First-Order Methods for Convex Constrained Optimization under Error Bound Conditions with Unknown Growth Parameters

Qihang Lin · Runchao Ma · Selvaprabu Nadarajah · Negar Soheili

Received: date / Accepted: date

Abstract We propose first-order methods based on a level-set technique for convex constrained optimization that satisfies an error bound condition with unknown growth parameters. The proposed approach solves the original problem by solving a sequence of unconstrained subproblems defined with different level parameters. Different from the existing level-set methods where the subproblems are solved sequentially, our method applies a first-order method to solve each subproblem independently and simultaneously, which can be implemented with either a single or multiple processors. Once the objective value of one subproblem is reduced by a constant factor, a sequential restart is performed to update the level parameters and restart the first-order methods. When the problem is non-smooth, our method finds an $\epsilon$-optimal and $\epsilon$-feasible solution by computing at most $O\left(\frac{G^2}{d^2} \ln^3\left(\frac{1}{\epsilon}\right)\right)$ subgradients where $G > 0$ and $d \geq 1$ are the growth rate and the exponent, respectively, in the error bound condition. When the problem is smooth, the complexity is improved to $O\left(\frac{G^1}{d^1} \ln^3\left(\frac{1}{\epsilon}\right)\right)$. Our methods do not require knowing $G$, $d$ and any problem dependent parameters.

Keywords first-order method · level-set method · convex constrained optimization · error bound condition

Mathematics Subject Classification (2010) 90C25 · 90C26

1 Introduction.

In this paper, we consider a convex optimization problem with inequality constraints:

$$f^* := \min_{x \in X} \{f(x) := f_0(x) \text{ s.t. } g(x) := \max_{i=1, \ldots, m} f_i(x) \leq 0\}, \quad (1)$$

where $f_i$ for $i = 0, \ldots, m$ are convex real-valued functions and $X \subset \mathbb{R}^n$ is a closed convex set onto which the projection mapping is computationally easy. We say a solution $\bar{x}$ to (1) is $\epsilon$-optimal if $f(\bar{x}) - f^* \leq \epsilon$ and $\epsilon$-feasible if $\bar{x} \in X$ and $g(\bar{x}) \leq \epsilon$. The goal of this paper is to find an $\epsilon$-optimal and $\epsilon$-feasible solution of (1) using first-order methods, namely, the algorithms that only utilize the objective value.
and the (sub)gradient of $f_i$ for $i = 0, \ldots, m$. We also assume (1) satisfies an error bound condition with parameters $G > 0$ and $d \geq 1$, namely,

$$\text{dist}(\mathbf{x}, \mathcal{X}^*)^d \leq G \max \{f(\mathbf{x}) - f^*, g(\mathbf{x})\}, \quad \forall \mathbf{x} \in \mathcal{X}, \quad (2)$$

where $\mathcal{X}^*$ is the set of optimal solutions to (1) and $\text{dist}(\mathbf{x}, \mathcal{X}^*)$ is the Euclidean distance of $\mathbf{x}$ to $\mathcal{X}^*$. Our methods do not require knowing or estimating the values of $f^*$, $G$ and $d$. Condition (2) is prevalent in convex optimization. For example, it holds with $d = 1$ if $f_i$ has a polyhedron epigraph (e.g. when $f_i$ is linear) for $i = 0, \ldots, m$ and holds with $d = 2$ if $f_i$ is strongly convex on $\mathcal{X}$ for $i = 0, \ldots, m$. Note that when $f_i \equiv 0$ for $i = 1, \ldots, m$, condition (2) is reduced to a more common version for unconstrained or simply constrained\footnote{Here, being simply constrained means $f_i \equiv 0$ for $i = 1, \ldots, m$ in (1) and $\mathcal{X}$ is a simple set, e.g., $\mathbb{R}^n$, a box or a ball, that allows for a projection mapping of a low cost.} optimization:

$$\text{dist}(\mathbf{x}, \mathcal{X}^*)^d \leq G(f(\mathbf{x}) - f^*), \quad \forall \mathbf{x} \in \mathcal{X}. \quad (3)$$

Although there has been a large volume of literature on convex constrained optimization, the studies on complexity of optimization algorithms under condition (2) remain rare. In fact, the impact of error bound condition to an algorithm’s complexity was mostly studied in the setting of unconstrained optimization in the form (3). This paper fills in the gap by showing the complexity of a class of first-order methods for constrained problems under the more general condition (2).

Our method is closely related to the level-set method which solves (1) by solving a sequence of unconstrained or simply constrained subproblems defined with different level parameters. In contrast to the existing level-set methods which solve one subproblem after another, our method solves them simultaneously by applying an independent first-order method to each subproblem. This can be implemented using multiple processors with each processor solving one subproblem, or using a single processor that rotates through the iterations of all first-order methods. Depending on if $f_i$ is non-smooth or smooth, each subproblem is solved by a subgradient method or a smoothing accelerated gradient method. The total complexity for our methods to find an $\epsilon$-optimal and $\epsilon$-feasible solution is established in terms of the total number of (sub)gradients computed across all processors. Our methods require the access to a solution strictly satisfying all inequality constraints in (1) but do not need to know other problem-dependent parameters such as $f^*$, $G$ and $d$ in (2), the Lipschitz constant of $f_i$ or its gradient (if $f_i$ is smooth), which are required by many other methods in order to accelerate their convergence under error bound conditions.

The scheme of using multiple processors is motivated by [32] for unconstrained optimization and our method can be described more easily under this scheme. Same as [32], our algorithms can be implemented with one processor and the total complexity we defined above is independent of the number of processors. The total wall-clock runtime if implemented with one processor will just be the runtime with multiple processors multiplied by the number of processors and, fortunately, the number of processors we need is in the order of $O(\ln(\frac{1}{\epsilon}))$. In fact, this is the source of one $O(\ln(\frac{1}{\epsilon}))$ factor in our complexity.

Our main contributions are summarized as follows:

- We propose a new class of first-order methods to solve convex constrained optimization problem (1), which is easy to implement and does not require knowing or estimating any problem-dependent parameters other than a lower bound of $f^*$ and a strictly feasible solution. To the best of our knowledge, this paper is the first work on first-order methods for convex constrained optimization under condition (2) with unknown $G$ and $d$.
- When $f_i$, $i = 0, 1, \ldots, m$, in (1) are non-smooth, a subgradient method is used to solve the level-set subproblems. In this case, we show that the total complexity of our method to achieve an $\epsilon$-optimal and $\epsilon$-feasible solution is $O\left(\frac{G^{2/d}}{\epsilon^2} \ln^3\left(\frac{1}{\epsilon}\right)\right)$, where $G$ and $d$ are from (2).
- When $f_i$, $i = 0, 1, \ldots, m$, in (1) are smooth, a smoothing accelerated gradient method is used to solve the level-set subproblems. In this case, we show that the total complexity of our method to achieve an $\epsilon$-optimal and $\epsilon$-feasible solution is $O\left(\frac{G^{2/d}}{\epsilon^{3/d}} \ln^3\left(\frac{1}{\epsilon}\right)\right)$, which is lower than the complexity for the non-smooth case.
2 Related Work.

In this section, we review the works on first-order methods under error bound condition and discuss our connection to them.

2.1 Unconstrained Convex Optimization under Error Bound Condition.

Since the fundamental studies by Hoffman [13], error bound condition has been actively studied during the past decades and widely applied to optimization, feasibility problem, and variational analysis [33, 26, 2, 23, 22, 24, 25]. For unconstrained or simply constrained optimization, numerous evidences have shown that an optimization algorithm can be significantly accelerated through utilizing an error bound condition (3). An non-exhaustive list of those findings include [32, 27, 48, 49, 14, 8, 7, 34, 5, 45, 15, 10, 40, 41, 42, 21, 39, 35, 44, 43, 12]. However, all of those studies only focus on an unconstrained or simply constrained problem and thus cannot be applied to (1).

Generally speaking, an optimization algorithm will not be automatically accelerated if an error bound condition holds. Instead, the acceleration is often achieved after modifying the algorithm according to the error bound condition, e.g., by using a special step length or restarting the algorithm in a particular frequency. However, most of those techniques usually require knowing some problem-dependent parameters such as $G$ and $d$ in (3) or the optimal objective value $f^*$, which are unknown and hard to estimate. Some recent techniques have been developed to address this issue by dynamically estimating $G$ during an algorithm [29, 20, 21, 7] or enumerating the key parameters of an algorithm (e.g. step size or restarting frequency) from a grid of a logarithmic size to make the algorithm adaptive to the values of $G$ or $d$ [34, 15, 14, 32, 8].

Our approach is motivated by the adaptive method from [32] which runs first-order algorithms in logarithmically many processors with each processor using a different step size. In the non-smooth case, the method by [32] has complexity of $O\left(\frac{G^{1/d}}{\sqrt{\epsilon}}\ln\left(\frac{1}{\epsilon}\right)\right)$ when $d > 1$ and $O\left(G^2\ln^2\left(\frac{1}{\epsilon}\right)\right)$ when $d = 1$, both of which are better than our method by logarithmic factors. In the unconstrained and smooth case, the exponent $d$ in (3) must satisfy $d \geq 2$. In this case, the complexity by [32] becomes $O\left(\frac{G^{1/d}}{\sqrt{\epsilon}}\ln\left(\frac{1}{\epsilon}\right)\right)$ when $d > 2$ and $O\left(G^{1/2}\ln^{2}\left(\frac{1}{\epsilon}\right)\right)$ when $d = 2$, which are also better than our complexity in the same setting. Their method requires knowing as few problem-dependent parameters as our method. However, their method is for unconstrained or simply constrained problem while our method allows for nonlinear constraints. Note that our method is not a generalization of [32] to the constrained case. In fact, each setting in their method is for unconstrained or simply constrained problem and thus cannot be applied to (1).

