30]). The classic result of Robinson [34] on polyhedral multifunctions further implies that Luque's error bound condition [27, (2.1)] associated with ∂f is satisfied, i.e., there exist positive constants δ and κ such that

$$\operatorname{dist}(z, \partial f^{-1}(0)) \leq \kappa\|u\|, \quad \forall z \in \partial f^{-1}(u), \quad \forall \|u\| \leq \delta. \tag{25}$$

Thus we can prove the global and local (super)linear convergence of Algorithm ALM by adapting the proofs in [37, Theorem 4], [27, Theorem 2.1] and [9, Theorem 4.2]. The next theorem shows that for the convex QP problem (**P**), one can always expect the KKT residual of the sequence generated by the ALM to converge at least R-(super)linearly.

Let the objective function $g : \Re^{n\times n} \times \operatorname{Ran}(\mathcal{Q}) \to (-\infty, +\infty]$ associated with (**D**) be given by

$$g(Z, W) := \delta^*_{\mathfrak{B}_n}(Z) + \frac{1}{2}\langle W,\ \mathcal{Q}W\rangle, \quad \forall (Z, W) \in \Re^{n\times n} \times \operatorname{Ran}(\mathcal{Q}).$$

Theorem 4 *The sequence $\{(Z^k, W^k, X^k)\}$ generated by Algorithm ALM under the stopping criterion (A) for all $k \geq 0$ is bounded, and $\{X^k\}$ converges to an optimal* *solution X^∞ of* (**P**)*. In addition,* $\{(Z^k, W^k)\}$ *converges to the unique optimal solution of* (**D**)*. Moreover, for all $k \geq 0$, it holds that*

$$\begin{aligned} & g(Z^{k+1}, W^{k+1}) - \inf\,(\mathbf{D}) \\ & \quad \leq \Psi_k(Z^{k+1}, W^{k+1}) - \inf \Psi_k + (1/2\sigma_k)(\|X^k\|^2 - \|X^{k+1}\|^2). \end{aligned}$$

Let Ω be the nonempty compact optimal solution set of (**P**)*. Suppose that the algorithm is executed under criteria* (A) *and* (B) *for all $k \geq 0$. Then, for all k sufficiently large, it holds that*

$$\begin{aligned} \mathrm{dist}(X^{k+1}, \Omega) &\leq \theta_k \mathrm{dist}(X^k, \Omega), \\ \|Z^{k+1} - \mathcal{Q}W^{k+1} - G\| &\leq \tau_k \mathrm{dist}(X^k, \Omega), \\ g(Z^{k+1}, W^{k+1}) - \inf\,(\mathbf{D}) &\leq \tau_k' \mathrm{dist}(X^k, \Omega), \end{aligned}$$

where $0 \leq \theta_k, \tau_k, \tau_k' < 1$ and $\theta_k \to \theta_\infty = \kappa/\sqrt{\kappa^2 + \sigma_\infty^2}$, $\tau_k \to \tau_\infty = 1/\sigma_\infty$ and $\tau_k' \to \tau_\infty' = \|X^\infty\|/\sigma_\infty$ with κ given in (25). Moreover, $\theta_\infty = \tau_\infty = \tau_\infty' = 0$ if $\sigma_\infty = \infty$.

Remark 3 We also note that if the ALM is used to solve an equivalent reformulation of the primal form of (**P**) and the corresponding subproblems are solved exactly, then the global linear convergence of a certain constraint norm to zero can be established via using the results developed in [7,10].

Next, we shall discuss how to solve the subproblems (23) efficiently. Given $\sigma > 0$ and $\widehat{X} \in \Re^{n\times n}$, since $\mathcal{L}_\sigma(Z, W; \widehat{X})$ is strongly convex on $\Re^{n\times n} \times \mathrm{Ran}(\mathcal{Q})$, we have that, for any $\alpha \in \Re$, the level set $\mathcal{L}_\alpha := \{(Z, W) \in \Re^{n\times n} \times \mathrm{Ran}(\mathcal{Q}) \mid \mathcal{L}_\sigma(Z, W; \widehat{X}) \leq \alpha\}$ is a closed and bounded convex set. Moreover, the optimization problem

$$\min \left\{\mathcal{L}_\sigma(Z, W; \widehat{X}) \mid (Z, W) \in \Re^{n\times n} \times \mathrm{Ran}(\mathcal{Q})\right\} \tag{26}$$

admits a unique optimal solution, which we denote as $(\overline{Z}, \overline{W}) \in \Re^{n\times n} \times \mathrm{Ran}(\mathcal{Q})$. Define

$$\psi(W) := \inf_Z \mathcal{L}_\sigma(Z, W; \widehat{X}) \quad \text{and} \quad Z(W) := \widehat{X} - \sigma(\mathcal{Q}W + G), \quad \forall\, W \in \mathrm{Ran}(\mathcal{Q}).$$

It is not difficult to see that $\inf_{W\in\mathrm{Ran}(\mathcal{Q})} \psi(W) = \inf_{Z,W\in\mathrm{Ran}(\mathcal{Q})} \mathcal{L}_\sigma(Z, W; \widehat{X})$ and

$$\sigma^{-1}\big(Z(W) - \Pi_{\mathfrak{B}_n}(Z(W))\big) = \arg\min_Z \mathcal{L}_\sigma(Z, W; \widehat{X}), \quad \forall\, W \in \mathrm{Ran}(\mathcal{Q}).$$

Therefore, $(\overline{Z}, \overline{W})$ solves the minimization problem (26) if and only if

$$\begin{aligned} & \overline{W} = \arg\min \left\{\psi(W) \mid W \in \mathrm{Ran}(\mathcal{Q})\right\}, \\ & \overline{Z} = \sigma^{-1}\big(Z(\overline{W}) - \Pi_{\mathfrak{B}_n}(Z(\overline{W}))\big) = \arg\min_Z \mathcal{L}_\sigma(Z, \overline{W}; \widehat{X}). \end{aligned} \tag{27}$$

