Inexact proximal splitting methods for Euclidean distance matrix optimization

Xin Jiang[1[0000−0003−1231−8529]] and Chaorui Yao[2]

[1] Lehigh University, Bethlehem, PA 18015, USA
xjiang@lehigh.edu, corresponding author
[2] University of California, Los Angeles, Los Angeles, CA 90095, USA
chaorui@ucla.edu

Abstract. This paper presents an inexact first-order proximal splitting method for matrix optimization problems involving Euclidean distance matrices. In the proposed method, the large number of linear (in)equality constraints in the problems are handled efficiently by exploiting problem structure. The presented method also replaces the Euclidean distance matrix projection at each iteration with a low-rank approximation, and the introduced inexactness errors are compensated by an adaptive choice of stepsizes. We test the algorithm on the non-metric multidimensional scaling problem, in which the proposed low-rank approximation helps construct a low-dimensional embedding desirable for practical purposes. Numerical results validate the efficiency and scalability of the algorithm.

Keywords: First-order methods · Proximal algorithms · Euclidean distance matrices · Dimension reduction

1 Introduction

This paper presents a first-order proximal splitting method for a class of large-scale optimization problems involving Euclidean distance matrices (EDMs). The problems often include a large set of linear (in)equality constraints, and have EDMs as the optimization variable. A symmetric $n \times n$ matrix X is a *Euclidean distance matrix* (EDM) if its entries can be expressed as squared pairwise Euclidean distances of a set of points: there exist vectors $y_1, \dots, y_n$ such that

$$X_{ij} = \|y_i - y_j\|^2, \quad i, j = 1, \dots, n. \tag{1}$$

The set of points $\{y_1, \dots, y_n\} \subset \mathbb{R}^d$ is called a *realization* (or *embedding*) of the EDM X. The set of $n \times n$ EDMs forms a convex cone $\mathbb{D}^n$.

Properties and theories of EDMs have been extensively studied in linear algebra; see, *e.g.*, [14,27]. Optimization over the EDM cone also has a wide variety of applications in graph theory [2], bioinformatics [4], signal processing [5,15,34], machine learning [1,10,29,47,49], *etc.* In particular, many dimension reduction techniques are developed to build a low-dimensional embedding that satisfies certain application-specific constraints [1,48]. In these applications, the number of linear (in)equality constraints scales very quickly with the matrix size. Thus, even when the matrix size is tractable, the huge number of linear constraints might prevent the use of general-purpose solvers. Moreover, classical ML approaches often use the embedding as the optimization variable and formulate the problem as a semidefinite program (SDP). But it might be more efficient to handle the EDM structure directly. In this paper, we use the EDM matrix as the variable, and reformulate the ML model as an EDM optimization problem. We show that such an EDM reformulation facilitates the use of proximal splitting methods. These algorithms iteratively decompose the EDM optimization problem into simpler problems, and separately handle the EDM conic constraint and other application-specific constraints. At each iteration, the conic constraint requires projection onto the (reduced) EDM cone, while the linear constraints can be handled efficiently by exploiting specific problem structure. Therefore, proximal methods can address the scalability concern mentioned in [1, 48], and are able to solve large-scale EDM optimization problems that cannot be handled by general-purpose solvers.

Moreover, in these problems, a solution corresponding to a low-dimensional embedding is of great interest. Thus, a regularization term is often added to promote low rank [1, 29, 50]. In comparison, for the EDM reformulation, it is tempting to replace the EDM projection at each iteration of the proximal method with a low-rank approximation. The modified algorithm then generates a sequence of low-rank matrices, and retrieves a low-dimensional realization at optimum. On the other hand, however, such modification introduces inexactness errors and in our case, the errors are not necessarily summable. Thus, classical analyses of inexact proximal algorithms cannot provide convergence guarantees, and new techniques and analyses are needed for the modified (inexact) proximal method.

Contributions. In this work, we study an inexact primal–dual proximal splitting method. In the presented algorithm, the inexactness errors are not necessarily summable (as required in most previous work on inexact optimization methods) and are compensated by an adaptive choice of stepsizes at each iteration. The proposed algorithm is applicable to a wide variety of convex optimization problems (not limited to EDM optimization), and is of independent interest as a first-order proximal method.

As an example, we apply the proposed method to non-metric multidimensional scaling (NMDS), a classical ML model for dimension reduction. Our method handles the large number of linear constraints efficiently by exploiting problem structure. Moreover, motivated by the need of a low-rank EDM solution, our method replaces the EDM projection at each iteration with a low-rank approximation. Although such modification introduces inexactness errors, convergence is still guaranteed by our analysis. Numerical experiments are conducted to verify the efficiency and scalability of the proposed algorithm.

Outline. The rest of the paper is organized as follows. The proposed algorithm, iPDHG, is presented and analyzed in Section 2. In Section 3, we apply the proposed method to NMDS, and explain how to further improve the efficiency of iPDHG by exploiting data structure. Section 4 contains numerical results.

2 Inexact preconditioned PDHG with line search

This section presents the inexact preconditioned primal–dual hybrid-gradient method with line search (short: iPDHG). iPDHG solves a general form of nonsmooth, convex problems, and is of independent interest as a proximal method.

2.1 Inexact proximal operators

The optimization problem studied in this paper has the canonical form

$$\text{minimize} \quad f(x) + g(Ax), \tag{2}$$

where f, g are closed convex functions and potentially nonsmooth. This general problem is a useful formulation for many important structures arising from various large-scale applications in machine learning, signal and image processing [8,12,18,25,35]. In the general problem (2), the structure of the nonsmooth function f (and g) is often exploited via its *(preconditioned) proximal operator*:

$$\mathbf{prox}_f^P(y,a) = \underset{x}{\operatorname{argmin}} \left(f(x) + \langle a, x\rangle + \tfrac{1}{2}\|x-y\|_P^2 \right), \tag{3}$$

where $P \succ 0$. When $P = I$, it reduces to the classical proximal operator [32], and the preconditioner P is often added to better capture the structure of f.