New first-order methods with non-traditional computational complexity are proposed by [9] for unconstrained or simply constrained optimization that satisfies the following growth condition:

$$\text{dist}(x,X^*) \leq G(f(x) - f_{\text{slb}}), \quad \forall x \in X,$$

where $G > 0$ and $f_{\text{slb}}$ is a known strict lower bound of $f^*$. As shown by Theorem 1.1 in [9], property (4) holds under very mild condition. Their methods find an $\epsilon$-relative solution, namely, a solution $\bar{x} \in X$ satisfying $f(\bar{x}) - f^* \leq \epsilon(f^* - f_{\text{slb}})$ with complexity $O\left(\frac{G^2}{\epsilon^2}\right)$, $O\left(\frac{G}{\epsilon}\right)$, and $O\left(\frac{G^2}{\epsilon}\right)$, when the problem is non-smooth, smooth, and structured non-smooth, respectively. Their results improve the complexity of the traditional first-order methods when the initial solution is far away from $X^*$. This is reflected by the dependency of their complexity on the initial solution which is hidden in the $O$ notation above for simplicity. In spite of the big differences in the settings between our paper and [9], we borrow some analysis from [9] to show that, when solving a level-set subproblem, the objective value will be reduced by a constant factor after certain number of iterations if the initial objective gap is large.

2.2 Constrained Convex Optimization.

The recent development in the first-order methods for convex optimization with convex constraints include [1, 35, 38, 37, 36, 47, 17, 18, 3, 6, 11] for deterministic constraints and [16, 47, 19] for stochastic...
constraints. Those works analyze the complexity of different first-order methods without assuming any error bound condition. To find an $\epsilon$-optimal and $\epsilon$-feasible solution for (1), the best complexity achieved by first-order methods in literature is $O(\frac{1}{\epsilon})$ when $f_i$ for $i = 0, 1, \ldots, m$ are non-smooth and $O(\frac{1}{\epsilon^2})$ when smooth. See [18] for example. Assuming (2), our methods have complexity $O(\frac{G^{3/4}}{\epsilon^2} \ln^3(\frac{1}{\epsilon}))$ and $O(\frac{G^{1/4}}{\epsilon^2} \ln^3(\frac{1}{\epsilon}))$ in the non-smooth and smooth cases, respectively. In either case, our complexity has better dependence on $\epsilon$ for any $d \geq 1$.

When $f_i$ for $i = 0, 1, \ldots, m$ are all strongly convex, the error bound condition (2) holds with $d = 2$. In this case, the best complexity achieved by the previous works is $O(\frac{1}{\epsilon})$ for the non-smooth case and $O(\frac{1}{\epsilon^2})$ for the smooth case. See [18] for example. Before our paper, [46] and [31] are the only works that consider error bound conditions when solving (1). A penalty method is proposed by [46], where they model the constraint $g(x) \leq 0$ in (1) as a part of the domain constraint by defining $X' := \{ x \in X | g(x) \leq 0 \}$. As a result, they consider an error bound condition in a form similar to (3), that is, $\text{dist}(x, X')^d \leq G(f(x) - f^*)$ for any $x \in X'$ (instead of any $x \in X$). The methods by [46] have complexity $O(\frac{G^{3/4}}{\epsilon^2} \ln(\frac{1}{\epsilon}))$ and $O(\frac{G^{1/4}}{\epsilon^2} \ln(\frac{1}{\epsilon}))$ in the non-smooth and smooth cases, respectively. By reformulating (1) into a non-smooth convex optimization problem with only affine equality constraints, [31] developed a subgradient method for (1) that has complexity $O(\text{dist}(x, S)^d \ln(\frac{1}{\epsilon}))$ when the reformulated problem satisfies the error bound condition with $d = 1$ (e.g. when (1) is a linear program).

When the same smoothness and the same value of $d$ are assumed, our complexities are higher than [46] and [31] by logarithmic factors. However, to achieve the aforementioned complexities, [46] and [31] require knowing $d$, $G$, $f^*$, and/or other problem-dependent parameters such as the Lipschitz constant of each $f_i$ and its gradient, which are unknown and hard to estimate in general. On the contrary, our methods do not require knowing $d$, $G$, $f^*$, or any problem-dependent parameters. Our methods do require a lower bound of $f^*$ and a strictly feasible solution, which can be obtained easily for many applications. We want to point out that a strictly feasible solution is also required by [46] and [31]. In addition, the algorithms by [46] have to perform exact projections onto $X'$, which is computationally challenging, while our methods only require projection onto the simple set $X$.

3 Preliminaries.

In this section, we introduce some basic definitions in convex constrained optimization and describe a simple level-set method which helps us to illustrate the idea behind the algorithms we propose. Let $\| \cdot \|$ be the Euclidean norm, $\mathcal{X}^*$ be the set of optimal solutions to (1), and $\text{dist}(x, S) := \min_{y \in S} \| y - x \|$ be the distance of $x$ to set $S$.

3.1 Assumptions.

Throughout the whole paper, we assume that there exists a feasible solution of (1) that strictly satisfies the constraint $g(x) \leq 0$ and is accessible to our algorithms.

**Assumption 1** There exists an accessible solution $\hat{x}$ such that $\hat{x} \in \mathcal{X}$ and $g(\hat{x}) < 0$.

We define the level-set function corresponding to (1) as:

$$H(r) := \min_{x \in \mathcal{X}} P(x; r),$$

where $r \in \mathbb{R}$ is called a level parameter and

$$P(x; r) := \max\{ f_0(x) - r, f_1(x), \ldots, f_m(x) \} = \max\{ f(x) - r, g(x) \}.$$  \hspace{1cm} (6)

It is easy to see that $\mathcal{X}^* = \arg\min_{x \in \mathcal{X}} P(x; f^*)$. The main properties of $H(r)$ are summarized below:

**Lemma 1 ([30, 18])** Function $H(r)$ defined in (5) has the following properties:

1. $H(r)$ is non-increasing and convex in $r$;
2. $H(f^*) = 0$;
3. $H(r) > 0$ when $r < f^*$ and $H(r) \leq 0$ when $r > f^*$. If Assumption 1 holds, $H(r) < 0$ when $r > f^*$;

4. For any $x \in \mathcal{X}$ and $r > 0$, we have $H(r) = \min_{y \in \mathcal{X}} P(y; r)$;
Algorithm 1: Level-Set Method

1: Input: $\alpha \in (0, 1)$, $\epsilon > 0$ and $r_0 < f^*$.
2: for $k = 0, 1, 2, \ldots$ do
3:   Find $x_k \in X$ that satisfies $\alpha P(x_k; r_k) \leq f^* - r_k$. \hfill (9)
4:   if $P(x_k; r_k) \leq \epsilon$ then
5:     Terminate and return $x_k$.
6:   end if
7:   Set $r_{k+1} = r_k + \alpha P(x_k; r_k)$.
8: end for
9: Output: $x_k$

4. $H(r) - \delta \leq H(r + \delta) \leq H(r)$ for any $\delta \geq 0$.

According to Lemma 1, $f^*$ is the unique root of $H(\cdot)$ under Assumption 1. Roughly speaking, a level-set method essentially applies a root-finding scheme to $H(\cdot)$ to find a $r \approx f^*$. Classical root-finding schemes, e.g., bisection search, require the exact value of $H(r)$, which requires solving (5) exactly. However, a level-set method allows solving (5) only approximately and uses the returned estimation of $H(r)$ to update $r$ towards $f^*$. Once a $r \approx f^*$ is found, a nearly optimal and nearly feasible solution to (1) can be found by solving $\min_{x \in X} P(x; r)$. We state this formally in the following lemma which can be easily proved by the definition of $P(x; r)$.

**Lemma 2** If $x \in X$ satisfies $P(x; r) \leq \epsilon$ for some $r$, $x$ is $(r - f^* + \epsilon)$-optimal and $\epsilon$-feasible.

Let $\theta$ be the negation of the left-hand derivative of $H(r)$ at $r = f^*$, namely,

$$
\theta := -\lim_{r \rightarrow f^*} \frac{H(r) - H(f^*)}{r - f^*} = \lim_{r \rightarrow f^*} \frac{H(r)}{f^* - r}.
$$

(7)

The following lemma can be easily derived from (7) and properties 1 and 4 in Lemma 1.

**Lemma 3** It holds that $0 < \theta \leq \frac{H(r)}{r - f^*} \leq 1$ for any $r < f^*$ and $0 \leq \frac{H(r)}{r - f^*} \leq \theta$ for any $r > f^*$.

Depending on what level-set method is used, the complexity for finding an $\epsilon$-optimal and $\epsilon$-feasible solution may depend on $\theta$ with smaller $\theta$ leading to high complexity.

Throughout the paper, we also make the following assumption.

