J. Nonlinear Var. Anal. 9 (2025), No. 6, pp. 885-906
Available online at http://jnva.biemdas.com
https://doi.org/10.23952/jnva.9.2025.6.04

AN EFFICIENT CGA_ADMM FOR THE METRIC NEARNESS PROBLEM

BO JIANG[1], PEIPEI TANG[1,*], CHENGJING WANG[2], YANGKAI WU[3]

[1]*School of Computer and Computing Science, Hangzhou City University, Hangzhou, China*
[2]*School of Mathematics, Southwest Jiaotong University, Chengdu, China*
[3]*Hangzhou Maida Intelligence Technology Co., Ltd, Hangzhou, China*

Abstract. The metric nearness problem aims to find a metric matrix nearest to a given dissimilarity matrix with the triangle inequalities valid. In this paper, we consider the metric nearness problem with the distance measured by the vector ℓ_p $(p = 1,2,\infty)$ norm. Due to the $O(n^3)$ constraints and $O(n^2)$ variables, the main difficulty of solving this kind of large scale problems is the high memory requirement. We design a constraint generation based alternating direction method of multipliers (CGA_ADMM) and take full advantage of the special structure of the constraint matrix so that the memory requirement of the CGA_ADMM is moderate. Numerical experiments of the real world graph data sets involving up to 10^8 variables and 10^{12} constraints demonstrate that our algorithm has a better performance than the current state-of-the-art algorithms.

Keywords. Alternating direction method of multipliers; Constraint generation algorithm; Metric nearness problem.

1. INTRODUCTION

As we know, clustering is one of the most fundamental tasks in machine learning. Among various kinds of clustering algorithms, an indispensable step is to get the metric distances of the corresponding clustering objects, which shows the pairwise dissimilarity of every two objects. Other applications including image processing, metric-based indexing of databases and computer vision show that quantities satisfying the metric property should be given for further purpose (see, e.g., [1, 2]). However the measurements usually do not represent the actual distance due to the noise of the input data or errors of the measurements. Although it is easy to obtain pairwise dissimilarities of every two objects, the metric distance should satisfy the specific feature of the triangular inequality. A given matrix $M = (m_{ij})$ is called a metric matrix if $m_{ij} \leq m_{ik} + m_{jk}$ holds for every triple (i, j, k). We denote n as the number of given nodes and $\mathscr{M}_n$ as the set of all the metric matrices of order n.

A natural idea is to get the dissimilarity matrix first and then try to find a metric matrix nearest to the dissimilarity matrix. Given a dissimilarity matrix $D = \left(d_{ij}\right)_{n\times n}$, the metric nearness problem, which dates back to [3], aims to find a metric matrix $X = \left(x_{ij}\right)_{n\times n}$ nearest to D. A weight matrix is usually used to capture the confidence in individual dissimilarity measures. Recently,

*Corresponding author.
E-mail address: tangpp@hzcu.edu.cn (P. Tang).

Received 14 April 2024; Accepted 21 February 2025; Published online 10 August 2025.

The work of [4] and [5] reformulated the problem as the metric violation distance problem and the sparse metric repair problem, respectively. For a given distance matrix, they tried to modify as few entries as possible so that the derived solution satisfies a metric. Furthermore, for a given weighted graph [6] proposed to find the smallest number of modifications to the weights so that the resulting weighted graph distances satisfy a metric. Another kind of metric learning aims to find a tree metric which minimizes the given error (see, e.g., [7, 8]).

The metric nearness problem is also a branch of matrix nearness problem (see, e.g., [9]), whose main purpose is to find a nearest matrix of the given matrix with some properties such as symmetry, positive definiteness, and normality. There are various kinds of measurements such as the vector ℓ_p $(p \geq 1)$ norms and Kullback-Leibler divergence to qualify the approximation error, which is also known as the nearness. The classical correlation clustering problem (see, e.g., [10]) aims to cluster the nodes in a way that minimizes the total quantity of mistakes. The mistake at the pair (i,j) is w^+_{ij} if the i-th and j-th nodes are separated but w^-_{ij} if the i-th and j-th nodes are clustered together. It can be written formally as the following integer linear programming problem

$$\begin{aligned} \min_{X\in\mathscr{M}_n} \quad & \sum_{1\leq i<j\leq n} \left(w^+_{ij} x_{ij} + w^-_{ij}(1-x_{ij}) \right) \\ \text{s.t.} \quad & x_{ij} \in \{0,1\}, \ \forall\, 1 \leq i < j \leq n. \end{aligned} \tag{1.1}$$

As mentioned in [11], the correlation clusting problem (1.1) can be relaxed as an ℓ_1 norm based metric nearness problem. In this paper, we consider the weighted ℓ_p norm based metric nearness problem, that is, we solve the following optimization problem

$$\min_{X\in\mathscr{M}_n} \left(\sum_{1\leq i<j\leq n} |\bar{w}_{ij}(x_{ij}-d_{ij})|^p \right)^{1/p}. \tag{1.2}$$

We deal with the problem by stacking the columns of the strict upper triangular part of the matrix to a long column vector.

General algorithms for solving the metric nearness problem are to reformulate them as a linear programming problem or a quadratic programming problem, which is confronted with the difficulty of memory exhaustion due to $O(n^3)$ constraints when dealing with large graphs. During the last decades, researchers designed specific algorithms to solve the metric nearness problem. [12] designed a triangle-fixing algorithm based on the inherit specific structure of the metric constraints. Several years later, [13] presented the implementation details of the triangle-fixing algorithm for the ℓ_1, ℓ_2 and ℓ_∞ norms based metric nearness problem. Furthermore, by adding a proximal term to the objective function, [11] applied Dykstra's projection method to solve the corresponding quadratic programming problem, which is closely related to the original ℓ_1 norm based metric nearness problem. Due to the specific structure of the triangle inequalities, the Dykstra's projection method has moderate memory requirement and it can be used to solve the metric nearness problem with variables up to 6×10^7 as shown in [11]. [14] presented a parallel version of Dykstra's projection method, coded the algorithm in Julia and tested the algorithm using matrices with variables up to 1.6×10^8. Although Dykstra's projection method converges rapidly for small scale data problems and it has a linear convergence rate (see e.g., [15]), we need to slightly modify the objective function by adding a proximal term with an extra parameter, which is not known in advance, when dealing with ℓ_1 and ℓ_∞ norms based metric nearness problems. For each iteration, the computation cost of Dykstra's projection method is $O(n^3)$, which results in high time cost when solving large scale problems. [16] proposed an active set algorithm, PROJECT AND FORGET, based on Floyd's Shortest path algorithm and Bregman projections.

Although metric nearness problem (1.2) is convex, it involves $n(n-1)/2$ variables, $n(n-1)(n-2)/2$ constraints and its objective function is nonsmooth. It is still of great challenge to obtain an approximate solution with moderate accuracy efficiently. For the ℓ_1, ℓ_∞ norms based metric nearness problems, we can reformulate (1.2) to a linear programming problem and solve it by classical interior point solvers such as Gurobi and Mosek when n is not too large. However, the memory cost becomes unacceptable when n is larger than 10^3 due to the $O(n^3)$ constraints. As we know, Dykstra's projection method is a well known algorithm that can solve problem (1.2) , but the accuracy of the obtained solution depends on the proximal term which is hard to determine. An alternative approach is the alternating direction method of multipliers (ADMM) (see e.g., [17, 18]) which deals with the corresponding block variables alternatively in each iteration. One may see [19] for the history of the ADMM.

In this paper, we design a constraint generation algorithm based ADMM named by CGA_A DMM to solve problem (1.2). The main difficulty of applying the ADMM to solve the metric nearness problem lies in the high memory requirement due to the $O(n^3)$ constraints and $O(n^2)$ variables. If we apply the ADMM directly, it will run out of memory for large scale problems. The basic idea of the constraint generation algorithm, also known as the cutting plane algorithm (see, e.g., [20, 21]), is that the feasible set is approximated by using only a subset of constraints and more constraints are added if the desired solution is infeasible. Since the number of active constraints at the solution point is at most $n(n-1)/2$, we combine the ADMM with the constraint generation strategy and design the CGA_ADMM so that the storage requirement of each iteration is much lower than that of the problem with whole constraints. Furthermore, we also take full advantage of the special structure of the constraint matrix such that it does not need to be stored in memory and the corresponding subproblem of the ADMM can be solved efficiently. Thus we can solve practical problems with n up to 10^5 and the memory requirement is acceptable. Numerical experiments demonstrate that our algorithm is efficient and has a better performance than the current state-of-the-art algorithms.

2. PRELIMINARIES

In this section, we first give some basic preliminaries for further use.