Simple calculations show that for all $W \in \mathrm{Ran}(\mathcal{Q})$,

$$\psi(W) = \frac{1}{2}\langle W,\ \mathcal{Q}W\rangle + \frac{1}{\sigma}\langle Z(W),\ \Pi_{\mathfrak{B}_n}(Z(W))\rangle - \frac{1}{2\sigma}(\|\Pi_{\mathfrak{B}_n}(Z(W))\|^2 + \|\widehat{X}\|^2).$$

Note that ψ is strongly convex and continuously differentiable on $\mathrm{Ran}(\mathcal{Q})$ with

$$\nabla\psi(W) = \mathcal{Q}W - \mathcal{Q}\Pi_{\mathfrak{B}_n}(Z(W)).$$

Thus, $\overline{W}$, the optimal solution of (27), can be obtained through solving the following nonsmooth piecewise affine equation:

$$\nabla\psi(W) = 0, \quad W \in \mathrm{Ran}(\mathcal{Q}).$$

Given $\widehat{W}$, define the following linear operator $\mathcal{M} : \Re^{n\times n} \to \Re^{n\times n}$ by

$$\mathcal{M}(\Delta W) := (\mathcal{Q} + \sigma\mathcal{Q}\mathcal{P}\mathcal{Q})\Delta W, \quad \forall \Delta W \in \Re^{n\times n},$$

where $\mathcal{P}$ is the HS-Jacobian of $\Pi_{\mathfrak{B}_n}$ at $Z(\widehat{W})$ as given in (14) and it is self-adjoint and positive semidefinite. Moreover, since $\mathcal{Q}$ is self-adjoint and positive definite on $\mathrm{Ran}\mathcal{Q}$, it follows that $\mathcal{M}$ is also self-adjoint and positive definite on $\mathrm{Ran}\mathcal{Q}$. Similarly as in Sect. 3, we propose to solve the subproblem (27) by an inexact semismooth Newton method and $\mathcal{M}$ will be regarded as a computable generalized Hessian of ψ at $\widehat{W}$.

Algorithm SSNCG2: **A semismooth Newton-CG algorithm for solving (27)**.

Given $\mu \in (0, 1/2)$, $\bar{\eta} \in (0, 1)$, $\tau \in (0, 1]$, and $\delta \in (0, 1)$, choose $W^0 \in \mathrm{Ran}(\mathcal{Q})$. Iterate the following steps for $j = 0, 1, \ldots$:

Step 1. Let $\mathcal{M}_j := \mathcal{Q} + \sigma\mathcal{Q}\mathcal{P}_j\mathcal{Q}$ where $\mathcal{P}_j$ is the HS-Jacobian of $\Pi_{\mathfrak{B}_n}$ at $Z(W^j)$ given in (14). Apply the CG algorithm to find an approximate solution dW^j to the following linear system

$$\mathcal{M}_j dW + \nabla\psi(W^j) = 0, \quad dW \in \mathrm{Ran}(\mathcal{Q}) \tag{28}$$

such that

$$\|\mathcal{M}_j dW^j + \nabla\psi(W^j)\| \le \min(\bar{\eta}, \|\nabla\psi(W^j)\|^{1+\tau}).$$

Step 2. (Line search) Set $\alpha_j = \delta^{m_j}$, where m_j is the first nonnegative integer m for which

$$\psi(W^j + \delta^m dW^j) \le \psi(W^j) + \mu\delta^m\langle\nabla\psi(W^j), dW^j\rangle.$$

Step 3. Set $W^{j+1} = W^j + \alpha_j\, dW^j$.

Similar to Theorem 2 and Theorem 3, it is not difficult to obtain the following theorem on the global and local superlinear (quadratic) convergence for the above Algorithm Ssncg2. Its proof is omitted for brevity.

Theorem 5 *Let $\{W^j\}$ be the infinite sequence generated by Algorithm* Ssncg2*. Then $\{W^j\}$ converges to the unique optimal solution $\overline{W} \in Ran(\mathcal{Q})$ to problem (27) and*

$$\|W^{j+1} - \overline{W}\| = O(\|W^j - \overline{W}\|^{1+\tau}).$$

Remark 4 Note that in the above theorem, since $\mathcal{Q}$ is positive definite on $\mathrm{Ran}(\mathcal{Q})$, we know that for each $j \geq 0$, $\mathcal{M}_j$ is also positive definite on $\mathrm{Ran}(\mathcal{Q})$. Therefore, we do not need any nondegeneracy condition assumption here as is required in Theorem 3.

Remark 5 The restriction of $dW \in \mathrm{Ran}(\mathcal{Q})$ appears to introduce severe numerical difficulties when we need to solve (28). Fortunately, we can overcome these difficulties via a careful examination of our algorithm and some numerical techniques. Indeed, at the jth iteration of Algorithm Ssncg2, instead of dealing with (28), we propose to solve the following simpler linear system

$$(\mathcal{I} + \sigma\mathcal{P}\mathcal{Q})dW = \Pi_{\mathfrak{B}_n}(Z(W^j)) - W^j, \tag{29}$$

where $\mathcal{I}$ is the identity operator defined on $\Re^{n\times n}$. Then, the approximate solution to (29) can be safely used as a replacement of dW^j in the execution of Algorithm Ssncg2. We omit the details here for brevity. Interested readers may refer to Sect. 4 in [23] for a detailed discussion on why this procedure is legitimate.