**Assumption 2** There exists $G > 0$ and $d \geq 1$ such that

$$
\text{dist}(x, X^*)^d \leq GP(x; f^*), \quad \forall x \in X.
$$

(8)

Because $H(f^*) = \min_{x \in X} P(x; f^*) = 0$, (8) means that $P(x; f^*)$ satisfies the global error bound condition with exponent $d$, or equivalently, that $P(x; f^*)$ has Hölderian growth with a degree of growth $d$. In this paper, we do not assume any of $f^*$, $G$ or $d$ is known.

3.2 Level-Set Method.

We present a level-set method in Algorithm 1. A sequence of level parameters $\{r_k\}_{k \geq 0}$ with $r_k \leq f^*$ is generated in Algorithm 1. For each $r_k$, Algorithm 1 applies another convex optimization algorithm to solve $\min_{x \in X} P(x; r_k)$ and computes a solution $x_k \in X$ such that $\alpha P(x_k; r_k) \leq f^* - r_k$. Note that such a solution always exists because $\min_{x \in X} P(x; r_k) = H(r_k) \leq f^* - r_k \leq \frac{1}{\theta}(f^* - r_k)$ by Lemma 3. Then the next level parameter is computed as $r_{k+1} = r_k + \alpha P(x_k; r_k)$. Algorithm 1 is slightly different from the existing level set methods in [1, 17, 18, 19] in the stopping criterion when solving the subproblem and how $r_k$ is updated. The following theorem characterizes the number of main iterations needed for Algorithm 1 to find an $\epsilon$-optimal and $\epsilon$-feasible point.
Theorem 1 Algorithm 1 terminates in at most
\[ \hat{K} := \left\lceil \frac{1}{a\theta} \ln \left( \frac{f^* - r_0}{\alpha c} \right) \right\rceil + 1 \]  
iterations and it returns an \( \epsilon \)-optimal and \( \epsilon \)-feasible solution to (1).

Proof If \( r_k \leq f^* \), there exists \( x_k \in X \) such that \( \alpha P(x_k; r_k) \leq f^* - r_k \) according to the discussion above. This means Line 3 of Algorithm 1 is well defined. Then Algorithm 1 guarantees
\[ f^* - r_{k+1} = f^* - r_k - \alpha P(x_k; r_k) \geq f^* - r_k - (f^* - r_k) = 0. \]  
(11)

This means \( r_{k+1} \leq f^* \) as long as \( r_k \leq f^* \). Since \( r_0 < f^* \), it holds by induction that \( r_k \leq f^* \) for all \( k \) during Algorithm 1. On the other hand, it also holds that
\[ f^* - r_{k+1} = f^* - r_k - \alpha P(x_k; r_k) \leq f^* - r_k - \alpha H(r_k) \leq (1 - \alpha \theta)(f^* - r_k), \]  
(12)

where the first inequality holds because \( H(r_k) \leq P(x_k; r_k) \) and the second inequality is because of Lemma 3 and the fact that \( r_k \leq f^* \). Since \( \alpha \theta \in (0, 1) \), recursively applying (12) for \( k = 0, 1, \ldots \) gives
\[ 0 \leq f^* - r_k \leq (1 - \alpha \theta)^k (f^* - r_0), \quad \forall k \geq 0. \]  
(13)

Therefore, we have \( P(x_k; r_k) \leq \frac{L - r_0}{\alpha} \leq \frac{1}{\alpha} (1 - \alpha \theta)^k (f^* - r_0) \), which implies \( P(x_k; r_k) \leq \epsilon \) and Algorithm 1 terminates in at most \( \left\lfloor \frac{1}{a\theta} \ln \left( \frac{L - r_0}{\alpha \epsilon} \right) \right\rfloor + 1 \) iterations. Since \( r_k \leq f^* \), the output \( x_k \) is \( \epsilon \)-optimal and \( \epsilon \)-feasible by Lemma 2. \( \Box \)

Theorem 1 shows that the convergence rate of Algorithm 1 depends on \( \theta \). Although the exact value of \( \theta \) is usually unknown in practice, we show by next lemma that \( \theta \) has a non-trivial (positive) computable lower bound under Assumption 1.

Lemma 4 Suppose Assumption 1 holds and \( \hat{\theta} \) is defined as in Assumption 1. We have
\[ \hat{\theta} := f(\hat{x}) - g(\hat{x}) > f^* \quad \text{and} \quad 0 < \hat{\theta}(r) := \frac{g(\hat{x})}{r - \hat{r}} \leq \theta, \quad \forall r < f^*. \]

Proof Since \( \hat{x} \in X \) and \( g(\hat{x}) < 0 \), by the definitions of \( \hat{\theta} \) and \( H(\cdot) \), we have
\[ H(\hat{\theta}) \leq \max \{ f(\hat{x}) - \hat{r}, g(\hat{x}) \} = g(\hat{x}) < 0. \]  
(14)

By property 3 in Lemma 1, we must have \( \hat{\theta} > f^* \) so that \( 0 < \hat{\theta}(r) \) for any \( r < f^* \). By the convexity of \( H(\cdot) \) from property 1 of Lemma 1 and the definition of \( \theta \) in (7), we have
\[ H(r) \geq H(f^*) - \theta(r - f^*) = -\theta(r - f^*), \quad \forall r. \]  
(15)

This further implies that, for \( r < f^* \),
\[ \hat{\theta}(r) = -\frac{g(\hat{x})}{r - \hat{r}} \leq -\frac{-H(\hat{r})}{\hat{r} - \hat{r}} \leq \frac{\theta(\hat{r} - f^*)}{\hat{r} - f^*} < \frac{\theta(\hat{r} - r)}{\hat{r} - r} = \theta, \]  
(16)

where the first inequality is from (14), second from (15) with \( r = \hat{r} \), and last because \( r < f^* < \hat{r} \). \( \Box \)
4 A Restarting First-Order Method.

In iteration $k$ of Algorithm 1, one way to find $x_k \in \mathcal{X}$ satisfying (9) is to solve the level-set subproblem (5) with $r = r_k$, namely,

$$\min_{x \in \mathcal{X}} P(x; r_k).$$

Let $\text{fom}$ represent the first-order method applied to (17). Although there are many choices for $\text{fom}$, it is difficult to numerically verify (9) because $f^*$ is unknown. As a result, we are not able to terminate $\text{fom}$ at the right time. If $\text{fom}$ is terminated too soon, the returned solution $x_k$ will not satisfy (9) so that Algorithm 1 may not converge. If $\text{fom}$ is terminated too late, Algorithm 1 will converge but consume longer runtime than it actually needs.

To address this issue, we propose an adaptive restarting technique. Suppose $\text{fom}$ is initialized at $x^{(0)}_k \in \mathcal{X}$. Let $x^{(t)}_k$ be the solution generated by $\text{fom}$ in iteration $t$. Whenever $x^{(t)}_k$ satisfies

$$P(x^{(t)}_k; r_k) \leq BP(x^{(0)}_k; r_k)$$

for a constant $B \in (0, 1)$, we overwrite $x^{(0)}_k$ with $x^{(t)}_k$, restart $\text{fom}$ at the new $x^{(0)}_k$, and generate $r_{k+1} = r_k + \alpha P(x^{(0)}_k; r_k)$. In other words, whenever $\text{fom}$ reduces the objective value of (17) by a constant factor, we pretend that the current solution in $\text{fom}$ satisfies (9) and use it to update the level parameter as in Algorithm 1. After that, we restart $\text{fom}$ at the current solution.

The remaining question is what will happen if, in the scheme above, we generate $r_{k+1} = r_k + \alpha P(x^{(0)}_k; r_k)$ with $x^{(0)}_k$ that does not satisfy (9). To answer this question, we require $\text{fom}$ to have a special property that, if $\text{fom}$ was initialized at a solution $x^{(0)}_k$ not satisfying (9), $\text{fom}$ will guarantee (18) for a large enough $t$. The $\text{fom}$ with this property will be introduced later. With this property, even if $r_{k+1}$ was generated by $x^{(0)}_k$ not satisfying (9), inequality (18) will hold in a later iteration so that $x^{(0)}_k$ and $r_{k+1}$ will be updated. Since we overwrite $x^{(0)}_k$ with $x^{(t)}_k$ when (18) holds and restart $\text{fom}$ at the new $x^{(0)}_k$, $P(x^{(0)}_k; r_k)$ keeps decreasing geometrically and thus $x^{(0)}_k$ will satisfy (9) after finitely many restarts of $\text{fom}$.

Although this restarting technique eventually will guarantee an solution $x^{(0)}_k$ satisfying (9), it requires solving multiple level-set subproblems like (17) in parallel. The reason is that, when $\text{fom}$ is restarted, we do not know if the new $x^{(0)}_k$ has satisfied (9) or not. In the former case, we want to move on the next subproblem, i.e., $\min_{x \in \mathcal{X}} P(x; r_{k+1})$ so that we can further generate $r_{k+2}$. In the latter case, we have to keep solving $\min_{x \in \mathcal{X}} P(x; r_k)$ by $\text{fom}$ initialized at $x^{(0)}_k$ so that we are able to correct $r_{k+1}$ later.