Let $\mathscr{X}$ be a finite dimensional Hilbert space equipped with an inner product $\langle\cdot,\cdot\rangle$ and the corresponding induced norm $\|\cdot\|$, and let $f:\mathscr{X}\longrightarrow[-\infty,+\infty]$ be an extended real valued function with its conjugate function f^* defined as $f^*(x):=\sup_{u\in\mathrm{dom}(f)}\left\{\langle x,u\rangle-f(u)\right\}$, where $\mathrm{dom}(f)=\{x\mid f(x)<+\infty\}$. The extended real valued function f is said to be proper if there exists at least one $x'\in\mathscr{X}$ such that $f(x')<+\infty$ and $f(x)>-\infty$ for all $x\in\mathscr{X}$, otherwise f is improper. For a proper lower semicontinuous function f, the proximal mapping $\mathrm{Prox}_{\sigma f}$ associated with f at x for a given $\sigma>0$ is defined by

$$\mathrm{Prox}_{\sigma f}(x):=\underset{u\in\mathscr{X}}{\mathrm{argmin}}\left\{f(u)+\frac{1}{2\sigma}\|u-x\|^2\right\}.$$

The proximal mapping $\mathrm{Prox}_{\sigma f}$ is single-valued and continuous. According to Moreau's identity (see, e.g., [22, Theorem 31.5]),

$$\mathrm{Prox}_{\sigma f}(x) + \sigma \mathrm{Prox}_{\sigma^{-1} f^*}(\sigma^{-1} x) = x,\ \forall\, x \in \mathscr{X}.$$

For a given vector $v \in \mathscr{R}^n$, we denote v_i as the ith component of the vector v, $\mathrm{Diag}(v)$ as the diagonal matrix whose diagonal vector is v, $\mathscr{S}^n$ as the space of symmetric matrices of order n, and I_n as the identity matrix of order n. Let C be a subset of $\mathscr{X}$. We define the indicator function δ_C of C as $\delta_C(x) = 0$ if $x \in C$ and otherwise $\delta_C(x) = +\infty$, and denote the Euclidean projection of x onto C by $\Pi_C(x) := \mathrm{argmin}_{y\in C} \|y - x\|$. For simplicity, let $n_1 = n(n-1)/2$, $n_2 = n(n-1)(n-2)/2$ and $\mathrm{trivec} : \mathscr{S}^n \longrightarrow \mathscr{R}^{n_1}$ be a function that stacks the columns of the strict upper triangular part of an input matrix to a column vector with $\mathrm{trivec}(X) := [x_{12}, x_{13}, x_{23}, \ldots, x_{1n}, x_{2n}, \ldots, x_{n-1,n}]^T$ for any $X \in \mathscr{S}^n$. By introducing a slack variable $y = \mathrm{trivec}(X - D)$, metric nearness problem (1.2) is equivalent to the following composite optimization problem

$$\min_{y\in\mathscr{R}^{n_1}} \Big\{ h(Ay - b) + q(Wy) \Big\}, \tag{2.1}$$

where $h(\cdot) = \delta_{\mathscr{R}_-^{n_2}}(\cdot)$, $q(\cdot) = \|\cdot\|_p$, A is the constraint matrix corresponding to the triangle inequalities, $b = -A\mathrm{trivec(D)}$, $W = \mathrm{Diag}(\mathrm{trivec}(\overline{W}))$ with $\overline{W} = (\bar{w}_{ij})$ and $\mathscr{R}_-^{n_2}$ is an n_2-dimensional negative half quadrant cone. Furthermore, we can rewrite problem (2.1) equivalently as

$$\min_{\xi\in\mathscr{R}^{n_2}, y,\eta\in\mathscr{R}^{n_1}} \Big\{ h(\xi) + q(\eta) \;\Big|\; Ay - \xi = b,\ Wy - \eta = 0 \Big\}. \tag{2.2}$$

The dual problem related to problem (2.2) takes the following form

$$\max_{u\in\mathscr{R}^{n_2}, v\in\mathscr{R}^{n_1}} \Big\{ -h^*(u) - q^*(v) - \langle u, b\rangle \;\Big|\; A^T u + W^T v = 0 \Big\}. \tag{2.3}$$

We can also write out the Lagrangian function related to problem (2.2) as

$$L(y,\xi,\eta;u,v) = h(\xi) + q(\eta) + \langle Ay - \xi - b, u\rangle + \langle Wy - \eta, v\rangle.$$

The Karush-Kuhn-Tucker (KKT) condition associated with problem (2.2) is as follows

$$Ay - \xi - b = 0, \quad Wy - \eta = 0, \quad A^T u + W^T v = 0, \quad u \in \partial h(\xi), \quad v \in \partial q(\eta).$$

3. The Constraint Generation Based ADMM

In this section, we introduce the constraint generation algorithm based ADMM to solve primal problem (2.2). The ADMM is a classical algorithm for solving many large scale convex optimization problems derived from machine learning. It solves the two blocks of variables alternatively so that the difficulty for solving the subproblem is reduced. However, the computation cost related to each iteration of the ADMM is still very huge and unacceptable, especially when n is large. Therefore we have to take full advantage of the special structure of the constraint matrix and adopt the strategy of constraint generation to reduce both the time complexity and memory requirement at each iteration of the ADMM in the CGA_ADMM. On the other hand, we add a part of the violated constraints into the constraint set successively so that the number of the outer iterations of the CGA_ADMM is small, which guarantees the high efficiency of our algorithm.

3.1. **The ADMM for solving problem (2.2).** As the subproblems we consider below (see Subsection 3.2) are all of the same type of problem (2.2), we only introduce the ADMM for problem (2.2) in this subsection. Given $\sigma > 0$, the augmented Lagrangian function of problem (2.2) can be written as

$$\begin{aligned}\mathscr{L}_\sigma(y,\xi,\eta;u,v) = h(\xi)+q(\eta)+\frac{\sigma}{2}\|Wy-\eta+\sigma^{-1}v\|^2-\frac{1}{2\sigma}\|v\|^2 \\ +\frac{\sigma}{2}\|Ay-\xi-b+\sigma^{-1}u\|^2-\frac{1}{2\sigma}\|u\|^2.\end{aligned}$$

Now we list the details of the ADMM in Algorithm 1. In the implementation of the ADMM, it is important to solve the corresponding subproblem efficiently. We can see that the main idea of the ADMM is to update the two blocks of variables alternately. At the kth iteration, it is easy to show that

$$\begin{aligned}&\xi^{k+1} = \mathrm{Prox}_{\sigma^{-1}h}(Ay^k-b+\sigma^{-1}u^k) = \Pi_{\mathscr{R}_-^{n_2}}(Ay^k-b+\sigma^{-1}u^k),\\ &\eta^{k+1} = \mathrm{Prox}_{\sigma^{-1}q}(Wy^k+\sigma^{-1}v^k),\\ &y^{k+1} \approx (A^TA+W^TW)^{-1}\Big(A^T(\xi^{k+1}+b-\sigma^{-1}u^k)+W^T(\eta^{k+1}-\sigma^{-1}v^k)\Big).\end{aligned}$$

Based on Moreau's identity and the result of [23], we can compute the proximal mapping of $\sigma^{-1}q$ as $\mathrm{Prox}_{\sigma^{-1}q}(z) = z-\Pi_{C_p^{1/\sigma}}(z)$, $\forall\, z\in\mathscr{R}^{n_1}$, where

$$C_p^{1/\sigma} := \left\{ z \;\middle|\; \|z\|_{p'} \le \frac{1}{\sigma} \right\},\quad \frac{1}{p}+\frac{1}{p'} = 1.$$

The details of $\Pi_{C_p^{1/\sigma}}$ for $q(\cdot) = \|\cdot\|_p$ with $p = 1,2,\infty$ are listed below.

(1) $q(\cdot) = \|\cdot\|_1$, we have

$$\left(\Pi_{C_1^{1/\sigma}}(z)\right)_i = \begin{cases} z_i, & \text{if } |z_i| \le 1/\sigma,\\ \mathrm{sign}(z_i)/\sigma, & \text{otherwise.}\end{cases}$$

(2) $q(\cdot) = \|\cdot\|_2$, we have

$$\Pi_{C_2^{1/\sigma}}(z) = \begin{cases} z, & \text{if } \|z\|_2 \le 1/\sigma,\\ \frac{z}{\sigma\|z\|_2}, & \text{otherwise.}\end{cases}$$

(3) $q(\cdot) = \|\cdot\|_\infty$, we have

$$\Pi_{C_\infty^{1/\sigma}}(z) = \begin{cases} z, & \text{if } \|z\|_1 \le 1/\sigma,\\ P_z\Pi_\triangle(P_zz), & \text{otherwise,}\end{cases}$$

where $P_z = \mathrm{Diag}(\mathrm{sign}(z))$, $\triangle = \{z \mid z_1+\ldots+z_{n_1} = 1/\sigma\}$ and $\Pi_\triangle(P_zz)$ can be computed in $O(n_1\log n_1)$ operations. One may see [23] for more details.

In order to update the variable y, we need to solve a large scale linear system of equations. We apply the Sherman-Morrison-Woodbury formula [24] if it is necessary, i.e.,

$$(B+Y^TY)^{-1} = B^{-1}-B^{-1}Y^T(I_t+YB^{-1}Y^T)^{-1}YB^{-1},$$

where $B\in\mathscr{R}^{s\times s}$ and $Y\in\mathscr{R}^{t\times s}$ are given matrices. We solve the corresponding linear system by an iterative solver such as the preconditioned conjugate gradient (PCG) method.

Algorithm 1 The ADMM for problem (2.2)

```
Given $\rho \in (0, \frac{1+\sqrt{5}}{2})$, $\sigma > 0$ and $\{\varepsilon_k\}_{k\geq 0}$ be a summable sequence of nonnegative numbers, choose $(y^0, u^0, v^0) \in \mathscr{R}^{n_1} \times \mathscr{R}^{n_2} \times \mathscr{R}^{n_1}$. Set $k = 0$.
  repeat
    Compute
        $(\xi^{k+1}, \eta^{k+1}) = \operatorname{argmin}_{\xi,\eta} \mathscr{L}_\sigma(y^k, \xi, \eta; u^k, v^k)$,
        $y^{k+1} \approx \operatorname{argmin}_y \left\{ \phi_k(y) := \mathscr{L}_\sigma(y, \xi^{k+1}, \eta^{k+1}; u^k, v^k) \right\}$,
    with $\mathscr{M} := \sigma(A^TA + W^TW)$ and $d^k \in \partial\phi_k(y^{k+1})$ such that $\|\mathscr{M}^{-\frac{1}{2}} d^k\| \leq \varepsilon^k$.
    Update the variables $u$ and $v$
        $u^{k+1} = u^k + \rho\sigma(Ay^{k+1} - \xi^{k+1} - b)$,    $v^{k+1} = v^k + \rho\sigma(Wy^{k+1} - \eta^{k+1})$
    and set $k := k+1$.
  until A desired stopping criterion is satisfied.
```

The ADMM in algorithm 1 is an inexact version. For more details about the exact version of ADMM, one refer to [17, 25]. We give a brief proof to the convergence results of Algorithm 1 in the Appendix. One refer to [26] for more details.

