Greedy coordinate descent method on non-negative quadratic programming

Chenyu Wu
Department of Mathematical Sciences
Rensselaer Polytechnic Institute
Troy, NY 12180, USA
wuc10@rpi.edu

Yangyang Xu
Department of Mathematical Sciences
Rensselaer Polytechnic Institute
Troy, NY 12180, USA
xuy21@rpi.edu

***Abstract*—The coordinate descent (CD) method has recently become popular for solving very large-scale problems, partly due to its simple update, low memory requirement, and fast convergence. In this paper, we explore the greedy CD on solving non-negative quadratic programming (NQP). The greedy CD generally has much more expensive per-update complexity than its cyclic and randomized counterparts. However, on the NQP, these three CDs have almost the same per-update cost, while the greedy CD can have significantly faster overall convergence speed. We also apply the proposed greedy CD as a subroutine to solve linearly constrained NQP and the non-negative matrix factorization. Promising numerical results on both problems are observed on instances with synthetic data and also image data.**

***Index Terms*—greedy coordinate descent, quadratic programming, nonnegative matrix factorization**

I. Introduction

The coordinate descent (CD) method is one of the most classic iterative methods for solving optimization problems. It dates back to 1950s [6] and is closely related to the Jacobi and Gauss-Seidel methods for solving a linear system. Compared to a full-update method, such as the gradient descent and the Newton's method, the coordinate update is simpler and cheaper, and also the CD method has lower memory requirement. Partly due to this reason, the CD method and its variants (such as coordinate gradient descent) have recently become particularly popular for solving very large-scale problems, under both convex and nonconvex settings (e.g., see [3], [4], [7], [17], [22], [25], [29], [33], [35], [38]–[40]). Roughly speaking, the CD method, at each iteration, picks one (by a certain rule) out of possibly many coordinates and then updates it (in a certain way) to decrease the objective value.

Early works (e.g., [6], [16], [32]) on CD chose the coordinates cyclicly, or greedily such that the change to the variable or the decrease of the objective value is maximized [2]. Since the pioneering work [17] that introduces random selection of the updated coordinate, many recent works focus on randomized CD methods (e.g., [14], [15], [22], [26], [36]). Theoretically, the randomized CD can have faster convergence than the cyclic CD [30]. Computationally, the greedy CD is generally more expensive than the randomized and cyclic CD, namely, the latter two can be coordinate-friendly (CF) [21] while the greedy one may fail to. However, for some special-structured problems such as the ℓ_1 minimization, the greedy CD is CF and can be faster than both the randomized and cyclic CD (e.g., [12], [18], [19], [23]) in theory and practice.

In this paper, we first explore the greedy CD to the non-negative quadratic programming (NQP). Similar to the cyclic and randomized CD methods, we show that the greedy CD is also CF for solving the NQP, by maintaining the full gradient of the objective and renewing it with $O(1)$ flops after each coordinate-update. Numerically, we demonstrate that the greedy CD can be significantly faster than the cyclic and randomized counterparts. We then apply the greedy CD as a subroutine in the framework of the augmented Lagrangian method (ALM) for the linearly constrained NQP and in the framework of the alternating minimization for the nonnegative matrix factorization (NMF) [11], [20]. On both problems with synthetic and/or real-world data, we observe promising numerical performance of the proposed methods.

Notation. The i-th component of a vector $\mathbf{x}$ is denoted as x_i. $\mathbf{x}_{<i}$ denotes the subvector of $\mathbf{x}$ of all components with indices less than i and $\mathbf{x}_{>i}$ with indices greater than i. Given a matrix $\mathbf{P}$, we use $\mathbf{p}_i$ for its i-th column and $\mathbf{p}_{i:}$ for its i-th row. For a twice differentiable function f on $\mathbb{R}^n$, $\nabla_i f$ denotes the partial derivative about the i-th variable and $\nabla_i^2 f$ for the second-order partial derivative. $[n]$ represents the set $\{1,\dots,n\}$.

II. Greedy Coordinate Descent method

We first briefly introduce the greedy CD method on a general coordinate-constrained optimization problem and then show the details on how to apply it to the NQP.