Since we do not know which case happens, we have to solve $\min_{x \in \mathcal{X}} P(x; r_k)$ and $\min_{x \in \mathcal{X}} P(x; r_{k+1})$ in parallel by two separated $\text{fom}$, denoted by $\text{fom}_k$ and $\text{fom}_{k+1}$, respectively. If $\text{fom}_k$ is restarted earlier than $\text{fom}_{k+1}$, a new $r_{k+1}$ will be generated which changes the subproblem $\min_{x \in \mathcal{X}} P(x; r_{k+1})$. Hence, we also have to restart $\text{fom}_{k+1}$ at its most recent starting point, denoted by $x^{(0)}_k$. If $\text{fom}_{k+1}$ is restarted earlier than $\text{fom}_k$, a new level parameter $r_{k+2} = r_{k+1} + \alpha P(x^{(0)}_{k+1}; r_{k+1})$ will be generated and the next subproblem, $\min_{x \in \mathcal{X}} P(x; r_{k+2})$, will need to be solved by another $\text{fom}$ denoted by $\text{fom}_{k+2}$. We can start this argument at $k = 0$ and repeat until there will be $K$ level-set subproblems, namely, $\min_{x \in \mathcal{X}} P(x; r_k)$ for $k = 0, 1, \ldots, K-1$, to be solved simultaneously. The $k$th subproblem will be solved by a separated $\text{fom}$, denoted by $\text{fom}_k$. An appropriate value for $K$ needs to be determined in advance. In fact, by Theorem 1, $x^{(0)}_{K-1}$ will be $\epsilon$-optimal and $\epsilon$-feasible when $K \geq \bar{K}$ ($\bar{K}$ defined in (10)) and $x^{(0)}_k$ satisfies (9) for each $k$.

In the approach described above, subproblem $\min_{x \in \mathcal{X}} P(x; r_{k+1})$ and its solver $\text{fom}_{k+1}$ are not defined until $\text{fom}_k$ generates $r_{k+1}$ at least once. However, it can happen that $r_k < f^*$ and $\text{fom}_k$ is initialized at a high-quality solution $x^{(0)}_k$; for example, an optimal solution of $\min_{x \in \mathcal{X}} P(x; r_k)$. If this happens, (18) will never happen during $\text{fom}_k$ so that $r_i$ with $i > k$ will not be generated. To address this issue, we introduce an initialization scheme, where we choose a level parameter $r_{mi} < f^*$ and a solution $x_{mi} \in \mathcal{X}$, and then we set $r_{i} = r_{mi}$, $x^{(0)}_i = x_{mi}$ and $r_{k+1} = r_k + \alpha P(x^{(0)}_k; r_k)$ for $k = 0, \ldots, K-2$. After that, we start $\text{fom}_k$ at $x^{(0)}_k$ for all $k$’s at the same time. In our approach, we run all $\text{fom}$ in parallel with synchronization. In particular, each $\text{fom}_k$ will perform iteration $t+1$ only when all $\text{fom}$ complete iteration $t$.

Recall that, when $r_{k+1}$ is updated as a result of (18), we also have to restart $\text{fom}_{k+1}$ because its subproblem has changed. At this moment, the issue mentioned above may still happen in $\text{fom}_{k+1}$. In
Specifically speaking, when (18) holds in we perform the sequential updates similar to the initialization scheme described above when (18) holds. Moreover, if statement A holds, the index \( r_k \) is updated. Observe that, once there exists some \( i \) such that \( r_i = 0 \), we must have \( f_r \geq f_{i} \) for \( i = k, \ldots, K-1 \). Hence, the index \( r_k \) is updated. We extend the hypothesis from \( k \) to \( k+1 \). Since Algorithm 2 will be called when (18) holds in some \( fom_k \), its input must include the index \( k \) in addition to \( r_k \) and \( (x_j^{(0)})_{j=k}^{K-1} \). Its output will be the new \( (x_j^{(0)})_{j=k}^{K-1} \) after this sequential update.

The following observation is very important to the design of our main algorithm.

**Proposition 1** Let \( K \geq \hat{K} \) where \( \hat{K} \) is defined in (10). Suppose \( r_0 < f^* \) and \( r_i+1 = r_i + \alpha P(x_i; r_i) \) for some \( x_i \in X \) for \( i = 0, 1, \ldots, K-2 \). Then there exists \( k \) such that at least one of the following two statements holds:

A. \( r_k < f^* \) and \( \alpha P(x_k; r_k) \geq f^* - r_k \).

B. \( x_k \) is an \( \epsilon \)-optimal and \( \epsilon \)-feasible solution.

Moreover, if statement A holds, the index \( k \) satisfying statement A is unique and we have \( r_0 < \cdots < r_k < f^* \) and \( \alpha P(x_k; r_k) < f^* - r_k \) for \( i = 0, 1, \ldots, k-1 \).

**Proof** Suppose statement A does not hold for any index \( k \). We want to show that statement B holds for some \( k \). Since \( r_0 < f^* \) by assumption, we must have \( \alpha P(x_0; r_0) < f^* - r_0 \), which serves as the base case. Suppose \( r_i < f^* \) and \( \alpha P(x_i; r_i) < f^* - r_i \) for \( i = 0, 1, \ldots, k \). We have \( r_{k+1} = r_k + \alpha P(x_k; r_k) < f^* \). Since statement A does not hold for any index, we must have \( \alpha P(x_{k+1}; r_{k+1}) < f^* - r_{k+1} \), which allows us to extend the hypothesis from \( k \) to \( k+1 \). By induction, we then have \( r_k < f^* \) and \( \alpha P(x_k; r_k) < f^* - r_k \) for \( k = 0, 1, \ldots, K-1 \). Following the proof of (13), we have

\[
 f^* - r_{k+1} = f^* - r_k - \alpha P(x_k; r_k) \leq f^* - r_k - \alpha H(r_k) \leq (1 - \alpha \theta)(f^* - r_k),
\]

which further implies \( f^* - r_k \leq (1 - \alpha \theta)^k (f^* - r_0) \) for any \( k \). Therefore, we have \( P(x_{K-1}; r_{K-1}) < (\frac{f^* - r_0}{1 - \alpha \theta})^{K-1} \). Since \( K \geq \hat{K} \) with \( \hat{K} \) defined in (10), we have \( P(x_{K-1}; r_{K-1}) \leq \epsilon \) and \( x_{K-1} \) is an \( \epsilon \)-optimal and \( \epsilon \)-feasible solution by Lemma 2.

Next we prove the rest of the conclusion. Suppose statement A holds for some index \( k \). Since \( P(x; r) \geq H(r) \geq 0 \) for any \( r < f^* \) and \( r_0 < f^* \), we must have \( r_1 > r_0 \). If \( \alpha P(x_0; r_0) < f^* - r_0 \), we also have \( r_1 < f^* \), which further implies \( r_2 > r_1 \). Repeat this argument until we find an index \( k' \) such that \( r_0 < \cdots < r_{k'} < f^* \), \( \alpha P(x_i; r_i) < f^* - r_i \) for \( i \leq k' - 1 \), and \( \alpha P(x_{k'}; r_{k'}) \geq f^* - r_{k'} \). Note that such an \( k' \) must exist since statement A will not hold otherwise. The last inequality above implies \( r_{k'+1} \geq f^* \). Observe that, once there exists \( r_i \geq f^* \), we must have

\[
 r_{i+1} = f^* - r_i - f^* + \alpha P(x_i; r_i) \geq r_i - f^* + \alpha H(r_i) \geq (1 - \alpha \theta)(r_i - f^*),
\]

where the first inequality is by the definition of \( H(\cdot) \) and the second by Lemma 3. Given that \( r_{k'+1} \geq f^* \), we must have \( r_i \geq f^* \) for any \( i \geq k' + 1 \). Given that \( \alpha P(x_i; r_i) < f^* - r_i \) for \( i \leq k' - 1 \), we must have \( k = k' \). Hence, the index \( k \) has the desired properties and it is unique.

In the following sections, we will show that there is \( fom \) for \( \min_{x \in X} P(x; r) \) with the desired property, namely, ensuring (18) for large enough \( t \) if initialized at a solution \( x_j^{(0)} \) not satisfying (9). We will present different \( foms \) with this property for the non-smooth and smooth problems and instantiate the algorithm described above based on those \( foms \).
5 Non-Smooth Case

In this section, we make the following assumptions in additions to Assumption 1 and 2.

**Assumption 3** There exists $M$ such that \( \max_{\xi \in \partial P_i(x')} \|\xi\| \leq M \) for any \( x \in X \) and \( i = 0, 1, \ldots, m \).

Since \( f_i \)'s are non-smooth, we will use the standard subgradient descent (SGD) method to solve the level-set subproblems. When applied to \( \min_{x \in X} P(x; r) \) with an initial solution \( z^{(0)} \in X \), the SGD method can be presented as

\[
z^{(t+1)} = \text{Proj}_X \left( z^{(t)} - \eta^{(t)} \xi^{(t)} \right), \quad t = 0, 1, \ldots,
\]

(19)

where \( \text{Proj}_X(\cdot) \) is the projection onto \( X \), \( \eta^{(t)} > 0 \) is a step size, \( \xi^{(t)} \in \partial P(z^{(t)}; r) \), and \( \partial P(x; r) \) is the subdifferential of \( P(x; r) \) with respect to \( x \). It is clear that \( \|\xi\| \leq M \) for any \( \xi \in \partial P(x; r) \) for \( x \in X \) and \( r \) under Assumption 3. The output of SGD can be chosen as the historically best iterate, i.e.,

\[
x^{(t)} := \arg \min_{x \in \{z^{(s)}\}_{s=0}^t} P(x; r).
\]

(20)

Before we formally introduce the main algorithm, we first present a well-known convergence result of the SGD method.