For many convex functions, the computation of their proximal operators is expensive and often involves solving a nontrivial convex subproblem. This motivates the use of inexact proximal operators [3,16,30,37,38,42]. In this paper, we define the inexact proximal operator $\hat{x} \approx_\epsilon \mathbf{prox}_f^P(y,a)$ to satisfy [16,30,37]

$$f(u) \geq f(\hat{x}) + \langle u - \hat{x}, P(y-\hat{x}) + a\rangle - \epsilon \ \text{ for all } u \in \mathbf{dom}\, f, \tag{4}$$

where $\epsilon > 0$ is a small tolerance for inexactness. When $\epsilon = 0$, we recover (3): $\hat{x} = \mathbf{prox}_f^P(y,a)$. To incorporate inexact prox-operators in proximal methods, additional conditions are often needed to bound the error ϵ.

2.2 Algorithm description

Now we present the iPDHG algorithm. Select starting points $x_{-1} = x_0 \in \mathbf{dom}\, f$ and $z_0 \in \mathbf{dom}\, g^*$. For $k = 0, 1, \ldots$, repeat:

$$\bar{x}_{k+1} = x_k + \theta_k(x_k - x_{k-1}) \tag{5a}$$

$$z_{k+1} \approx_{\epsilon_k} \mathbf{prox}_{\sigma_k g^*}^Q(z_k, -A\bar{x}_{k+1}) \tag{5b}$$

$$x_{k+1} = \mathbf{prox}_{\tau_k f}^P(x_k, \tau A^T z_k), \tag{5c}$$

where $g^*(z) = \sup_y (\langle y, z\rangle - g(y))$ is the convex conjugate of g^*, and the error ϵ_k in z-update must satisfy

$$\epsilon_k \leq \tfrac{\eta}{2\sigma_k}\|z_{k+1} - z_k\|_Q^2. \tag{6}$$

The parameters τ_k, σ_k, θ_k are determined by a backtracking line search. We select the initial parameters $\bar{\theta}_{-1} = 1$, $\tau_{-1} > 0$, $\sigma_{-1} > 0$, and tolerance $0 < \eta <$

$\delta \leq 1$. To start line search at iteration k, we choose $\bar{\theta}_k \in [1, \sqrt{1+\theta_{k-1}}]$. For $i = 0, 1, 2, \ldots$, we set $\theta_k = 2^{-i}\bar{\theta}_k$, $\tau_k = \theta_k \tau_{k-1}$, $\sigma_k = \theta_k \tau_{k-1}$, and compute $\bar{x}_{k+1}$, z_{k+1}, x_{k+1} using (5). If

$$\langle z_{k+1} - z_k, A(\bar{x}_{k+1} - x_{k+1})\rangle \leq \tfrac{1}{2\tau_k}\|\bar{x}_{k+1} - x_k\|_P^2 + \tfrac{\delta-\eta}{2\sigma_k}\|z_{k+1} - z_k\|_Q^2, \tag{7}$$

we accept the computed iterates $\bar{x}_{k+1}$, x_{k+1}, z_{k+1} and stepsizes τ_k, σ_k, and terminate the line search. If (7) does not hold, we increment i and continue the line search. In practice, δ is chosen as one, and $\delta < 1$ is only needed for some convergence analysis (see Theorem 1). The constant η is used to control the error bound on ϵ_k so that the inexactness error can be compensated by a smaller stepsize chosen in (7). The convergence analysis can be easily extended to handle the case where η varies in different iterations (as η_k).

The iteration (5) with $\epsilon_k = 0$ is called the preconditioned primal–dual hybrid-gradient (PDHG) method, and has been widely studied in the literature [7,9,11]. The error bound condition (6) is new and different from the conditions used in most inexact proximal methods. In particular, it does not require the errors are summable (*i.e.*, $\sum_k \epsilon_k < \infty$). Instead, the error is required to be bounded at each iteration, and convergence is guaranteed via an adaptive choice of stepsizes. The condition (6) is similar to the one used in [30], but iPDHG has substantial difference from the algorithm proposed in [30]. In [30], the inexactness error is due to an early-stopped subproblem solver while for problems in Section 3, an inexact prox-evaluation is motivated by the need for a low-rank solution. In addition, the paper [30] only studies the case $P = I$ and $Q = AA^T$, and their analysis requires f to be strongly convex. In comparison, our convergence results hold for any preconditioners and for general convex problems (2).

2.3 Convergence analysis

We now present the analysis of iPDHG, of which the key idea follows [23,24,31]. We start with a lemma that guarantees the line search exits at each iteration.

Lemma 1. *The stepsizes τ_k, σ_k generated by* iPDHG *are bounded below:*

$$\tau_k \geq \tau_{\min} = \min\left\{\tau_{-1}, \tfrac{\sqrt{\delta-\eta}}{2\sqrt{\beta}\|A\|}\right\}, \qquad \sigma_k \geq \beta\tau_{\min}, \tag{8}$$

for all $k \in \mathbb{N}_{\geq 1}$, where $\beta := \sigma_{-1}/\tau_{-1}$, and $\|A\| := \sup_{u\neq 0} \|Au\|_{Q^{-1}}/\|u\|_P$.