Remark 6 At the kth iteration of Algorithm ALM, given X^k and σ_k, we first obtain W^{k+1} via executing Algorithm Ssncg2. Then, we have that

$$Z^{k+1} = \sigma_k^{-1}(X^k - \sigma_k(\mathcal{Q}W^{k+1} + G) - \Pi_{\mathcal{B}_n}(X^k - \sigma_k(\mathcal{Q}W^{k+1} + G))).$$

Therefore, it is easy to see that the multiplier update step (24) in Algorithm ALM can be equivalently recast as:

$$X^{k+1} = X^k - \sigma_k(Z^{k+1} + \mathcal{Q}W^{k+1} + G) = \Pi_{\mathcal{B}_n}(X^k - \sigma_k(\mathcal{Q}W^{k+1} + G)).$$

5 Numerical experiments

In this section, we evaluate the performance of our algorithms from various aspects. We have implemented all our algorithms in Matlab. Unless otherwise specifically stated, all our computational results are obtained from a 12-core workstation with Intel Xeon E5-2680 processors at 2.50GHz and 128GB memory. The experiments are run in Matlab 8.6 and Gurobi 6.5.2 [16] (with an academic license) under the 64-bit Windows Operating System. It is well known that Gurobi is an extremely powerful solver for solving generic quadratic programming problems. It is our view that any credible algorithms designed for solving a specialized class of QP problems should be benchmarked against Gurobi and be able to demonstrate its advantage over Gurobi. But we should note that as a general QP solver, Gurobi does not necessarily fully exploit the specific structure of the Birkhoff polytope, although it can fully exploit the sparsity of the constraint matrices and variables.

5.1 Numerical results for the projection onto the Birkhoff polytope

First we compare our Algorithm SSNCG1 with the state-of-the-art solver, Gurobi, for solving large scale instances of the projection problems (12) and its dual (15). Note that dual problem (15) is an unconstrained smooth convex optimization problem. For solving such a problem, the accelerated proximal gradient (APG) method of Nesterov [29] has become very popular due to its simplicity in implementation and strong iteration complexity. As it is a very natural method for one to adopt in the first attempt to solve (15), we also implement the APG method for solving (15) for comparison purposes.

Recall from (15) that $C = \{X \in \Re^{n\times n} \mid X \geq 0\}$ and define the function $h : \Re^{n\times n} \to \Re$ by $h(Z) = \frac{1}{2}\|\Pi_C(Z)\|^2$. Note that

$$\nabla h(Z) = \Pi_C(Z) \quad \text{and} \quad \|\nabla h(Y) - \nabla h(Z)\| \leq \|Y - Z\|, \quad \forall\, Y, Z \in \Re^{n\times n}.$$

Given $\hat{y} \in \Re^{2n}$, the Lipschitz continuity of ∇h implies that for all $y \in \Re^{2n}$,

$$\begin{aligned}\frac{1}{2}\|\Pi_C(\mathcal{B}^*y + G)\|^2 &\leq \frac{1}{2}\|\Pi_C(\mathcal{B}^*\hat{y} + G)\|^2\\ &\quad + \langle \Pi_C(\mathcal{B}^*\hat{y} + G),\ \mathcal{B}^*(y - \hat{y})\rangle + \frac{1}{2}\|\mathcal{B}^*(y - \hat{y})\|^2.\end{aligned}$$

From the above inequality, we can derive the following simple upper bound for φ:

$$\varphi(y) \leq \hat{\varphi}(y; \hat{y}) := \varphi(\hat{y}) + \langle \nabla\varphi(\hat{y}),\ y - \hat{y}\rangle + \frac{1}{2}\|\mathcal{B}^*y - \mathcal{B}^*\hat{y}\|^2, \quad \forall\, y \in \Re^{2n}. \tag{30}$$

The APG method we implemented here is based on (30). The detailed steps of the APG method for solving (15) are given as follows.

Algorithm APG: An accelerated proximal gradient algorithm for (15).

Given $y^0 \in \mathrm{Ran}(\mathcal{B})$, set $z^1 = y^0$ and $t_1 = 1$. For $j = 1, \ldots,$ perform the following steps in each iteration:

Step 1. Compute $\nabla\varphi(z^j) = \mathcal{B}\Pi_C(\mathcal{B}^*z^j + G) - b$. Then compute

$$y^j = \operatorname{argmin}\left\{\hat{\varphi}(y; z^j) \ \mid y \in \mathrm{Ran}(\mathcal{B})\right\}$$

via solving the following linear system:

$$\mathcal{B}\mathcal{B}^*y = \mathcal{B}\mathcal{B}^*z^j - \nabla\varphi(z^j), \quad y \in \mathrm{Ran}(\mathcal{B}). \tag{31}$$

Step 2. Set $t_{j+1} = \frac{1+\sqrt{1+4t_j^2}}{2}$, $\beta_j = \frac{t_j - 1}{t_{j+1}}$. Compute $z^{j+1} = y^j + \beta_j(y^j - y^{j-1})$.

Note that since $b \in \operatorname{Ran}(\mathcal{B})$, the solution $y^j \in \operatorname{Ran}(\mathcal{B})$ to equation (31) is in fact unique. Hence, Algorithm APG is well defined. In our implementation, we further use the restarting technique to accelerate the convergence of the algorithm.

In our numerical experiments, we measure the accuracy of an approximate optimal solution (X, y) for problem (12) and its dual problem (15) by using the following relative KKT residual:

$$\eta = \max\{\eta_P, \eta_C\},$$

where

$$\eta_P = \frac{\|\mathcal{B}X - b\|}{1 + \|b\|}, \quad \eta_C = \frac{\|X - \Pi_C(\mathcal{B}^* y + G)\|}{1 + \|X\|}.$$

We note that for the Gurobi solver, the primal infeasibility η_P associated with the computed approximate solution is usually very small. On the other hand, for Algorithm Ssncg1 and Algorithm APG, since the solution X is obtained through the dual approach, i.e., $X = \Pi_C(\mathcal{B}^* y + G)$, we have that for these two algorithms, $\eta_C = 0$.

Let $\varepsilon > 0$ be a given tolerance. We terminate both algorithms Ssncg1 and APG when $\eta < \varepsilon$. The algorithms will also be stopped when they reach the maximum number of iterations (1000 iterations for Ssncg1 and 20,000 iterations for APG) or the maximum computation time of 3 h. For the Gurobi solver, we use the default parameter settings, i.e., using the default stopping tolerance and all 12 computing cores.

