PROJECTION-FREE NON-SMOOTH CONVEX PROGRAMMING

KAMIAR ASGARI* AND MICHAEL J. NEELY*

Abstract. In this paper, we provide a sub-gradient based algorithm to solve general constrained convex optimization without taking projections onto the domain set. The well studied Frank-Wolfe type algorithms also avoid projections. However, they are only designed to handle smooth objective functions. The proposed algorithm treats both smooth and non-smooth problems and achieves an $O(1/\sqrt{T})$ convergence rate (which matches existing lower bounds). The algorithm yields similar performance in expectation when the deterministic sub-gradients are replaced by stochastic sub-gradients. Thus, the proposed algorithm is a projection-free alternative to the Projected sub-Gradient Descent (PGD) and Stochastic projected sub-Gradient Descent (SGD) algorithms.

Key words. Projection-free optimization, Non-smooth convex programming, Frank-Wolfe method

MSC codes. 90C25

1. Introduction. In order to solve general convex optimization, many versions of Black-Box first-order algorithms have been proposed. However, most of those commonly used algorithms require a Euclidean (L_2) projection onto the domain set at each iteration, such as the famous Projected sub-Gradient Descent (PGD) algorithm.[1] Unless there is special structure, it is not always computationally easy to perform the projection steps. This can make PGD less attractive for large-scale constrained optimization problems. This paper develops an optimization algorithm that replaces projections with (computationally easier) Linear Optimization steps.

Specifically, let $\mathcal{X} \in \mathbb{R}^n$ be a closed and convex set. The problem of this paper is to minimize a general convex function $f : \mathbb{R}^n \to \mathbb{R}$ over all $x \in \mathcal{X}$. The Euclidean projection, i.e. L_2 projection, is defined as follows. Let $\mathcal{X}$ be a closed convex set. Define

$$\pi_{\mathcal{X}}(y) \stackrel{def}{=} \text{Argmin}_{x\in\mathcal{X}} \|x - y\|, \quad \forall y \in \mathbb{R}^n$$

We call $\pi_{\mathcal{X}}(y)$ the Euclidean projection of the point y onto the set $\mathcal{X}$ (as $\mathcal{X}$ is closed and convex, the projection exists and is unique). As our algorithm is similar to PGD, let us review PGD:

Algorithm 1.1 Projected sub-Gradient Descent (PGD)

1: Require $T \in \{1, 2, \ldots\}, \beta > 0, x_0 \in \mathcal{X}$
2: **for** $k = 0 \cdots T - 1$ **do**
3: Calculate $g_k \in \partial f(x_k)$
4: $x_{k+1} = \pi_{\mathcal{X}}(x_k - \beta g_k)$
5: **end for**
6: **return** $\bar{x} = \frac{1}{T+1}\sum_{k=0}^{T} x_k$

It can be shown that the PGD's update rule is equivalent to solve the following quadratic optimization problem:

$$x_{k+1} = \pi_{\mathcal{X}}(x_k - \beta g_k) = \text{Argmin}_{x\in\mathcal{X}} \left\{ \langle g_k, x\rangle + \frac{1}{2\beta}\|x - x_k\|^2 \right\}$$

*University of Southern California, Los Angeles, CA (Kamiaras@usc.edu, Mjneely@usc.edu).

[1] We use the more common acronym PGD for projected sub-gradient descent instead of PSD.

where $\langle x, y\rangle$ is the inner product of x and y in $\mathbb{R}^n$.

For a general convex set, the projection step may involve a numerical procedure that can be computationally expensive. Different algorithms to compute a projection of a point onto a convex set have been suggested [16, 17, 9, 13]. However, for many convex sets, a linear optimization, defined by the following equation:

$$\min_{x\in\mathcal{X}} \{\langle g, x\rangle\}, \quad \text{for a given } g \in \mathbb{R}^n \tag{1.1}$$

can be carried out easier than a projection. Let us call sets $\mathcal{X}$ for which problems of the type Equation (1.1) can be solved more easily than a projection as Appropriate sets. Our algorithm works for any domain set that is convex. However, it shows its potential when the domain set $\mathcal{X}$ is an Appropriate set.

Many examples of Appropriate sets are known and studied. The work [14] unifies many Appropriate sets under the umbrella of *Atomic sets*. One famous example is the set of matrices with bounded *trace norm*. The trace norm of a matrix A, shown as $\|A\|_*$, is the sum of its singular values. For a fixed $\tau > 0$, define set $\mathcal{X} \overset{def}{=} \{A \in \mathbb{R}^{m\times n} : \|A\|_* \leq \tau\}$. The projection of matrix $X \in \mathbb{R}^{m\times n}$ on the set $\mathcal{X}$ requires a Singular Value Decomposition (SVD) of X which costs $O\left(mn\min\{m, n\}\right)$ time. Meanwhile, a linear optimization: $\min_{X\in\mathcal{X}}\langle P, X\rangle$, requires the calculation of the largest singular value and its respective singular vectors of the matrix P. This calculation costs linear time in the number of non-zero entries in that matrix.

1.1. Main Result. Similar to Frank-wolf type algorithms that are projection-free but only work for smooth objective functions, this paper offers a projection-free algorithm for general convex programming. This is an alternative for PGD when the Equation (1.1) is easier then a projection. The algorithm is also compatible with stochastic (noisy) sub-gradients, which makes it an alternative for SGD.

1.2. History and Related Work. The original Frank-Wolfe algorithm [6] was introduced and analyzed for polyhedral domains in $\mathbb{R}^n$, relying on line-search on a quadratic upper bound on f. A general framework to analyze Frank-wolf type algorithms is achieved in [14] which also provides a comprehensive overview and comparison between many existing algorithms of that type. Using the idea of Variance-Reduced, [11] gets better bounds for both smooth convex functions and smooth strongly-convex functions while presenting another comparison of different variants of Frank-Wolfe algorithms.

Using randomized smoothing, [10] proposed a variant of Frank-Wolfe for online optimization of non-smooth and smooth convex functions but not achieving the optimal convergence rates. This result has been improved in [12] using the so-called blocking technique.

In the case when the domain only consists of convex functional constrains ($\mathcal{X} = \{\, x \mid h_i(x) \leq 0, \forall i \in \{1, \cdots, I\}\,\}$), [18] develops a unique algorithm with only one projection which (with high probability) achieves an $O(1/\sqrt{T})$ rate for general convex optimization, and an $O(\log(T)/T)$ rate for strongly convex optimization. Using similar assumptions (and adding an extra assumption of smoothness of constraint functions h_i), [15] achieves a one projection algorithm with $O(1/\sqrt{T})$ regret for general online convex optimization, and an $O(\log(T)/T)$ regret for online strongly convex optimization. Similar functional constraints are treated in [21] with $O(1/T)$ convergence for general composite smooth convex and/or separable convex programs using knowledge of a Lagrange Multiplier bound, and [22] removes the need to know the Lagrange Multiplier bound.

In another line of work, [1] proposed a generalization of the conditional gradient algorithm achieving rates similar to the standard stochastic gradient algorithm using only zeroth-order information.

2. Preliminaries. $\mathbb{R}$ denotes the set of real numbers, and $\mathbb{R}^n$ is the usual vector space of real n-tuples $x = (x^{(1)}, \cdots, x^{(n)})$. The inner product of x and y in $\mathbb{R}^n$ is expressed by

$$\langle x, y \rangle \stackrel{def}{=} \sum_{i=1}^{n} x^{(i)} y^{(i)}$$

In this paper the norm is standard Euclidean which is defined as:

$$\|x\| \stackrel{def}{=} \sqrt{\sum_{i=1}^{n} \left(x^{(i)}\right)^2} = \sqrt{\langle x, x \rangle}$$

Consider an arbitrary function f defined on the set S. Fix the point $x \in S$. A vector g is called a *subgradient* of the function f at the point x if

$$f(y) \geq f(x) + \langle g, y - x \rangle, \quad \forall y \in S \tag{2.1}$$

The set of all subgradients of f at x, $\partial f(x)$, is called the *subdifferential* of the function f at the point x. Function f is called G-Lipschitz continuous on the set S if $\forall x, y \in S$

$$|f(x) - f(y)| \leq G\|x - y\| \tag{2.2}$$

A subset C of $\mathbb{R}^n$ is said to be *convex* if $(1 - \lambda)x + \lambda y \in C$ whenever $x \in C$, $y \in C$ and $0 \leq \lambda \leq 1$. A function f is called *convex* on set C if C is a convex set and for all $x, y \in C$ and $\alpha \in [0, 1]$ the following inequality holds:

$$f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha) f(y).$$

2.1. Extension of Convex Function. An important assumption used in this paper is that f is convex and G-Lipschitz continuous on the *whole* $\mathbb{R}^n$. While this is a valid assumption in many practical cases, there are cases which can violate it.

The following theorem collects important parts from a larger *McShane-Whitney extension theorem* of [20]. It shows that if f is convex and G-Lipschitz continuous on the set $\mathcal{X}$, then f can be extended to a convex and G-Lipschitz continuous function on the whole of $\mathbb{R}^n$.

THEOREM 2.1. *Let $\mathcal{X}$ be a convex set in $\mathbb{R}^n$, $f : \mathcal{X} \to \mathbb{R}$ a convex G-Lipschitz continuous function defined on $\mathcal{X}$. Then the function $\tilde{f}$ defined by:*

$$\tilde{f}(w) \stackrel{def}{=} \inf_{x \in \mathcal{X}} \{f(x) + G\|x - w\|\}$$

satisfies the following:

(i) $\tilde{f}$ is a real valued function.
(ii) $\tilde{f}$ is a G-Lipschitz continuous function on $\mathbb{R}^n$.
(iii) For any $x \in \mathbb{R}^n$, any sub-gradient $g \in \partial \tilde{f}(x)$ is bounded by G: $\|g\| \leq G$.
(iv) $\tilde{f}$ is a convex function on $\mathbb{R}^n$.
(v) $f(w) = \tilde{f}(w)$, $\forall w \in \mathcal{X}$

Proof. This theorem is a special case of the *McShane-Whitney extension theorem* and it is fully borrowed from different parts of [20]. However, for the sake of convenience, the proof is collected and presented at Appendix A. ∎

2.2. Problem setup. We want to provide a projection-free algorithm to solve the following optimization:

$$\min_{x\in\mathcal{X}} f(x)$$

We assume that the set $\mathcal{X}$ is contained in an Euclidean ball centered at $x_1 \in \mathcal{X}$ and of radius R. This has two consequences:

$$\|x - y\| \leq 2R, \quad \forall x, y \in \mathcal{X} \tag{2.3}$$

$$\|x - x_1\| \leq R, \quad \forall x \in \mathcal{X} \tag{2.4}$$

Furthermore we assume that f is convex and G-Lipschitz on $\mathbb{R}^n$ (if not we can use its extension $\tilde{f}$ according to the Theorem 2.1).

3. Projection-free Algorithm for Deterministic Optimization. We assume that the function f is represented by a Black-Box first-order oracle. This means that we can only get information by requesting the oracle to return a sub-gradient of the requested point x which we call $g(x) \in \partial f(x)$. We provide the upper bound the on number of requests from the oracle to achieve a desired accuracy.