Proof. First note that the exit condition (7) holds if $\sqrt{\sigma_k\tau_k}\|A\| \leq \sqrt{\delta-\eta}$:

$$\begin{aligned}\langle z_{k+1} - z_k, A(\bar{x}_{k+1} - x_{k+1})\rangle &\leq \|A\|\|z_{k+1} - z_k\|_Q\|\bar{x}_{k+1} - x_{k+1}\|_P \\ &\leq \tfrac{\sqrt{\sigma_k\tau_k}\|A\|}{\delta-\eta}\Big(\tfrac{\|\bar{x}_{k+1}-x_{k+1}\|_P^2}{\tau_k}\tfrac{(\delta-\eta)\|z_{k+1}-z_k\|_Q^2}{\sigma_k}\Big)^{1/2} \\ &\leq \tfrac{1}{2\tau_k}\|\bar{x}_{k+1} - x_{k+1}\|_P^2 + \tfrac{(\delta-\eta)}{2\sigma_k}\|z_{k+1} - z_k\|_Q^2.\end{aligned}$$

Based on this observation, we use induction to establish the lower bounds (8). Suppose $\tau_{k-1} \geq \tau_{\min}$ and $\sigma_{k-1} \geq \sigma_{\min}$, which holds at $k = 0$. The first value of θ_k tested in the line search is $\theta_k = \bar{\theta}_k \geq 1$. If this value is accepted, then

$$\tau_k = \bar{\theta}_k\tau_{k-1} \geq \tau_{k-1} \geq \tau_{\min}, \qquad \sigma_k = \bar{\theta}_k\sigma_{k-1} \geq \sigma_{k-1} \geq \sigma_{\min}.$$

If $\theta = \bar{\theta}_k$ is rejected, one or more backtracking steps are taken. Denote by $\tilde{\theta}$ the last rejected value. Then, $\tilde{\theta}\sqrt{\sigma_{k-1}\tau_{k-1}}\|A\| > \sqrt{\delta - \eta}$, and the accepted θ_k satisfies

$$\theta_k = \frac{\tilde{\theta}}{2} > \frac{\sqrt{\delta-\eta}}{2\sqrt{\sigma_k\tau_k}\|A\|} = \frac{\sqrt{\delta-\eta}}{2\tau_k\sqrt{\beta}\|A\|}.$$

Thereby, it holds that $\tau_k = \theta_k\tau_{k-1} > \frac{\sqrt{\delta-\eta}}{2\sqrt{\beta}\|A\|} \geq \tau_{\min}$, and $\sigma_k = \beta\tau_k \geq \beta\tau_{\min}$, which completes the proof.

Theorem 1. *If $\eta \in (0,1)$, the iterates $\{(x_k, z_k)\}$ generated by* iPDHG *converge to a pair of primal–dual optimal solutions $(x^\star, z^\star)$.*

Proof. The optimality conditions for the updates (5b) and (5c) imply that

$$g^*(z_{k+1}) - g^*(z) \leq \tfrac{1}{\sigma_k}\langle z - z_{k+1}, z_{k+1} - z_k\rangle_Q - \langle z - z_{k+1}, A\bar{x}_{k+1}\rangle + \epsilon_k \tag{9}$$
$$f(x_{k+1}) - f(x) \leq \tfrac{1}{\tau_k}\langle x_{k+1} - x_k, x - x_{k+1}\rangle_P + \langle z_{k+1}, A(x - x_{k+1})\rangle \tag{10}$$

for all $(x,z) \in \mathbf{dom}\, f \times \mathbf{dom}\, g^*$. Then, at previous iteration, it holds that

$$\begin{aligned} f(x_k) - f(x) &\leq \tfrac{1}{\tau_{k-1}}\langle x_k - x_{k-1}, x - x_k\rangle_P + \langle z_k, A(x - x_k)\rangle \\ &= \tfrac{\theta_k}{\tau_k}\langle x_k - x_{k-1}, x - x_k\rangle_P + \langle z_k, A(x - x_k)\rangle \end{aligned}$$

for all $x \in \mathbf{dom}\, f$. We evaluate this inequality at $x := x_{k+1}$ and add it to the inequality at $x := x_{k-1}$ multiplied by θ_k:

$$\begin{aligned} &(1+\theta_k)f(x_k) - \theta_k f(x_{k-1}) - f(x_{k+1}) \\ &\leq \tfrac{\theta_k}{\tau_k}\langle x_k - x_{k-1}, x_{k+1} - \bar{x}_{k+1}\rangle_P + \langle z_k, A(x_{k+1} - \bar{x}_{k+1})\rangle \\ &= \tfrac{1}{\tau_k}\langle \bar{x}_{k+1} - x_k, x_{k+1} - \bar{x}_{k+1}\rangle + \langle z_k, A(x_{k+1} - \bar{x}_{k+1})\rangle \\ &= \tfrac{1}{2\tau_k}\big(\|x_{k+1} - x_k\|_P^2 - \|\bar{x}_{k+1} - x_k\|_P^2 - \|\bar{x}_{k+1} - x_{k+1}\|_P^2\big) \\ &\quad + \langle z_k, A(x_{k+1} - \bar{x}_{k+1})\rangle, \end{aligned} \tag{11}$$

where in the second step we use $\bar{x}_{k+1} = x_k + \theta_k(x_k - x_{k-1})$.