In this subsection, we test 17 instances of the given matrix G for (12) with dimensions n ranging from 10^3 to 3.2×10^4. Among these test instances, 6 of them are similarity matrices derived from the LIBSVM datasets [6]: **gisette**, **mushrooms**, **a6a**, **a7a**, **rcv1** and **a8a**. Similarly as in [43], we first normalize each data point to have a unit l_2-norm and use the following Gaussian kernel to generate G, i.e.,

$$G_{ij} = \exp\left(-\|x_i - x_j\|^2\right), \quad \forall\, 1 \le i, j \le n.$$

The other 11 instances are randomly generated using the Matlab command: `G = randn(n)`.

In Table 1, we report the numerical results obtained by Ssncg1, APG and Gurobi in solving various instances of the projection problem (12). Here, we terminate algorithms APG and Ssncg1 when $\eta < 10^{-9}$. In order to further demonstrate the ability of Ssncg1 in computing highly accurate solutions, we also report the results obtained by Ssncg1 in solving the instances to the accuracy of 10^{-15}. In the table, the first two columns give the name of problems and the size of G in (12). The number of iterations, the relative KKT residual η and computation times (in the format hours:minutes:seconds) are listed in the last twelve columns. For Gurobi, we also list the relative primal feasibility η_P. As one can observe, although Gurobi can produce a very small η_P, the corresponding relative KKT residual η can only reach the accuracy about 10^{-5} to 10^{-6}. In other words, comparing to Gurobi, the solutions produced by Ssncg1 and APG with the tolerance of $\varepsilon = 10^{-9}$ are already more accurate.

Table 1 The performance of SSNCG1, APG, and Gurobi on the projection problem (12) and its dual (15). In the table, “a” and “c1” stand for APG and SSNCG1 with the tolerance $\varepsilon = 10^{-9}$; “b” stands for Gurobi; “c2” stands for SSNCG1 with $\varepsilon = 10^{-15}$. The entry “*” indicates out of memory. The computation time is in the format of “hours:minutes:seconds”

		Iter	η	Time
problem	n	a | b | c1 | c2	a | b(η_P) | c1 | c2	a | b | c1 | c2
rand1	1000	1350 | 15 | 12 | 13	9.7–10 | **4.1–5** (1.2–15) | 8.7–12 | 5.2–16	18 | 06 | 01 | 01
rand2	2000	2630 | 17 | 13 | 14	9.9–10 | **2.4–5** (1.5–15) | 4.4–12 | 4.5–16	2:21 | 31 | 02 | 02
rand3	4000	3544 | 21 | 14 | 15	9.9–10 | **8.4–6** (2.5–15) | 2.2–13 | 4.1–16	12:31 | 2:30 | 07 | 08
rand4	8000	6454 | 25 | 14 | 16	9.9–10 | **2.3–6** (1.8–14) | 4.2–10 | 4.3–16	1:30:05 | 13:02 | 27 | 34
rand5	10000	8234 | 25 | 14 | 16	**3.1–7** | **6.2–6** (3.4–15) | 8.6–11 | 4.5–16	3:00:00 | 21:27 | 44 | 58
rand6	12000	5565 | 25 | 15 | 17	**2.6–4** | **4.6–6** (3.8–15) | 2.0–11 | 4.6–16	3:00:00 | 33:31 | 1:14 | 1:33
rand7	16000	3061 | ∗ | 15 | 16	**1.6–3** | ∗ | 3.7–12 | 5.9–16	3:00:02 | ∗ | 2:26 | 2:55
rand8	20000	1646 | ∗ | 16 | 17	**6.8–3** | ∗ | 1.7–11 | 9.5–16	3:00:07 | ∗ | 4:08 | 4:45
rand9	24000	1014 | ∗ | 16 | 17	**1.9–2** | ∗ | 2.9–13 | 4.9–16	3:00:04 | ∗ | 6:14 | 7:15
rand10	30000	622 | ∗ | 16 | 17	**4.9–2** | ∗ | 3.8–12 | 5.9–16	3:00:14 | ∗ | 9:53 | 12:01
rand11	32000	559 | ∗ | 16 | 18	**6.3–2** | ∗ | 2.0–11 | 4.8–16	3:00:17 | ∗ | 11:57 | 14:10
gisette	6000	928 | 24 | 11 | 12	9.8–10 | **3.3–6** (2.5–15) | 9.1–12 | 6.5–16	7:19 | 6:58 | 14 | 16
mushrooms	8124	763 | 20 | 11 | 13	9.8–10 | **9.5–5** (4.8–15) | 2.8–10 | 1.9–16	11:07 | 11:58 | 27 | 32
a6a	11220	1227 | 26 | 13 | 14	9.9–10 | **4.7–6** (4.0–15) | 5.4–12 | 3.8–16	34:29 | 31:21 | 59 | 1:03
a7a	16100	1377 | ∗ | 14 | 15	9.9–10 | ∗ | 7.5–13 | 2.9–16	1:28:14 | ∗ | 2:14 | 2:34
rcv1	20242	1583 | ∗ | 17 | 18	**1.3–6** | ∗ | 2.0–12 | 1.9–16	3:00:03 | ∗ | 4:33 | 5:02
a8a	22696	1330 | ∗ | 14 | 16	**2.7–4** | ∗ | 9.7–10 | 2.5–16	3:00:03 | ∗ | 5:12 | 6:15

One can also observe from Table 1 that only our algorithm Ssncg1 can solve all the test problems to the required accuracies of $\eta < 10^{-9}$ and $\eta < 10^{-15}$. Indeed, APG can only solve 8 smaller instances out of 17 to the desired accuracy after 3 h and Gurobi reported out of memory when the size of G is larger than 12,000. Moreover, Ssncg1 is much faster than APG and Gurobi for all the test instances. For example, for the instance **rand4**, Ssncg1 is at least 26 times faster than Gurobi and 180 times faster than APG. In addition, Ssncg1 can solve **rand11**, a quadratic programming problem with over 1 billion variables and nonnegative constraints, to the extremely high accuracy of 5×10^{-16} in about 14 min while APG consumed 3 h to only produce a solution with an accuracy of 6×10^{-2}. We also emphasize here that from the accuracy of 10^{-9} to the much higher accuracy of 10^{-15}, Ssncg1 only needs one or two extra iterations and consumes insignificant additional time. The latter observation truly confirmed the power of the quadratic (or at least superlinear) convergence property of Algorithm Ssncg1 and the power of exploiting the second order sparsity property of the underlying projection problem within the algorithm.