**Proposition 2** Let \( z^* = \text{Proj}_X(z^{(0)}) \) and \( x^{(t)} \) be defined as in (20). The SGD method in (19) guarantees, for any \( t \geq 0 \),

\[
P(x^{(t)}; r) - P(z^*; r) \leq \frac{\text{dist}(z^{(0)}, X^*)^2 + \sum_{s=0}^t \eta^{(s)})^2 \|\xi^{(s)}\|^2}{2\sum_{s=0}^t \eta^{(s)}}.
\]

**Proof** By the non-expansion property of \( \text{Proj}_X(\cdot) \) and the fact that \( z^* \in X \), it holds that

\[
\|z^{(t+1)} - z^*\|^2 = \|\text{Proj}_X(z^{(t)} - \eta^{(t)} \xi^{(t)}) - z^*\|
\leq \|z^{(t)} - \eta^{(t)} \xi^{(t)} - z^*\|^2
= \|z^{(t)} - z^*\|^2 - 2\eta^{(t)} (\xi^{(t)})^\top (z^{(t)} - z^*) + \eta^{(t)})^2 \|\xi^{(t)}\|^2
\leq \|z^{(t)} - z^*\|^2 - 2\eta^{(t)} \left( P(z^{(t)}; r) - P(z^*; r) \right) + \eta^{(t)})^2 \|\xi^{(t)}\|^2,
\]

(21)

where last inequality follows from the convexity of \( P(x; r) \) in \( x \). Summing up (21) for \( t = 0, 1, \ldots \) and arranging terms, we have

\[
\left( \sum_{s=0}^t \eta^{(s)} \right) \left[ \min_{x \in \{z^{(s)}\}_{s=0}^t} P(z^{(s)}; r) - P(z^*; r) \right] \leq \sum_{s=0}^t \eta^{(s)} \left( P(z^{(s)}; r) - P(z^*; r) \right)
\leq \frac{1}{2} \|z^{(0)} - z^*\|^2 + \frac{1}{2} \sum_{s=0}^t \eta^{(s)})^2 \|\xi^{(s)}\|^2,
\]

which implies the conclusion we want after organizing terms. \( \square \)

Base on Proposition 2, we can show in Proposition 3 that, if the step sizes are chosen appropriately, the SGD method has the property we want and thus can be used as \( f_{om} \) in our main algorithm. Some argument we use in the proof is borrowed from the proof of Proposition 3.2 in [9].

**Proposition 3** Suppose \( r \leq f^* \) and \( \eta^{(t)} = \frac{(B - \alpha)P(z^{(0)}; r)}{M^2 G^2/d} \) for constants \( \alpha \) and \( B \) satisfying \( 0 < \alpha < B < 1 \). Let \( z^* = \text{Proj}_X(z^{(0)}) \) and \( x^{(t)} \) be defined as in (20). The SGD method in (19) guarantees either (i) \( \alpha P(x^{(t)}; r) < B(f^* - r) \) or (ii) \( P(x^{(t)}; r) \leq BP(x^{(0)}; r) \) for

\[
t = \left\lfloor \frac{M^2 G^2/d}{(B - \alpha)^2 P(z^{(0)}; r)2 - 2/d} \right\rfloor - 1.
\]

(22)

As a consequence, if \( \alpha P(x^{(0)}; r) \geq f^* - r \), we must have \( P(x^{(t)}; r) \leq BP(x^{(0)}; r) \) for \( r \) satisfying (22).
Proof Since $z^* \in X^*$ and $r \leq f^*$, we have $P(z^*; r) = f^* - r$ by the definition of $P(x; r)$. Plugging the definition of $\eta(i)$ into the conclusion of Proposition 2, we have

\[
P(x^{(t)}; r) \leq P(z^*; r) + \frac{\text{dist}(z^{(0)}, X^*)^2}{2(B - \alpha)P(z^{(0)}; r)} \sum_{i=0}^{t-1} \|\xi^{(i)}\|^2 + \frac{B - \alpha}{2} P(z^{(0)}; r) \\
\leq f^* - r + \frac{M^2 \text{dist}(z^{(0)}, X^*)^2}{2(t + 1)(B - \alpha)P(z^{(0)}; r)} + \frac{B - \alpha}{2} P(z^{(0)}; r) \\
\leq f^* - r + \frac{M^2 G^2/d P(z^{(0)}; r)^{2/d}}{2(t + 1)(B - \alpha)P(z^{(0)}; r)} + \frac{B - \alpha}{2} P(z^{(0)}; r),
\]

where the first inequality follows from the definition of $\eta(i)$, the second from Assumption 3, the third from Assumption 2, and the last from the fact that $P(z^{(0)}; f^*) \leq P(z^{(0)}; r)$ which holds because $r \leq f^*$.

Suppose $\alpha P(x^{(t)}; r) \geq B(f^* - r)$ and $t$ satisfies (22). Applying $z^{(0)} = x^{(0)}$ and (22) to (23) gives

\[
P(x^{(t)}; r) \leq f^* - r + \frac{B - \alpha}{2} P(z^{(0)}; r) + \frac{B - \alpha}{2} P(z^{(0)}; r) \leq \frac{\alpha}{B} P(x^{(t)}; r) + (B - \alpha) P(x^{(t)}; r),
\]

which implies $P(x^{(t)}; r) \leq BP(x^{(0)}; r)$. We thus have proved that, if (i) does not hold, (ii) must hold.

Suppose $\alpha P(x^{(0)}; r) \geq f^* - r$ but $P(x^{(t)}; r) > BP(x^{(0)}; r)$ for $t$ satisfying (22). According to the first conclusion of this proposition, we must have $\alpha P(x^{(t)}; r) < B(f^* - r)$, which implies $\alpha P(x^{(0)}; r) < B(f^* - r)$, contradicting with the assumption that $\alpha P(x^{(0)}; r) \geq f^* - r$. Hence, we must have $P(x^{(t)}; r) \leq BP(x^{(0)}; r)$ if $\alpha P(x^{(t)}; r) \geq f^* - r$. \qed

Now we are ready to present our main algorithm in Algorithm 3. In the discussion at the beginning of Section 4, we simply used $x_i^{(t)}$ and $r_i$ to represent the $t$th iterate and the level parameter in $\text{fom}_i$, which will be overwritten every time after $\text{fom}_i$ is restarted. In the theoretical proof, we need to use different notations to represent the values of $x_i^{(t)}$ and $r_i$ after different restarts. Hence, we introduce one more superscript $l$ so that $x_i^{(l; t)}$ represent the solution in the $t$th iteration after $l$th restart of $\text{fom}_i$, and $r_i^{(l; t)}$ represent the level parameter after $l$th restart of $\text{fom}_i$. Since $\text{fom}_i$ may be restarted at different times with different frequencies, we use $t_i$ and $l_i$ to represent $t$ and $l$ for $\text{fom}_i$ in Algorithm 3.

In Algorithm 3, we choose the SGD method in (19) to be $\text{fom}$. Suppose $\text{fom}$ has been restarted $l_i$ restarts. The level-set subproblem which $\text{fom}$ is applied to becomes $\min_{x \in X} P(x; r_i^{(l_i)})$ and $\text{fom}$ initialized at $x_i^{(l_i; 0)}$. In fact, Line 11-12 of Algorithm 3 performs the $t_i$th iteration of $\text{fom}_i$ for each $i$ using the step length in Proposition 3. Line 13-15 ensure the final output $\bar{x}$ is the historically best solution, i.e., the solution with the smallest objective value among all $\epsilon$-feasible solutions in the history of all $\text{fom}$. At the beginning of each iteration of Algorithm 3, we check if $P(x_i^{(l; t_i); r_i^{(l; t_i)}}) \leq BP(x_i^{(l_i; 0); r_i^{(l_i)}})$ holds for some $i$. If yes, we restart $\text{fom}_i$ at $x_i^{(l_i; 0)}$ and restart $\text{fom}_j$ at $x_j^{(l_j; 0)}$ for $j = i + 1, \ldots, K - 1$. Then, we increase the restart counter $t_j$ by one and reset the iteration counter $t_j$ to zero for those restarted $\text{fom}_j$. Finally, we update the level parameters in the restarted $\text{fom}_i$ using SeqUpdate. These steps correspond to Line 5-8. Note that $\text{fom}_i$ with $j = 0, \ldots, i - 1$ are not affected and will continue their iterations.

Remark 1 (Implementation with $K$ processors.) Algorithm 3 is presented for the setting of one processor. To implement it equivalently with $K$ processors, we just need to remove the “for loop” and perform Line 11-15 with the $i$th processor in parallel and synchronize $\text{fom}_i$ every iteration.

The following definitions and lemmas are needed for our complexity analysis.

Lemma 5 $r_0^{(l)} = r_{\text{init}} < f^*$ for any $l$ during Algorithm 3.