For notation simplicity, we define

$$F(x) = f(x) - f(x^\star) - \langle z^\star, A(x - x^\star)\rangle, \quad G(z) = g^*(z) - g^*(z^\star) - \langle z - z^\star, Ax^\star\rangle.$$

Both functions are nonnegative for all $(x,z) \in \mathbf{dom}\, f \times \mathbf{dom}\, g^*$, because $(x^\star, z^\star)$ is a pair of primal–dual optimal solutions. Then, adding (9) with $z := z^\star$, (10) with $x := x^\star$, and (11) gives

$$\begin{aligned} &\tau_k\big((1+\theta_k)F(x_k) - \theta_k F(x_{k-1}) + G(z_{k+1})\big) \\ &= (1+\theta_k)f(x_k) - \theta_k f(x_{k-1}) - f(x^\star) + g^*(z_{k+1}) - g^*(z^\star) \\ &\leq \tfrac{1}{2}\big(\|x^\star - x_k\|_P^2 - \|\bar{x}_{k+1} - x_k\|_P^2 - \|x^\star - x_{k+1}\|_P^2 - \|\bar{x}_{k+1} - x_{k+1}\|_P^2\big) \\ &\quad + \tfrac{1}{2\beta}\big(\|z^\star - z_k\|_Q^2 - \|z^\star - z_{k+1}\|_Q^2 - \|z_{k+1} - z_k\|_Q^2\big) \\ &\quad + \tau_k\langle z_{k+1} - z_k, A(\bar{x}_{k+1} - x_{k+1})\rangle + \epsilon_k. \end{aligned} \tag{12}$$

The last two terms on the right-hand side can be further bounded using the exit condition in line search (7) and the error bound (6):

$$\tau_k\langle z_{k+1}-z_k, A(\overline{x}_{k+1}-x_{k+1})\rangle+\epsilon_k \le \tfrac{1}{2}\|\overline{x}_{k+1}-x_{k+1}\|_P^2+\tfrac{\delta}{2\beta}\|z_{k+1}-z_k\|_Q^2.$$

Substituting this upper bound into (12) and telescoping:

$$\begin{aligned}
&\tau_k(1+\theta_k)F(x_k)+\tfrac{1}{2}\|x^\star-x_{k+1}\|_P^2+\tfrac{1}{2\beta}\|z^\star-z_{k+1}\|_Q^2\\
&\le \tau_{k-1}(1+\theta_{k-1})F(x_{k-1})+\tfrac{1}{2}\|x^\star-x_k\|_P^2+\tfrac{1}{2\beta}\|z^\star-z_{k+1}\|_Q^2\\
&\quad -\tfrac{1}{2}\|\overline{x}_{k+1}-x_{k+1}\|_P^2-\tfrac{1-\delta}{2\beta}\|z_{k+1}-z_k\|_Q^2\\
&\le \tau_{-1}F(x_{-1})+\tfrac{1}{2}\|x^\star-x_0\|_P^2+\tfrac{1}{2\beta}\|z^\star-z_0\|_Q^2\\
&\quad -\tfrac{1}{2}\sum_{j=0}^{k}\left(\|\overline{x}_{j+1}-x_{j+1}\|_P^2+\tfrac{1-\delta}{\beta}\|z_{j+1}-z_j\|_Q^2\right).
\end{aligned} \tag{13}$$

This shows that the sequences $\{x_k\}$ and $\{z_k\}$ are bounded, and

$$\tfrac{1}{\tau_{k-1}}\|x_k-x_{k-1}\|_P=\tfrac{1}{\tau_k}\|x_k-\overline{x}_k\|_P\to 0, \qquad \|z_{k+1}-z_k\|_Q\to 0. \tag{14}$$

Let $(\hat{x},\hat{z})$ be a limit point of a converging subsequence (x_{k_i}, z_{k_i}). Then, the proof for the uniqueness of $(\hat{x},\hat{z})$ follows from standard analysis for PDHG [23, §3.3.4], and is omitted here due to space limit.

3 Application to EDM optimization

In this section, we apply iPDHG to EDM optimization problems. We take the generalized non-metric multidimensional scaling (NMDS) problem as an example here, and many other potential applications of the proposed iPDHG exist in machine learning [10, 43, 47–49], graph signal processing [34], *etc.*

Section 3.1 gives a brief review of EDMs. For more properties of EDMs, we refer interested readers to recent surveys [15, 27] and the book [14]. Then in Section 3.2, we reformulate NMDS as an EDM problem and then apply iPDHG. We explain in detail how to further improve efficiency by exploiting data structure.

3.1 Euclidean distance matrices

The definition (1) of an EDM can be rewritten in matrix notation as $X = \mathbf{diag}(Y)\mathbf{1}^T+\mathbf{1}\,\mathbf{diag}(Y)^T-2Y$, where $Y=\begin{bmatrix}y_1\ y_2\ \cdots\ y_n\end{bmatrix}^T\begin{bmatrix}y_1\ y_2\ \cdots\ y_n\end{bmatrix}$ is positive semidefinite (PSD) (*i.e.*, $Y\succeq 0$). Therefore, X is an EDM if and only if there exists a real symmetric matrix $Y\in\mathbb{S}^n$ such that

$$Y\succeq 0, \qquad \mathbf{diag}(Y)\mathbf{1}^T+\mathbf{1}\,\mathbf{diag}(Y)^T-2Y=X, \tag{15}$$

where $\mathbf{diag}(Y)$ returns the diagonal entries of Y as a column vector. We refer to the matrix Y that satisfies (15) as a *Gram matrix* of X.

An equivalent representation of EDMs is due to Schoenberg [39, 40]: X is an EDM if and only if $\mathbf{diag}(X)=0$ and $c^TXc\le 0$ for all c with $\mathbf{1}^Tc=0$. Schoenberg's conditions can be rewritten as

$$\mathbf{diag}(X)=0, \qquad V^TXV\preceq 0, \tag{16}$$

where V is any matrix whose columns span the orthogonal complement of $\mathbf{1}$, *i.e.*, range$(V) = \{c \in \mathbb{R}^n \mid \mathbf{1}^T c = 0\}$. A common choice of V is $V = I - (1/n)\mathbf{1}\mathbf{1}^T$.

We denote by $\mathbb{D}_0^n$ the *reduced* EDM cone: $\mathbb{D}_0^n = \{X \in \mathbb{S}^n \mid V^T X V \preceq 0\}$. Then, the EDM cone can be written as $\mathbb{D}^n = \{X \in \mathbb{D}_0^n \mid \mathbf{diag}(X) = 0\}$. Projection onto the reduced EDM cone $\mathbb{D}_0^n$ is much easier than projection onto $\mathbb{D}^n$, and the computational complexity is dominated by an eigendecomposition [17, 19, 21, 45].