Since the worst-case iteration complexity of APG is only sublinear, it is not surprising that the performance of APG is relatively poor compared to Ssncg1. We also note that comparing to small scale problems, APG needs much more iterations to obtain relatively accurate solutions for large scale problems. For example, for the instance **rand6**, APG took 3 h and 5565 iterations to only generate a relatively inaccurate solution with an accuracy of 3×10^{-4}. Despite this, for small scale instances (especially the instances **gisette**, **mushrooms** and **a6a**), APG, although much slower than Ssncg1, can obtain accurate solutions with computation time comparable to the powerful commercial solver Gurobi. Thus, as a first-order method, it is already quite powerful.

Figure 1a plots the KKT residual η against the iteration count of Ssncg1 for solving the instance **a8a**. Clearly, our algorithm Ssncg1 exhibits at least a superlinear convergence behavior when approaching the optimal solution. In Fig. 1b, we compare the computational complexities of Ssncg1 and Gurobi when used to solve the 17 projection problems in Table 1. It shows that the time t (in s) taken to solve a problem of dimension n is given by $t = \exp(-16)n^{2.1}$ for Ssncg1 and $t = \exp(-14)n^{2.3}$ for Gurobi. One can further observe that on the average, for a given n in the range from $[\exp(6), \exp(11)]$, our algorithm is at least $7n^{0.2}$ times faster than Gurobi.

In Table 2, we report the detailed results obtained by our algorithm Ssncg1 and a recently developed algorithm (called PPROJ) in [18]. PPROJ is an extremely fast implementation of an algorithm which utilizes the sparse reconstruction by separable approximation [44] and the dual active set algorithm (DASA). We used the code downloaded from the authors' homepage[2]. Since PPROJ is implemented in C and it depends on some C libraries for linear system solvers, we have to compile PPROJ under the Linux system. Therefore, we compare the performance of Ssncg1 and PPROJ on the high performance computing (HPC[3]) cluster at the National University of Singapore. Due to the memory limit imposed for each user, we are only able to test instances with the matrix dimensions less than 21,000. Default parameter values for PPROJ are used

[2] https://www.math.lsu.edu/~hozhang/Software.html.

[3] https://comcen.nus.edu.sg/services/hpc/about-hpc/.

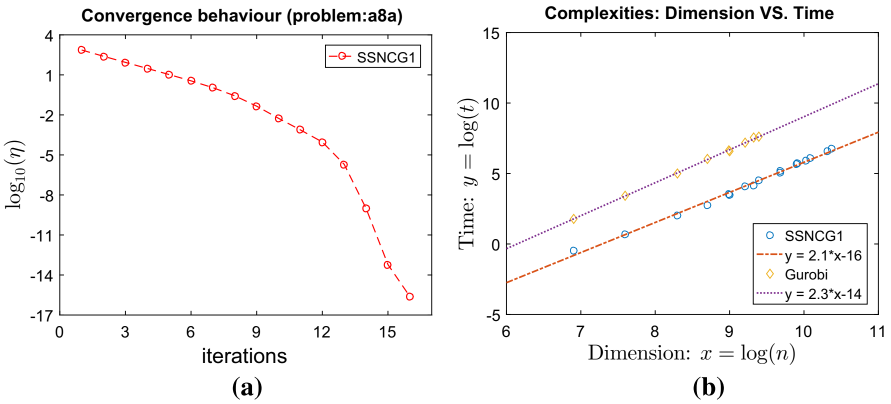

Fig. 1 Performance evaluations of SSNCG1. **a** Convergence behaviour (problem:a8a). **b** Complexities: dimension versus time

Table 2 The performance of SSNCG1 and PPROJ on the projection problems (12) and its dual (15). In the table, “pp” stands for PPROJ; “c2” stands for SSNCG1 with $\varepsilon = 10^{-15}$. The computation time is in the format of “hours:minutes:seconds”

<table>
<tr><th></th><th></th><th>η</th><th>Time</th></tr>
<tr><td>problem</td><td>n</td><td>pp | c2</td><td>pp | c2</td></tr>
<tr><td>rand1</td><td>1000</td><td>**7.5–14** | 5.2–16</td><td>03 | 00</td></tr>
<tr><td>rand2</td><td>2000</td><td>**2.9–12** | 4.4–16</td><td>07 | 02</td></tr>
<tr><td>rand3</td><td>4000</td><td>**1.5–11** | 4.0–16</td><td>32 | 05</td></tr>
<tr><td>rand4</td><td>8000</td><td>**4.0–13** | 4.3–16</td><td>2:32 | 22</td></tr>
<tr><td>rand5</td><td>10000</td><td>**5.8–12** | 4.5–16</td><td>3:45 | 39</td></tr>
<tr><td>rand6</td><td>12000</td><td>**1.6–12** | 4.7–16</td><td>5:31 | 59</td></tr>
<tr><td>rand7</td><td>16000</td><td>**5.8–13** | 5.9–16</td><td>19:34 | 1:20</td></tr>
<tr><td>rand8</td><td>20000</td><td>**4.3–13** | 9.5–16</td><td>34:01 | 2:10</td></tr>
<tr><td>gisette</td><td>6000</td><td>**1.4–14** | 6.5–16</td><td>2:39 | 11</td></tr>
<tr><td>mushrooms</td><td>8124</td><td>**6.9–7** | 2.0–16</td><td>16:22:46 | 21</td></tr>
<tr><td>a6a</td><td>11220</td><td>**3.3–14** | 3.8–16</td><td>14:32 | 39</td></tr>
<tr><td>a7a</td><td>16100</td><td>**3.7–14** | 2.9–16</td><td>43:53 | 1:21</td></tr>
<tr><td>rcv1</td><td>20242</td><td>**1.3–13** | 1.9–16</td><td>2:01:32 | 2:19</td></tr>
</table>