Let $f:\mathbb{R}^n\to\mathbb{R}$ be a differentiable function and for each $i\in[n]$, $X_i\subseteq\mathbb{R}$ be a closed convex set. The greedy CD for solving $\min_{\mathbf{x}\in\mathbb{R}^n}\left\{f(\mathbf{x}) : x_i\in X_i, \forall i\in[n]\right\}$ iteratively performs the update: $x_i^{k+1}=x_i^k$ if $i\neq i_k$, and

$$x_i^{k+1}=\arg\min_{x_i\in X_i} f(\mathbf{x}_{<i}^k, x_i, \mathbf{x}_{>i}^k),\ \text{if } i=i_k, \tag{1}$$

where i_k is selected greedily by

$$i_k=\arg\min_{i\in[n]}\left\{\min_{x_i\in X_i} f\left(\mathbf{x}_{<i}^k, x_i, \mathbf{x}_{>i}^k\right)\right\}. \tag{2}$$

Notice that here we follow [2] and greedily choose i_k based on the objective value. In the literature, there are a few other

This work in partly supported by the NSF grant DMS-1719549.

ways to greedily choose i_k, based on the magnitude of partial derivatives or the change of coordinate update. We refer the readers to the review paper [29].

In general, to choose i_k by (2), we need to solve n one-dimensional minimization problem, and thus the per-update complexity of the greedy CD can be as high as n times of that of a cyclic or randomized CD. Similar to [12] that considers the LASSO, we show that the greedy CD on the NQP can have similar per-update cost as the cyclic and randomized CD.

A. Greedy CD on the NQP

Now we derive the details on how to apply the aforementioned greedy CD on the following NQP:

$$\min_{\mathbf{x}\geq\mathbf{0}} \ F(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^\top\mathbf{P}\mathbf{x} + \mathbf{d}^\top\mathbf{x}. \tag{3}$$

Here, $\mathbf{P} \in \mathbb{R}^{n\times n}$ is a given symmetric positive semidefinite (PSD) matrix, and $\mathbf{d} \in \mathbb{R}^n$ is given. To have well-defined coordinate updates, we assume $P_{ii} > 0, \forall\, i \in [n]$. The work [31] has studied a greedy block CD on *unconstrained* QPs, and it requires the matrix $\mathbf{P}$ to be positive definite. Hence, our setting is more general, and more importantly, our algorithm can be used as a subroutine to solve a larger class of problems such as the linear equality-constrained NQP and the NMF, discussed in sections III and IV, respectively. We emphasize that our discussion on the greedy CD may be extended to other applications with separable non-smooth regularizers.

Suppose that the value of the k-th iterate is $\mathbf{x}^k$. We define $G_i^{(k)}(x_i) = F(\mathbf{x}^k_{<i}, x_i, \mathbf{x}^k_{>i})$. Since F is a quadratic function, by the Taylor expansion, it holds

$$G_i^{(k)}(x_i) = F(\mathbf{x}^k) + \nabla_i F(\mathbf{x}^k)(x_i - x_i^k) + \tfrac{P_{ii}}{2}(x_i - x_i^k)^2. \tag{4}$$

Let $\hat{x}_i^k$ be the minimizer of $G_i^{(k)}(x_i)$ over $x_i \geq 0$. Then

$$\hat{x}_i^k = \max\left(0,\ x_i^k - \tfrac{\nabla_i F(\mathbf{x}^k)}{P_{ii}}\right), \forall\, i \in [n], \tag{5}$$

and thus the best coordinate by the rule in (2) is

$$i_k = \arg\min_{i\in[n]} \left\{\nabla_i F(\mathbf{x}^k)(\hat{x}_i^k - x_i^k) + \tfrac{P_{ii}}{2}(\hat{x}_i^k - x_i^k)^2\right\}. \tag{6}$$

Notice that the most expensive part in (5) lies in computing $\nabla_i F(\mathbf{x}^k)$, which takes $O(n)$ flops. Hence, to obtain the best i_k and thus a new iterate $\mathbf{x}^{k+1}$, it costs $O(n^2)$. Therefore, performing n coordinate updates will cost $O(n^3)$, which is order of magnitude larger than the per-update cost $O(n^2)$ by the gradient descent method. However, this naive implementation does not exploit the coordinate update, i.e., any two consecutive iterates differ at most at one coordinate. Using this fact, we show below that the greedy CD method on solving (3) can be CF [21] by maintaining a full gradient, i.e., n coordinate updates cost similarly as a gradient descent update.

Let $\mathbf{g}^k = \nabla F(\mathbf{x}^k)$. Provided that $\mathbf{g}^k$ is stored in the memory, we only need $O(1)$ flops to have $\hat{x}_i^k$ defined in (5), and thus to obtain the best i_k, it costs $O(n)$. Furthermore, notice that $\nabla F(\mathbf{x}) = \mathbf{P}\mathbf{x} + \mathbf{d}$. Hence,

$$\nabla F(\mathbf{x}^{k+1}) = \nabla F(\mathbf{x}^k) + \mathbf{P}(\mathbf{x}^{k+1} - \mathbf{x}^k) = \mathbf{g}^k + (\hat{x}_{i_k}^k - x_{i_k}^k)\mathbf{p}_{i_k}.$$

Therefore, to renew $\mathbf{g}$, it takes an additional $O(n)$ flops. This way, we need $O(n)$ flops to update the iterate and maintain full gradient from $(\mathbf{x}^k, \mathbf{g}^k)$ to $(\mathbf{x}^{k+1}, \mathbf{g}^{k+1})$, and thus completing n greedy coordinate updates costs $O(n^2)$, which is similar to the cost of a gradient descent update.