Theorem 3.1. *Assume that all elements of the weight matrix $\overline{W}$ are positive and the solution sets of primal problem (2.2) and dual problem (2.3) are nonempty. Let $\{(y^k, \xi^k, \eta^k, u^k, v^k)\}$ be the sequence generated by Algorithm 1. Then the sequence $\{(y^k, \xi^k, \eta^k)\}$ converges to an optimal solution of primal problem (2.2) and $\{(u^k, v^k)\}$ converges to an optimal solution of dual problem (2.3).*

Theorem 3.2. *Suppose the same assumption as that of Theorem 3.1 holds. Let $\{(\xi^k, \eta^k, y^k, u^k, v^k)\}$ be the sequence generated by the ADMM and $(\bar{\xi}, \bar{\eta}, \bar{y}, \bar{u}, \bar{v})$ be a solution of the corresponding KKT system. Let $r(\xi, \eta, y, u, v) = dist^2(0, \partial h(\xi) - u) + dist^2(0, \partial q(\eta) - v) + \|A^T u + W^T v\|^2 + \|Ay - \xi - b\|^2 + \|Wy - \eta\|^2$. Then there exists a constant $\kappa > 0$ such that*

$$\min_{1\leq k\leq T} \{r(\xi^{k+1}, \eta^{k+1}, y^{k+1}, u^{k+1}, v^{k+1})\} \leq \kappa/T,$$

$$\lim_{T\to\infty} \{T \min_{1\leq k\leq T} \{r(\xi^{k+1}, \eta^{k+1}, y^{k+1}, u^{k+1}, v^{k+1})\}\} = 0.$$

3.2. **The constraint generation algorithm.** In this subsection, we introduce the constraint generation algorithm. We adopt the strategy of solving the related problem with a subset of constraints and then adding the violated constraints successively until all the constraints satisfy a given stopping criterion. For each iteration of the constraint generation algorithm, we apply Algorithm 1 to solve the corresponding subproblem. The implementation details are presented in Algorithm 2.

In the implementation of the constraint generation algorithm, we need to update the constraint set so that a desired stopping criterion is satisfied. Let $\bar{S}$ be the complement set of S. In each iteration of Algorithm 2, we set $\xi_{\bar{S}} = \Pi_{\mathscr{R}_-^{|\bar{S}|}}(A_{\bar{S}}\hat{y} - b_{\bar{S}})$, $u_{\bar{S}} = 0$ and then check the desired stopping criterion for the whole problem. During the iterations, we need to add a subset of

Algorithm 2 CGA_ADMM: A constraint generation algorithm based ADMM

```
Choose a subset $S$ of $\{1,2,\ldots,n_2\}$ with at least one element $j \in S$ such that $b_j < 0$.
repeat
    Apply Algorithm 1 to obtain an approximate solution $\hat{y}$ of the following problem
        $$\min_{y\in\mathscr{R}^{n_1}} \Big\{h(A_S y - b_S) + q(Wy)\Big\}.$$
    Let $S' = \{j \mid A_j\hat{y} - b_j > 0,\ j \in \{1,2,\ldots,n_2\} \setminus S\}$. If $S' \neq \emptyset$, then set $S := S \bigcup$ subset of $S'$.
until A desired stopping criterion is satisfied.
```

the violated constraints to the constraint set. The strategy needs to balance the cost of solving each subproblem and the number of constraint generation iterations. Adding too few violated constraints may lead to too many of outer iterations and increase the total cost of computation. As mentioned in [20], we can also take an idea of dropping some of the elements of S, i.e., we can drop some of those constraints that are not active. In our implementation, we add $\min\{|S|,|S'|\}$ violated constraints, which are among the first $\min\{|S|,|S'|\}$ largest values of the violated constraints, into the constraint set at each iteration.

During the constraint generation of Algorithm 2, the desired stopping criterion of the whole problem (defined in Section 4) is also satisfied if $S' = \emptyset$, which indicates that the constraint generation algorithm can be terminated in finite iterations.

Based on the fact that $y = 0$ is the unique optimal solution of problem (2.2) if $-b \le 0$ holds, a natural choice of the initial value of y is the zero vector. In the implementation of the CGA_ADMM, we set the initial constraints set $S = \{i \mid b_i < 0\}$. Furthermore, let J be the set of nonzero column indices of A_S with its complement set as $\bar{J}$, we only need to consider the variables y_J and $y_{\bar{J}} = 0$. Therefore, the actual number of variables related to each subproblem is $|J|$, which may be less than n_1 if $|S| \ll n_2$. As the number of constraints at each iteration is much less than n_2, the computation cost of each iteration is lower than that of applying ADMM to the original problem directly, and the running time of our algorithm will not grow too fast when n grows, especially when the number of the active constraints is much less than n_1. We use the solution (y,u,v) of the previous iteration to warm start the next iteration, which reduces the iteration number of the ADMM.

Now we give a brief summary of the efficiency of our algorithm. We have taken a deep consideration of the special structure of the problem. The zero vector (a natural guess solution) is used to generate a violated constraint set, which is also considered as the initial constraint set. Although $O(n^3)$ cost is needed in the step of computing S', numerical experiments show that the number of the constraint generation iterations is usually small, hence the computing cost is entirely acceptable. Since the constraint matrix is highly sparse, that is, each row has only 3 nonzero values, the number of the variables related to the constraints, especially for the first few iterations, may be less than the number of columns of the constraint matrix. The values of the variables not related to the constraints are all zeros because of the special structure of this problem. Furthermore, we can also apply the Sherman-Morrison-Woodbury formula so that the dimension of the linear system of equations that needs to be solved is less than n_1 when the number of the constraints is less than n_1. Therefore, the cost of each iteration for the

ADMM drops rapidly and the memory requirement is acceptable, which is demonstrated by the numerical experiments in Section 4.

4. Numerical Experiments

In this section, we test numerical experiments for our algorithm. The experiments are implemented on a Windows workstation (two 16-core, Intel Xeon E5-2667 @3.20GHz CPU, 64GB RAM) with all the algorithms written in C++ language except the PROJECT AND FORGET algorithm[1], which is originally written in Julia language.

In the implementation of our algorithm CGA_ADMM, we normalize each row of the constraint matrix A. Let R_p, R_d, R_c and R_g be the relative primal infeasibility, the relative dual infeasibility, the relative complementarity and the relative gap, respectively, which are defined as

$$R_p := \max\left\{\frac{\|Ay-\xi-b\|}{1+\|\xi\|}, \frac{\|Wy-\eta\|}{1+\|\eta\|}\right\}, \quad R_d := \frac{\|A^T u+W^T v\|}{1+\|W^T v\|},$$
$$R_g := \frac{|\text{pobj}-\text{dobj}|}{1+\text{dobj}}, \quad R_c := \max\left\{\frac{\|\xi-\text{Prox}_h(\xi+u)\|}{1+\|\xi\|}, \frac{\|\eta-\text{Prox}_q(\eta+v)\|}{1+\|\eta\|}\right\}$$

with pobj and dobj as the primal and dual objective values, respectively. The relative KKT residual is defined by $\theta_{kkt} := \max\{R_p, R_d, R_c\}$.

We adopt the relative KKT residual θ_{kkt} and the relative gap R_g to measure the accuracy of our algorithm. For each iteration of the constraint generation algorithm, we stop the ADMM with both θ_{kkt} and R_g of the inner subproblem being less than 10^{-3} or the number of the iterations reaching the maximum of 10000. The outer iteration of the constraint generation is stopped if the relative KKT residual of the whole problem is less than 10^{-3}. For the ADMM without constrain generation, it is terminated if both the relative KKT residual $\theta_{kkt} < 10^{-3}$ and the relative gap $R_g < 10^{-3}$ or its number of iterations reaches the maximum of 10000. The parameter $\rho = 1.618$.

In the numerical experiments of the ℓ_p $(p = 1,\infty)$ norm based metric nearness problem, we first reformulate the problem to a linear programming problem and then compare our algorithm with Dykstra's projection method, the Gurobi software and the PROJECT AND FORGET algorithm $(p = 1)$. As for Dykstra's projection method, we adopt the same stopping criterion as that used in [11] with the same tolerance. We also use the parallelization technology mentioned in [14] to improve the efficiency of the projection method. As the original version of Dykstra's projection method[2] is written in Julia, we rewrite it in C++ and compare the efficiency of both versions. In all of the following tables, we report the name of the dataset (Graph), the number of vertices ($|V|$), the number of edges ($|E|$), the performing time (time) in the format of hours:minutes:seconds and the objective function value (obj). For the CGA_ADMM and CGA_Gurobi we also report the number of iterations of constraint generation (iter). The comparison between C++ and Julia is listed in Table 1. Table 1 shows that the C++ version of Dykstra's Projection method runs faster than the Julia version for almost all datasets. The parallelization strategy of the C++ version differs from the Julia version. This difference may cause extra iterations in some datasets, such as caGrQc, but in most cases it makes the algorithm more

[1]https://www.dropbox.com/sh/lq5nnhi4je2lh89/AABUUW7k5z3lXTSm8x1hhN1Da?dl=0
[2]https://github.com/nveldt/ParallelDykstras

TABLE 1. The performances of Dykstra's projection method written by Julia and C++ for the ℓ_1 norm based metric nearness problem.

Graph	$\|V\|$	$\|E\|$	time	
			Julia	C++
minnesota	2640	3302	0:12:01	**0:06:26**
caGrQc	4158	13422	**0:54:59**	1:13:15
power	4941	6594	2:10:44	**1:03:24**
p2p-Gnutella08	6299	20776	8:19:28	**3:11:34**
caHepTh	8638	24806	30:50:28	**14:25:46**

efficient in each single loop. Thus in the following experiments when comparing with Dykstra's projection method, we use the C++ version and all of these algorithms are written in C++.

The Gurobi software is an efficient software package for solving linear programming problems. In the comparison, we apply the same strategy of constraint generation for the triangle inequalities constraints as that used in our algorithm to improve the efficiency of the Gurobi (CGA_Gurobi), while the constraints related to the reformulations of the ℓ_1 or ℓ_∞ norm are always in the constraint set. The iteration is stopped if the relative KKT residual of the corresponding linear programming problem is less than 10^{-3}. For each subproblem of the constraint generation algorithm, we use the barrier algorithm of the Gurobi and set 10^{-3} for both the tolerances BarConvTol and FeasibilityTol as the stop criterion for the Gurobi package. While applying Gurobi without constraint generation, the corresponding parameters are set the same as that used in the constraint generation version.