Algorithm 3.1 Projection-free Algorithm

Require integer $T \geq 1$
Choose constants $\alpha > 0$ and $\eta > 0$ and the initial point $x_1 \in \mathcal{X}$
Choose $y_1 = x_1$
Choose $Q_0 = \mathbf{0}$
for $1 \leq k \leq T - 1$ **do**
  Choose $Q_k = Q_{k-1} + y_k - x_k$
  Choose $g_k = g(y_k) \in \partial f(y_k)$
  Choose $x_{k+1} \in \text{Argmin}_{x\in\mathcal{X}} \langle -Q_k, x\rangle$
  Choose $y_{k+1} = \frac{1}{\alpha+\eta}(\alpha y_k + \eta x_{k+1} - \eta Q_k - g_k)$
end for
return $\bar{x} = \frac{1}{T}\sum_{k=1}^{T} x_k$

THEOREM 3.1 (Convergence of Algorithm 3.1). *Define* $x^* \in \text{Argmin}_{x\in\mathcal{X}}\{f(x)\}$. *Then with the following parameters,*

$$\alpha = \frac{G\sqrt{T}}{R}$$

$$\eta = \frac{G}{2R\sqrt{T}}$$

the Algorithm 3.1 ensures

$$f(\bar{x}) - f(x^*) \leq \frac{3RG}{\sqrt{T}}$$

for any integer $T \geq 1$.

Proof. See Appendix B. ∎

Let us remember the Algorithm 1.1 convergence too:

THEOREM 3.2 (PGD convergence rate). *Define* $x^* \in \operatorname{Argmin}_{x\in\mathcal{X}}\{f(x)\}$. *With the following parameter,*

$$\beta = \frac{R}{G\sqrt{T}}$$

the Algorithm 1.1 ensures

$$f(\bar{x}) - f(x^*) \leq \frac{RG}{\sqrt{T}}$$

for any integer $T \geq 1$.

Proof. See Theorem 3.2. from [5]. ∎

Upper bound on the error in both algorithms are of the same order (up to a constant) regarding the problem's parameters. However, this does not mean that they will always perform similarly as these are only upper bounds. Simulations, provided on the section 5, compare these two algorithms numerically.

4. Projection-free Algorithm for Stochastic Optimization. In many applications, function f is represented by a first-order *stochastic* oracle. The oracle takes Y as input and returns $\hat{g}(Y)$ such that $\mathbb{E}\{\hat{g}(Y)|Y\} \in \partial f(Y)$, meaning we only have access to a noisy but unbiased version of a sub-gradient. We also need to add an extra condition:

$$\mathbb{E}\{\|\hat{g}(Y)\|^2|Y\} \leq B^2$$

for some known fixed $B \geq G$ (this simply means that the noise must have a bounded variance). Surprisingly the algorithm works for this case too, only the $g_k = g(y_k)$ in Line 13 of Algorithm 3.1 is replaced with $g_k = \hat{g}(Y_k)$.

THEOREM 4.1 (Convergence of Algorithm 3.1 with stochastic sub-gradient). *Define* $x^* \in \operatorname{Argmin}_{x\in\mathcal{X}}\{f(x)\}$.

- *Then with the following parameters,*

$$\alpha = \frac{B\sqrt{T}}{R}$$

$$\eta = \frac{G}{2R\sqrt{T}}$$

the Algorithm 3.1 with stochastic sub-gradient ensures

$$\mathbb{E}\left\{f(\bar{X})\right\} - f(x^*) \leq \frac{BR + 2GR}{\sqrt{T}}$$

for any integer $T \geq 1$.

- *Then with following parameters,*

$$\alpha = \frac{B\sqrt{T}}{R}$$

$$\eta = \frac{2B}{R\sqrt{T}}$$

the Algorithm 3.1 with stochastic sub-gradient ensures

$$\mathbb{E}\left\{f(\bar{X})\right\} - f(x^*) \leq \frac{3GD}{2\sqrt{T}}$$

for any integer $T \geq 1$.

Proof. See Appendix C. ∎

SGD algorithm is also very similar to PGD (Algorithm 1.1). The $g_k = g(x_k)$ in Line 3 of Algorithm 1.1 is replaced with $g_k = \hat{g}(X_k)$.

THEOREM 4.2 (SGD convergence rate). *Define* $x^* \in \text{Argmin}_{x\in\mathcal{X}}\{f(x)\}$. *With the following parameter,*

$$\beta = \frac{R}{B\sqrt{T}}$$

the SGD algorithm ensures

$$\mathbb{E}\{f(\bar{X})\} - f(x^*) \leq \frac{BR}{\sqrt{T}}$$

for any integer $T \geq 1$.

Proof. See Theorem 6.3. from [5]. ∎

Upper bound of the average error in both algorithms are of the same order (up to a constant) regarding the problem's parameters. However, if in addition to parameter B the parameter G is provided, unlike PGD/SGD, our algorithm is able to take advantage of it. Simulations, provided on the section 5, compare these two algorithms numerically.

5. Numerical results. In this section we provide two examples where our algorithm is compared with PGD and SGD.

5.1. Hypercube domain with $L1$-norm objective function . This example is a good case study to see the numerical accuracy of our algorithm as the exact minima is analytically calculable. However, this is not an example to illustrate the computational gains. Our domain set is a n-dimensional hypercube:

$$\mathcal{X} = \left\{x \in \mathbb{R}^n : |x^{(i)}| \leq 1,\ \forall i \in \{1, \cdots, n\}\right\}$$

The set $\mathcal{X}$ is inside the Euclidean ball with radius $R = 2\sqrt{n}$ centered at $x_1 = \mathbf{0}$. Define the L_1-norm as: $\|x\|_1 = \sum_{i=1}^n |x^{(i)}|$. Let us choose the objective function $f_\omega(x) = \|x-\omega\|_1$. It is easy to see that this function is convex and Lipschitz continuous over $\mathbb{R}^n$ with coefficient

$$G = \sqrt{n}$$

When the exact sub-gradient is given to us without any error, we choose the parameters as follows: $\beta = \frac{R}{G\sqrt{T}}$, $\alpha = \frac{G\sqrt{T}}{R}$, $\eta = \frac{G}{2R\sqrt{T}}$.

In order to simulate an inexact sub-gradient, we added a mean-zero normal distribution with covariance matrix $\sigma^2 I_n$ to the exact sub-gradient, i.e., $\hat{g}(y_k) = g(y_k) + N_k$ where $\{N\}_{k=1}^T$ are i.i.d, n dimensional vector, samples of $\mathcal{N}(0, \sigma^2 I_n)$. Thus,

$$\mathbb{E}\{\|\hat{g}(x)\|^2\} = G^2 + n\sigma^2$$

so

$$B = \sqrt{G^2 + n\sigma^2}$$

We choose the parameters as following: $\beta = \frac{R}{B\sqrt{T}}$, $\alpha = \frac{B\sqrt{T}}{R}$, $\eta = \frac{G}{2R\sqrt{T}}$.

The experiments were done for different noise levels, σ, different dimensions, n, and for different values of T. Two cases are considered in which the parameter ω is

inside or outside of the domain set, $\mathcal{X}$. Clearly, in all of the figures, by increasing T, the error decreases monotonically for both algorithms.

Four first figure show the results of the simulations for $\omega \notin \mathcal{X}$. First observation from Figure 5.1 is that both algorithms are sensitive to the noise levels in low dimension (i.e. parameter n). Another observation is that our algorithm gives a slightly better result in high noise levels while the PGD/SGD works better in the low noises. In the case of no-noise, they have the same performance but our algorithm shows a more smooth convergence curve. By gradually increasing the dimension from 10 in Figure 5.1 to 500 in the next three figures: Figure 5.2, Figure 5.3, and Figure 5.4, both algorithms start to show less sensitivity to the noise levels. Still our algorithm gives a slightly better result in high noise levels while the PGD/SGD works better in the low noise levels.

The next four first figure show the results of the simulations for $\omega \in \mathcal{X}$. First observation from Figure 5.5 is that both algorithms are sensitive to the noise levels in low dimension (i.e. parameter n). While both algorithms perform similarly in the presence of the noise, PGD/SGD algorithm shows a very non-smooth convergence curve in the absence of the noise. By gradually increasing the dimension from 10 in Figure 5.5 to 500 in the next three figures: Figure 5.6, Figure 5.7, and Figure 5.8, both algorithms perform similarly in the presence of the noise. PGD/SGD algorithm shows a non-smooth and inferior convergence curve in the absence of the noise.

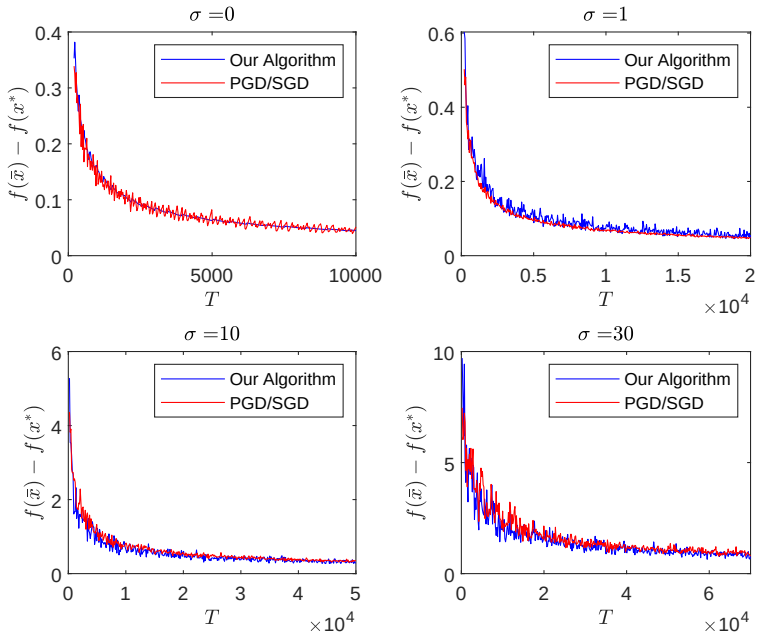

FIG. 5.1. *Dimension $n = 10$ and ω is outside the set $\mathcal{X}$. Both algorithms show a lot of sensitivity to the noise levels in low dimension. Another observation is that our algorithm gives a slightly better result in high noise levels while the PGD/SGD works better in the low noises. In the case of no-noise, they have the same performance but our algorithm shows a more smooth convergence curve.*

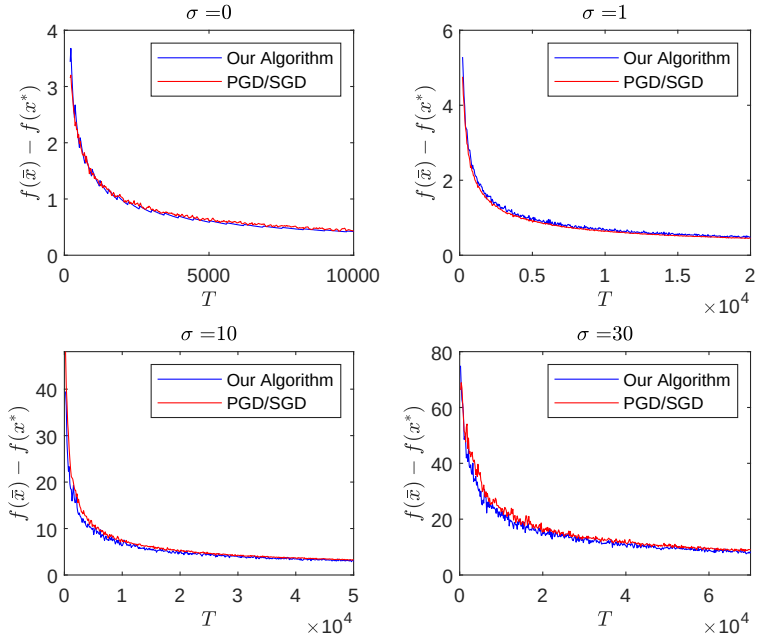

FIG. 5.2. *Dimension $n = 100$ and ω is outside the set $\mathcal{X}$. Both algorithms show less sensitivity to the noise levels in a higher dimension. Again, our algorithm gives a slightly better result in high noise levels while the PGD/SGD works better in the low noises.*