B. Stopping Condition

A point $\mathbf{x}^* \geq \mathbf{0}$ is an optimal solution to (3) *if and only if* $\mathbf{0} \in \nabla F(\mathbf{x}^*) + \mathcal{N}_+(\mathbf{x}^*)$, where $\mathcal{N}_+(\mathbf{x}) = \left\{\mathbf{g} \in \mathbb{R}^n : g_i x_i \leq 0, \forall\, i \in [n]\right\}$ denotes the normal cone of the non-negative orthant at $\mathbf{x} \geq \mathbf{0}$. Therefore, $\mathbf{x}^*$ should satisfy the following conditions for each $i \in [n]$: $\nabla_i F(\mathbf{x}^*) = 0$ if $x_i^* > 0$, and $\nabla_i F(\mathbf{x}^*) \geq 0$ if $x_i^* = 0$. Let $I_0^k = \{i \in [n] : x_i^k = 0\}$, $I_+^k = \{i \in [n] : x_i^k > 0\}$, and

$$\delta_k = \sqrt{\textstyle\sum_{i\in I_0^k}\left[\min(0, \nabla_i F(\mathbf{x}^k))\right]^2 + \sum_{i\in I_+^k}\left[\nabla_i F(\mathbf{x}^k)\right]^2}. \tag{7}$$

Then if δ_k is smaller than a pre-specified error tolerance, we can stop the algorithm.

C. Pseudocode of the greedy CD

Summarizing the above discussions, we have the pseudocode for solving the NQP (3) in Algorithm 1.

Algorithm 1: Greedy CD for (3): $\mathrm{GCD}(\mathbf{P}, \mathbf{d}, \varepsilon, \mathbf{x}^0)$

Input: a PSD matrix $\mathbf{P} \in \mathbb{R}^{n\times n}$, $\mathbf{d} \in \mathbb{R}^n$, initial point $\mathbf{x}^0 \geq \mathbf{0}$, and an error tolerance $\varepsilon > 0$.
Overhead: let $k = 0$, $\mathbf{g}^0 = \nabla F(\mathbf{x}^0)$ and set δ_0 by (7).
while $\delta_k > \varepsilon$ **do**
  Compute $\hat{x}_i^k$ for all $i \in [n]$ by (5);
  Find $i_k \in [n]$ by the rule in (6);
  Let $x_i^{k+1} = x_i^k$ for $i \neq i_k$ and $x_i^{k+1} = \hat{x}_i^k$ for $i = i_k$;
  Update $\mathbf{g}^{k+1} = \mathbf{g}^k + (x_{i_k}^{k+1} - x_{i_k}^k)\mathbf{p}_{i_k}$;
  Increase $k \leftarrow k + 1$ and compute δ_k by (7).
Output $\mathbf{x}^k$

D. Convergence result

The greedy CD that chooses coordinates based on the objective decrease has been analyzed in [2]. We apply the results there to obtain the convergence of Algorithm 1.

Theorem 1. *Let $\{\mathbf{x}^k\}$ be the sequence from Algorithm 1. Suppose that the lower level set $\mathcal{L}_0 = \left\{\mathbf{x} \geq \mathbf{0} : F(\mathbf{x}) \leq F(\mathbf{x}^0)\right\}$ is compact. Then $F(\mathbf{x}^k) \to F^*$ as $k \to \infty$, where F^* is the optimal objective value of* (3)*.*

Proof. Since $F(\mathbf{x}^k)$ is decreasing with respect to k and $\mathcal{L}_0$ is compact, the sequence $\{\mathbf{x}^k\}$ has a finite cluster point $\bar{\mathbf{x}}$, and $F(\mathbf{x}^k) \to F(\bar{\mathbf{x}})$ as $k \to \infty$. Now notice that the minimizer of $G_i^{(k)}$ in (4) is unique for each i since $P_{ii} > 0$. Hence, it follows from [2, Theorem 3.1] that $\bar{\mathbf{x}}$ must be a stationary point of (3) and thus an optimal solution because $\mathbf{P}$ is PSD. Therefore, $F(\bar{\mathbf{x}}) = F^*$, and this completes the proof. □

Remark 1. *It is not difficult to show that $F(\mathbf{x}^{k+1}) - F(\mathbf{x}^k) \leq -\frac{P_{ii}}{2}\|\mathbf{x}^{k+1} - \mathbf{x}^k\|^2$. In addition, by [34, Theorem 18], the quadratic function F satisfies the so-called global*

error bound. Hence, it is possible to show a globally linear convergence result of Algorithm 1. Due to the page limitation, we do not extend the detailed discussion here, but instead we will explore it in an extended version of the paper.