In this section, it would be more convenient to represent an matrix variable by a vector variable. Given $X \in \mathbb{S}^n$, we define

$$\mathbf{vec}(X) = \big(X_{11}, \ldots, X_{nn}, \sqrt{2}X_{21}, \ldots, \sqrt{2}X_{n1}, \sqrt{2}X_{32}, \ldots, \sqrt{2}X_{n,n-1}\big).$$

The function $\mathbf{vec}$ maps $X \in \mathbb{S}^n$ to a vector of length $p = \frac{n(n+1)}{2}$ that contains the lower triangular entries of X. Note that the strictly lower triangular entries of X are scaled by $\sqrt{2}$. The scaling ensures that $\mathbf{tr}(XY) = \langle \mathbf{vec}(X), \mathbf{vec}(Y)\rangle$.

3.2 Generalized non-metric multidimensional scaling

Non-metric multidimensional scaling (NMDS) is a classical problem in statistics and machine learning [13,28,41], and it aims to find a low-dimensional realization such that the Euclidean distances between the realization points satisfy a predetermined ordering. In this section, we consider the following generalization of NMDS [1], which is formulated as a semidefinite program (SDP):

$$\begin{array}{ll} \text{minimize} & \sum_{(i,j,s,t)\in\Omega} w_{ijst} + \lambda\, \mathbf{tr}(Y) \\ \text{subject to} & Y_{ss} - 2Y_{st} + Y_{tt} - Y_{ii} + 2Y_{ij} - Y_{jj} \geq w_{ijst},\ (i,j,s,t) \in \Omega \\ & w_{ijst} \leq 1, \quad (i,j,s,t) \in \Omega \\ & \mathbf{1}^T Y \mathbf{1} = 0, \quad Y \succeq 0 \end{array} \tag{17}$$

with variables $Y \in \mathbb{S}^n$ and $w \in \mathbb{R}^m$ (with $m = |\Omega|$). The variable Y is the Gram matrix of the realization $\{y_1, \ldots, y_n\} \subset \mathbb{R}^d$ (with $n \gg d$). The second term in the objective, $\mathbf{tr}\, Y = \sum_i \lambda_i(Y)$, is the sum of eigenvalues and serves as a convex regularization promoting low rank (or a convex relaxation for the low-rank constraint) [1]. The first two sets of inequalities in (17) describe the *soft* ordering constraints, and very often in applications, one has $m = O(n^2)$. Note that each w_{ijst} is constrained by only two upper bounds, so at the optimum,

$$w_{ijst} = \min\{1, Y_{ss} - 2Y_{st} + Y_{tt} - Y_{ii} + 2Y_{ij} - Y_{jj}\}, \quad \text{for } (i,j,s,t) \in \Omega.$$

Since the variable Y in (17) is the Gram matrix in an EDM parametrization, the problem is more easily written in terms of the corresponding EDM X with elements $X_{ij} = Y_{ii} - 2Y_{ij} + Y_{jj}$. Using Schoenberg's parametrization (16) and the vector notation $x = \mathbf{vec}(X)$, we can write (17) equivalently as

$$\begin{array}{ll} \text{minimize} & \sum_{k=1}^m \max\{0, 1 + (Bx)_k\} + \lambda \mathbf{1}^T x \\ \text{subject to} & Cx = 0, \quad x \in \mathbf{vec}(\mathbb{D}_0^n), \end{array} \tag{18}$$

where the optimization variable is $x \in \mathbb{R}^p$ with $p = \frac{n(n+1)}{2}$. The data matrix B has size $m \times p$, and each row of B represents a tuple $(i,j,s,t) \in \Omega$. If the kth row represents (i,j,s,t), then $(Bx)_k = X_{ij} - X_{st}$ for $x = \mathbf{vec}(X)$. For simplicity, a factor $1/(\sqrt{2}n)$ was absorbed in the parameter λ in (18).

To apply iPDHG, we formulate problem (18) in the form of (2) with

$$\begin{aligned} f(x) &= \delta_{\mathcal{C}}(x), \qquad g(u,v) = g_1(u) + \lambda \mathbf{1}^T v + \delta_{\mathbf{vec}(\mathbb{D}_0^n)}(v), \\ \mathcal{C} &= \{x \mid Cx = 0\}, \quad g_1(u) = \sum_{k=1}^{m} \max\{0, 1 + u_k\}, \quad A = \begin{bmatrix} B \\ I \end{bmatrix}. \end{aligned} \tag{19}$$

The basic PrePDHG iterations ((5) with $\epsilon_k = 0$) are

$$\bar{x}_{k+1} = x_k + \theta(x_k - x_{k-1}) \tag{20a}$$

$$z_{k+1}^{(1)} = \mathbf{prox}_{\sigma g_1^*}(z_k^{(1)} + \sigma B \bar{x}_{k+1}) \tag{20b}$$

$$z_{k+1}^{(2)} = z_k^{(2)} + \sigma \bar{x}_{k+1} - \mathit{\Pi}_{\mathbf{vec}(\mathbb{D}_0^n)}\big(z_k^{(2)} + \sigma \bar{x}_{k+1} - \tfrac{\lambda}{\sigma}\mathbf{1}\big) \tag{20c}$$

$$x_{k+1} = \mathbf{prox}_{\tau f}^{P}\left(x_k, \tau(B^T z_{k+1}^{(1)} + z_{k+1}^{(2)})\right). \tag{20d}$$

The $z^{(1)}$-update is separable and has a closed-form formula, and the $z^{(2)}$-update involves projection onto $\mathbf{vec}(\mathbb{D}_0^n)$ (dominated by an eigen-decomposition). Both updates are simple, and preconditioning does not help improve efficiency.