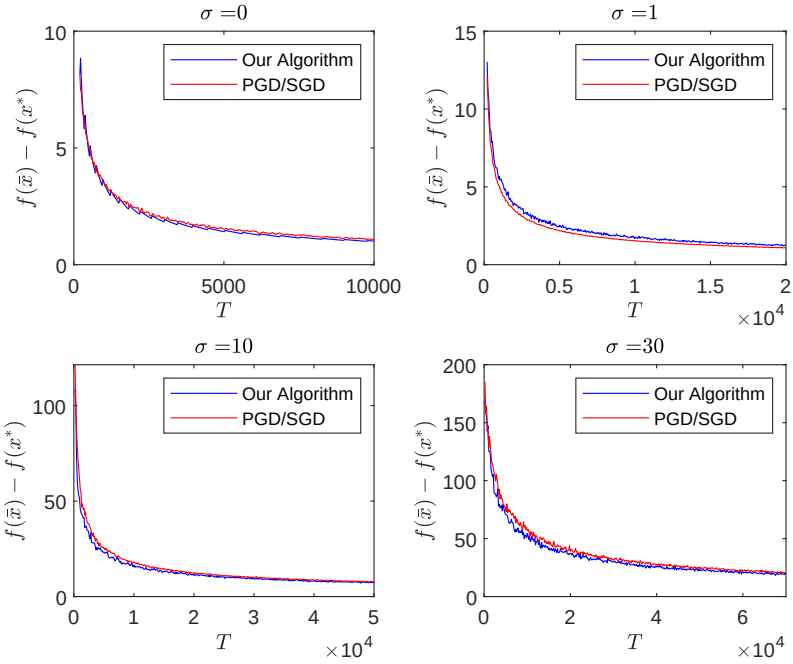

FIG. 5.3. *Dimension $n = 250$ and ω is outside the set $\mathcal{X}$. Both algorithms show less sensitivity to the noise levels in a higher dimension. Again, our algorithm gives a slightly better result in high noise levels while the PGD/SGD works better in the low noises.*

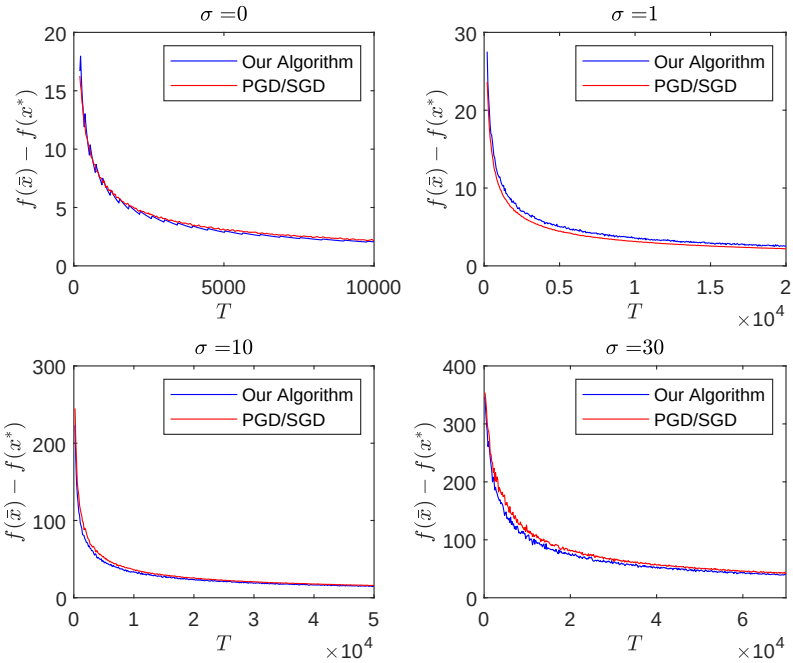

FIG. 5.4. *Dimension $n = 500$ and ω is outside the set $\mathcal{X}$. Both algorithms show less sensitivity to the noise levels in a higher dimension. Again, our algorithm gives a slightly better result in high noise levels while the PGD/SGD works better in the low noises.*

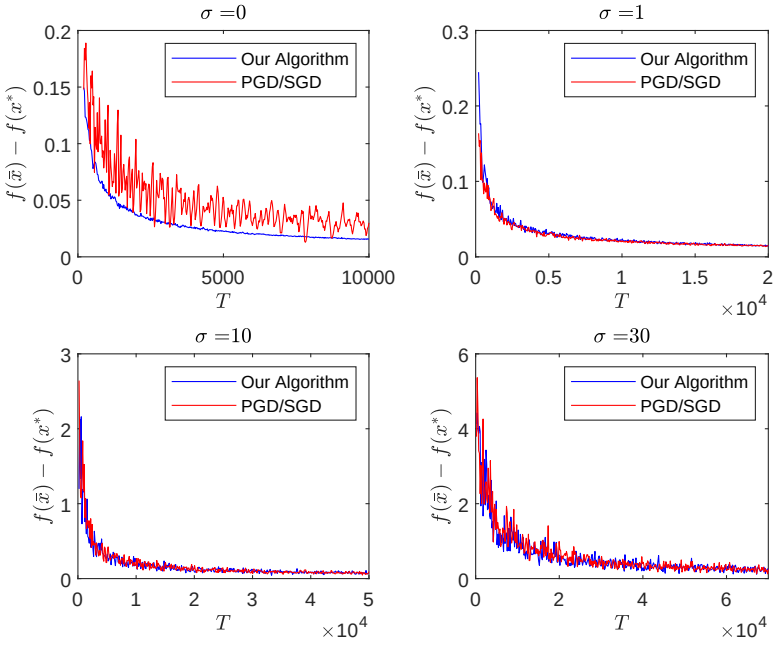

FIG. 5.5. *Dimension $n = 10$ and ω is inside the set $\mathcal{X}$. Both algorithms show a lot of sensitivity to the noise levels in low dimension. While both algorithms perform similarly in the presence of the noise, PGD/SGD algorithm shows a very non-smooth convergence curve in the absence of the noise.*

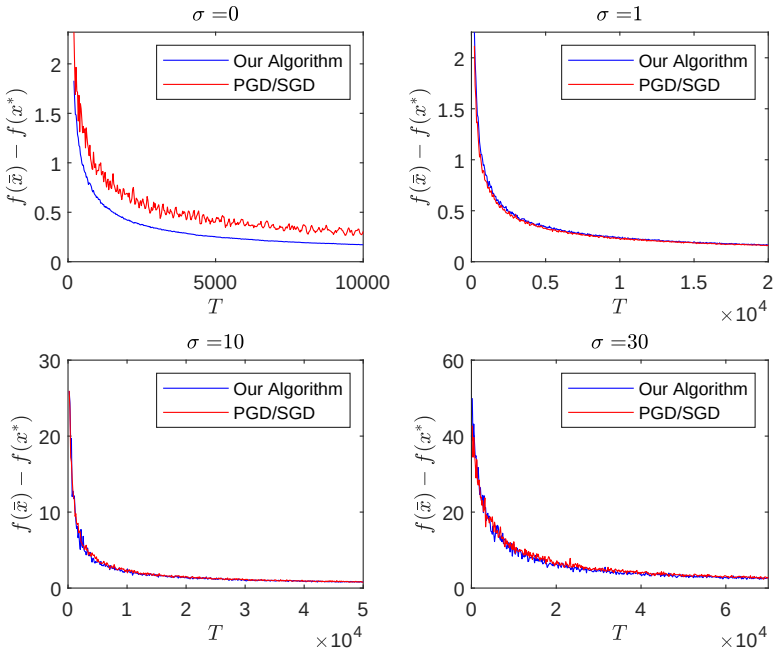

FIG. 5.6. *Dimension $n = 100$ and ω is inside the set $\mathcal{X}$. Again, both algorithms perform similarly in the presence of the noise, PGD/SGD algorithm shows a non-smooth and inferior convergence curve in the absence of the noise.*

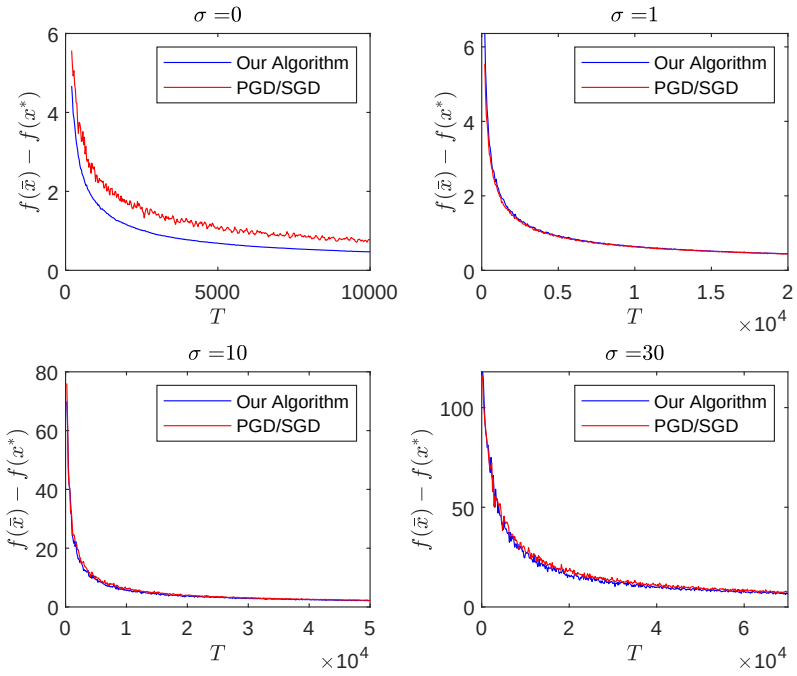

FIG. 5.7. *Dimension $n = 250$ and ω is inside the set $\mathcal{X}$. Again, both algorithms perform similarly in the presence of the noise, PGD/SGD algorithm shows a non-smooth and inferior convergence curve in the absence of the noise.*

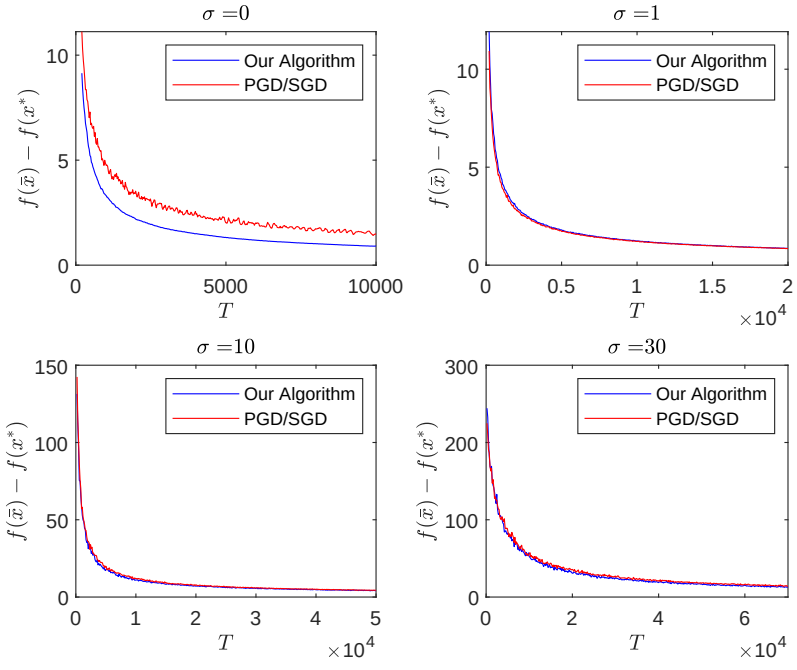

FIG. 5.8. *Dimension $n = 500$ and ω is inside the set $\mathcal{X}$.Again, both algorithms perform similarly in the presence of the noise, PGD/SGD algorithm shows a non-smooth and inferior convergence curve in the absence of the noise.*

5.2. Bounded nuclear norm domain with $L1$-norm objective function. The bounded nuclear norm domain, is a well known example in which the linear optimization has significant computational advantage over projection.