E. Comparison to the cyclic and randomized CD methods

We compare the greedy CD to its cyclic and randomized counterparts and also the accelerated projected gradient method FISTA [1]. Two random NQP instances were generated. For the first one, we set $n = 5,000$ and generated a symmetric PSD matrix $\mathbf{P}$ and the vector $\mathbf{d}$ by the normal distribution. For the second one, we set $n = 1,000$, $\mathbf{P} = 0.1\mathbf{I} + 0.9\mathbf{E}$ and $\mathbf{d} = -10\mathbf{e}$, where $\mathbf{I}$ denotes the identity matrix, and $\mathbf{E}$ and $\mathbf{e}$ are all-ones matrix and vector. The $\mathbf{P}$ in the second instance was used to construct a difficult unconstrained QP for the cyclic CD in [10], [30]. Figure 1 plots the objective error produced by the three different CD methods and FISTA. From the plots, we clearly see that the greedy CD performs significantly better than the other two CDs and FISTA on both instances.

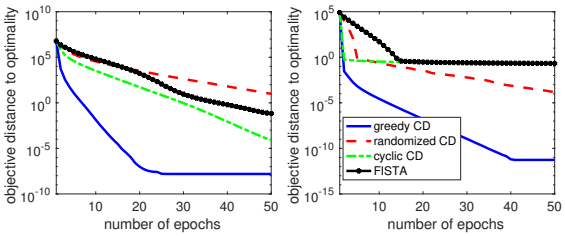

Fig. 1. Comparison of three different CDs and FISTA on two NQP instances. Left: dimension $n = 5,000$ and $\mathbf{P}$ generated by normal distribution; Right: $n = 1,000$ and $\mathbf{P} = 0.1\mathbf{I} + 0.9\mathbf{E}$.

III. Linear equality-constrained non-negative quadratic programming

In this section, we apply Algorithm 1 as a subroutine in the inexact ALM framework to solve the NQP with linear constraints. More specifically, we consider the problem

$$\min_{\mathbf{x}\geq\mathbf{0}} F(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^\top\mathbf{Q}\mathbf{x} + \mathbf{c}^\top\mathbf{x}, \text{ s.t. } \mathbf{A}\mathbf{x} = \mathbf{b}, \quad (8)$$

where $\mathbf{Q} \in \mathbb{R}^{n\times n}$ is a PSD matrix, and $\mathbf{A} \in \mathbb{R}^{m\times n}$. The ALM [24], [27] is perhaps the most popular method for solving nonlinear functional constrained problems. Applied to (8), it iteratively performs the updates:

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}\geq\mathbf{0}} L_{\beta_k}(\mathbf{x}, \mathbf{y}^k) \quad (9a)$$

$$\mathbf{y}^{k+1} = \mathbf{y}^k + \beta_k(\mathbf{A}\mathbf{x}^{k+1} - \mathbf{b}). \quad (9b)$$

Here, $\mathbf{y} \in \mathbb{R}^m$ is the Lagrange multiplier, and

$$L_\beta(\mathbf{x}, \mathbf{y}) = F(\mathbf{x}) + \mathbf{y}^\top(\mathbf{A}\mathbf{x} - \mathbf{b}) + \tfrac{\beta}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 \quad (10)$$

is the AL function with a penalty parameter $\beta > 0$.

A. inexact ALM with greedy CD

In the ALM update, the $\mathbf{y}$-update is easy. However, the $\mathbf{x}$-subproblem (9a) in general requires an iterative solver. On solving (8), we can rewrite the AL function as

$$L_\beta(\mathbf{x}, \mathbf{y}) = \tfrac{1}{2}\mathbf{x}^\top(\mathbf{Q} + \beta\mathbf{A}^\top\mathbf{A})\mathbf{x} + (\mathbf{c} + \mathbf{A}^\top\mathbf{y} - \beta\mathbf{A}^\top\mathbf{b})^\top\mathbf{x},$$

which is a quadratic function. Hence, (9a) is an NQP, and we propose to apply the greedy CD derived previously to solve it. The pseudocode of the proposed method is shown in Algorithm 2.