during the experiments. Note that since the stopping criterion of PPROJ is slightly different from ours, we report the accuracy measure η corresponding to the solutions obtained by PPROJ. One can observe that, except the instance **mushrooms**, PPROJ can obtain highly accurate solutions. In fact, we observe from the detailed output file of PPROJ that it does not solve the instance **mushrooms** to the required accuracy while spending excessive amount of time on the DASA in computing Cholesky factorizations. One can also observe from Table 2 that SSNCG1 is much faster than PPROJ, especially for large scale problems. For example, for the instance **rcv1**, SSNCG1 is at least 52 times faster than PPROJ. Therefore, one can safely conclude that SSNCG1 is robust and highly efficient for solving projections problems over the Birkhoff polytope. However, we should emphasize here that PPROJ is a solver aiming at computing the projection onto a general polyhedral convex set and does not necessarily fully exploit the specific structure of the Birkhoff polytope. In fact, it would be an interesting future research topic to investigate whether PPROJ can take advantage of both the sparsity of the constraint matrix B and the second order sparsity of the underlying problem to further accelerate its performance.

5.2 Numerical results for quadratic programming problems arising from relaxations of QAP problems

Given matrices $A, B \in \mathcal{S}^n$, the quadratic assignment problem (QAP) is given by

$$\min\{\langle X,\ AXB\rangle \mid X \in \{0,1\}^{n\times n} \cap \mathfrak{B}_n\},$$

where $\{0,1\}^{n\times n}$ denotes the set of matrices with only 0 or 1 entries. It has been shown in [1] that a reasonably good lower bound for the above QAP can often be obtained by solving the following convex QP problem:

$$\min\{\langle X,\ \mathcal{Q}X\rangle \mid X \in \mathfrak{B}_n\}, \tag{32}$$

where the self-adjoint positive semidefinite linear operator $\mathcal{Q}$ is defined by

$$\mathcal{Q}(X) := AXB - SX - XT, \quad \forall X \in \Re^{n\times n},$$

and $S, T \in \mathcal{S}^n$ are given as follows. Consider the eigenvalue decompositions, $A = V_A D_A V_A^T$, $B = V_B D_B V_B^T$, where V_A and $D_A = \mathrm{diag}(\alpha_1,\ldots,\alpha_n)$ correspond to the eigenvectors and eigenvalues of A, and V_B and $D_B = \mathrm{diag}(\beta_1,\ldots,\beta_n)$ correspond to the eigenvectors and eigenvalues of B, respectively. We assume that $\alpha_1 \geq \ldots \geq \alpha_n$ and $\beta_1 \leq \ldots \leq \beta_n$. Let $(\bar{s},\bar{t})$ be an optimal solution to the LP: $\max\{e^T s + e^T t \mid s_i + t_j \leq \alpha_i\beta_j,\ i,j = 1,\ldots,n\}$, whose solution can be computed analytically as shown in [1]. Then $S = V_A\mathrm{diag}(\bar{s})V_A^T$ and $T = V_B\mathrm{diag}(\bar{t})V_B^T$. In our numerical experiments, the test instances A and B are obtained from the QAP Library [5]. We measure the accuracy of an approximate optimal solution X for problem (32) by using the following relative KKT residual:

$$\eta = \frac{\|X - \Pi_{\mathfrak{B}_n}(X - \mathcal{Q}X)\|}{1 + \|X\| + \|\mathcal{Q}X\|}.$$

Table 3 reports the performance of the ALM designed in Sect. 4 against Gurobi in solving the QP (32). In the fourth and fifth columns of Table 3, “alm (itersub)” denotes the number of outer iterations with itersub in the parenthesis indicating the number of inner iterations of ALM. One can observe from Table 3 that our algorithm is much faster than Gurobi, especially for large scale problems. For example, for the instance **tai150b**, ALM only needs 13 s to reach the desired accuracy while Gurobi needs about two and half hours. One can easily see that Algorithm ALM is highly efficient because each of its subproblems can be solved by the powerful semismooth Newton-CG algorithm SSNCG2 based on the efficient computations of $\Pi_{\mathfrak{B}_n}$ and its

Table 3 The performance of ALM and Gurobi on the quadratic programming problems (32). In the table, “gu” stands for Gurobi; “alm” stands for ALM (accuracy $\eta < 10^{-7}$). The entry “*” indicates out of memory. The computation time is in the format of “hours:minutes:seconds”. “00” in the time column means less than 0.5 s