DEFINITION 5.1 (Singular Value Decomposition). *SVD of a real valued $m \times n$ matrix A is a factorization of the form $A = USV^\top$, where U is an $m \times m$ orthogonal matrix, S is an diagonal matrix with non-negative real numbers on the diagonal, and V is an $n \times n$ orthogonal matrix. The diagonal entries $s_i = S_{i,i}$ are uniquely determined by A and are known as the singular values of A. Also, let us call the i-th column of matrix U by u_i and the i-th column of matrix V by v_i.*

DEFINITION 5.2 (Nuclear norm). *Nuclear norm or Trace norm of a $m \times n$ matrix A with SVD $A = USV^\top$ is defined as:*

$$\|A\|_* = \sum_{i=1}^{\min(m,n)} s_i$$

Now let us define the set $\mathcal{X}$ of all real valued $m \times n$ matrices with nuclear norm smaller that τ. Fix $\tau > 0$,

$$\mathcal{X} \stackrel{def}{=} \{A \in \mathbb{R}^{m \times n} : \|A\|_* \leq \tau\}$$

Until this section, algorithms were defined in the vector form. However, they can work for matrices just by rearranging the matrices into vector forms. Frobenius norm of a matrix is equal to its vector form L_2-norm.

DEFINITION 5.3 (Frobenius norm). *Frobenius norm can be defined as:*

$$\|A\|_F = \sqrt{\sum_{i,j=1}^{m,n} |a_{i,j}|^2}$$

LEMMA 5.4. *Frobenius norm can be upper bounded by Nuclear norm:*

$$\|A\|_F \leq \|A\|_*$$

Proof. See [8]. ∎

Now we are ready to come up with a enclosing Euclidean ball for the set $\mathcal{X}$. If we choose $X_1 = \mathbf{0}$ then

$$\|X_1 - Z\|_F = \|Z\|_F \leq \|Z\|_* \leq \tau, \quad \forall Z \in \mathcal{X}$$

Let us choose the objective function

$$f_W(X) = \sum_{i=1}^{m} \sum_{j=1}^{n} |X_{i,j} - W_{i,j}|$$

It is easy to see that this function is convex and Lipschitz continuous with coefficient

$$G = \sqrt{nm}$$

When the exact sub-gradient is given to us without any error, we choose the parameters as before: $\beta = \frac{R}{G\sqrt{T}}$, $\alpha = \frac{G\sqrt{T}}{R}$, $\eta = \frac{G}{2R\sqrt{T}}$.

In order to simulate inexact sub-gradients, we added a mean-zero normal distribution with covariance matrix $\sigma^2 I_{nm}$ to the exact sub-gradients, i.e., $\hat{g}(y_k) = g(y_k) + N_k$ where $\{N_k\}$ are i.i.d, nm vectors, samples of $\mathcal{N}(0, \sigma^2 I_{nm})$. Thus,

$$\mathbb{E}\{\|\hat{g}(x)\|^2\} = G^2 + nm\sigma^2$$

so

$$B = \sqrt{G^2 + nm\sigma^2}$$

Thus the parameters are: $\beta = \frac{R}{B\sqrt{T}}$, $\alpha = \frac{B\sqrt{T}}{R}$, $\eta = \frac{G}{2R\sqrt{T}}$.

We used the two following lemmas to implement our algorithm and PGD/SGD:

LEMMA 5.5 (Linear optimization on nuclear-norm ball). *Fix a parameter $\tau > 0$. Let A be a $m \times n$ matrix and consider its singular-value decomposition $A = USV^\top$. Then*

$$\text{Argmin}_{\|X\|_* \leq \tau} \langle A, X \rangle_F = \tau u_1 v_1^\top$$

Proof. See [14]. ∎

LEMMA 5.6 (Projection onto the nuclear-norm ball). *Fix a parameter $\tau > 0$. Let A be a $m \times n$ matrix and consider its singular-value decomposition $A = USV^\top$. If $\|A\|_* \geq \tau$, then the Euclidean projection of A onto the nuclear-norm ball is given by*

$$\pi_{\|X\|_* \leq \tau}(A) = \sum_{i=1}^{\min(m,n)} \max(0, s_i - \lambda) u_i v_i^\top$$

where $\lambda \geq 0$ is the solution to the equation

$$\sum_{i=1}^{\min(m,n)} \max(0, s_i - \lambda) = \tau$$

Proof. See [2], [7]. ∎

The experiments were done for different noise levels, σ, different dimensions, and for different values of T. Two cases are considered in which the parameter W is inside or outside of the domain set, $\mathcal{X}$. When W is inside, it is the optimal point. However, when the W is outside, then the optimal point is not calculable analytically (unlike the first experiment) so we only demonstrate the value of the function $f(\bar{x})$ instead of the error $f(\bar{x}) - f(x^*)$. Clearly, in all of the figures, by increasing T, the error decreases monotonically for both algorithms.

Four first figure show the results of the simulations for $W \notin \mathcal{X}$. First observation from Figure 5.9 is that PGD/SGD shows a superior result over our algorithm in the low noise levels. This superiority vanishes in the higher noise levels. By gradually increasing the dimensions in the next three figures: Figure 5.10, Figure 5.11, and Figure 5.12, PGD/SGD, still, shows a superior result over our algorithm in the low noise levels. This superiority vanishes in the higher noise levels and our algorithm starts to do a better job. Clearly, PGD/SGD is more sensitive to the noise level.

The next four first figure show the results of the simulations for $W \in \mathcal{X}$. First observation from Figure 5.13 is that in the absence of the noise, PGD/SDG does an inferior job. However, introducing small amount of noise improves the SGD/PGD performance dramatically. In other words, our algorithm is showing a superior result over PGD/SDG in the low noise levels, this superiority vanishes in the higher noise

levels. By gradually increasing the dimension in the next three figures: Figure 5.14, Figure 5.15, and Figure 5.16, still, in the absence of the noise, PGD/SDG does an inferior job. Again, this superiority vanishes in the higher noise levels. One more time, introducing small amount of noise improve the SGD/PGD performance dramatically.

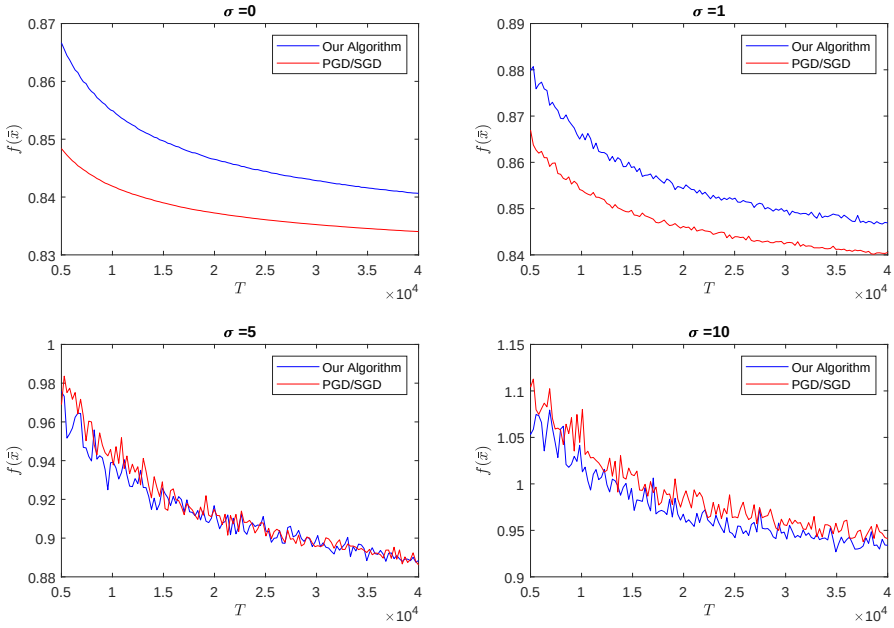

FIG. 5.9. *Parameters are $n = 5$, $m = 5$. W is outside of the set $\mathcal{X}$. PGD/SGD is showing a superior result over our algorithm in the low noise levels. This superiority vanishes in the higher noise levels.*

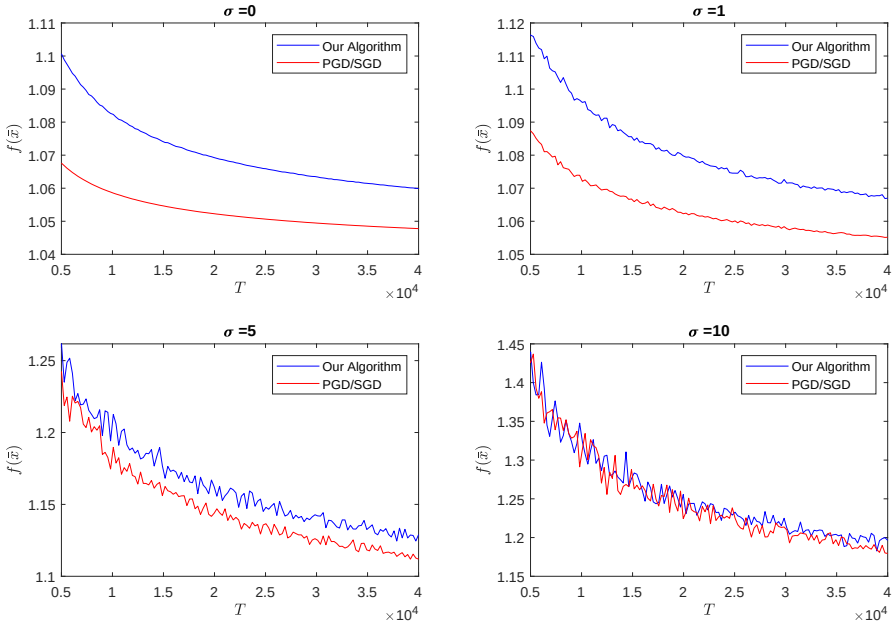

FIG. 5.10. *Parameters are $n = 5$, $m = 10$. W is outside of the set $\mathcal{X}$. PGD/SGD is showing a superior result over our algorithm in the low noise levels. This superiority vanishes in the higher noise levels.*

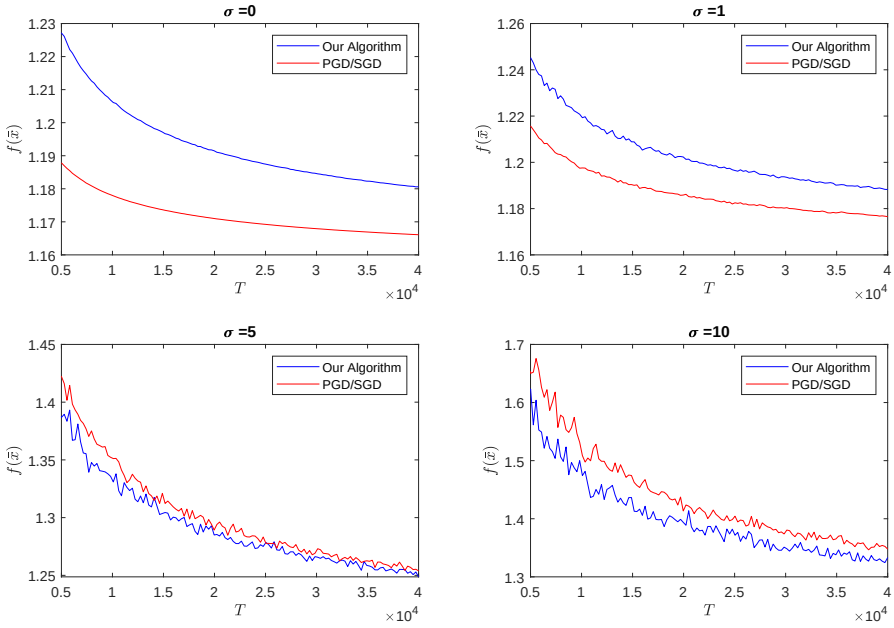

FIG. 5.11. *Parameters are $n = 10$, $m = 10$. W is outside of the set $\mathcal{X}$. PGD/SGD is showing a superior result over our algorithm in the low noise levels. This superiority vanishes in the higher noise levels and our algorithm does a better job. Clearly, PGD/SGD is more sensitive to the noise level.*