Algorithm 2: inexact ALM with greedy CD for (8)

Input: a PSD matrix $\mathbf{Q} \in \mathbb{R}^{n\times n}$, $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$, and $\varepsilon > 0$.
Initialization: $\mathbf{x}^0 \in \mathbb{R}^n, \mathbf{y}^0 = \mathbf{0}$, and $\beta_0 > 0$; set $k = 0$.
while *a stopping condition not satisfied* **do**
  Let $\mathbf{P} = \mathbf{Q} + \beta_k\mathbf{A}^\top\mathbf{A}$ and $\mathbf{d} = \mathbf{c} + \mathbf{A}^\top\mathbf{y} - \beta_k\mathbf{A}^\top\mathbf{b}$.
  Choose $\varepsilon_k \leq \varepsilon$ and let $\mathbf{x}^{k+1} = \mathrm{GCD}(\mathbf{P}, \mathbf{d}, \varepsilon_k, \mathbf{x}^k)$.
  Obtain $\mathbf{y}^{k+1}$ by (9b).
  Choose $\beta_{k+1} \geq \beta_k$, and increase $k \leftarrow k + 1$.

Notice that each $\mathbf{x}^{k+1}$ satisfies $\mathrm{dist}\big(\mathbf{0}, \nabla_{\mathbf{x}} L_{\beta_k}(\mathbf{x}^{k+1}, \mathbf{y}^k) + \mathcal{N}_+(\mathbf{x}^{k+1})\big) \leq \varepsilon_k$. Hence, by the update of $\mathbf{y}^{k+1}$, it holds

$$\mathrm{dist}\left(\mathbf{0}, \nabla F(\mathbf{x}^{k+1}) + \mathbf{A}^\top\mathbf{y}^{k+1} + \mathcal{N}_+(\mathbf{x}^{k+1})\right) \leq \varepsilon_k,$$

namely, the dual residual is always no larger than ε_k. Since $\varepsilon_k \leq \varepsilon$, the output $(\mathbf{x}^{k+1}, \mathbf{y}^{k+1})$ violates the KKT conditions at most ε in terms of both primal and dual feasibility, if we stop the algorithm once $\|\mathbf{A}\mathbf{x}^{k+1} - \mathbf{b}\| \leq \varepsilon$.

B. Convergence result

We can apply the results in [13], [27], [37] to obtain the convergence of Algorithm 2 based on the actual iterate.

Theorem 2. *Suppose that for each $i \in [n]$, $Q_{ii} > 0$ or $\mathbf{a}_i \neq \mathbf{0}$. Let $\{\mathbf{x}^k\}_{k\geq 0}$ be the sequence from Algorithm 2. If $\varepsilon_k \to 0$ and $\beta_k \to \infty$, then $|F(\mathbf{x}^k) - F^*| \to 0$ and $\|\mathbf{A}\mathbf{x}^k - \mathbf{b}\| \to 0$, where F^* denotes the optimal objective value of* (8).

IV. Non-negative matrix factorization

In this section, we consider the non-negative matrix factorization (NMF). It aims to factorize a given non-negative matrix $\mathbf{M} \in \mathbb{R}_+^{m\times n}$ into two low-rank non-negative matrices $\mathbf{X}$ and $\mathbf{Y}$ such that $\mathbf{M} \approx \mathbf{X}\mathbf{Y}^\top$. The factor matrix $\mathbf{X}$ plays a role of basis while $\mathbf{Y}$ contains the coefficient. Due to the non-negativity, NMF can be used to learn local features of an objective [11] and has better interpretability than the principal component analysis (PCA). Measuring the approximation error by the Frobenius norm, one can model the NMF as

$$\min_{\mathbf{X},\mathbf{Y}} \tfrac{1}{2}\|\mathbf{X}\mathbf{Y}^\top - \mathbf{M}\|_F^2, \text{ s.t. } \mathbf{X} \in \mathbb{R}_+^{m\times r}, \ \mathbf{Y} \in \mathbb{R}_+^{n\times r}, \quad (11)$$

where $\mathbf{M} \in \mathbb{R}_+^{m\times n}$ is given, and r is a user-specified rank.

A. Alternating minimization with greedy CD

The objective of (11) is non-convex jointly with respect to $\mathbf{X}$ and $\mathbf{Y}$. However, it is convex with respect to one of $\mathbf{X}$ and $\mathbf{Y}$ while the other is fixed. For this reason, one natural way to solve (11) is the alternating minimization (AltMin), which iteratively performs the update

$$\mathbf{X}^{k+1} = \arg\min_{\mathbf{X}\geq\mathbf{0}} \tfrac{1}{2}\|\mathbf{X}(\mathbf{Y}^k)^\top - \mathbf{M}\|_F^2, \quad (12a)$$

$$\mathbf{Y}^{k+1} = \arg\min_{\mathbf{Y}\geq\mathbf{0}} \tfrac{1}{2}\|(\mathbf{X}^{k+1})\mathbf{Y}^\top - \mathbf{M}\|_F^2. \quad (12b)$$