		Iter	η	Time
problem	n	gu | alm (itersub)	gu | alm	gu | alm
lipa50a	50	11 | 21 (58)	**1.8–6** | 7.3–8	11 | 01
lipa50b	50	11 | 17 (123)	**2.2–6** | 5.0–8	11 | 05
lipa60a	60	11 | 19 (54)	**1.4–6** | 6.7–8	30 | 01
lipa60b	60	11 | 18 (104)	**1.7–6** | 9.9–8	29 | 05
lipa70a	70	11 | 19 (52)	**1.7–6** | 6.0–8	1:17 | 01
lipa70b	70	11 | 19 (103)	**1.4–6** | 6.0–8	1:20 | 06
lipa80a	80	11 | 25 (68)	**1.3–6** | 7.3–8	2:46 | 01
lipa80b	80	12 | 18 (141)	*6.3–7* | 9.3–8	2:52 | 14
lipa90a	90	11 | 20 (54)	**2.7–6** | 8.8–8	5:32 | 01
lipa90b	90	12 | 19 (134)	*5.5–7* | 2.5–8	5:46 | 15
sko100a	100	14 | 26 (95)	**8.5–6** | 8.5–8	2:06 | 11
sko100b	100	14 | 27 (93)	**8.3–6** | 7.9–8	2:06 | 10
sko100c	100	15 | 27 (93)	**4.5–6** | 9.0–8	2:11 | 11
sko100d	100	15 | 26 (91)	**4.8–6** | 8.8–8	2:06 | 10
sko100e	100	14 | 27 (98)	**5.8–6** | 8.5–8	2:06 | 11
sko100f	100	16 | 27 (93)	**6.1–6** | 9.6–8	2:15 | 09
sko64	64	13 | 27 (91)	**7.3–6** | 9.0–8	13 | 04
sko72	72	13 | 26 (86)	**8.1–6** | 7.6–8	22 | 04
sko81	81	14 | 26 (89)	**4.4–6** | 7.6–8	43 | 06
sko90	90	14 | 26 (95)	**4.4–6** | 7.8–8	43 | 08
tai100a	100	11 | 18 (52)	**1.3–6** | 9.5–8	10:31 | 02
tai100b	100	11 | 27 (98)	**1.3–6** | 9.1–8	10:31 | 13
tai50a	50	11 | 20 (55)	**1.1–6** | 6.1–8	09 | 01
tai50b	50	13 | 25 (89)	**5.9–6** | 8.5–8	10 | 03
tai60a	60	10 | 19 (54)	**5.4–6** | 9.6–8	27 | 01
tai60b	60	10 | 28 (102)	**5.4–6** | 6.6–8	27 | 06
tai80a	80	11 | 21 (59)	**1.2–6** | 7.9–8	2:36 | 01
tai80b	80	11 | 27 (98)	**1.2–6** | 8.5–8	2:36 | 07
tai256c	256	$*$ | 2 (4)	$*$ | 2.1–16	$*$ | 00
tai150b	150	19 | 27 (94)	***4.3–7*** | 9.3–8	2:46:17 | 13
tho150	150	16 | 24 (96)	**5.6–6** | 9.9–8	18:52 | 22
wil100	100	13 | 25 (82)	**9.1–6** | 8.8–8	2:14 | 07
wil50	50	13 | 29 (99)	**3.9–6** | 8.4–8	05 | 03
esc128	128	17 | 2 (4)	8.2–11 | 2.2–16	09 | 00

corresponding HS-Jacobian. Note that problem (32) is in fact a quadratic programming with n^2 variables. It is thus not surprising that the interior-point method based solver Gurobi reports out of memory for problem **tai265c** with $n = 256$.

6 Conclusion

In this paper, we study the generalized Jacobians in the sense of Han and Sun [17] of the Euclidean projector over a polyhedral convex set with an emphasis on the Birkhoff polytope. A special element in the set of the generalized Jacobians, referred as the HS-Jacobian, is successfully constructed. Armed with its simple and explicit formula, we are able to provide a highly efficient procedure to compute the HS-Jacobian. To ensure the efficiency of our procedure, a dual inexact semismooth Newton method is designed and implemented to find the projection over the Birkhoff polytope. Numerical comparisons between the state-of-the-art solvers Gurobi and PPROJ have convincingly demonstrated the remarkable efficiency and robustness of our algorithm and implementation. To further demonstrate the importance of the fast computations of the projector and its corresponding HS-Jacobian, we also incorporate them in the augmented Lagrangian method for solving a class of Birkhoff polytope constrained convex QP problems. Extensive numerical experiments on a collection of QP problems arising from the relaxation of quadratic assignment problems show the large benefits of our second order nonsmooth analysis based procedure.

Acknowledgements We would like to thank Professor Jong-Shi Pang at University of Southern California for his helpful comments on an early version of this paper and the referees for helpful suggestions to improve the quality of this paper.

References

1. Anstreicher, K.M., Brixius, N.W.: A new bound for the quadratic assignment problem based on convex quadratic programming. Math. Program. **89**, 341–357 (2001)
2. Beck, A., Teboulle, M.: A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM J. Imaging Sci. **2**, 183–202 (2009)
3. Birkhoff, G.: Three observations on linear algebra. Universidad Nacional de Tucumán, Revista, Serie A **5**, 147–151 (1946)
4. Bonnans, J.F., Shapiro, A.: Perturbation Analysis of Optimization Problems. Springer, New York (2000)
5. Burkard, R.E., Karisch, S.E., Rendl, F.: QAPLIB—a quadratic assignment problem library. J. Glob. Optim. **10**, 391–403 (1997)
6. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines. ACM Trans. Intell. Syst. Technol. **2**, 27:1–27:27 (2011)
7. Chiche, A., Gilbert, JCh.: How the augmented Lagrangian algorithm can deal with an infeasible convex quadratic optimization problem. J. Convex Anal. **23**, 425–459 (2016)
8. Clarke, F.H.: Optimization and Nonsmooth Analysis. Wiley, New York (1983)
9. Cui, Y., Sun, D.F., Toh, K.-C.: On the asymptotic superlinear convergence of the augmented Lagrangian method for semidefinite programming with multiple solutions (2016). arXiv:1610.00875
10. Delbos, F., Gilbert, JCh.: Global linear convergence of an augmented Lagrangian algorithm to solve convex quadratic optimization problems. J. Convex Anal. **12**, 45–69 (2005)
11. Dykstra, R.L.: An algorithm for restricted least squares regression. J. Am. Stat. Assoc. **78**, 837–842 (1983)