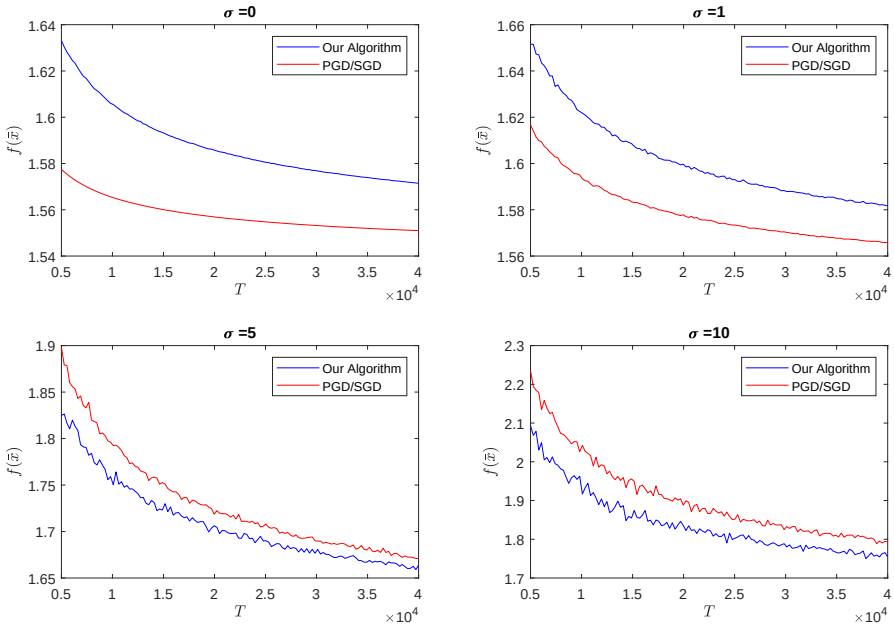

FIG. 5.12. *Parameters are $n = 10$, $m = 20$. W is outside of the set $\mathcal{X}$. PGD/SGD is showing a superior result over our algorithm in the low noise levels. This superiority vanishes in the higher noise levels and our algorithm does a better job. Clearly, PGD/SGD is more sensitive to the noise level.*

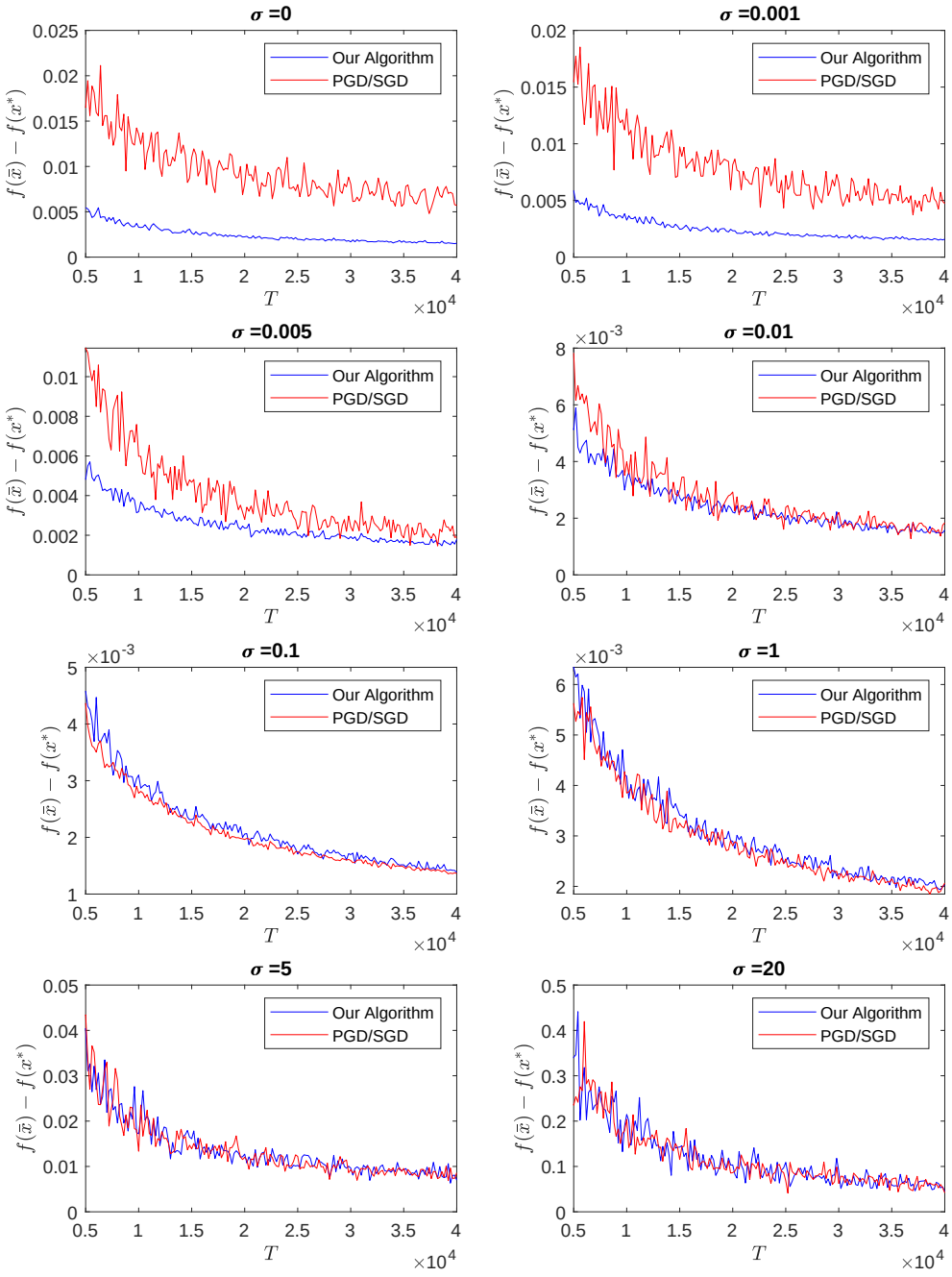

FIG. 5.13. *Parameters are $n = 5$, $m = 5$. W is inside of the set $\mathcal{X}$. In the absence of the noise, PGD/SDG does an inferior job. However, introducing small amount of noise improves the SGD/PGD performance dramatically. In other words, our algorithm is showing a superior result over PGD/SDG in the low noise levels, this superiority vanishes in the higher noise levels.*

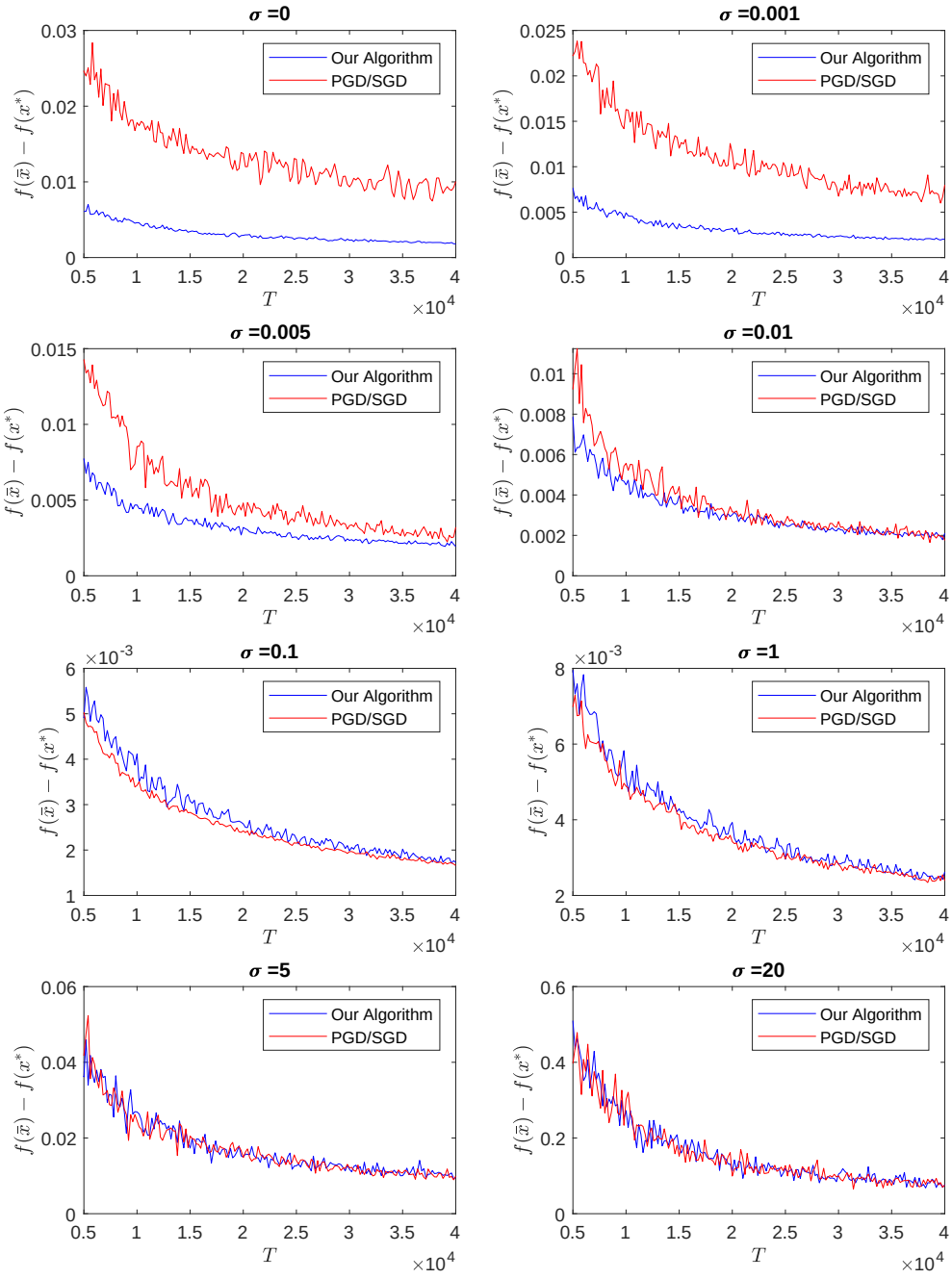

FIG. 5.14. *Parameters are $n = 5$, $m = 10$. W is inside of the set $\mathcal{X}$. In the absence of the noise, PGD/SDG does an inferior job. However, introducing small amount of noise improve the SGD/PGD performance dramatically. In other words, our algorithm is showing a superior result over PGD/SDG in the low noise levels, this superiority vanishes in the higher noise levels.*