Preconditioning. We discuss one preconditioner for x-update: $P = B^T B + \gamma I$. This preconditioner has been widely used in applications arising from image processing and inverse problems [22, 30, 36]. The positive constant γ is added to guarantee the positive-definiteness of P and to improve numerical stability. With this choice of P, the x-update involves solving the subproblem

$$\text{minimize} \quad \delta_{\mathcal{C}}(x) + \tfrac{1}{2}\|Bx - Bx_k\|^2 + \tfrac{\gamma}{2\tau}\|x - x_k + \tfrac{\tau}{\sqrt{\gamma}}(B^T z_k^{(1)} + z_k^{(2)})\|^2$$

with variable $x \in \mathbb{R}^p$. Note that the vector $(Bx)_k = X_{ij} - X_{st}$ does not contain the first n entries of x. Thus, the first n entries of x_{k+1} are zero, and the other entries equal to the last $\frac{n(n-1)}{2}$ entries of the vector

$$\tilde{x} = x_k - \tfrac{\tau}{\sqrt{\gamma}}(B^T z_k^{(1)} + z_k^{(2)}) + \tfrac{\tau}{\sqrt{\gamma}} B^T (BB^T + \gamma I)^{-1} B (B^T z_k^{(1)} + z_k^{(2)}).$$

The matrix $BB^T + \gamma I$ does not change during optimization. So we only need to factorize the matrix once and reuse the factor in every x-update. This matrix is also very sparse (sparser than $B^T B$) as each row of B has only two entries.

In comparison, when $P = I$, the x-update reduces to a projection on $\mathcal{C}$; *i.e.*, setting the first n elements to zero. Thus, preconditioning slightly increases the computation complexity, and leads to an improvement in the numerical performance. In particular, we observe in experiments (see Section 4) that with preconditioning, the total number of PDHG iterations dramatically decreases.

Low-rank approximation as inexact prox-operator. It is well known that if an EDM $X \in \mathbb{D}^n$ has a realization in $\mathbb{R}^d$ $(d < n)$, then the rank of X is at most $d + 2$ [20, Thm. 5]. Thus, if problem (17) has a d-dimensional realization at optimum, then the EDM problem (18) has a solution of rank at most $d+2$. This motivates us to replace the projection to $\mathbb{D}_0^n$ involved in (20c) with a rank-$(d+2)$ PSD approximation: we only keep the leading $(d+2)$ eigenvalues and remove the rest. In this case, the advantage of a low-rank PSD approximation is threefold. First, this modification decreases the computational complexity in $z^{(2)}$-update. Also, the low-rank regularization term $\mathbf{tr}\, Y$ is no longer needed, so we avoid tuning λ, which is difficult but critical in practice [1]. The third benefit is on practical performance. It is observed that the use of low-rank approximation often helps to generate an embedding that is "visually clearer"; see results in Section 4. This is because iPDHG is more likely to obtain a low-rank EDM solution.

On the other hand, however, low-rank approximation introduces inexactness error. This error is actually computable, and the complexity is dominated by an eigen-decomposition. In practice, however, it is not worth the effort. In the implementation, we fix η in advance, and check the condition (6) at each iteration. We only take an inexact update if the condition is satisfied; otherwise we perform an *exact* $z^{(2)}$-update at this iteration. In all the numerical experiments in Section 4, it is observed that the condition (6) is always satisfied, and iPDHG generates a sequence of low-rank iterates converging to an EDM solution.

4 Numerical experiments

We evaluate the performance of iPDHG applied to NMDS (18). The numerical results verify that the proposed method efficiently solves large-scale EDM optimization problems that cannot be handled by general-purpose solvers (*e.g.*, SeDuMi [44], SDPT3 [46]). In addition, it is observed that iPDHG generates virtually better realizations with inexact (low-rank) updates than without.

The experiments are carried out in MATLAB 2021b on a desktop with an Intel Core i5 2.4GHz CPU and 16GB RAM. We compare iPDHG with the SCS solver [33], which is built on the alternating direction method of multipliers (ADMM) [6] and is able to solve large-scale quadratic cone programs. Four datasets are used in the experiments: Scurve, Cifar-10, MNIST, and EMNIST. Scurve is a small dataset suggested in [26], and is used here for visualization of the realizations. In our experiments we only take part of each dataset. (Recall that the most expensive step in iPDHG is the eigen-decomposition, so for iPDHG, the scale of solvable problems is limited by the size of matrices for which an eigen-decomposition can be performed on a desktop.) For all the experiments, we set the initial embedding dimension $d=2$, the control parameter for inexactness $\eta=0.5$, and the parameter in the preconditioner $\gamma=0.1$.

Table 1 reports the number of iterations and CPU time for SCS and all variants of iPDHG. Our method (iPDHG) outperforms SCS in terms of both the number of iterations and runtime. It is also shown that preconditioning significantly reduces the number of iterations, and thus the computation time. Moreover, the use of inexact updates does not affect the number of iterations much. Recall that only a small portion of eigenvalues is needed in inexact updates, so for larger datasets, iPDHG converges faster than its exact variant. We also note that

data	n	m	algo	precond.	inexact	number of iterations	CPU time (in sec.)
Cifar-10	3000	$4,498,499$	iPDHG	Y	Y	$1,026$	9.43×10^3
			iPDHG	Y	N	974	9.40×10^3
			iPDHG	N	Y	$18,625$	2.82×10^5
			SCS	-	-	$1,328$	1.64×10^4
MNIST	5000	$12,497,499$	iPDHG	Y	Y	$1,563$	5.45×10^4
			iPDHG	Y	N	$1,494$	6.03×10^4
			iPDHG	N	Y	$30,054$	1.62×10^6
			SCS	-	-	$1,998$	9.28×10^4
EMNIST	8000	$31,995,999$	iPDHG	Y	Y	$1,154$	2.27×10^5
			iPDHG	Y	N	$1,026$	3.64×10^5
			iPDHG	N	Y	$20,245$	7.26×10^6
			SCS	-	-	-	OOM

Table 1. Four algorithms are tested for NMDS on three datasets. n indicates matrix size and m indicates the number of linear constraints. The 5th and 6th columns indicate whether preconditioning and inexact updates are used. OOM indicates out of memory.