12. Fischer, A., Kanzow, C.: On finite termination of an iterative method for linear complementarity problems. Math. Program. **74**, 279–292 (1996)
13. Fogel, F., Jenatton, R., Bach, F., d'Aspremont, A.: Convex relaxations for permutation problems. In: Advances in Neural Information Processing Systems, pp. 1016–1024 (2013)
14. Gabay, D., Mercier, B.: A dual algorithm for the solution of nonlinear variational problems via finite element approximations. Comput. Math. Appl. **2**, 17–40 (1976)
15. Glowinski, R., Marroco, A.: Sur approximation, par elements finis dordre un, et la resolution, par penalisation-dualite, dune classe de problemes de Dirichlet non lineares. Revue Francaise dAutomatique, Informatique et Recherche Operationelle **9**(R–2), 41–76 (1975)
16. Optimization, I. Gurobi: Gurobi Optimizer Reference Manual (2016)
17. Han, J.Y., Sun, D.F.: Newton and quasi-Newton methods for normal maps with polyhedral sets. J. Optim. Theory Appl. **94**, 659–676 (1997)
18. Hager, W.W., Zhang, H.: Projection onto a polyhedron that exploits sparsity. SIAM J. Optim. **26**, 1773–1798 (2016)
19. Higham, N.: Computing the nearest symmetric correlation matrix-a problem from finance. IMA J. Numer. Anal. **22**, 329–343 (2002)
20. Haraux, A.: How to differentiate the projection on a convex set in Hilbert space. Some applications to variational inequalities. J. Math. Soc. Jpn. **29**, 615–631 (1977)
21. Hiriart-Urruty, J.-B., Strodiot, J.-J., Nguyen, V.H.: Generalized Hessian matrix and second-order optimality conditions for problems with $C^{1,1}$ data. Appl. Math. Optim. **11**, 43–56 (1984)
22. Jiang, B., Liu, Y.F., Wen, Z.W.: L_p-norm regularization algorithms for optimization over permutation matrices. SIAM J. Optim. **26**, 2284–2313 (2016)
23. Li, X.D., Sun, D.F., Toh, K.-C.: QSDPNAL: a two-phase augmented Lagrangian method for convex quadratic semidefinite programming. Math. Program. Comput. **10**, 703–743 (2018)
24. Li, X.D., Sun, D.F., Toh, K.-C.: A highly efficient semismooth Newton augmented Lagrangian method for solving Lasso problems. SIAM J. Optim. **28**, 433–458 (2018)
25. Lim, C.H., Wright, S.J.: *Beyond the Birkhoff polytope: convex relaxations for vector permutation problems*. In: Advances in Neural Information Processing Systems, pp. 2168–2176 (2014)
26. Malick, J.: A dual approach to semidefinite least-squares problems. SIAM J. Matrix Anal. Appl. **26**, 272–284 (2004)
27. Luque, F.J.: Asymptotic convergence analysis of the proximal point algorithm. SIAM J. Control Optim. **22**, 277–293 (1984)
28. Mifflin, R.: Semismooth and semiconvex functions in constrained optimization. SIAM J. Control Optim. **15**, 959–972 (1977)
29. Nesterov, Y.: A method of solving a convex programming problem with convergence rate $O(1/k^2)$. Sov. Math. Dokl. **27**, 372–376 (1983)
30. Pang, J.-S.: Newton's method for B-differentiable equations. Math. Oper. Res. **15**, 311–341 (1990)
31. Pang, J.-S., Ralph, D.: Piecewise smoothness, local invertibility, and parametric analysis of normal maps. Math. Oper. Res. **21**, 401–426 (1996)
32. Qi, H., Sun, D.F.: A quadratically convergent Newton method for computing the nearest correlation matrix. SIAM J. Matrix Anal. Appl. **28**, 360–385 (2006)
33. Qi, L., Sun, J.: A nonsmooth version of Newton's method. Math. Program. **58**, 353–367 (1993)
34. Robinson, S.M.: Some continuity properties of polyhedral multifunctions. In: Mathematical Programming at Oberwolfach, vol. 14 of Mathematical Programming Studies, pp. 206–214 . Springer, Berlin Heidelberg (1981)
35. Robinson, S.M.: Implicit B-differentiability in generalized equations. Technical Report #2854, Mathematics Research Center, University of Wisconsin, Madison (1985)
36. Rockafellar, R.T.: Convex Analysis. Princeton University Press, Princeton (1970)
37. Rockafellar, R.T.: Augmented Lagrangians and applications of the proximal point algorithm in convex programming. Math. Oper. Res. **1**, 97–116 (1976)
38. Rockafellar, R.T., Wets, R.J.-B.: Variational Analysis. Springer, New York (1998)
39. Sun, D.F., Han, J.Y., Zhao, Y.: On the finite termination of the damped-Newton algorithm for the linear complementarity problem. Acta Math. Appl. Sin. **21**, 148–154 (1998)
40. Sun, J.: *On Monotropic Piecewise Quadratic Programming*. Ph.D. thesis, Department of Mathematics, University of Washington (1986)
41. Trefethen, L.N., Bau III, D.: Numerical Linear Algebra. SIAM, Philadelphia (1997)

42. Von Neumann, J.: A certain zero-sum two-person game equivalent to an optimal assignment problem. Ann. Math. Stud. **28**, 5–12 (1953)
43. Wang, F., Li, P., Konig, A.C.: *Learning a bi-stochastic data similarity matrix*. In: 2010 IEEE 10th International Conference on Data Mining (ICDM), pp 551–560
44. Wright, S.J., Nowak, R.D., Figueiredo, M.A.T.: Sparse reconstruction by separable approximation. IEEE Trans. Signal Process. **57**, 2479–2493 (2009)
45. Zhao, X., Sun, D.F., Toh, K.-C.: A Newton-CG augmented Lagrangian method for semidefinite programming. SIAM J. Optim. **20**, 1737–1765 (2010)

Affiliations

Xudong Li[1] · Defeng Sun[2] · Kim-Chuan Toh[3]

Xudong Li
lixudong@fudan.edu.cn

Defeng Sun
defeng.sun@polyu.edu.hk

[1] School of Data Science and Shanghai Center for Mathematical Sciences, Fudan University, Shanghai, China

[2] Department of Applied Mathematics, The Hong Kong Polytechnic University, Hung Hom, Hong Kong

[3] Department of Mathematics and Institute of Operations Research and Analytics, National University of Singapore, 10 Lower Kent Ridge Road, Singapore, Singapore