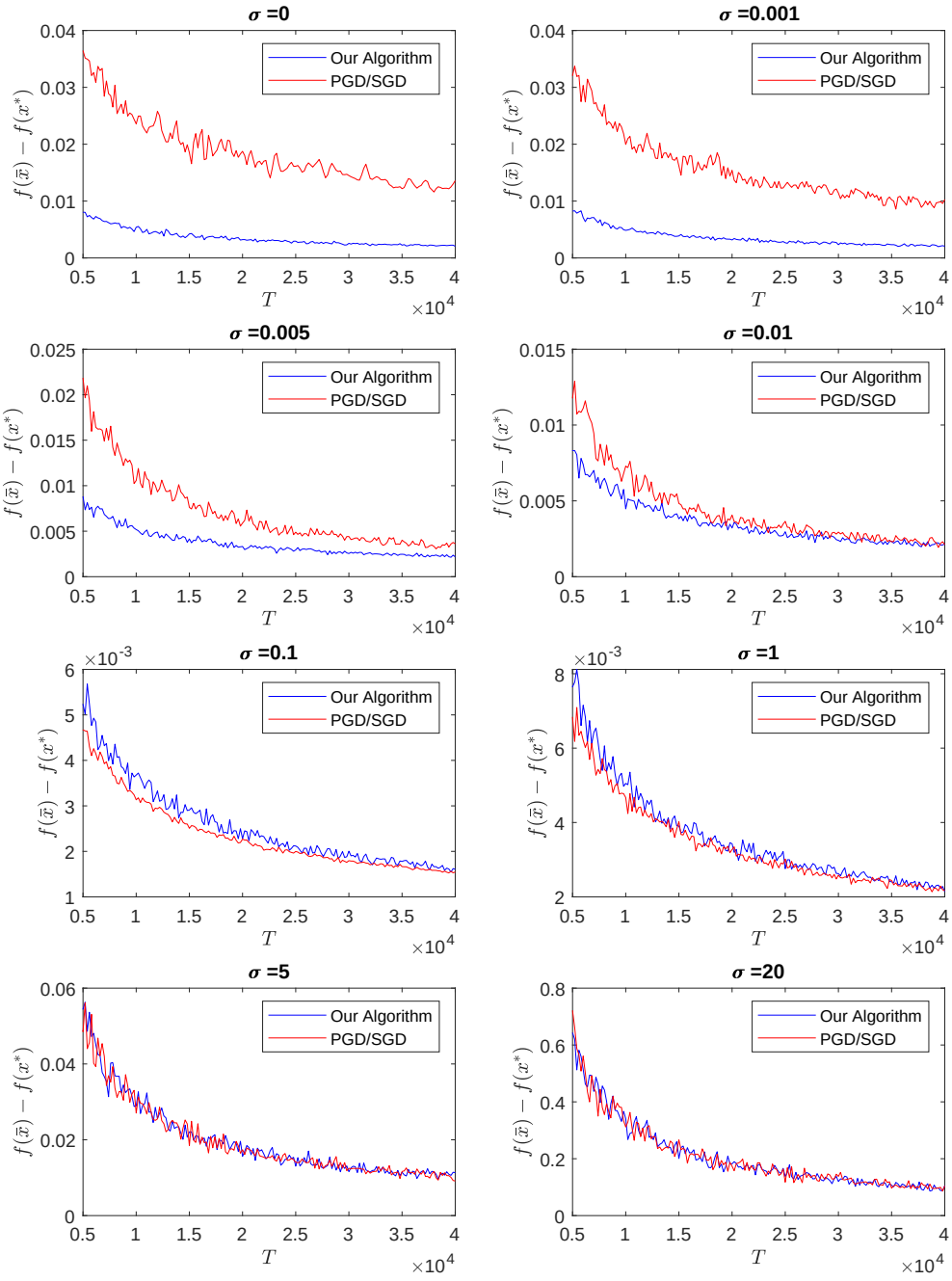

FIG. 5.15. *Parameters are $n = 10$, $m = 10$. W is inside of the set $\mathcal{X}$. In the absence of the noise, PGD/SDG does an inferior job. However, introducing small amount of noise improve the SGD/PGD performance dramatically. In other words, our algorithm is showing a superior result over PGD/SDG in the low noise levels, this superiority vanishes in the higher noise levels.*

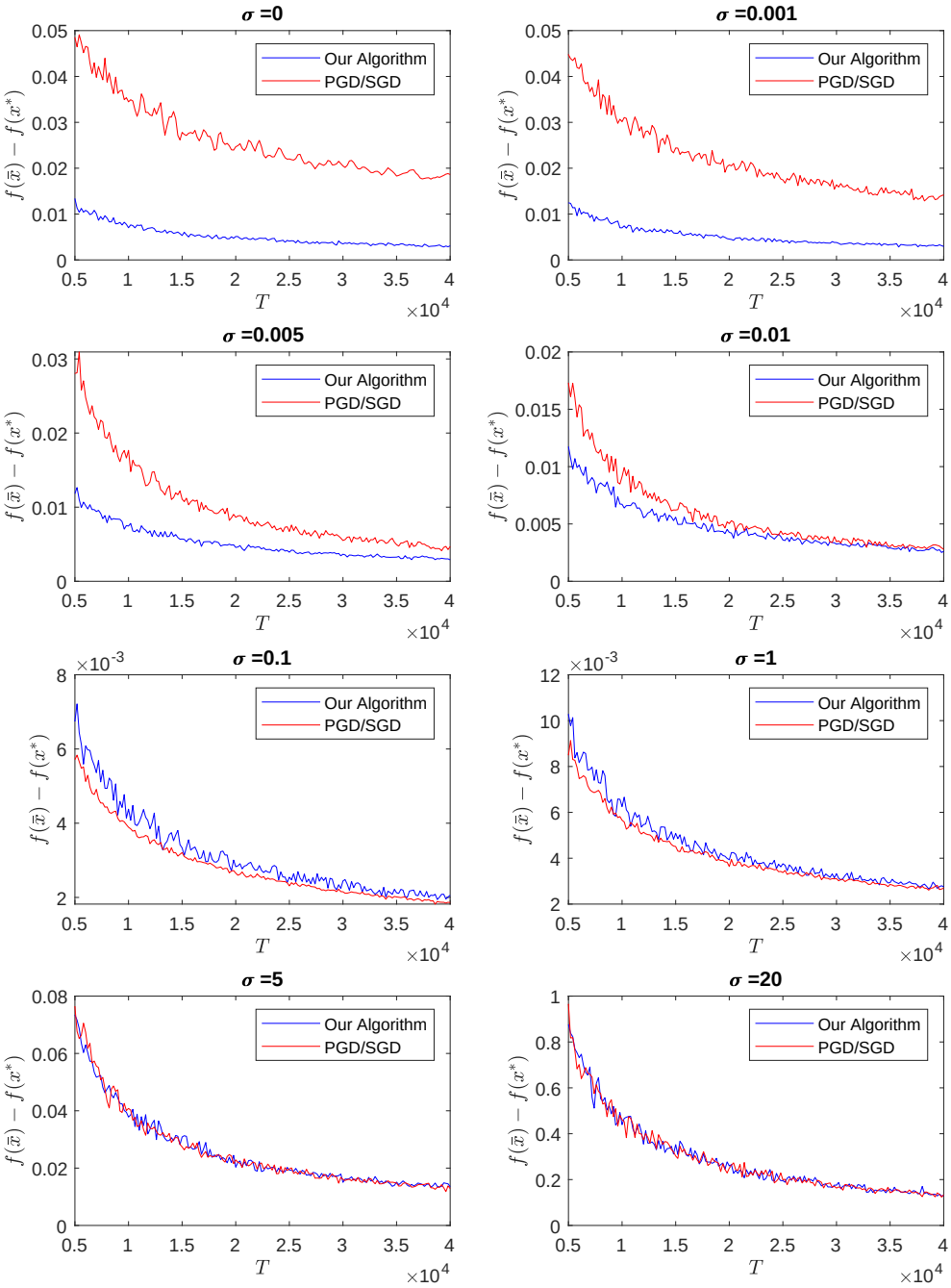

FIG. 5.16. *Parameters are $n = 10$, $m = 20$. W is inside of the set $\mathcal{X}$. In the absence of the noise, PGD/SDG does an inferior job. However, introducing small amount of noise improve the SGD/PGD performance dramatically. In other words, our algorithm is showing a superior result over PGD/SDG in the low noise levels, this superiority vanishes in the higher noise levels.*

6. Conclusions and Open Problems. This paper addresses the problem of solving general constrained convex optimization without using projection. Prior works on projection-free algorithms focus mainly on smooth problems and/or problems with special constraint structure. The simulations and the convergence theorems show that our algorithm's performance is comparable with the PGD/SGD algorithm, thus it can be a substitute for the PGD/SGD algorithm if projection-free algorithms are favoured.

An open problem is whether our algorithm can be modified to exploit the gains of mirror descent [4, 3]. A well known algorithm that replaces the Euclidean norm in PGD/SGD with a Bregman divergence which results in improved performance with respect to system dimension in certain cases (such as when the domain set is a probability simplex). Another question arises considering that PGD/SGD with time varying step-size reaches the optimal rate, $O(1/T)$, for non-smooth strongly convex optimization. It is unclear how to extend our algorithm in this scenario.

Appendix A. Proof of the Theorem 2.1.

Proof. This proof is collected from a larger *McShane-Whitney extension theorem* of [20] and presented in a condense format.

(i) Fix w in $\mathbb{R}^n$. For fixed z in the set $\mathcal{X}$ we have for every x in the set $\mathcal{X}$:

$$\begin{aligned}
&f(x) + G\|x - w\| \\
&\geq f(z) - G\|z - x\| + G\|x - w\|, \quad \text{(By Lipschitz)} \\
&\geq f(z) - G\|z - x\| + G\|x - z\| - G\|z - w\|, \quad \text{(triangle inequality)} \\
&= f(z) - G\|z - w\|
\end{aligned}$$

Then taking the infimum over x in the set $\mathcal{X}$ gives

$$f(w) \geq f(z) - G\|z - w\| > -\infty$$

(ii) Let $w_1, w_2 \in \mathbb{R}^n$ and $x \in \mathcal{X}$. Then $\|w_1 - x\| \leq \|w_2 - x\| + \|w_1 - w_2\|$, therefore

$$\begin{aligned}
&f(x) + G\|w_1 - x\| \leq f(x) + G\|w_2 - x\| + G\|w_1 - w_2\| \Rightarrow \\
&\tilde{f}(w_1) \leq f(x) + G\|w_2 - x\| + G\|w_1 - w_2\| \Rightarrow \\
&\tilde{f}(w_1) - G\|w_1 - w_2\| \leq f(x) + G\|w_2 - x\| \Rightarrow \\
&\tilde{f}(w_1) - G\|w_1 - w_2\| \leq \tilde{f}(w_2) \Rightarrow \\
&\tilde{f}(w_1) - \tilde{f}(w_2) \leq G\|w_1 - w_2\|
\end{aligned}$$

If we start from the inequality $\|w_2 - x\| \leq \|w_1 - x\| + \|w_1 - w_2\|$ and work as above, we get $\tilde{f}(w_1) - \tilde{f}(w_2) \leq G\|w_1 - w_2\|$ therefore $|\tilde{f}(w_1) - \tilde{f}(w_2)| \leq G\|w_1 - w_2\|$.