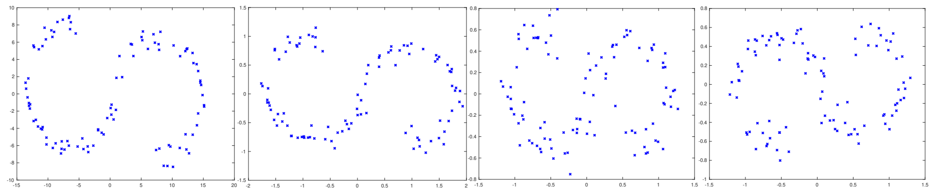

Fig. 1. Plot of Scurve realizations generated by iPDHG, iPDHG without preconditioning, exact PrePDHG ((5) with $\epsilon_k = 0$), and SCS, respectively.

SCS fails for EMNIST dataset ($n = 8000$) while iPDHG is able to solve it efficiently. This could be because SCS needs to generate a large number of auxiliary variables to represent the EDM constraint, so the solver runs out of memory.

For the illustrative dataset Scurve, we plot the constructed realizations from iPDHG, PrePDHG and SCS in Fig. 1. We see that algorithms with inexact (low-rank) updates generate a clearer and less noisy "S" curve compared with algorithms with exact prox-evaluations.

5 Conclusion

We present a primal–dual proximal algorithm (iPDHG) that allows for inexact prox-evaluations, and apply it to Euclidean distance matrix (EDM) optimization problems. The inexact prox-evaluation is motivated by the existence of a low-rank EDM solution, and backtracking line search is used to compensate the inexactness errors and to guarantee convergence. The proposed method is then applied to the EDM reformulation of non-metric multidimensional scaling. Data structure is exploited to further improve the efficiency and scalability of iPDHG, and numerical experiments are conducted to verify its superiority. Future work is needed to address the scalability of eigen-decomposition used in the algorithm.

Acknowledgement. The authors would like to thank Lieven Vandenberghe for his constructive feedback and helpful discussions during the preparation of this manuscript.

References

1. Agarwal, S., Wills, J., Cayton, L., Lanckriet, G., Kriegman, D., Belongie, S.: Generalized non-metric multidimensional scaling. In: Meila, M., Shen, X. (eds.) Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 2, pp. 11–18. PMLR (2007)
2. Alfakih, A.Y., Khandani, A., Wolkowicz, H.: Solving Euclidean distance matrix completion problems via semidefinite programming. Computational Optimization and Applications **12**(1), 13–30 (1999)
3. Bauschke, H.H., Combettes, P.: Convex Analysis and Monotone Operator Theory in Hilbert Spaces. Springer Publishing Company, Incorporated, 1st edn. (2011)
4. Biswas, P., Toh, K.C., Ye, Y.: A distributed SDP approach for large-scale noisy anchor-free graph realization with applications to molecular conformation. SIAM Journal on Scientific Computing **30**(3), 1251–1277 (2008)
5. Biswas, P., Ye, Y.: Semidefinite programming for Ad-Hoc wireless sensor network localization. In: Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks. pp. 46–54 (2004)
6. Boyd, S.P., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends in Machine Learning **3**(1), 1–122 (2011)
7. Chambolle, A., Pock, T.: A first-order primal–dual algorithm for convex problems with applications to imaging. Journal of Mathematical Imaging and Vision **40**(1), 120–145 (2011)
8. Chambolle, A., Pock, T.: An introduction to continuous optimization for imaging. Acta Numerica **25**, 161–319 (2016)
9. Chambolle, A., Pock, T.: On the ergodic convergence rates of a first-order primal–dual algorithm. Mathematical Programming **159**(1), 253–287 (2016)
10. Cheng, Z., Zhang, J., Agrawal, A., Boyd, S.P.: Joint graph learning and model fitting in Laplacian regularized stratified models. arXiv e-prints **arXiv:2305.02573** (2023)
11. Condat, L.: A primal–dual splitting method for convex optimization involving Lipschitzian, proximable and linear composite terms. Journal of Optimization Theory and Applications **158**(2), 460–479 (2013)
12. Condat, L., Kitahara, D., Contreras, A., Hirabayashi, A.: Proximal splitting algorithms for convex optimization: A tour of recent advances, with new twists. SIAM Review **65**(2), 375–435 (2023)
13. Cox, T.F., Cox, M.A.A.: Multidimensional Scaling. Chapman & Hall (2000)
14. Dattorro, J.: Convex Optimization & Euclidean Distance Geometry. Meboo Publishing (2005)
15. Dokmanic, I., Parhizkar, R., Ranieri, J., Vetterli, M.: Euclidean distance matrices: Essential theory, algorithms, and applications. IEEE Signal Processing Magazine **32**(6), 12–30 (2015)
16. Eckstein, J.: Approximate iterations in Bregman-function-based proximal algorithms. Mathematical Programming **83**(1), 113–123 (1998)