In the implementation of Dykstra's projection method, we adopt the same stopping criterion as that in [11], that is,

$$\max\left\{\frac{|\text{pobj}-\text{dobj}|}{|\text{dobj}|},\ \max\{Ay-b\}\right\} < 10^{-2}.$$

Although the stopping criterion for Dykstra's projection method seems stricter than that in our algorithm, the accuracy is lower, which can be seen from the objective values listed in the numerical experiments. The reason lies in that Dykstra's projection method can only be applied to solve an approximate problem with an additional proximal term for the ℓ_1 and ℓ_∞ norms based metric nearness problems.

We implement the comparisons on some real world graph data sets from SuiteSparse Matrix Collection [27] including collaboration networks, web-based graphs and others. The same approach as that used in [11] is adopted to generate the input dissimilarity matrix and the weight matrix for our problem. For all the graphs, we remove the weights and directions of all edges to ensure that they are undirected and unweighted.

4.1. **Numerical experiments for the ℓ_1 norm based metric nearness problem.** In this section, we perform our numerical experiments for the ℓ_1 norm based metric nearness problem. We compare our CGA_ADMM algorithm with Dykstra's projection method, the Gurobi software and the PROJECT AND FORGET algorithm. We run the experiments of the PROJECT AND

FORGET algorithm with the default parameters. All of the following experiments achieve the specified accuracy requirements unless being out of memory.

In order to apply Dykstra's projection method and Gurobi to solve the ℓ_1 norm based metric nearness problem, we need to rewrite the problem as a linear programming problem. By introducing some slack variables, we first reformulate the ℓ_1 norm based metric nearness problem (2.1) as the following form:

$$\min_{z\in\mathscr{R}^{2n_1}} \left\{ \langle c, z\rangle \;\middle|\; \widetilde{A}z \le \widetilde{b} \right\}, \tag{4.1}$$

where c is a column vector with $c = [\mathbf{0}_{n_1}; \mathrm{trivec}(\overline{W})]$ and

$$\widetilde{A} = \begin{bmatrix} A & O_{n_2\times n_1} \\ I_{n_1} & -I_{n_1} \\ -I_{n_1} & -I_{n_1} \end{bmatrix}, \quad \widetilde{b} = \begin{bmatrix} b \\ \mathbf{0}_{n_1} \\ \mathbf{0}_{n_1} \end{bmatrix}.$$

The Lagrangian function of problem (4.1) takes the form

$$l(z;u) = \langle c, z\rangle + \langle u, \widetilde{A}z - \widetilde{b}\rangle$$

and the related KKT condition is

$$\widetilde{A}^T u + c = 0, \; u \in \mathscr{R}_+^{2n_1+n_2}(\widetilde{A}z - \widetilde{b}).$$

Then we can apply the constraint generation based algorithm with each subproblem solved by the Gurobi package to obtain an approximate solution of the linear programming problem (4.1). For each iteration of the constraint generation, we set $u_{\bar{S}} = 0$. The constraint generation algorithm is stopped if the relative KKT residual of problem (4.1)

$$\max\left\{ \frac{\|\widetilde{A}^T u + c\|}{1 + \|u\| + \|c\|}, \frac{\|u - \Pi_{\mathscr{R}_+^{2n_1+n_2}}(u + \widetilde{A}z - \widetilde{b})\|}{1 + \|u\| + \|\widetilde{A}z\| + \|\widetilde{b}\|} \right\} < 10^{-3}.$$

As for Dijkstra's projection method, we need to add an extra proximal term to the objective function of problem (4.1) to get the corresponding quadratic programming problem in the following form:

$$\min_{z\in\mathscr{R}^{2n_1}} \left\{ \langle c, z\rangle + \frac{1}{2\gamma} z^T \widetilde{W} z \;\middle|\; \widetilde{A}z \le \widetilde{b} \right\}, \tag{4.2}$$

where $\widetilde{W}$ is a diagonal matrix defined as $\widetilde{W} := \mathrm{Diag}([\mathrm{trivec}(\overline{W}); \mathrm{trivec}(\overline{W})])$, which is the same as that in [11]. Though it has been proved in [28] that there exists $\gamma_0 > 0$ such that for all $\gamma > \gamma_0$ the optimal solution of problem (4.2) is also an optimal solution of problem (4.1) when $\widetilde{W} = I_{2n_1}$. However, it is unknown how to determine γ_0 exactly for these problems and Dykstra's projection method converges more slowly when γ grows larger. In numerical experiments of the ℓ_1 norm based metric nearness problem, we set $\gamma = 1$, which is the same as the setting in [11].

As for the ADMM and Gurobi, we first compare them with their constraint generation versions CGA_ADMM and CGA_Gurobi for the ℓ_1 norm based metric nearness problem and the results are listed in Table 2.

Table 2 demonstrates that the stopping criterion applied in the CGA_ADMM is appropriate to obtain an approximate solution of the problem with the whole constraints. The gaps of

TABLE 2. The performances of the ADMM, CGA_ADMM, Gurobi, CGA_Gurobi for the ℓ_1 norm based metric nearness problem.

Graph	ADMM		CGA_ADMM		Gurobi		CGA_Gurobi	
$(\lvert V\rvert,\lvert E\rvert)$	time	obj	time(iter)	obj	time	obj	time(iter)	obj
jazz (198,2742)	0:03:12	2.46e+02	0:00:22(2)	2.46e+02	0:07:44	2.46e+02	0:01:02(2)	2.46e+02
SmallW (233,994)	0:06:37	7.89e+02	0:06:02(4)	7.88e+02	0:14:45	7.87e+02	0:05:49(4)	7.88e+02
celegansneural (297,2148)	0:09:01	7.15e+02	0:03:14(2)	7.15e+02	0:54:33	7.14e+02	0:10:50(2)	7.14e+02
USAir97 (332,2126)	0:13:13	7.33e+02	0:03:22(3)	7.33e+02	0:43:34	7.32e+02	0:11:11(3)	7.32e+02

the objective values between ADMM and CGA_ADMM, Gurobi and CGA_Gurobi are small. We can also see from Table 2 that both of the constraint generation versions are more efficient. Therefore, in the following numerical experiments, we only adopt the CGA_ADMM and CGA_Gurobi for further comparisons.

TABLE 3. The performances of Dykstra's projection method, CGA_ADMM, CGA_Gurobi and PROJECT AND FORGET for the ℓ_1 norm based metric nearness problem.

Graph	Dykstra		CGA_ADMM		CGA_Gurobi		PROJECT AND FORGET	
$(\lvert V\rvert,\lvert E\rvert)$	time	obj	time(iter)	obj	time(iter)	obj	time(iter)	obj
minnesota (2640,3302)	0:06:26	1.01e+03	0:00:39(2)	9.53e+02	0:00:14(2)	9.54e+02	0:03:31(11)	1.11e+03
caGrQc (4158,13422)	1:13:15	3.07e+03	0:08:42(1)	2.82e+03	out of memory		1:28:23(99)	4.04e+03
power (4941,6594)	1:03:24	2.15e+03	0:01:39(1)	1.94e+03	0:01:03(1)	1.94e+03	0:39:30(26)	2.59e+03
caHepTh (8638,24806)	14:25:46	7.39e+03	0:25:57(1)	6.97e+03	out of memory		out of memory	
caHepPh (11204,117619)	34:26:56	1.39e+04	5:23:58(2)	1.39e+04	out of memory		out of memory	
caAstroPh (17903,196972)	180:15:39	3.11e+04	6:01:23(1)	2.87e+04	out of memory		out of memory	

We perform the comparisons for several real data sets and list the results in Table 3. The experiments show that the Gurobi package runs out of memory when the number of vertices is large, especially larger than 6000. For Dykstra's projection method, though it works well for the data sets with n small, the running time grows quickly as n grows larger due to the $O(n^3)$ cost of each iteration. For a given γ, the KKT condition for the original problem is unknown. Though the stopping criteria for these algorithms have some differences, the function value obtained by our algorithm is more accurate comparing with Dykstra's projection method and the PROJECT AND FORGET algorithm. And the PROJECT and FORGET algorithm does not obtain the desired results for the data sets with the number of vertices larger than 8000 due to the limitation of memory. For each data set, it is interesting that the constraint generation stops after one or two iterations and the constraints involved in each iteration are much less than the original problem, which may explain the high efficiency of our algorithm.

4.2. **Numerical experiments for the ℓ_2 norm based metric nearness problem.**

In this section we present the numerical experiments for the ℓ_2 norm based metric nearness problem. To obtain the same primal infeasibility level as that of Dykstra's projection method, we add an extra stopping criterion $\max\{Ay-b\} < 10^{-2}$ to the constraint generation method.

TABLE 4. The performances of the ADMM, CGA_ADMM for the ℓ_2 norm based metric nearness problem.

Graph	ADMM		CGA_ADMM	
	time	obj	time(iter)	obj
jazz	0:02:38	1.19e+01	0:00:26(2)	1.19e+01
SmallW	0:03:18	1.74e+01	0:00:44(2)	1.74e+01
celegansneural	0:24:08	1.73e+01	0:00:50(1)	1.73e+01
USAir97	0:07:56	1.80e+01	0:01:10(2)	1.80e+01

As shown in Table 4, the CGA_ADMM is more efficient than the ADMM and therefore we only list the results of the CGA_ADMM for large scale data sets. We first square the objective function and reformulate it as a quadratic programming problem, then we apply Dykstra's projection method and Gurobi. All of the following experiments achieve the specified accuracy requirements unless it runs out of memory.

We can see from Table 5 that the CGA_ADMM can solve the ℓ_2 norm based metric nearness problem efficiently. It can obtain a desired solution with the given accuracy and the number of iterations of the constraint generation is less than 5. The constraint generation strategy and the efficiency of the ADMM guarantee the high efficiency of our algorithm.

TABLE 5. The performances of Dykstra's projection method, CGA_ADMM and CGA_Gurobi for the ℓ_2 norm based metric nearness problem.