(iii) Fix x, and $g \in \partial\tilde{f}(x)$. For any y We have

$$\tilde{f}(x) + \langle g, y - x\rangle \leq \tilde{f}(y) \leq \tilde{f}(x) + G\|y - x\|$$

Now let $y = x + \epsilon g$ for $\epsilon > 0$, then

$$\langle g, \epsilon g\rangle = \epsilon\|g\|^2 \leq G\|\epsilon g\|$$

so

$$\|g\| \leq G$$

(iv) Let $w_1, w_2 \in \mathbb{R}^n$ and $0 \leq \theta \leq 1$

$$\begin{aligned}
&\theta \tilde{f}(w_1) + (1-\theta)\tilde{f}(w_2) \\
=&\theta \inf_{x_1 \in \mathcal{X}} \{f(x_1) + G\|w_1 - x_1\|\} + (1-\theta) \inf_{x_2 \in \mathcal{X}} \{f(x_2) + G\|w_2 - x_2\|\} \\
=& \inf_{x_1, x_2 \in \mathcal{X}} \{\theta f(x_1) + \theta G\|w_1 - x_1\| + (1-\theta) f(x_2) + (1-\theta) G\|w_2 - x_2\|\} \\
\geq& \inf_{x_1, x_2 \in \mathcal{X}} \{f(\theta x_1 + (1-\theta)x_2) + G\|\theta w_1 - \theta x_1 + (1-\theta)w_2 - (1-\theta)x_2\|\} \\
=& \inf_{y \in \mathcal{X}} \{f(y) + G\|\theta w_1 + (1-\theta)w_2 - y\|\} \\
=&\tilde{f}(\theta w_1 + (1-\theta)w_2)
\end{aligned}$$

(v) We show that $\tilde{f}$ extends f. If $w \in \mathcal{X}$, then

$$\tilde{f}(w) = \inf_{x \in \mathcal{X}} \{f(x) + G\|w - x\|\} \leq f(w) + G\|w - w\| = f(w)$$

on the other hand, if $w \in \mathcal{X}$, then

$$\tilde{f}(w) = \inf_{x \in \mathcal{X}} \{f(x) + G\|w - x\|\} \overset{a}{\geq} \inf_{x \in \mathcal{X}} \{f(w)\} = f(w)$$

where (a) is a direct use of Lipschitz continuity's definition. Thus, $\forall w \in \mathcal{X}$ we have $\tilde{f}(w) = f(w)$ ∎

Appendix B. Proof of the Theorem 3.1.

LEMMA B.1. *Let function f be convex on the convex set C. Fix $\alpha > 0$, $x_0 \in C$. Suppose*

$$x' = \text{Argmin}_{x \in C} \left\{f(x) + \theta\|x - x_0\|^2\right\}$$

then $\forall z \in C$

$$f(x') + \theta\|x' - x_0\|^2 \leq f(z) + \theta\|z - x_0\|^2 - \theta\|z - x'\|^2$$

Proof. A special case of the Lemma 6 of [19]. ∎

LEMMA B.2. *In Algorithm 3.1, $\forall t \geq 1$ we have:*

$$Q_t = \sum_{k=1}^{t} y_k - \sum_{k=1}^{t} x_k$$

Proof. Just combine Line 12 and Line 10 from Algorithm 3.1 and sum over time. ∎

LEMMA B.3. *For any $a, b \in \mathbb{R}^n$ and $\theta > 0$,*

$$\langle a, b\rangle + \frac{\theta}{2}\|a\|^2 \geq -\frac{\|b\|^2}{2\theta}$$

Proof. Combine Cauchy–Schwarz inequality:

$$\langle a, b\rangle \geq -\|a\|\|b\|$$

with

$$\frac{\theta}{2}\|a\|^2 + \frac{\|b\|^2}{2\theta} - \|a\|\|b\| = \left(\sqrt{\frac{\theta}{2}}\|a\| + \sqrt{\frac{1}{2\theta}}\|b\|\right)^2 \geq 0$$

∎

Proof of the Theorem 3.1. It is easy to verify that Line 15 from Algorithm 3.1 can be written as:

$$y_{k+1} = \text{Argmin}_{y\in\mathbb{R}^n} \left\{ \langle \eta Q_k + g_k, y\rangle + \frac{\alpha}{2}\|y - y_k\|^2 + \frac{\eta}{2}\|y - x_{k+1}\|^2 \right\} \tag{B.1}$$

Notice the function:

$$\langle \eta Q_k + g_k, y\rangle + \frac{\eta}{2}\|y - x_{k+1}\|^2$$

is convex, thus Equation (B.1) satisfies the Lemma B.1 so we get:

$$\begin{aligned}\langle \eta Q_k + g_k, y_{k+1}\rangle + \frac{\alpha}{2}\|y_{k+1} - y_k\|^2 + \frac{\eta}{2}\|y_{k+1} - x_{k+1}\|^2 \\ \leq \langle \eta Q_k + g_k, x^*\rangle + \frac{\alpha}{2}\|x^* - y_k\|^2 - \frac{\alpha}{2}\|x^* - y_{k+1}\|^2 + \frac{\eta}{2}\|x^* - x_{k+1}\|^2\end{aligned}$$

We have $\|x_{k+1} - x^*\| \leq 2R$ as $x_{k+1}, x^* \in \mathcal{X}$,

$$\begin{aligned}\langle \eta Q_k + g_k, y_{k+1}\rangle + \frac{\alpha}{2}\|y_{k+1} - y_k\|^2 + \frac{\eta}{2}\|y_{k+1} - x_{k+1}\|^2 \\ \leq \langle \eta Q_k + g_k, x^*\rangle + \frac{\alpha}{2}\|x^* - y_k\|^2 - \frac{\alpha}{2}\|x^* - y_{k+1}\|^2 + 2\eta R^2\end{aligned}$$

Add $\langle \eta Q_k, -x_{k+1}\rangle$ and $\langle g_k, -y_k\rangle$ to both sides

$$\begin{aligned}\langle \eta Q_k, y_{k+1} - x_{k+1}\rangle + \langle g_k, y_{k+1} - y_k\rangle + \frac{\alpha}{2}\|y_{k+1} - y_k\|^2 \\ \leq \langle \eta Q_k, x^* - x_{k+1}\rangle + \langle g_k, x^* - y_k\rangle + \frac{\alpha}{2}\|x^* - y_k\|^2 - \frac{\alpha}{2}\|x^* - y_{k+1}\|^2 + 2\eta R^2\end{aligned}$$

An easy consequence of Line 14 from Algorithm 3.1 is: $\langle Q_k, x_{k+1}\rangle \geq \langle Q_k, x^*\rangle$ for all $x^* \in \mathcal{X}$, thus

$$\begin{aligned}\langle \eta Q_k, y_{k+1} - x_{k+1}\rangle + \langle g_k, y_{k+1} - y_k\rangle + \frac{\alpha}{2}\|y_{k+1} - y_k\|^2 + \frac{\eta}{2}\|y_{k+1} - x_{k+1}\|^2 \\ \leq \langle g_k, x^* - y_k\rangle + \frac{\alpha}{2}\|x^* - y_k\|^2 - \frac{\alpha}{2}\|x^* - y_{k+1}\|^2 + 2\eta R^2\end{aligned}$$

A simple conclusion from Equation Line 12 from Algorithm 3.1 is:

$$\langle Q_k, y_{k+1} - x_{k+1}\rangle = \frac{1}{2}\|Q_{k+1}\|^2 - \frac{1}{2}\|Q_k\|^2 - \frac{1}{2}\|y_{k+1} - x_{k+1}\|^2$$

so we get:

$$\begin{aligned}\frac{\eta}{2}\|Q_{k+1}\|^2 - \frac{\eta}{2}\|Q_k\|^2 + \langle g_k, y_{k+1} - y_k\rangle + \frac{\alpha}{2}\|y_{k+1} - y_k\|^2 \\ \leq \langle g_k, x^* - y_k\rangle + \frac{\alpha}{2}\|x^* - y_k\|^2 - \frac{\alpha}{2}\|x^* - y_{k+1}\|^2 + 2\eta R^2\end{aligned}$$

Using Lemma B.3 we get:

$$\begin{aligned}\frac{\eta}{2}\|Q_{k+1}\|^2 - \frac{\eta}{2}\|Q_k\|^2 - \frac{\|g_k\|^2}{2\alpha} \\ \leq \langle g_k, x^* - y_k\rangle + \frac{\alpha}{2}\|x^* - y_k\|^2 - \frac{\alpha}{2}\|x^* - y_{k+1}\|^2 + 2\eta R^2\end{aligned}$$

Sum from $k=1$ to $k=T-1$ to get

$$\frac{\eta}{2}\|Q_T\|^2 - \frac{\eta}{2}\|Q_1\|^2 - (T-1)2\eta R^2 - \frac{1}{2\alpha}\sum_{k=1}^{T-1}\|g_k\|^2 \\ \leq \sum_{k=1}^{T-1}\langle g_k, x^* - y_k\rangle + \frac{\alpha}{2}\|x^* - y_1\|^2 - \frac{\alpha}{2}\|x^* - y_T\|^2$$

Combining Line 9 and (2.4) from Algorithm 3.1 gives us: $\|x^* - y_1\| \leq R$, thus

$$\frac{\eta}{2}\|Q_T\|^2 - \frac{\eta}{2}\|Q_1\|^2 - (T-1)2\eta R^2 - \frac{1}{2\alpha}\sum_{k=1}^{T-1}\|g_k\|^2 \\ \leq \sum_{k=1}^{T-1}\langle g_k, x^* - y_k\rangle + \frac{\alpha R^2}{2} - \frac{\alpha}{2}\|x^* - y_T\|^2$$

Combining Line 12 and Line 9 from Algorithm 3.1 gives us: $Q_1 = Q_0 + y_1 - x_1 = \mathbf{0}$, thus

$$\frac{\eta}{2}\|Q_T\|^2 - (T-1)2\eta R^2 - \frac{1}{2\alpha}\sum_{k=1}^{T-1}\|g_k\|^2 \\ \leq \sum_{k=1}^{T-1}\langle g_k, x^* - y_k\rangle + \frac{\alpha R^2}{2} - \frac{\alpha}{2}\|x^* - y_T\|^2$$

Add and subtract $\langle g_T, x^* - y_T\rangle$ to the right hand side,

$$\frac{\eta}{2}\|Q_T\|^2 - (T-1)2\eta R^2 - \frac{1}{2\alpha}\sum_{k=1}^{T-1}\|g_k\|^2 \\ \leq \sum_{k=1}^{T}\langle g_k, x^* - y_k\rangle + \frac{\alpha R^2}{2} - \frac{\alpha}{2}\|x^* - y_T\|^2 - \langle g_T, x^* - y_T\rangle$$

Again using Lemma B.3 we get,

$$\frac{\eta}{2}\|Q_T\|^2 - (T-1)2\eta R^2 - \frac{1}{2\alpha}\sum_{k=1}^{T-1}\|g_k\|^2 \leq \sum_{k=1}^{T}\langle g_k, x^* - y_k\rangle + \frac{\alpha R^2}{2} + \frac{\|g_T\|^2}{2\alpha}$$

Rearranging,

$$\sum_{k=1}^{T}\langle g_k, x^* - y_k\rangle + \frac{\eta}{2}\|Q_T\|^2 \leq \frac{\alpha R^2}{2} + \frac{1}{2\alpha}\sum_{k=1}^{T}\|g_k\|^2 + (T-1)2\eta R^2 \tag{B.2}$$

Using convexity of f we get

$$\sum_{k=0}^{T}(f(y_k) - f(x^*)) + \frac{\eta}{2}\|Q_T\|^2 \leq \frac{\alpha R^2}{2} + \frac{1}{2\alpha}\sum_{k=1}^{T}\|g_k\|^2 + (T-1)2\eta R^2$$

Using G-Lipschitz continuity of f we get

$$\sum_{k=0}^{T}(f(y_k)-f(x^*))+\frac{\eta}{2}\|Q_T\|^2\leq\frac{\alpha R^2}{2}+T\frac{G^2}{2\alpha}+(T-1)2\eta R^2$$

Dividing by T and using Jensen inequality

$$\begin{aligned}&f\left(\frac{1}{T}\sum_{k=1}^{T}y_k\right)-f(x^*)+\frac{1}{T}\frac{\eta}{2}\|Q_T\|^2\\ \leq&\frac{\alpha R^2}{2T}+\frac{G^2}{2\alpha}+\frac{T-1}{T}2\eta R^2\\ \leq&\frac{\alpha R^2}{2T}+\frac{G^2}{2\alpha}+2\eta R^2\end{aligned}$$

Similar to Line 17 from Algorithm 3.1, define $\bar{y}=\frac{1}{T}\sum_{k=1}^{T}$. Add and subtract $f(\bar{x})$ to the left hand side (LHS),

$$f(\bar{x})-f(x^*)+f(\bar{y})-f(\bar{x})+\frac{1}{T}\frac{\eta}{2}\|Q_T\|^2\leq\frac{\alpha R^2}{2T}+\frac{G^2}{2\alpha}+2\eta R^2$$

Using Lemma B.2 and G-Lipschitz continuity of f we get

$$f(\bar{x})-f(x^*)-G\|\bar{y}-\bar{x}\|+T\frac{\eta}{2}\|\bar{y}-\bar{x}\|^2\leq\frac{\alpha R^2}{2T}+\frac{G^2}{2\alpha}+2\eta R^2$$

By completing the square we have

$$f(\bar{x})-f(x^*)\leq\frac{\alpha R^2}{2T}+\frac{G^2}{2\alpha}+2\eta R^2+\frac{G^2}{2\eta T}$$

Choose

$$\alpha=\frac{G\sqrt{T}}{R}$$

and

$$\eta=\frac{2G}{R\sqrt{T}}$$

which means

$$f(\bar{x})-f(x^*)\leq\frac{3RG}{\sqrt{T}}$$

□

Appendix C. Proof of the Theorem 4.1.