Proof This property holds trivially because SeqUpdate will never change the first level parameter according to Algorithm 2 (even if the first input of SeqUpdate is zero). \qed
At the beginning of each iteration of Algorithm 3, we have
\[ r_{l+1}^{(i)} = r_i^{(l)} + \alpha P(x_i^{(l)}, r_i^{(l)}) \text{ for } i = 0, 1, \ldots, K - 1. \] (25)

If \( K \geq \hat{K} \), either \( x_i^{(0, 0)} \) is \( \epsilon \)-optimal and \( \epsilon \)-feasible for some \( i \) or there exists a unique index \( k \) such that
\[ r_0^{(l_0)} < \cdots < r_k^{(l_k)} < f^*, \alpha P(x_k^{(l_k)}, r_k^{(l_k)}) < f^* - r_k^{(l_k)} \text{ for } i = 0, \ldots, K - 1, \text{ and } \alpha P(x_k^{(l_k)}, r_k^{(l_k)}) \geq f^* - r_k^{(l_k)}. \] (26)

**Proof** It is easy to verify (25) according to Line 2 and Line 5-8 in Algorithm 3. The second part of the conclusion is implied by Proposition 1. □

**Definition 1** The following terms are defined for each iteration of Algorithm 3.
- At the beginning of each iteration, the unique index \( k \) satisfying (26) is called a **critical index**.
- Line 5-8 in Algorithm 3 is called **sequential restart from \( \text{fom} \)**. For the sake of convenience, we call the initialization step (Line 2) a sequential restart from \( \text{fom}_0 \).
- Suppose \( k \) is the critical index for at the beginning of an iteration and a sequential restart happens in this iteration. We say this sequential restart is **effective** if it is from \( \text{fom} \), with \( i \leq k \). We say this sequential restart is **critical** if it is effective and, when it finishes (i.e., right after Line 8), either (I) \( x_i^{(l, 0)} \) is \( \epsilon \)-optimal and \( \epsilon \)-feasible for some \( i \), or (II) the unique index \( k \) satisfying (26) increases. For the sake of convenience, we say the initialization step is effective and critical.

**Lemma 7** Suppose \( K \geq \hat{K} \). At most \( \hat{K} \) critical sequential restarts can happen before Algorithm 3 finds an \( \epsilon \)-optimal and \( \epsilon \)-feasible solution.

**Proof** The critical index \( k \) will never decrease during Algorithm 3 and will increase after a critical sequential restart by definition. Suppose \( K = \hat{K} \) so that \( 0 \leq k \leq \hat{K} - 1 \). By its definition, a critical sequential restart can only happen \( K \) times before finding an \( \epsilon \)-optimal and \( \epsilon \)-feasible solution. Suppose \( K \geq \hat{K} + 1 \) and, after \( \hat{K} \) critical sequential restarts, an \( \epsilon \)-optimal and \( \epsilon \)-feasible solution still has not been found. At that moment, the critical index \( k \) should be at least \( \hat{K} \). By (25) and (26), we have
\[ f^* - r_i^{(l+i+1)} = f^* - r_i^{(l)} - \alpha P(x_i^{(l)}, r_i^{(l)}) \leq f^* - r_i^{(l)} - \alpha H(i, t_i) \leq (1 - \alpha \theta) (f^* - r_i^{(l)}), \]
\footnote{The increased \( k \) will be the critical index for the next iteration.}
which further implies
\[ \alpha P(x^{(l+1)}_{k-1}, r^{(l+1)}_{k-1}) < f^* - r^{(l+1)}_{k-1} \leq (1 - \alpha \theta)^{k-1} (f^* - r^{(l)}_{0}) = (1 - \alpha \theta)^{k-1} (f^* - r^{(l)}_{0}) \leq \alpha \epsilon, \]
where the equality is by Lemma 5, the first inequality is from (26), and the last inequality is because \( k \geq \hat{K} \). This means \( x^{(l+1)}_{k-1} \) is \( \epsilon \)-optimal and \( \epsilon \)-feasible by Lemma 2 which leads to a contradiction. Hence, an \( \epsilon \)-optimal and \( \epsilon \)-feasible solution must have been found within \( \hat{K} \) critical sequential restarts.

\[ \square \]

We present the complexity of Algorithm 3 as follows, which is our first main result.

**Theorem 2** Suppose \( K = \left\lceil \frac{1}{\alpha \theta (r_{\text{ini}})} \ln \left( \frac{\epsilon - r_{\text{ini}}}{\alpha \epsilon} \right) \right\rceil + 1 \) in Algorithm 3, where \( \hat{r} \) and \( \hat{\theta} \) are defined as in Lemma 4. Algorithm 3 finds an \( \epsilon \)-optimal and \( \epsilon \)-feasible solution by calculating at most

\[ (m + 1) \cdot K \cdot \left[ \frac{M^2 G^2/d}{(B - \alpha)^2 \epsilon^2 - 2/d} \right] \cdot C_{\text{all}} = O \left( \log^3 \left( \frac{1}{\epsilon} \right) \frac{1}{\epsilon^2 - 2/d} \right) \]

subgradients of \( f_0, f_1, \ldots, \) or \( f_m \) in total, where \( \hat{K} \) is defined in (10) and

\[ C_{\text{all}} := \hat{K} \cdot \ln \left( \frac{2(f^* - r_{\text{ini}})}{\alpha \epsilon} \right) / \ln \left( 1 + \frac{\alpha}{2} (1 - B) \theta \right) + \hat{K} \ln \left( \frac{1}{1 - \alpha} \right) + \ln \left( \frac{f^* - r_{\text{ini}}}{\alpha (1 - B) \epsilon} \right) + 2 \ln \left( \frac{P(x_{\text{ini}}; r_{\text{ini}})}{\epsilon} \right) / \ln(1/B) \]

\[ = O \left( \log^2 \left( \frac{1}{\epsilon} \right) \right). \]

**Proof Step 1: Bound the number of iterations between two consecutive effective sequential restarts.**

Suppose an effective sequential restart happens and, after that, no \( x^{(l,0)}_i \) is \( \epsilon \)-optimal and \( \epsilon \)-feasible for any \( i \). By Lemma 6, a critical index \( k \) is well defined at the beginning of the next iteration. Then we consider when the next effective sequential restart will happen. Note that any ineffective sequential restart must be from \( \text{fom}_j \) with \( j \geq k \) so that it has no impact on \( \text{fom}_i \) with \( i \leq k \) and thus will not change what time the next effective sequential restart will happen. Since a critical index will only change after a critical sequential start, which is a special case of effective sequential restarts, the critical index will always be \( k \) before the next effective sequential restart. According to (26) and Proposition 3, we must have

\[ P(x^{(l,0)}_i; r^{(l,0)}_i) \leq B P(x^{(l,i)}_i; r^{(l,i)}_i) \]

iterations. In fact, this statement is at least true for \( i = k \) if no sequential restart from \( \text{fom}_k \) with \( i \leq k - 1 \) happens before the next sequential restart from \( k \). Since \( x^{(l,0)}_k \) is not \( \epsilon \)-optimal and \( \epsilon \)-feasible and \( r^{(l,k)}_i < f^* \) according to (26), it must holds that \( P(x^{(l,k)}_k; r^{(l,k)}_k) > \epsilon \) by Lemma 2, which means we must have

\[ P(x^{(l,0)}_i; r^{(l,0)}_i) \leq B P(x^{(l,i)}_i; r^{(l,i)}_i) \]

iterations so that an effective sequential restart will happen also within the number of iterations given in (27).

**Step 2: Bound the total number of effective sequential restarts.**

We first bound the number of effective sequential restarts from \( \text{fom}_k \), where \( k \) is the critical index, between two consecutive effective sequential restart. Suppose a critical sequential restart happens and, after that, no \( x^{(l,0)}_i \) is \( \epsilon \)-optimal and \( \epsilon \)-feasible for any \( i \). By Lemma 6, a critical index \( k \) is well defined at the beginning of the next iteration and will not change until the next critical sequential start happens. Because \( k \) does not change between two consecutive critical sequential restarts, the effective sequential restarts during that period are all defined with respect to the same \( k \).
Let $C_k$ be the number of effective sequential restarts from $f_{0m_k}$ before the next critical sequential restart happens. We claim that

$$C_k \leq \left[ \tilde{K} \ln \left( \frac{1}{1-\alpha} \right) + \ln \left( \frac{f^* - r_{\text{ini}}}{\alpha(1-B)\epsilon} \right) + 2 \ln \left( \frac{P(x_{\text{ini}}; r_{\text{ini}})}{\epsilon} \right) \right] / \ln(1/B).$$

We state this claim formally in Proposition 5 in Appendix. If $k = 0$, after $C_k$ effective sequential restarts, there will be a critical sequential restart and $k$ will increase. If $k > 0$, however, the number of effective sequential restarts before the next critical sequential restart can be more than $C_k$ because some effective sequential restarts may be from $f_{0m}$, with $i < k$ and is not counted by $C_k$.

Hence, our next task is to bound the total number of effective sequential restarts from $f_{0m}$ with $i < k$, where $k$ is the critical index. Note that the critical index $k$ is dynamic and never decreases during Algorithm 3. This means, once an index $i$ becomes less than the critical index, it remains so for the rest of the algorithm. Hence, we define $D_i$ as the total number of effective sequential restarts from $f_{0m}$, between the iteration where $i$ just becomes less than the critical index and the iteration where an $\epsilon$-optimal and $\epsilon$-feasible solution is found. We claim that

$$D_i \leq \ln \left( \frac{2(K - f^* - r_{\text{ini}})}{\alpha \epsilon} \right) / \ln \left( 1 + \frac{\alpha}{2} (1 - B) \beta \right) \quad \text{for } i = 0, 1, \ldots, K - 1,$$

We also state this claim formally in Proposition 6 in Appendix.