Both subproblems are in the form of $\min_{\mathbf{Z}\in\mathbb{R}^{r\times p}} \frac{1}{2}\|\mathbf{AZ}-\mathbf{B}\|_F^2$, which is equivalent to solving p independent NQPs

$$\min_{\mathbf{z}_i\in\mathbb{R}^r} \tfrac{1}{2}\|\mathbf{A}\mathbf{z}_i-\mathbf{b}_i\|^2,\ i\in[p].$$

Hence, we can apply the greedy CD derived in section II to solve the $\mathbf{X}$-subproblem and $\mathbf{Y}$-subproblem in (12), by breaking them respectively into m and n independent NQPs. The pseudocode is shown in Algorithm 3. In Lines 4 and 8, we rescale those two factor matrices such that they have balanced norms. This way, neither of them will blow up or diminish, and the resulting NQP subproblems are relatively well-conditioned. Numerically, we observe that the rescaling technique can significantly speed up the convergence.

Notice that it is straightforward to extend our method to the non-negative tensor decomposition [28]. Due to the page limitation, we do not give the details here but leave it to an extended version of this paper.

Algorithm 3: AltMin with greedy CD for (11)

1 **Input**: $\mathbf{M}\in\mathbb{R}_+^{m\times n}$, and rank r.
2 **Initialization**: $\mathbf{X}^0\in\mathbb{R}_+^{m\times r}$ and $\mathbf{Y}^0\in\mathbb{R}_+^{n\times r}$.
3 **for** $k=0,1,\ldots$ **do**
4 Rescale $\mathbf{X}^k$ and $\mathbf{Y}^k$ to $\|\mathbf{x}_i^k\|=\|\mathbf{y}_i^k\|, \forall\, i\in[r]$.
5 Let $\mathbf{P}=(\mathbf{Y}^k)^\top\mathbf{Y}^k$ and $\mathbf{D}=-(\mathbf{Y}^k)^\top\mathbf{M}^\top$.
6 Choose $\varepsilon_k>0$.
7 Compute $\mathbf{x}_{i:}^{k+1}=\mathrm{GCD}(\mathbf{P},\mathbf{d}_i,\varepsilon_k,\mathbf{x}_{i:}^k)$, for $i\in[m]$.
8 Rescale $\mathbf{X}^{k+1}$ and $\mathbf{Y}^k$ to $\|\mathbf{x}_i^{k+1}\|=\|\mathbf{y}_i^k\|, \forall\, i\in[r]$.
9 Let $\mathbf{P}=(\mathbf{X}^{k+1})^\top\mathbf{X}^{k+1}$ and $\mathbf{D}=-(\mathbf{X}^{k+1})^\top\mathbf{M}$.
10 Compute $\mathbf{y}_{i:}^{k+1}=\mathrm{GCD}(\mathbf{P},\mathbf{d}_i,\varepsilon_k,\mathbf{y}_{i:}^k)$, for $i\in[n]$.

B. Convergence result

The convergence of the AltMin has been well-studied; see [5] for example. Although we rescale the two factor matrices and inexactly solve each subproblem, it is not difficult to adapt the existing analysis and obtain the convergence of Algorithm 3 as follows.

Theorem 3. *Let $\{(\mathbf{X}^k,\mathbf{Y}^k)\}_{k\geq 0}$ be the sequence generated from Algorithm 3 with $\varepsilon_k\to 0$. Then any finite limit point of the sequence is a stationary point of* (11).

V. Numerical Experiments

In this section, we test Algorithm 2 on Gaussian random-generated instances of (8) and compare it to the MATLAB built-in solver `quadprog`. Also, we test Algorithm 3 on three instances of the NMF (11), two with synthetic data and another with face image data. For the tests on (8), we generated three different-sized instances. In Algorithm 2, we set $\varepsilon_k=10^{-3}, \forall\, k$, for the subroutine GCD, and the algorithm was stopped once $\|\mathbf{A}\mathbf{x}^k-\mathbf{b}\|\leq\varepsilon$ with $\varepsilon=10^{-2}$ or 10^{-3}. For the tests on (11) with synthetic data, we obtain $\mathbf{M}=\mathbf{L}\mathbf{R}^\top$, where $\mathbf{L}\in\mathbb{R}_+^{m\times r}$ and $\mathbf{R}\in\mathbb{R}_+^{n\times r}$ were respectively generated by MATLAB's code `max(0,randn(m,r))` and `max(0,randn(n,r))`. One dataset was generated with $m=n=1,000, r=50$ and the other with $m=n=5,000, r=100$. For the other test on (11), we used the CBCL face image data [8], which consists of 6,977 images of size 19×19. We picked the first 2,000 images and vectorized each image into a vector. This way, we formed a non-negative $\mathbf{M}\in\mathbb{R}^{361\times 2000}$, and we set $r=30$. In Algorithm 3, we set $\varepsilon_k=10^{-3}, \forall\, k$, for the subroutine GCD on the test with synthetic data and $\varepsilon_k=0.1, \forall\, k$, on the face image data. Different tolerances were adopted here because the image data does not admit an exact factorization.