Proof of the Theorem 4.1. The analysis is the same up to Equation (B.2). Taking expectations of both sides of Equation (B.2):

$$\frac{\eta}{2}\mathbb{E}\left\{\|Q_T\|^2\right\}-(T-1)2\eta R^2\leq\sum_{k=1}^{T}\mathbb{E}\left\{\langle g_k,x^*-Y_k\rangle\right\}+\frac{\alpha R^2}{2}+\frac{1}{2\alpha}\sum_{k=1}^{T}\mathbb{E}\left\{\|g_k\|^2\right\}$$

Replacing $g_k=\hat{g}(Y_k)$,

$$\begin{aligned}&\frac{\eta}{2}\mathbb{E}\left\{\|Q_T\|^2\right\}-(T-1)2\eta R^2\\ &\qquad\leq\sum_{k=1}^{T}\mathbb{E}\left\{\langle\hat{g}(Y_k),x^*-Y_k\rangle\right\}+\frac{\alpha R^2}{2}+\frac{1}{2\alpha}\sum_{k=1}^{T}\mathbb{E}\left\{\|\hat{g}(Y_k)\|^2\right\}\end{aligned}$$

Using the following basic equivalence: $\mathbb{E}\{A\} = \mathbb{E}\{\mathbb{E}\{A|B\}\}$

$$\frac{\eta}{2}\mathbb{E}\left\{\|Q_T\|^2\right\} - (T-1)2\eta R^2 \leq \sum_{k=1}^{T}\mathbb{E}\left\{\langle\mathbb{E}\{\hat{g}(Y_k)|Y_k\}, x^* - Y_k\rangle\right\} + \frac{\alpha R^2}{2} + \frac{1}{2\alpha}\sum_{k=1}^{T}\mathbb{E}\left\{\mathbb{E}\left\{\|\hat{g}(Y_k)\|^2|Y_k\right\}\right\}$$

thus

$$\frac{\eta}{2}\mathbb{E}\left\{\|Q_T\|^2\right\} - (T-1)2\eta R^2 \leq \sum_{k=1}^{T}\mathbb{E}\left\{\langle\mathbb{E}\{\hat{g}(Y_k)|Y_k\}, x^* - Y_k\rangle\right\} + \frac{\alpha R^2}{2} + \frac{TB^2}{2\alpha}$$

We know $\mathbb{E}\{\hat{g}(Y_k)|Y_k\} = g(Y_k)$ so using convexity of f we get,

$$\frac{\eta}{2}\mathbb{E}\left\{\|Q_T\|^2\right\} - (T-1)2\eta R^2 \leq \sum_{k=1}^{T}\mathbb{E}\left\{f(x^*) - f(Y_k)\right\} + \frac{\alpha R^2}{2} + \frac{TB^2}{2\alpha}$$

Rearranging,

$$\sum_{k=1}^{T}\mathbb{E}\left\{f(Y_k) - f(x^*)\right\} + \frac{\eta}{2}\mathbb{E}\left\{\|Q_T\|^2\right\} \leq \frac{\alpha R^2}{2} + \frac{TB^2}{2\alpha} + (T-1)2\eta R^2$$

Dividing by T and using Jensen inequality,

$$\mathbb{E}\left\{f\left(\frac{1}{T}\sum_{k=0}^{T}Y_k\right) - f(x^*)\right\} + \frac{1}{T}\frac{\eta}{2}\mathbb{E}\left\{\|Q_T\|^2\right\} \leq \frac{\alpha R^2}{2T} + \frac{B^2}{2\alpha} + 2\eta R^2$$

Add and subtract $\mathbb{E}\{f(\bar{x})\}$ to the LHS,

$$\mathbb{E}\left\{f(\bar{X}) - f(x^*) + f(\bar{Y}) - f(\bar{X})\right\} + \frac{1}{T}\frac{\eta}{2}\mathbb{E}\left\{\|Q_T\|^2\right\} \leq \frac{\alpha R^2}{2T} + \frac{B^2}{2\alpha} + 2\eta R^2$$

Use Lemma B.2 and G-Lipschitz continuity of f we get,

$$\mathbb{E}\left\{f(\bar{X}) - f(x^*) + G\|\bar{Y} - \bar{X}\|\right\} + T\frac{\eta}{2}\mathbb{E}\left\{\|\bar{Y} - \bar{X}\|^2\right\} \leq \frac{\alpha R^2}{2T} + \frac{B^2}{2\alpha} + 2\eta R^2$$

By completing the square we have,

$$\mathbb{E}\left\{f(\bar{X})\right\} - f(x^*) \leq \frac{\alpha R^2}{2T} + \frac{B^2}{2\alpha} + \frac{\eta D^2}{2} + \frac{G^2}{2\eta T} \tag{C.1}$$

Choose

$$\alpha = \frac{B\sqrt{T}}{R}$$

and

$$\eta = \frac{2G}{R\sqrt{T}}$$

which means

$$\mathbb{E}\left\{f(\bar{X})\right\} - f(x^*) \leq \frac{BR + 2RG}{\sqrt{T}}$$

However, if we do not have access to G independent from B then replacing the inequality $G \leq B$ in Equation (C.1) we get,

$$\mathbb{E}\left\{f(\bar{X})\right\} - f(x^*) \leq \frac{\alpha R^2}{2T} + \frac{B^2}{2\alpha} + \frac{\eta D^2}{2} + \frac{B^2}{2\eta T}$$

Choose

$$\alpha = \frac{B\sqrt{T}}{R}$$

and

$$\eta = \frac{2G}{R\sqrt{T}}$$

which means

$$\mathbb{E}\left\{f(\bar{X})\right\} - f(x^*) \leq \frac{3BR}{\sqrt{T}}$$

□

REFERENCES

[1] K. Balasubramanian and S. Ghadimi, *Zeroth-order (non)-convex stochastic optimization via conditional gradient and gradient updates*, in Advances in Neural Information Processing Systems, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds., vol. 31, Curran Associates, Inc., 2018, https://proceedings.neurips.cc/paper/2018/file/36d7534290610d9b7e9abed244dd2f28-Paper.pdf.

[2] A. Beck, *First-order methods in optimization*, SIAM, 2017.

[3] A. Beck and M. Teboulle, *Mirror descent and nonlinear projected subgradient methods for convex optimization*, Operations Research Letters, 31 (2003), pp. 167–175.

[4] C. Blair, *Problem complexity and method efficiency in optimization (a. s. nemirovsky and d. b. yudin)*, SIAM Review, 27 (1985), pp. 264–265, https://doi.org/10.1137/1027074, https://doi.org/10.1137/1027074, https://arxiv.org/abs/https://doi.org/10.1137/1027074.

[5] S. Bubeck et al., *Convex optimization: Algorithms and complexity*, Foundations and Trends® in Machine Learning, 8 (2015), pp. 231–357.

[6] M. Frank and P. Wolfe, *An algorithm for quadratic programming*, Naval Research Logistics Quarterly, 3 (1956), pp. 95–110, https://doi.org/https://doi.org/10.1002/nav.3800030109, https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800030109, https://arxiv.org/abs/https://onlinelibrary.wiley.com/doi/pdf/10.1002/nav.3800030109.

[7] D. Garber, *On the convergence of projected-gradient methods with low-rank projections for smooth convex minimization over trace-norm balls and related problems*, SIAM Journal on Optimization, 31 (2021), pp. 727–753.

[8] G. H. Golub and C. F. Van Loan, *Matrix computations*, JHU press, 2013.

[9] S.-P. Han, *A successive projection method*, Mathematical Programming, 40 (1988), pp. 1–14.

[10] E. Hazan and S. Kale, *Projection-free online learning*, arXiv preprint arXiv:1206.4657, (2012).

[11] E. Hazan and H. Luo, *Variance-reduced and projection-free stochastic optimization*, in Proceedings of The 33rd International Conference on Machine Learning, M. F. Balcan and K. Q. Weinberger, eds., vol. 48 of Proceedings of Machine Learning Research, New York, New York, USA, 20–22 Jun 2016, PMLR, pp. 1263–1271, https://proceedings.mlr.press/v48/hazana16.html.

[12] E. Hazan and E. Minasyan, *Faster projection-free online learning*, in Conference on Learning Theory, PMLR, 2020, pp. 1877–1893.

[13] A. N. Iusem and B. F. Svaiter, *A row-action method for convex programming*, Mathematical programming, 64 (1994), pp. 149–171.

[14] M. Jaggi, *Revisiting frank-wolfe: Projection-free sparse convex optimization*, in International Conference on Machine Learning, PMLR, 2013, pp. 427–435.

[15] K. Levy and A. Krause, *Projection free online learning over smooth sets*, in Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics, K. Chaudhuri and M. Sugiyama, eds., vol. 89 of Proceedings of Machine Learning Research, PMLR, 16–18 Apr 2019, pp. 1458–1466, https://proceedings.mlr.press/v89/levy19a.html.

[16] A. Lin, *A class of methods for projection on a convex set*, Advanced Modeling and Optimization (AMO), 5 (2003).

[17] A. Lin, *Projection algorithms in nonlinear programming*, The Johns Hopkins University, 2003.
[18] M. Mahdavi, T. Yang, R. Jin, S. Zhu, and J. Yi, *Stochastic gradient descent with only one projection*, in Advances in Neural Information Processing Systems, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds., vol. 25, Curran Associates, Inc., 2012, https://proceedings.neurips.cc/paper/2012/file/c52f1bd66cc19d05628bd8bf27af3ad6-Paper.pdf.
[19] Y. Nesterov, *Primal-dual subgradient methods for convex problems*, Mathematical programming, 120 (2009), pp. 221–259.
[20] I. Petrakis, *McShane-Whitney extensions in constructive analysis*, Logical Methods in Computer Science, Volume 16, Issue 1 (2020), https://doi.org/10.23638/LMCS-16(1:18)2020, https://lmcs.episciences.org/6105.
[21] H. Yu and M. J. Neely, *A primal-dual type algorithm with the o(1/t) convergence rate for large scale constrained convex programs*, in 2016 IEEE 55th Conference on Decision and Control (CDC), 2016, pp. 1900–1905, https://doi.org/10.1109/CDC.2016.7798542.
[22] H. Yu and M. J. Neely, *A primal-dual parallel method with $o(1/\epsilon)$ convergence for constrained composite convex programs*, 2017, https://doi.org/10.48550/ARXIV.1708.00322, https://arxiv.org/abs/1708.00322.