According to Lemma 7, there are at most $\tilde{K}$ critical sequential restarts before Algorithm 3 finds an $\epsilon$-optimal and $\epsilon$-feasible solution. Hence, the total number of effective sequential restarts from $f_{0m_k}$ with the critical index $k$ is at most $\sum_{k=0}^{\tilde{K}-1} C_k$. On the other hands, the total number effective sequential restarts from all $f_{0m_i}$’s with $i$ less than the critical index is at most $\sum_{i=0}^{\tilde{K}-1} D_i$. Hence, there are at most

$$\sum_{k=0}^{\tilde{K}-1} C_k + \sum_{i=0}^{\tilde{K}-1} D_i \leq C_{\text{all}}$$

effective sequential restarts before Algorithm 3 finds an $\epsilon$-optimal and $\epsilon$-feasible solution.

Step 3: Derive the total complexity.

By Step 1 above, the number of iterations between two effective sequential restart is bounded by (27). By Step 2 above, there are at most $C_{\text{all}}$ effective sequential restarts before an $\epsilon$-optimal and $\epsilon$-feasible solution is found. Moreover, each iteration of Algorithm 3 involves $K$ separated of SGD updates which require calculating $(m + 1)K$ subgradients. Overall, Algorithm 3 must find an $\epsilon$-optimal and $\epsilon$-feasible solution by calculating

$$(m + 1) \cdot K \cdot \left[ \frac{M^2 G^{2/d}}{(B - \alpha)^2 \epsilon^{2 - 2/d}} \right] \cdot C_{\text{all}}$$

subgradients of $f_0, f_1, \ldots$, or $f_m$ in total. □

6 Smooth Case

In this section, we assume functions $f_i$, $i = 0, 1, \ldots, m$ are differentiable and Assumption 3 means $\max \{ \| \nabla f_i(x) \|_2, i = 0, 1, \ldots, m \} \leq M$ for any $x \in X$. In addition to Assumption 3, we further make the following assumptions.

Assumption 4 Functions $f_i$, $i = 0, 1, \ldots, m$, are $L$-smooth on $X$ for some $L \geq 0$, namely, for $i = 0, 1, \ldots, m$, $f_i$ is differentiable and $f_i(x) \leq f_i(y) + \langle \nabla f_i(y), x - y \rangle + \frac{L}{2} \| x - y \|^2$ for any $x$ and $y$ in $X$.

Remember the sub-problem of level-set method is to solve following optimization problem:

$$\min_{x \in X} P(x; r) := \max \{ f_0(x) - r, f_1(x), \ldots, f_m(x) \}$$

(28)

where $r < f^*$ is a level parameter. Although Assumption 4 makes sure $f_i$ is smooth, $P(x; r)$ may still be non-smooth due to the $\max \{ \cdot \}$ operator. However, smoothing approximation method has been proposed to solve original non-smooth optimization problem [18, 4, 28], which can lead to faster convergence rate
Algorithm 4 Accelerated projected gradient step with line search: $\text{APG}(F, \mathbf{x}, v, \tilde{L}, A, \gamma_u)$

1: **Input:** A differentiable function $F: \mathbb{X}, v \in \mathbb{X}, \tilde{L} > 0, A \geq 0, \text{ and } \gamma_u > 1.$
2: $\tilde{L} \leftarrow \tilde{L}/\gamma_u$
3: **repeat**
4: $\tilde{L} \leftarrow \tilde{L} \cdot \gamma_u$
5: Find $\mathbf{a}$ from equation $\frac{\mathbf{a}^2}{\mathbf{a}^2} = \frac{\tilde{L}}{L}$.
6: $\mathbf{y} \leftarrow \frac{\mathbf{a} + v}{\mathbf{a} + v}$
7: $\hat{\mathbf{x}} \leftarrow \text{Proj}_X (y - \tilde{L}^{-1}\nabla F(y))$
8: **until** $\hat{\mathbf{x}} \in X$
9: **Output:** $\mathbf{x}, \tilde{L}, \text{ and } a.$

Algorithm 5 Accelerated gradient method by [29] applied to $\min_{x \in \mathbb{X}} P_\sigma(x; r)$

1: **Input:** $r \in \mathbb{R}, \sigma > 0, x^{(0)} \in \mathbb{X}, \tilde{L} > 0, \gamma_u > 1, \text{ and } \gamma_d > 1.$
2: $A^{(0)} \leftarrow 0$ and $v^{(0)} \leftarrow x^{(0)}$
3: **for** $t = 1, \ldots$ **do**
4: $(x^{(t+1)}, L, A^{(t+1)}) \leftarrow \text{APG}(P_\sigma(x; r), x^{(t)}, v^{(t)}, \tilde{L}, A^{(t)}, \gamma_u)$
5: $v^{(t+1)} \leftarrow \arg \min_{x \in \mathbb{X}} \{\sum_{i=1}^{m+1} \alpha_i \langle \nabla P_\sigma(x^{(i)}; r), x - x^{(i)} \rangle + \frac{1}{2} \|x - x^{(0)}\|_2^2\}$
6: $A^{(t+1)} \leftarrow A^{(t)} + a^{(t+1)}$
7: $\tilde{L} \leftarrow \tilde{L} / \gamma_d$
8: **end for**

in many settings. In this work, we consider exponentially smoothed function $P_\sigma(x; r)$, which is defined by

$$P_\sigma(x; r) := \frac{1}{\sigma} \ln \left( \exp(\sigma (f_0(x) - r)) + \sum_{i=1}^{m} \exp(\sigma f_i(x)) \right),$$  \hspace{1cm} (29)

where $\sigma > 0$ is a smoothing parameter. Lemma 8 characterizes the closeness of $P_\sigma(x; r)$ to $P(x; r)$.

Lemma 8 ([4]) Suppose Assumption 3 and Assumption 4 hold. Given $\sigma > 0$, the exponentially smoothed approximation $P_\sigma(x; r)$ is $(\sigma M^2 + L)$-smooth. Moreover, we have

$$0 \leq P_\sigma(x; r) - P(x; r) \leq \frac{\ln(m + 1)}{\sigma}, \forall x \in \mathbb{X}, \forall r.$$

We consider solving (29) approximately by solving its smooth approximation $\min P_\sigma(x; r)$ using the accelerated gradient method given in (4.9) (with $\mu = 0$) in [29]. That algorithm is presented in Algorithm 5 and its convergence property is given in Theorem 3.

Theorem 3 Algorithm 5 guarantees

$$P_\sigma(x^{(t)}; r) - P_\sigma(x; r) \leq \frac{\gamma_u(\sigma M^2 + L)}{t^2} \|x^{(0)} - x\|^2, \forall x \in \mathbb{X}, \forall t \geq 1.$$

Proposition 4 Suppose $r < f^*$ and $\sigma = \frac{3 \ln(m + 1)}{(B - \alpha)P(x^{(0)}; r)}$ for constants $\alpha$ and $B$ satisfying $0 < \alpha < B < 1$ in Algorithm 3. Let $x^* = \text{Proj}_X(x^{(0)})$. The smoothing APG method in Algorithm 5 guarantees either (i) $\alpha P(x^{(t)}; r) < B(f^* - r)$ or (ii) $P(x^{(t)}; r) \leq BP(x^{(0)}; r)$ for

$$t = \max \left\{ \frac{3\sqrt{\gamma_u} \ln(m + 1) M G^2 / d}{(B - \alpha)P(x^{(0)}; r)^{1 - 1 / d}}, \frac{3\gamma_u L G^2 / d}{(B - \alpha)P(x^{(0)}; r)^{1 - 2 / d}} \right\} \hspace{1cm} (30)$$

As a consequence, if $\alpha P(x^{(0)}; r) \geq f^* - r$, we must have $P(x^{(t)}; r) \leq BP(x^{(0)}; r)$ for $t$ satisfying (30).
Proof Plugging the definition of \( \sigma \) into the conclusion of Theorem 3, we have

\[
P_x(x^{(l)}; r) \leq P_x(x^*; r) + \frac{\gamma_0(\sigma M^2 + L)\text{dist}(x^{(0)}, x^*)^2}{r^2} + \gamma_0(\sigma M^2 + L)\text{dist}(x^{(0)}, x^*)^2 + \frac{\log (m + 1)}{\sigma} \\
\leq f^* - r + \frac{\gamma_0(\sigma M^2 + L)\text{dist}(x^{(0)}, x^*)^2}{r^2} + \frac{\log (m + 1)}{\sigma} \\
\leq f^* - r + \frac{3\ln(m + 1)\gamma_0(\sigma M^2 + L)\text{dist}(x^{(0)}, x^*)^2}{(B - \alpha)t^2} + \frac{\log (m + 1)}{\sigma} \\
\leq f^* - r + \frac{3\ln(m + 1)\gamma_0(\sigma M^2 + L)\text{dist}(x^{(0)}, x^*)^2}{(B - \alpha)t^2} + \frac{\log (m + 1)}{\sigma} \\
\leq f^* - r + \frac{3\ln(m + 1)\gamma_0(\sigma M^2 + L)\text{dist}(x^{(0)}, x^*)^2}{3} + \frac{(B - \alpha)\text{dist}(x^{(0)}, x^*)}{3},
\]

where the first inequality follows from the definition of \( \sigma \) in Theorem 3 and \( x^* \), the second inequality from Lemma 8, the third from Assumption 2, and the last inequality uses the definition of \( \sigma \) and the fact that \( P(x^{(0)}; f^*) \leq P(x^{(0)}; r) \) which holds because \( r < f^* \).