TABLE I
Results by Algorithm 2 on three different-sized instances of (8) and the speed comparison with MATLAB function quadprog. Here, tol. is the ε in Algorithm 2; obj. relerr is computed by $\frac{|F(\mathbf{x})-F^*|}{|F^*|}$; res. relerr is by $\frac{\|\mathbf{A}\mathbf{x}-\mathbf{b}\|}{\|\mathbf{b}\|}$; time1 is the running time (sec.) by the proposed method; time2 is by quadprog.

problem size	tol.	obj. relerr	res. relerr	time1	time2
$m=200$	0.01	2.758e-05	5.192e-04	0.42	0.32
$n=1000$	0.001	1.118e-06	2.811e-05	0.76	
$m=1000$	0.01	3.714e-06	1.138e-04	16.68	24.92
$n=5000$	0.001	2.257e-07	8.074e-06	30.85	
$m=2000$	0.01	4.361e-07	2.021e-05	71.90	217.41
$n=10000$	0.001	3.795e-07	1.419e-05	133.07	

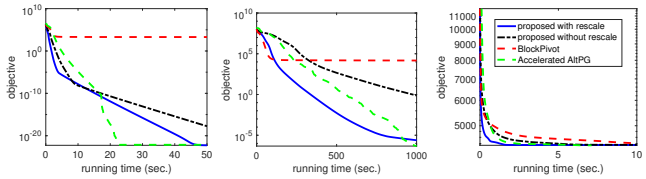

Fig. 2. Comparison of the proposed method (Algorithm 2) with and without the rescaling to the BlockPivot method in [9] and the accelerated AltPG in [38]. Left: synthetic data with $m=n=1,000, r=50$; Middle: synthetic data with $m=n=5,000, r=100$; Right: CBCL face image data set

The results for the tests on (8) are shown in Table I. From the results, we see that our method can yield medium-accurate solutions. For the middle-sized instance, our method can be faster than MATLAB's solver when $\varepsilon=10^{-2}$, and for the large-sized instance, our method is faster under both settings of $\varepsilon=10^{-2}$ and 10^{-3}. This implies that for solving large-scale linear-constrained NQP, the proposed method can outperform MATLAB's solver if a medium-accurate solution is required. The results for the tests on (11) are plotted in Figure 2, where we compared Algorithm 3 with the BlockPivot method in [9] and the accelerated AltPG method in [38]. The BlockPivot is also an AltMin, but different from our greedy CD, it uses an active-set-type method as a subroutine to exactly solve each subproblem. The accelerated AltPG performs block proximal gradient update to X and Y alternatingly, and it uses extrapolation technique for acceleration. From the results, we see that the proposed method performs significantly better than BlockPivot, which seems to be trapped at a local solution for the two synthetic cases. Compared to the accelerated AltPG, the proposed method can be faster in the beginning to obtain a medium accuracy. In addition, the proposed method converges faster with the rescaling than that without the rescaling on the instances with large-sized synthetic data and the face image data. That could be because the subproblems may be bad-conditioned if rescaling is not applied.

REFERENCES

[1] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[2] B. Chen, S. He, Z. Li, and S. Zhang. Maximum block improvement and polynomial optimization. *SIAM Journal on Optimization*, 22(1):87–107, 2012.

[3] C. D. Dang and G. Lan. Stochastic block mirror descent methods for nonsmooth and stochastic optimization. *SIAM Journal on Optimization*, 25(2):856–881, 2015.

[4] X. Gao, Y.-Y. Xu, and S.-Z. Zhang. Randomized primal–dual proximal block coordinate updates. *Journal of the Operations Research Society of China*, 7(2):205–250, 2019.

[5] L. Grippo and M. Sciandrone. On the convergence of the block nonlinear gauss–seidel method under convex constraints. *Operations research letters*, 26(3):127–136, 2000.

[6] C. Hildreth. A quadratic programming procedure. *Naval research logistics quarterly*, 4(1):79–85, 1957.

[7] M. Hong, X. Wang, M. Razaviyayn, and Z.-Q. Luo. Iteration complexity analysis of block coordinate descent methods. *Mathematical Programming*, 163(1-2):85–114, 2017.

[8] P. O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5(Nov):1457–1469, 2004.

[9] J. Kim and H. Park. Toward faster nonnegative matrix factorization: A new algorithm and comparisons. In *2008 Eighth IEEE International Conference on Data Mining*, pages 353–362. IEEE, 2008.