17. Gaffke, N., Mathar, R.: A cyclic projection algorithm via duality. Metrika **36**(1), 29–54 (1989)
18. Glowinski, R., Osher, S.J., Yin, W.: Splitting Methods in Communication, Imaging, Science, and Engineering. Scientific Computation, Springer (2017)
19. Glunt, W., Hayden, T.L., Hong, S., Wells, J.: An alternating projection algorithm for computing the nearest Euclidean distance matrix. SIAM Journal on Matrix Analysis and Applications **11**(4), 589–600 (1990)
20. Gower, J.C.: Properties of Euclidean and non-Euclidean distance matrices. Linear Algebra and its Applications **67**, 81–97 (1985)
21. Hayden, T.L., Wells, J.: Approximation by matrices positive semidefinite on a subspace. Linear Algebra and Its Applications **109**, 115–130 (1988)
22. Jacobs, M., Leger, F., Li, W., Osher, S.: Solving large-scale optimization problems with a convergence rate independent of grid size. SIAM Journal on Numerical Analysis **57**(3), 1100–1123 (2019)
23. Jiang, X., Vandenberghe, L.: Bregman primal–dual first-order method and applications to sparse semidefinite programming. Computational Optimization and Applications **81**(1), 127–159 (2022)
24. Jiang, X., Vandenberghe, L.: Bregman three-operator splitting methods. Journal of Optimization Theory and Applications **196**(3), 936–972 (2023)
25. Komodakis, N., Pesquet, J.: Playing with duality: an overview of recent primal–dual approaches for solving large-scale optimization problems. IEEE Signal Processing Magazine **32**(6), 31–54 (2015)
26. Kovan, I.: Multidimensional scaling (MDS) for dimensionality reduction and data visualization. available at `https://rb.gy/kmhvc` (2021)
27. Krislock, N., Wolkowicz, H.: Euclidean distance matrices and applications. In: Anjos, M.F., Lasserre, J.B. (eds.) Handbook on Semidefinite, Conic and Polynomial Optimization, pp. 879–914. Springer (2012)
28. Kruskal, J.B.: Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika **29**(1), 1–27 (1964)
29. Kulis, B., Surendran, A.C., Platt, J.C.: Fast low-rank semidefinite programming for embedding and clustering. In: Meila, M., Shen, X. (eds.) Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 2, pp. 235–242. PMLR (2007)
30. Liu, Y., Xu, Y., Yin, W.: Acceleration of primal–dual methods by preconditioning and simple subproblem procedures. Journal of Scientific Computing **86**(2), 21 (2021)
31. Malitsky, Y., Pock, T.: A first-order primal–dual algorithm with linesearch. SIAM Journal on Optimization **28**(1), 411–432 (2018)
32. Moreau, J.: Proximité et dualité dans un espace Hilbertien. Bulletin de la Société Mathématique de France **93**, 273–299 (1965)
33. O'Donoghue, B., Chu, E., Parikh, N., Boyd, S.: SCS: Splitting conic solver, version 3.2.3. `https://github.com/cvxgrp/scs` (2022)
34. Ortega, A.: Introduction to Graph Signal Processing. Cambridge University Press (2022)
35. Parikh, N., Boyd, S.: Proximal algorithms. Foundations and Trends in Optimization **1**(3), 127–239 (2014)
36. Pock, T., Chambolle, A.: Diagonal preconditioning for first order primal–dual algorithms in convex optimization. In: International Conference on Computer Vision. pp. 1762–1769 (2011)
37. Rasch, J., Chambolle, A.: Inexact first-order primal–dual algorithms. Computational Optimization and Applications **76**(2), 381–430 (2020)

38. Rockafellar, R.T.: Monotone operators and the proximal point algorithm. SIAM Journal on Control and Optimization **14**(5), 877–898 (1976)
39. Schoenberg, I.J.: Remarks to Maurice Fréchet's article "Sur la définition axiomatique d'une classe d'espaces vectoriels distanciés applicables vectoriellement sur l'espace de Hilbert". Annals of Mathematics **36**(3), 724–732 (1935)
40. Schoenberg, I.J.: Metric spaces and positive definite functions. Transactions of the American Mathematical Society **44**(3), 522–536 (1938)
41. Shepard, R.N.: The analysis of proximities: multidimensional scaling with an unknown distance function. i. Psychometrika **27**(2), 125–140 (1962)
42. Solodov, M.V., Svaiter, B.F.: A unified framework for some inexact proximal point algorithms. Numerical Functional Analysis and Optimization **22**(7-8), 1013–1035 (2001)
43. Song, L., Smola, A., Borgwardt, K., Gretton, A.: Colored maximum variance unfolding. In: Advances in Neural Information Processing Systems 20 (2007)
44. Sturm, J.F.: Using SeDuMi 1.02, A MATLAB toolbox for optimization over symmetric cones. Optimization Methods and Software **11**(1–4), 625–653 (1999)
45. Sun, Y., Vandenberghe, L.: Decomposition methods for sparse matrix nearness problems. SIAM Journal on Matrix Analysis and Applications **36**(4), 1691–1717 (2015)
46. Toh, K.C., Tütüncü, R.H., Todd, M.J.: SDPT3 version 3.02. A MATLAB software for semidefinite-quadratic-linear programming (2002), available at `www.math.nus.edu.sg/~mattohkc/sdpt3.html`
47. Tuck, J., Barratt, S., Boyd, S.P.: A distributed method for fitting Laplacian regularized stratified models. Journal of Machine Learning Research **22**(60), 1–37 (2021)
48. Weinberger, K.Q., Saul, L.K.: Unsupervised learning of image manifolds by semidefinite programming. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition. vol. 2 (2004)
49. Weinberger, K.Q., Saul, L.K.: An introduction to nonlinear dimensionality reduction by maximum variance unfolding. In: National Conference on Artificial Intelligence. pp. 1683–1686 (2006)
50. Witten, D.M., Tibshirani, R.: Supervised multidimensional scaling for visualization, classification, and bipartite ranking. Computational Statistics & Data Analysis **55**(1), 789–801 (2011)