Graph	Dykstra		CGA_ADMM		CGA_Gurobi	
	time	obj	time(iter)	obj	time(iter)	obj
minnesota	0:01:24	1.90e+01	0:00:23(3)	1.90e+01	0:00:11(3)	1.89e+01
caGrQc	0:14:43	3.65e+01	0:02:56(3)	3.65e+01	out of memory	
power	0:16:46	2.92e+01	0:01:54(3)	2.92e+01	0:01:16(3)	2.91e+01
caHepTh	2:48:32	5.68e+01	0:14:33(3)	5.68e+01	out of memory	
caHepPh	8:12:45	8.35e+01	2:06:05(4)	8.32e+01	out of memory	
caAstroPh	66:43:53	1.20e+02	3:37:07(3)	1.20e+02	out of memory	

4.3. **Numerical experiments for the ℓ_∞ norm based metric nearness problem.**

In this section we present the numerical experiments for the ℓ_∞ norm based metric nearness problem. All of the following experiments achieve the specified accuracy requirements unless being out of memory.

In order to apply Dykstra's projection method and Gurobi to solve the ℓ_∞ norm based metric nearness problem, we first rewrite it as a linear programming problem. By introducing slack variables, we can transform the ℓ_∞ norm based metric nearness problem as the following optimization problem.

$$\min_{z\in\mathscr{R}^{n_1+1}} \left\{ \langle \widehat{c}, z\rangle \;\middle|\; \widehat{A}z \le \widehat{b} \right\}, \tag{4.3}$$

where $\widehat{c} = [\mathbf{0}_{n_1}; 1]$ and

$$\widehat{A} = \begin{bmatrix} A & \mathbf{0}_{n_2} \\ \mathrm{Diag}(\mathrm{trivec}(\overline{W})) & -\mathbf{1}_{n_1} \\ -\mathrm{Diag}(\mathrm{trivec}(\overline{W})) & -\mathbf{1}_{n_1} \end{bmatrix}, \quad \widehat{b} = \begin{bmatrix} b \\ \mathbf{0}_{n_1} \\ \mathbf{0}_{n_1} \end{bmatrix},$$

and $\mathbf{1}_{n_1}$ is an n_1-dimensional column vector with all elements 1.

Therefore, we can define the corresponding Lagrangian function and the relative KKT residual in a similar way to that of problem (4.1). The constraint generation algorithm can also be applied with each subproblem solved by the Gurobi package. The stopping criterion is the same as that used in the ℓ_1 norm based problem.

Similar to the approach for problem (4.1), an extra proximal term is added to the objective function when we apply Dykstra's projection method to solve (4.3). The corresponding problem takes the following form

$$\min_{z \in \mathscr{R}^{n_1+1}} \left\{ \langle \widehat{c}, z \rangle + \frac{1}{2\gamma} z^T \widehat{W} z \;\middle|\; \widehat{A} z \leq \widehat{b} \right\}, \tag{4.4}$$

where $\widehat{W}$ is a diagonal matrix defined as $\widehat{W} := \mathrm{Diag}([\mathrm{trivec}(\overline{W}); 1])$.

TABLE 6. The performances of the ADMM, CGA_ADMM, Gurobi, CGA_Gurobi for the ℓ_∞ norm based metric nearness problem.

Graph	ADMM		CGA_ADMM		Gurobi		CGA_Gurobi	
	time	obj	time(iter)	obj	time	obj	time(iter)	obj
jazz	0:23:02	7.52e-02	0:02:32(3)	7.53e-02	0:05:30	7.45e-02	0:03:03(3)	7.50e-02
SmallW	0:17:31	8.33e-02	0:01:58(2)	8.32e-02	0:08:41	8.29e-02	0:02:17(2)	8.31e-02
celegansneural	0:58:37	7.62e-02	0:08:03(2)	7.59e-02	0:23:43	7.57e-02	0:07:32(2)	7.60e-02
USAir97	0:53:35	8.36e-02	0:27:07(3)	8.35e-02	0:39:49	8.30e-02	0:14:21(3)	8.33e-02

We first test the efficiency of the constraint generation algorithm for the ADMM and Gurobi in some small scale data sets. The results in Table 6 show that the constraint generation based algorithms are much more efficient. Then in the following implementation, the constraint generation versions are used for further comparisons.

We implement the Gurobi package which is based on the same constraint generation strategy of the ℓ_1 norm based metric nearness problem to solve the corresponding linear programming problem with the related relative KKT residual small enough as the stopping criterion. Similarly, an extra proximal term with a parameter γ is added to the objective function when we apply Dykstra's projection method. In order to get an appropriate value of parameter γ of the corresponding quadratic programming problem so that the approximate optimal objective value is not larger than 1.5 times of the actual value, we test a list of values of γ and obtain the corresponding objective values of problem (2.1) in some data sets. We report the objective value obtained by the Gurobi package (Gurobi_obj), γ, the objective value obtained by Dykstra's projection method (obj), the ratio of these objective values (ratio), the number of iterations (iter) and the running time (time) with the format of "seconds" for Dykstra's projection method. The results are listed in Table 7.

Table 7 shows that we need to choose a relatively larger value of γ (at least 500) so that the objective value obtained by Dykstra's projection method is relatively close to the actual optimal value. However, the number of iterations grows rapidly when γ increasing and so does the time

TABLE 7. The performances of Dykstra's projection method for the ℓ_∞ norm based metric nearness problem with different γ.

Graph	Gurobi_obj	γ	obj	ratio	iter	time(s)
Harvard500 $\lvert V\rvert = 500$ $\lvert E\rvert = 2043$	8.23e-02	1	7.00e-01	849.74%	55	9
		10	5.77e-01	700.62%	115	19
		100	3.97e-01	482.19%	1555	257
		200	3.20e-01	388.40%	3175	490
		500	2.60e-01	316.23%	10560	1539
email $\lvert V\rvert = 1133$ $\lvert E\rvert = 5451$	8.29e-02	1	3.02e-01	364.62%	40	43
		10	2.59e-01	312.34%	165	174
		100	2.02e-01	244.12%	1460	1537
		200	1.76e-01	212.76%	4530	4678
		500	1.44e-01	174.11%	15575	15863

TABLE 8. The performances of Dykstra's projection method, CGA_ADMM and Gurobi for the ℓ_∞ norm based metric nearness problem.

Graph	Dykstra		CGA_ADMM		CGA_Gurobi	
	time_per_iter(s)	estimated_time	time(iter)	obj	time(iter)	obj
minnesota	3.71	10h	0:01:09(4)	8.33e-02	0:00:30(4)	8.31e-2
caGrQc	16.29	45h	1:16:11(4)	8.40e-02	out of memory	
power	25.23	70h	0:01:43(1)	8.31e-02	0:00:38(1)	8.31e-2
caHepTh	158.87	441h	0:54:13(1)	8.42e-02	out of memory	

cost, which limits the application of Dykstra's projection method for the large scale ℓ_∞ norm based metric nearness problem. In the following comparison, we set $\gamma = 500$. We first list the time cost of one iteration (time_per_iter) of Dykstra's projection method and the results are shown in the second column of Table 8. Based on Table 7, although the actual value may be larger than 10000, we assume that the total number of iterations is 10000 and list the estimated time (estimated_time) of Dykstra's projection method.

The implementation results of our algorithm, Dykstra's projection method and the CGA_Gurobi are listed in Table 8. The comparisons show that the Gurobi package runs out of memory when the number of vertices is large, but our algorithm can solve the large scale ℓ_∞ norm based metric nearness problem efficiently. The number of the iterations of the constraint generation is small (not greater than 5) and the constraints involved in each iteration is much less than the original problem. We only need to implement a few iterations of constraint generation, hence our algorithm is very efficient. It seems that the ℓ_∞ norm based metric nearness problem is much more difficult and challenging than the other two problems. As we know, it is the first time that numerical implementations and experiments for the ℓ_∞ based metric nearness problem are conducted with n greater than 10^4. The maximal memory requirement for the caAstroPh data is about 20GB for the ℓ_1 and ℓ_2 norm based problems, and 50GB for the ℓ_∞ norm based problem.

5. Conclusion

In this paper, we introduced a constraint generation algorithm based ADMM, CGA_ADMM to solve the ℓ_p ($p = 1, 2, \infty$) norm based metric nearness problem. We developed a constraint generation strategy which balances the time cost of each iteration and the number of the iterations to make it possible for solving large scale metric nearness problems efficiently. We take full advantage of the special structure of the problem and make the memory requirement acceptable. Several numerical experiments on real data sets were implemented. The results demonstrate that our algorithm is much more efficient than the current state-of-the-art algorithms.

Acknowledgments
Chengjing Wang's work was supported in part by the National Natural Science Foundation of China (No. U21A20169) and Zhejiang Provincial Natural Science Foundation of China (Grant No. LTGY23H240002).