[10] C.-P. Lee and S. J. Wright. Random permutations fix a worst case for cyclic coordinate descent. *IMA Journal of Numerical Analysis*, 39(3):1246–1275, 2018.

[11] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788, 1999.

[12] Y. Li and S. Osher. Coordinate descent optimization for l1 minimization with application to compressed sensing; a greedy algorithm. *Inverse Problems and Imaging*, 3(3):487–503, 2009.

[13] Z. Li and Y. Xu. Augmented lagrangian based first-order methods for convex and nonconvex programs: nonergodic convergence and iteration complexity. *arXiv preprint arXiv:2003.08880*, 2020.

[14] Q. Lin, Z. Lu, and L. Xiao. An accelerated proximal coordinate gradient method. In *Advances in Neural Information Processing Systems*, pages 3059–3067, 2014.

[15] J. Liu and S. J. Wright. Asynchronous stochastic coordinate descent: Parallelism and convergence properties. *SIAM Journal on Optimization*, 25(1):351–376, 2015.

[16] Z.-Q. Luo and P. Tseng. On the convergence of the coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72(1):7–35, 1992.

[17] Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

[18] J. Nutini. *Greed is good: greedy optimization methods for large-scale structured problems*. PhD thesis, University of British Columbia, 2018.

[19] J. Nutini, M. Schmidt, I. Laradji, M. Friedlander, and H. Koepke. Coordinate descent converges faster with the gauss-southwell rule than random selection. In *International Conference on Machine Learning*, pages 1632–1641, 2015.

[20] P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994.

[21] Z. Peng, T. Wu, Y. Xu, M. Yan, and W. Yin. Coordinate-friendly structures, algorithms and applications. *Annals of Mathematical Sciences and Applications*, 1(1):57–119, 2016.

[22] Z. Peng, Y. Xu, M. Yan, and W. Yin. ARock: an algorithmic framework for asynchronous parallel coordinate updates. *SIAM Journal on Scientific Computing*, 38(5):A2851–A2879, 2016.

[23] Z. Peng, M. Yan, and W. Yin. Parallel and distributed sparse optimization. In *2013 Asilomar conference on signals, systems and computers*, pages 659–646. IEEE, 2013.

[24] M. J. Powell. *A method for non-linear constraints in minimization problems*. in Optimization, R. Fletcher Ed., Academic Press, New York, NY, 1969.

[25] M. Razaviyayn, M. Hong, and Z.-Q. Luo. A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM Journal on Optimization*, 23(2):1126–1153, 2013.

[26] P. Richtárik and M. Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1-2):1–38, 2014.

[27] R. T. Rockafellar. The multiplier method of hestenes and powell applied to convex programming. *Journal of Optimization Theory and applications*, 12(6):555–562, 1973.

[28] A. Shashua and T. Hazan. Non-negative tensor factorization with applications to statistics and computer vision. In *Proceedings of the 22nd international conference on Machine learning*, pages 792–799. ACM, 2005.

[29] H.-J. M. Shi, S. Tu, Y. Xu, and W. Yin. A primer on coordinate descent algorithms. *arXiv preprint arXiv:1610.00040*, 2016.

[30] R. Sun and Y. Ye. Worst-case complexity of cyclic coordinate descent: $O(n^2)$ gap with randomized version. *Mathematical Programming*, pages 1–34, 2019.

[31] G. Thoppe, V. S. Borkar, and D. Garg. Greedy block coordinate descent (gbcd) method for high dimensional quadratic programs. *arXiv preprint arXiv:1404.6635*, 2014.

[32] P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109(3):475–494, 2001.

[33] P. Tseng and S. Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, 117(1-2):387–423, 2009.

[34] P.-W. Wang and C.-J. Lin. Iteration complexity of feasible descent methods for convex optimization. *The Journal of Machine Learning Research*, 15(1):1523–1548, 2014.

[35] S. J. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

[36] Y. Xu. Asynchronous parallel primal–dual block coordinate update methods for affinely constrained convex programs. *Computational Optimization and Applications*, 72(1):87–113, 2019.

[37] Y. Xu. Iteration complexity of inexact augmented lagrangian methods for constrained convex programming. *Mathematical Programming, Series A*, pages 1–46, 2019.

[38] Y. Xu and W. Yin. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on Imaging Sciences*, 6(3):1758–1789, 2013.

[39] Y. Xu and W. Yin. Block stochastic gradient iteration for convex and nonconvex optimization. *SIAM Journal on Optimization*, 25(3):1686–1716, 2015.

[40] Y. Xu and W. Yin. A globally convergent algorithm for nonconvex optimization based on block coordinate update. *Journal of Scientific Computing*, 72(2):700–734, 2017.