References

[1] S. Baraty, A. S. Dan, C. Zara, The impact of triangular inequality violations on medoid-based clustering, In: Foundations of Intelligent Systems-19th International Symposium, Warsaw, 2011.
[2] F. Wang, J. M. Sun, Survey on distance metric learning and dimensionality reduction in data mining, Data Mining and Knowledge Discovery, 29 (2015), 534–564.
[3] I.S. Dhillon, S. Sra, J.A Tropp, The metric nearness problems with applications, Department of Computer Sciences Technical Report TR-03-23, The University of Texas at Austin, 2003.
[4] A. C Gilbert, L. Jain, If it ain't broke, don't fix it: Sparse metric repai, In: 2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton), 2017.
[5] C. L. Fan, B. Raichel, G. V. Buskirk, Metric violation distance: Hardness and approximation, Algorithmica, 84 (2022), 1441-1465.
[6] C. L. Fan, A. C. Gilbert, B. Raichel, R. Sonthalia, G. V. Buskirk, Generalized metric repair on graphs, Leibniz International Proceedings in Informatics, LIPIcs, 2020. DOI: 10.4230/LIPIcs.SWAT.2020.25
[7] V. Cohen-Addad, D. Das, E. Kipouridis, N. Parotsidis, M. Thorup, Fitting distances by tree metrics minimizing the total error within a constant factor, In: 2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS), pages 468-479, 2022.
[8] L. L. Cavalli-Sforza, A. W. F. Edwards, Phylogenetic analysis. models and estimation procedures, The American Journal of Human Genetics, 19 (1967), 233-257.
[9] N.J. Higham, Matrix Nearness Problems and Applications, In: M.J.C. Gover, S. Barnett, (Ed.), Applications of Matrix Theory, Oxford University Press, Oxford, 1989.
[10] N. Bansal, A. Blum, S. Chawla, Correlation clustering, Machine Learning, 56 (2004), 89-113.
[11] N. Veldt, D.F. Gleich, A. Wirth, J. Saunderson, Metric-constrained optimization for graph clustering algorithms, SIAM J. Math. Data Sci. 1 (2019), 333-355.
[12] S. Sra, J. Tropp, I.S. Dhillon, Triangle fixing algorithms for the metric nearness problem, Advances in Neural Information Processing Systems, 17 (2005), 361-368.
[13] J. Brickell, I.S. Dhillon, S. Sra, J.A. Tropp, The metric nearness problem, SIAM J. Matrix Anal. Appl. 30 (2008), 375-396.
[14] C. Ruggles, N. Veldt, D. F Gleich, A parallel projection method for metric constrained optimization, In: 2020 Proceedings of the SIAM Workshop on Combinatorial Scientific Computing, pp. 43–53, SIAM, 2020.
[15] R. Escalante, M. Raydan, Alternating Projection Methods, SIAM, Philadelphia, PA, 2011.
[16] R. Sonthalia, A.C. Gilbert, Project and forget: Solving large-scale metric constrained problems, arXiv preprint arXiv:2005.03853, 2020.

[17] D. Gabay, B. Mercier, A dual algorithm for the solution of nonlinear variational problems via finite element approximation, Comput. Math. Appl. 2 (1976), 17-40.

[18] R. Glowinski, Lectures on Numerical Methods for Non-Linear Variational Problems, Tata Institute of Fundamental Research 1980, Springer, Berlin, 1980.

[19] J. Eckstein, W. Yao, Understanding the convergence of the alternating direction method of multipliers: Theoretical and computational perspectives, Pacific J. Optim. 11 (2015), 619-644.

[20] D. Bertsimas, J. N. Tsitsiklis, Introduction to Linear Optimization, Athena Scientific Belmont, MA, 1997.

[21] M. X. Lin, D. F. Sun, K.C. Toh, An augmented lagrangian method with constraint generation for shape-constrained convex regression problems, Math. Program. Comput. 14 (2022), 223–270.

[22] R. T. Rockafellar, Convex Analysis, Princeton University Press, Princeton, 1970.

[23] M. Fornasier, H. Rauhut, Recovery algorithms for vector-valued data with joint sparsity constraints, SIAM J. Numer. Anal. 46 (2008), 577-613.

[24] G. Golub, C. F. Van Loan, Matrix Computationsm, 3nd Edition, Johns Hopkins University Press, Baltimore, MD, 1996.

[25] R. Glowinski, A, Marroco, Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de Dirichlet non linéaires, Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique, 9 (1975), 41-76.

[26] L. Chen, D. F. Sun, K.-C. Toh, An efficient inexact symmetric gauss-seidel based majorized admm for high-dimensional convex composite conic programming, Math. Program. 161 (2017), 237-270.

[27] T. A. Davis, Y. Hu, The university of florida sparse matrix collection, ACM Trans. Math. Softw. 38 (2011), 1-25.

[28] O. L. Mangasarian, Normal solutions of linear programs, In: Mathematical Programming at Oberwolfach II, Mathematical Programming Studies 22, pp. 206-216. Springer, Berlin, Heidelberg, 1984.

APPENDIX

A. PROOF OF THEOREM 3.1

The proof of Theorem 3.1 is a special case of that in [26], we give a brief proof for completeness.

Proof. To prove the result, we rewrite primal problem (2.2) as the following form

$$\begin{aligned} \min_{\substack{\zeta\in\mathscr{R}^{n_1+n_2}\\ y\in\mathscr{R}^{n_1}}} \quad & H(\zeta)+F(y) \\ \text{s.t.} \quad & \mathscr{A}\zeta+\mathscr{B}y=c, \end{aligned} \tag{A.1}$$

with $\zeta=(\xi,\eta)$, $H(\zeta)=h(\xi)+q(\eta)$, $F(y)=0$,

$$\mathscr{A}=\begin{bmatrix} -I_{n_2} & O_{n_2\times n_1} \\ O_{n_1\times n_2} & -I_{n_1} \end{bmatrix},\quad \mathscr{B}=\begin{bmatrix} A \\ W \end{bmatrix},\quad c=\begin{bmatrix} b \\ \mathbf{0}_{n_1} \end{bmatrix},$$

and $O_{n_2\times n_1}\in\mathscr{R}^{n_2\times n_1}$ a zero matrix and $\mathbf{0}_{n_1}\in\mathscr{R}^{n_1}$ a zero vector.

Due to the assumption that all elements of the weight matrix $\overline{W}$ are positive, we can easily obtain that $\mathscr{A}^*\mathscr{A}$ and $\mathscr{B}^*\mathscr{B}$ are positive definite. It follows from Theorem 5.1 of [26] that $\{(y^k,\xi^k,\eta^k)\}$ converges to an optimal solution of primal problem (2.2) and $\{(u^k,v^k)\}$ converges to an optimal solution of dual problem (2.3). $\square$

B. PROOF OF THEOREM 3.2

In this section, we present a proof for Theorem 3.2 in order to be self-contained. Although both the theorem and the proof can be regarded as a special case of those in [26], we deal with a special problem in this paper, thus we can have an individualized estimation of the bound. Denote $R_1(\xi,u):=\partial h(\xi)-u$, $R_2(\eta,v):=\partial q(\eta)-v$, $R_3(u,v):=A^Tu+W^Tv$, $R_4(\xi,y):=Ay-\xi-b$, $R_5(\eta,y):=Wy-\eta$. Let $\tau=(1+\rho/\min\{1+\rho,1+\rho^{-1}\})/2$, $\hat{\tau}=1-\tau\min\{\rho,\rho^{-1}\}$, $\iota=\min\{1,1-\rho+\rho^{-1}\}\tau-(1-\tau)\rho$ and $\mathscr{E}:=\sum_{k=0}^{\infty}\varepsilon^k$, $\mathscr{E}':=\sum_{k=0}^{\infty}(\varepsilon^k)^2$. For $x,y\in\mathscr{R}^n$ and a self-adjoint positive semidefinite matrix M of order n, define $\langle x,y\rangle_M:=\langle x,My\rangle$ and $\|x\|_M:=\sqrt{\langle x,Mx\rangle}=\|M^{\frac12}x\|$.

Lemma B.1. *Suppose the same assumption as that of Theorem 3.1 holds. Let $\{(\xi^k,\eta^k,y^k,u^k,v^k)\}$ be the sequence generated by the ADMM and $(\bar{\xi},\bar{\eta},\bar{y},\bar{u},\bar{v})$ be a solution of the corresponding KKT system. Then, for $k\geq 1$,*

$$\begin{aligned} &2\tau\langle d^k-d^{k-1},y^k-y^{k+1}\rangle-2\langle d^k,y^{k+1}-\bar{y}\rangle+\frac{(1-\tau)\sigma}{2}(\|\xi^k-\xi^{k+1}\|^2+\|\eta^k-\eta^{k+1}\|^2)\\ &\quad+\tau\min\{\rho,1+\rho-\rho^2\}\|y^k-y^{k+1}\|^2_{\mathscr{M}}+\iota\sigma(\|R_4(\xi^{k+1},y^{k+1})\|^2+\|R_5(\eta^{k+1},y^{k+1})\|^2)\\ &\leq\varphi_k(\bar{\xi},\bar{\eta},\bar{u},\bar{v})-\varphi_{k+1}(\bar{\xi},\bar{\eta},\bar{u},\bar{v}), \end{aligned}$$

where

$$\begin{aligned} \varphi_k(\xi,\eta,u,v) \;:=\; & \frac{1}{\rho\sigma}(\|u-u^k\|^2+\|v-v^k\|^2)+\sigma(\|R_4(\xi,y^k)\|^2+\|R_5(\eta,y^k)\|^2)\\ &+\hat{\tau}\sigma(\|R_4(\xi^k,y^k)\|^2+\|R_5(\eta^k,y^k)\|^2). \end{aligned}$$

Proof. From the optimality condition of the subproblem for the ADMM, we know

$$
\begin{aligned}
&0 \in \partial h(\xi^{k+1}) - \sigma(Ay^k - \xi^{k+1} - b + \sigma^{-1}u^k), && \text{(B.1)}\\
&0 \in \partial q(\eta^{k+1}) - \sigma(Wy^k - \eta^{k+1} + \sigma^{-1}v^k), && \text{(B.2)}\\
&d^k = \sigma W^T(Wy^{k+1} - \eta^{k+1} + \sigma^{-1}v^k) + \sigma A^T(Ay^{k+1} - \xi^{k+1} - b + \sigma^{-1}u^k).
\end{aligned}
$$

Hence

$$
\langle d^k - A^T(u^k + \sigma R_4(\xi^{k+1}, y^{k+1})) - W^T(v^k + \sigma R_5(\eta^{k+1}, y^{k+1})), y^{k+1} - \bar{y}\rangle = 0. \tag{B.3}
$$

By (B.1), (B.2) and the convexity of h and q, we have

$$
\begin{aligned}
h(\bar{\xi}) + \langle u^k + \sigma R_4(\xi^{k+1}, y^{k+1}) + \sigma A(y^k - y^{k+1}), \xi^{k+1} - \bar{\xi}\rangle &\geq h(\xi^{k+1}), && \text{(B.4)}\\
q(\bar{\eta}) + \langle v^k + \sigma R_5(\eta^{k+1}, y^{k+1}) + \sigma W(y^k - y^{k+1}), \eta^{k+1} - \bar{\eta}\rangle &\geq q(\eta^{k+1}). && \text{(B.5)}
\end{aligned}
$$

Furthermore, the optimal solution $(\bar{\xi}, \bar{\eta}, \bar{y}, \bar{u}, \bar{v})$ satisfies

$$
0 \in \partial h(\bar{\xi}) - \bar{u},\ 0 \in \partial q(\bar{\eta}) - \bar{v},\ A^T\bar{u} + W^T\bar{v} = 0,\ A\bar{y} - \bar{\xi} - b = 0,\ W\bar{y} - \bar{\eta} = 0.
$$

Therefore, by the convexity of h and q we obtain

$$
\begin{aligned}
&h(\xi^{k+1}) \geq h(\bar{\xi}) + \langle \bar{u}, \xi^{k+1} - \bar{\xi}\rangle, && \text{(B.6)}\\
&q(\eta^{k+1}) \geq g(\bar{\eta}) + \langle \bar{v}, \eta^{k+1} - \bar{\eta}\rangle, && \text{(B.7)}\\
&\langle A^T\bar{u} + W^T\bar{v}, y^{k+1} - \bar{y}\rangle = 0. && \text{(B.8)}
\end{aligned}
$$

Combing the above results (B.3)–(B.8), we obtain

$$
\begin{aligned}
&\langle u^k - \bar{u} + \sigma R_4(\xi^{k+1}, y^{k+1}), \xi^{k+1} - \bar{\xi}\rangle + \langle v^k - \bar{v} + \sigma R_5(\eta^{k+1}, y^{k+1}), \eta^{k+1} - \bar{\eta}\rangle\\
&\quad + \sigma\langle A^T(\xi^{k+1} - \bar{\xi}) + W^T(\eta^{k+1} - \bar{\eta}), y^k - y^{k+1}\rangle \geq 0, && \text{(B.9)}\\
&\langle d^k, y^{k+1} - \bar{y}\rangle - \langle u^k - \bar{u} + \sigma R_4(\xi^{k+1}, y^{k+1}), Ay^{k+1} - A\bar{y}\rangle\\
&\quad - \langle v^k - \bar{v} + \sigma R_5(\eta^{k+1}, y^{k+1}), Wy^{k+1} - W\bar{y}\rangle = 0. && \text{(B.10)}
\end{aligned}
$$

By adding (B.9) with (B.10), it follows that

$$
\begin{aligned}
&\langle d^k, y^{k+1} - \bar{y}\rangle - \langle u^k - \bar{u} + \sigma R_4(\xi^{k+1}, y^{k+1}), R_4(\xi^{k+1}, y^{k+1})\rangle\\
&\quad - \langle v^k - \bar{v} + \sigma R_5(\eta^{k+1}, y^{k+1}), R_5(\eta^{k+1}, y^{k+1})\rangle + \sigma\langle A(y^k - y^{k+1}), \xi^{k+1} - \bar{\xi}\rangle\\
&\quad + \sigma\langle W(y^k - y^{k+1}), \eta^{k+1} - \bar{\eta}\rangle \geq 0. && \text{(B.11)}
\end{aligned}
$$

Since

$$
\begin{aligned}
&\langle A(y^k - y^{k+1}, \xi^{k+1} - \bar{\xi})\rangle \\
&= \langle A(y^k - \bar{y}) - A(y^{k+1} - \bar{y}), A(y^{k+1} - \bar{y}) - R_4(\xi^{k+1}, y^{k+1})\rangle \\
&= \frac{1}{2}\left(\|A(y^k - \bar{y})\|^2 - \|A(y^k - \bar{y}) - A(y^{k+1} - \bar{y})\|^2 - \|A(y^{k+1} - \bar{y})\|^2\right) \\
&\quad - \langle A(y^k - \bar{y}) - A(y^{k+1} - \bar{y}), R_4(\xi^{k+1}, y^{k+1})\rangle \\
&= \frac{1}{2}\left(\|A(y^k - \bar{y})\|^2 - \|A(y^{k+1} - \bar{y})\|^2 + \|R_4(\xi^{k+1}, y^{k+1})\|^2 - |R_4(\xi^{k+1}, y^k)\|^2\right), \\
&\langle W(y^k - y^{k+1}, \eta^{k+1} - \bar{\eta})\rangle \\
&= \langle W(y^k - \bar{y}) - W(y^{k+1} - \bar{y}), W(y^{k+1} - \bar{y}) - R_5(\eta^{k+1}, y^{k+1})\rangle \\
&= \frac{1}{2}\left(\|W(y^k - \bar{y})\|^2 - \|W(y^k - \bar{y}) - W(y^{k+1} - \bar{y})\|^2 - \|W(y^{k+1} - \bar{y})\|^2\right) \\
&\quad - \langle W(y^k - \bar{y}) - W(y^{k+1} - \bar{y}), R_5(\eta^{k+1}, y^{k+1})\rangle \\
&= \frac{1}{2}\left(\|W(y^k - \bar{y})\|^2 - \|W(y^{k+1} - \bar{y})\|^2 + \|R_5(\eta^{k+1}, y^{k+1})\|^2 - \|R_5(\eta^{k+1}, y^k)\|^2\right), \\
&\langle u^k - \bar{u} + \sigma R_4(\xi^{k+1}, y^{k+1}), R_4(\xi^{k+1}, y^{k+1})\rangle \\
&= \frac{1}{\rho\sigma}\langle (u^{k+1} - \bar{u}) - (u^k - \bar{u}), u^k - \bar{u}\rangle + \sigma\|R_4(\xi^{k+1}, y^{k+1})\|^2 \\
&= \frac{1}{2\rho\sigma}\left(\|u^{k+1} - \bar{u}\|^2 - \|u^{k+1} - u^k\|^2 - \|u^k - \bar{u}\|^2\right) + \sigma\|R_4(\xi^{k+1}, y^{k+1})\|^2 \\
&= \frac{1}{2\rho\sigma}\left(\|u^{k+1} - \bar{u}\|^2 - \|u^k - \bar{u}\|^2\right) + \frac{(2-\rho)\sigma}{2}\|R_4(\xi^{k+1}, y^{k+1})\|^2
\end{aligned}
$$

and

$$
\begin{aligned}
&\langle v^k - \bar{v} + \sigma R_5(\eta^{k+1}, y^{k+1}), R_5(\eta^{k+1}, y^{k+1})\rangle \\
&= \frac{1}{\rho\sigma}\langle (v^{k+1} - \bar{v}) - (v^k - \bar{v}), v^k - \bar{v}\rangle + \sigma\|R_5(\eta^{k+1}, y^{k+1})\|^2 \\
&= \frac{1}{2\rho\sigma}\left(\|v^{k+1} - \bar{v}\|^2 - \|v^{k+1} - v^k\|^2 - \|v^k - \bar{v}\|^2\right) + \sigma\|R_5(\eta^{k+1}, y^{k+1})\|^2 \\
&= \frac{1}{2\rho\sigma}\left(\|v^{k+1} - \bar{v}\|^2 - \|v^k - \bar{v}\|^2\right) + \frac{(2-\rho)\sigma}{2}\|R_5(\eta^{k+1}, y^{k+1})\|^2,
\end{aligned}
$$

we obtain from (B.11) that

$$
\begin{aligned}
&\langle d^k, y^{k+1} - \bar{y}\rangle + \frac{1}{2\rho\sigma}\left(\|u^k - \bar{u}\|^2 - \|u^{k+1} - \bar{u}\|^2 + \|v^k - \bar{v}\|^2 - \|v^{k+1} - \bar{v}\|^2\right) \\
&\qquad + \frac{\sigma}{2}\left(\|A(y^k - \bar{y})\|^2 - \|A(y^{k+1} - \bar{y})\|^2 + \|W(y^k - \bar{y})\|^2 - \|W(y^{k+1} - \bar{y})\|^2\right) \\
&\geq \frac{\sigma}{2}\left(\|R_4(\xi^{k+1}, y^k)\|^2 + \|R_5(\eta^{k+1}, y^k)\|^2\right) \\
&+ \frac{(1-\rho)\sigma}{2}\left(\|R_4(\xi^{k+1}, y^{k+1})\|^2 + \|R_5(\eta^{k+1}, y^{k+1})\|^2\right). \qquad \text{(B.12)}
\end{aligned}
$$

Similar to Lemma 5.2 of [26], we can prove that

$$\begin{aligned}
&(1-\rho)\sigma(\|R_4(\xi^{k+1},y^{k+1})\|^2+\|R_5(\eta^{k+1},y^{k+1})\|^2)\\
&\quad+\sigma(\|R_4(\xi^{k+1},y^k)\|^2+\|R_5(\eta^{k+1},y^k)\|^2)+2\tau\left\langle d^{k-1}-d^k,y^k-y^{k+1}\right\rangle\\
&\geq \hat{\tau}\sigma(\|R_4(\xi^{k+1},y^{k+1})\|^2-\|R_4(\xi^k,y^k)\|^2+\|R_5(\eta^{k+1},y^{k+1})\|^2-\|R_5(\eta^k,y^k)\|^2)\\
&\quad+\iota\sigma(\|R_4(\xi^{k+1},y^{k+1})\|^2+\|R_5(\eta^{k+1},y^{k+1})\|^2)\\
&\quad+\frac{(1-\tau)\sigma}{2}(\|\xi^k-\xi^{k+1}\|^2+\|\eta^k-\eta^{k+1}\|^2)+\min\{\rho,1+\rho-\rho^2\}\tau\|y^k-y^{k+1}\|^2_{\mathscr{M}}. \qquad \text{(B.13)}
\end{aligned}$$

Noticing that

$$R_4(\bar{\xi},y)=A(y-\bar{y}),\quad R_5(\bar{\eta},y)=W(y-\bar{y}),$$

and applying (B.13) to (B.12), the desired result follows. □

Now we give the proof of Theorem 3.2.

Proof. In order to derive the desired result, we first give an upper bound for

$$\sum_{k=1}^{\infty}\Big\{\frac{(1-\tau)\sigma}{2}(\|\xi^k-\xi^{k+1}\|^2+\|\eta^k-\eta^{k+1}\|^2)+\tau\min\{\rho,1+\rho-\rho^2\}\|y^k-y^{k+1}\|^2_{\mathscr{M}}$$
$$+\iota\sigma(\|R_4(\xi^{k+1},y^{k+1})\|^2+\|R_5(\eta^{k+1},y^{k+1})\|^2)\Big\}.$$

Based on the assumption of Theorem 3.1, the solution set of the KKT system is bounded. Therefore the sequence $\{\varphi_k(\bar{\xi},\bar{\eta},\bar{u},\bar{v})\}$ is bounded and there exists a constant $\Lambda>0$ such that $\varphi_k(\bar{\xi},\bar{\eta},\bar{u},\bar{v})\leq\Lambda$ for all $k\geq 1$. Since $\sigma(\|R_4(\bar{\xi},y^k)\|^2+\|R_5(\bar{\eta},y^k)\|^2)=\|y^k-\bar{y}\|^2_{\mathscr{M}}$, we obtain

$$\frac{1}{\rho\sigma}(\|u^k-\bar{u}\|^2+\|v^k-\bar{v}\|^2)+\|y^k-\bar{y}\|^2_{\mathscr{M}}+\hat{\tau}\sigma(\|R_4(\xi^k,y^k)\|^2+\|R_5(\eta^k,y^k)\|^2)\leq\Lambda.$$

Furthermore, we have

$$\|y^k-\bar{y}\|_{\mathscr{M}}\leq\sqrt{\Lambda},\quad \|y^k-\bar{y}\|\leq\|\mathscr{M}^{-\frac{1}{2}}\|\sqrt{\Lambda},\quad \|y^k-y^{k+1}\|\leq 2\|\mathscr{M}^{-\frac{1}{2}}\|\sqrt{\Lambda}.$$

From the fact that $\|d^k\|\leq\|\mathscr{M}^{\frac{1}{2}}\|\varepsilon^k$, we obtain $\|d^k-d^{k-1}\|\leq\|\mathscr{M}^{\frac{1}{2}}\|(\varepsilon^k+\varepsilon^{k-1})$. Therefore by Lemma B.1, we derive

$$\begin{aligned}
&\sum_{k=1}^{\infty}\Big\{\frac{(1-\tau)\sigma}{2}\left(\|\xi^k-\xi^{k+1}\|^2+\|\eta^k-\eta^{k+1}\|^2\right)+\tau\min\{\rho,1+\rho-\rho^2\}\|y^k-y^{k+1}\|^2_{\mathscr{M}}\\
&\quad+\iota\sigma\left(\|R_4(\xi^{k+1},y^{k+1})\|^2+\|R_5(\eta^{k+1},y^{k+1})\|^2\right)\Big\}\\
&\leq\sum_{k=1}^{\infty}(\varphi_k(\bar{\xi},\bar{\eta},\bar{u},\bar{v})-\varphi_{k+1}(\bar{\xi},\bar{\eta},\bar{u},\bar{v}))+\sum_{k=1}^{\infty}\Big\{2\langle d^k,y^{k+1}-\bar{y}\rangle-2\tau\langle d^k-d^{k-1},y^k-y^{k+1}\rangle\Big\} \qquad \text{(B.14)}\\
&\leq\varphi_1(\bar{\xi},\bar{\eta},\bar{u},\bar{v})+2\|\mathscr{M}^{\frac{1}{2}}\|\sqrt{\Lambda}\sum_{k=1}^{\infty}\varepsilon^k+4\tau\|\mathscr{M}^{-\frac{1}{2}}\|\|\mathscr{M}^{\frac{1}{2}}\|\sqrt{\Lambda}\sum_{k=1}^{\infty}(\varepsilon^k+\varepsilon^{k-1})\\
&\leq\varphi_1(\bar{\xi},\bar{\eta},\bar{u},\bar{v})+2\|\mathscr{M}^{\frac{1}{2}}\|\sqrt{\Lambda}(1+4\tau\|\mathscr{M}^{-\frac{1}{2}}\|)\mathscr{E}:=\kappa_1. \qquad \text{(B.15)}
\end{aligned}$$

From the optimality conditions of the subproblem for the ADMM, we know

$$\begin{aligned}
&0 \in \partial h(\xi^{k+1}) - \sigma(Ay^k - \xi^{k+1} - b + \sigma^{-1}u^k),\\
&0 \in \partial q(\eta^{k+1}) - \sigma(Wy^k - \eta^{k+1} + \sigma^{-1}v^k),\\
&d^k = \sigma W^T(Wy^{k+1} - \eta^{k+1} + \sigma^{-1}v^k) + \sigma A^T(Ay^{k+1} - \xi^{k+1} - b + \sigma^{-1}u^k).
\end{aligned}$$

and it follows that

$$\begin{aligned}
\mathrm{dist}^2(0, R_1(\xi^{k+1}, u^{k+1})) &\le \|u^{k+1} - \sigma(Ay^k - \xi^{k+1} - b + \sigma^{-1}u^k)\|^2\\
&= \|\rho\sigma(Ay^{k+1} - \xi^{k+1} - b) - \sigma(Ay^k - \xi^k - b) - \sigma(\xi^k - \xi^{k+1})\|^2\\
&= \sigma^2\|(\xi^k - \xi^{k+1}) - \rho R_4(\xi^{k+1}, y^{k+1}) + R_4(\xi^k, y^k))\|^2\\
&\le 3\sigma^2\left(\|\xi^k - \xi^{k+1}\|^2 + \rho^2\|R_4(\xi^{k+1}, y^{k+1})\|^2 + \|R_4(\xi^k, y^k)\|^2\right).
\end{aligned} \tag{B.16}$$

Similarly, we have

$$\mathrm{dist}^2(0, R_2(\eta^{k+1}, v^{k+1})) \le 3\sigma^2\left(\|\eta^k - \eta^{k+1}\|^2 + \rho^2\|R_5(\eta^{k+1}, y^{k+1})\|^2 + \|R_5(\eta^k, y^k)\|^2\right) \tag{B.17}$$

and

$$\begin{aligned}
&\|R_3(u^{k+1}, v^{k+1})\|^2\\
&= \|\sigma A^T(\rho R_4(\xi^{k+1}, y^{k+1}) - R_4(\xi^k, y^k) - (\xi^k - \xi^{k+1}))\\
&\quad + \sigma W^T(\rho R_5(\eta^{k+1}, y^{k+1}) - R_5(\eta^k, y^k) - (\eta^k - \eta^{k+1})) + \mathscr{M}(y^k - y^{k+1}) + d^k\|^2\\
&\le 8\|\mathscr{M}\|(\|\mathscr{M}^{-\frac{1}{2}}d^k\|^2 + \|y^k - y^{k+1}\|^2_{\mathscr{M}})\\
&\quad + 8\sigma^2\|\mathscr{M}\|(\|\mathscr{M}^{-\frac{1}{2}}A^T\|^2 + \|\mathscr{M}^{-\frac{1}{2}}W^T\|^2)(\|\xi^k - \xi^{k+1}\|^2 + \|\eta^k - \eta^{k+1}\|^2)\\
&\quad + 8\sigma^2\|\mathscr{M}\|(\|\mathscr{M}^{-\frac{1}{2}}A^T\|^2 + \|\mathscr{M}^{-\frac{1}{2}}W^T\|^2)(\rho^2\|R_4(\xi^{k+1}, y^{k+1})\|^2 + \|R_4(\xi^k, y^k)\|^2\\
&\qquad + \rho^2\|R_5(\eta^{k+1}, y^{k+1})\|^2 + \|R_5(\eta^k, y^k)\|^2).
\end{aligned} \tag{B.18}$$

Combing (B.15)–(B.18) and $\|\mathscr{M}^{-\frac{1}{2}}d^k\| \le \varepsilon^k$, we obtain

$$\begin{aligned}
&\sum_{k=1}^{\infty} r(\xi^{k+1}, \eta^{k+1}, y^{k+1}, u^{k+1}, v^{k+1})\\
&\le \kappa_2 \sum_{k=1}^{\infty}\left\{\frac{(1-\tau)\sigma}{2}\left(\|\xi^k - \xi^{k+1}\|^2 + \|\eta^k - \eta^{k+1}\|^2\right) + \tau\min\{\rho, 1 + \rho - \rho^2\}\|y^k - y^{k+1}\|^2_{\mathscr{M}}\right.\\
&\quad \left. + \iota\sigma\left(\|R_4(\xi^{k+1}, y^{k+1})\|^2 + \|R_5(\eta^{k+1}, y^{k+1})\|^2\right)\right\} + 8\|\mathscr{M}\|\sum_{k=1}^{\infty}(\varepsilon^k)^2\\
&\quad + (3\sigma^2 + 8\sigma^2\|\mathscr{M}\|(\|\mathscr{M}^{-\frac{1}{2}}A^T\|^2 + \|\mathscr{M}^{-\frac{1}{2}}W^T\|^2))(\|R_4(\xi^1, y^1)\|^2 + \|R_5(\eta^1, y^1)\|^2)\\
&\le \kappa := (3\sigma^2 + 8\sigma^2\|\mathscr{M}\|(\|\mathscr{M}^{-\frac{1}{2}}A^T\|^2 + \|\mathscr{M}^{-\frac{1}{2}}W^T\|^2))(\|R_4(\xi^1, y^1)\|^2 + \|R_5(\eta^1, y^1)\|^2)\\
&\quad + \kappa_1\kappa_2 + 8\|\mathscr{M}\|\mathscr{E}',
\end{aligned}$$

where

$$\kappa_2 := \max\left\{\frac{8\|\mathscr{M}\|}{\tau\min\{\rho, 1+\rho-\rho^2\}}, \frac{2\sigma}{1-\tau}(3+8\|\mathscr{M}\|(\|\mathscr{M}^{-\frac{1}{2}}A^T\|^2+\|\mathscr{M}^{-\frac{1}{2}}W^T\|^2)),\right.$$
$$\left.\frac{\sigma^2(3+8\|\mathscr{M}\|(\|\mathscr{M}^{-\frac{1}{2}}A^T\|^2+\|\mathscr{M}^{-\frac{1}{2}}W^T\|^2))(1+\rho^2)+1}{\iota\sigma}\right\}.$$

Due to Lemma 6.1 of [26], the desired result follows. □