Suppose \( \alpha P(x^{(l)}; r) \geq B(f^* - r) \) and \( t \) satisfies (30). It follows from (31) that

\[
P(x^{(l)}; r) \leq f^* - r + \frac{3\ln(m + 1)\gamma_0(\sigma M^2 + L)\text{dist}(x^{(0)}, x^*)^2}{(B - \alpha)t^2} + \frac{\log (m + 1)}{\sigma} \\
\leq f^* - r + \frac{(B - \alpha)\text{dist}(x^{(0)}, x^*)}{3} + \frac{(B - \alpha)\text{dist}(x^{(0)}, x^*)}{3} \\
\leq \frac{n}{B}P(x^{(l)}; r) + (B - \alpha)P(x^{(l)}; r),
\]

where the second inequality follows from (30). We thus have proved that if (i) does not happen, (ii) must happens.

Suppose \( \alpha P(x^{(l)}; r) \geq f^* - r \) but \( P(x^{(l)}; r) > BP(x^{(l)}; r) \) for \( t \) satisfying (30). According to the first part of the conclusion of this proposition, we must have \( \alpha P(x^{(l)}; r) < B(f^* - r) \), which implies \( \alpha BP(x^{(l)}; r) < B(f^* - r) \), contradicting with the assumption that \( \alpha P(x^{(l)}; r) \geq f^* - r \). Hence, we must have \( P(x^{(l)}; r) \leq BP(x^{(l)}; r) \) if \( \alpha P(x^{(l)}; r) \geq f^* - r \). □

Now we are ready to present our main algorithm in Algorithm 6. Similar to Algorithm 3, we use a superscript \( l \) to distinguish from each other the iterates and the level-parameters after different times of sequential restarts. In Algorithm 6, we choose Algorithm 5 to be \( \text{fom} \). After \( l_i \) restarts, \( \text{fom} \) will be solving \( \min_{x \in X} P(x; r_i^{(l_i)}) \) from the initial solution \( x_i^{(l_i,0)} \). In fact, Line 11-16 of Algorithm 6 performs the \( l_i \)th iteration of \( \text{fom} \), for each \( i \) using the smoothing parameter \( \sigma_i \) suggested in Proposition 4. Similar to Algorithm 3, Algorithm 6 also check if \( P(x_i^{(l_i, k_i)}; r_i^{(l_i)}) \leq BP(x_i^{(l_i,0)}; r_i^{(l_i)}) \) at the beginning of each iteration. The only difference is that, when we restart \( \text{fom} \) at \( x_i^{(l_i, k_i)} \), we also need to reset \( \nu_i^{(l_i, k_i)} \), \( A_i^{(l_i,0)} \) and \( L_i \) accordingly. Similar to Algorithm 3, Algorithm 6 can be implemented equivalently with \( K \) processors by removing the “for loop” and performing Line 11-15 with the \( \text{rith} \) processor in parallel.

With the exactly same proofs, we can show that Lemma 5, Lemma 6, and Lemma 7 also hold for Algorithm 6. Moreover, all concepts in Definition 1 can be also defined for Algorithm 6. With those lemmas and definitions, the complexity of Algorithm 6 is then presented as follows, which is our second main result.

**Theorem 4** Suppose \( K = \left\lfloor \frac{1}{\alpha \theta (r_m)} \ln \left( \frac{\hat{r} - \hat{r} \alpha m}{\alpha m} \right) \right\rfloor + 1 \) in Algorithm 6, where \( \hat{r} \) and \( \hat{\theta} \) are defined as in Lemma 4. Algorithm 6 finds an \( \epsilon \)-optimal and \( \epsilon \)-feasible solution by calculating at most

\[
(m + 1) \cdot K \cdot \max \left\{ \frac{3\gamma_0(\epsilon^2 \ln m + 1)MG_1^{1/d}}{(B - \alpha)\epsilon^{1 - 1/d}}, \sqrt{\frac{3\gamma_0(\epsilon^2 \ln m + 1)MG_2^{1/d}}{(B - \alpha)\epsilon^{1 - 1/d}}} \right\} \cdot C_{all} = O \left( \log^3 \left( \frac{1}{\epsilon} \right) \right) \frac{1}{\epsilon^{1 - 1/d}}
\]

gradients of \( f_0, f_1, \ldots, \) or \( f_m \) in total, where \( \hat{K} \) is defined in (10) and \( C_{all} \) is defined as in Theorem 2.

**Proof** The proof is almost the same as that of Theorem 2 except that the number of iterations between two consecutive effective sequential restarts is at most

\[
\max \left\{ \frac{3\gamma_0(\epsilon^2 \ln m + 1)MG_1^{1/d}}{(B - \alpha)P(x_i^{(l_i, k_i)}; r_i^{(l_i)})^{1 - 1/d}}, \sqrt{\frac{3\gamma_0(\epsilon^2 \ln m + 1)MG_2^{1/d}}{(B - \alpha)P(x_i^{(l_i, k_i)}; r_i^{(l_i)})^{1 - 2/d}}} \right\}
\]
Algorithm 6 Level-Set Method Based on Restarted Accelerated Gradient Method

1: **Input:** Total number of *foms* $K$, initial Lipschitz constant $L$, initial solution $x_{ini} \in X$, initial level parameter $r_{ini} < f^*$, step size $\alpha \in (0, 1)$, and a reduction factor $B \in (\alpha, 1)$.

2: **Initialization:**

\[
(x_j^{(0,0)}, l_j, t_j, L_j) \leftarrow (x_{ini}, 0, 0, \hat{L}) \text{ for } j = 0, \ldots, K - 1,
\]

\[
x \leftarrow x_{ini}, \quad f_{best} \leftarrow +\infty, \quad (r_j^{(0)})_{j=0}^{K-1} \leftarrow \text{SeqUpdate}(0, (x_j^{(0,0)})_{j=0}^{K-1}, r_{ini}).
\]

3: **while** stopping condition is not satisfied **do**

4: **if** $P(x_i^{(l_i, t_i)}; r_i^{(l_i)}) \leq B P(x_i^{(l_i, t_i)}; r_i^{(l_i)})$ for some $i$ (choose any one if there are multiple) **then**

5: \[
(x_i^{(l_i+1,0)}, v_i^{(l_i+1,0)}, A_i^{(l_i+1)}, l_i) \leftarrow (x_i^{(l_i, t_i)}, x_i^{(l_i, t_i)}, 0, \hat{L})
\]

6: \[
(x_j^{(l_j+1,0)}, v_j^{(l_j+1,0)}, A_j^{(l_j+1)}, L_j) \leftarrow (x_j^{(l_j, t_j)}, x_j^{(l_j, t_j)}, 0, \hat{L}) \text{ for } j = i + 1, \ldots, K - 1
\]

7: \[
(l_j, t_j) \leftarrow (l_j + 1, 0) \text{ for } j = i, \ldots, K - 1
\]

8: \[
(r_j^{(l_j)})_{j=i}^{j=m} \leftarrow \text{SeqUpdate}(i, (x_j^{(l_j,0)})_{j=i}^{j=m}, r_i^{(l_i-1)})
\]

9: **end if**

10: **for** $i = 0, 1, \ldots, K - 1$ **do**

11: \[
\sigma_i \leftarrow \frac{2 \log(m+1)}{(B-\alpha) P(x_i^{(l_i, t_i)})}
\]

12: \[
(x_i^{(l_i, t_i+1)}, L_i, A_i^{(l_i+1)}) \leftarrow \text{APG}(P_{\sigma_i}(x_i^{(l_i)}), x_i^{(l_i, 0)}, v_i^{(l_i, 0)}, L_i, A_i^{(l_i)}, \gamma_u)
\]

13: \[
v_i^{(l_i+1)} \leftarrow \arg \min_{x \in X} \left\{ \sum_{s=1}^{l_i+1} \frac{1}{4^s} \left( \nabla P_{\sigma_i}(x_i^{(l_i+s)}; A_i^{(l_i+s)}), x - x_i^{(l_i+s)} \right)^2 + \frac{1}{2} \|x - x_i^{(l_i, 0)}\|^2 \right\}
\]

14: \[
A_i^{(l_i+1)} \leftarrow A_i^{(l_i+1)} + A_i^{(l_i)}
\]

15: \[
L_i \leftarrow L_i/\gamma_d
\]

16: \[
t_i \leftarrow t_i + 1
\]

17: **if** $g(x_i^{(l_i, t_i)}) \leq \epsilon$ and $f(x_i^{(l_i, t_i)}) \leq f_{best}$ **then**

18: \[
x \leftarrow x_i^{(l_i, t_i)}, \quad f_{best} \leftarrow f(x_i^{(l_i, t_i)})
\]

19: **end if**

20: **end for**

21: **end while**

22: **Output:** $x$

---

Iterations according to the iteration number (30) from Proposition 4, where $k$ is the critical index. Before an $\epsilon$-optimal and $\epsilon$-feasible solution is found, we have $P(x_k^{(l_k, 0)}; r_k^{(l_k)}) > \epsilon$ so that the number of iterations between two consecutive effective sequential restarts is at most

\[
\max \left\{ \frac{3 \sqrt{\gamma_u \ln (m+1)} M G^{1/d}}{(B-\alpha) \epsilon^{1-1/d}}, \sqrt{\frac{3 \gamma_u L G^2/d}{(B-\alpha) \epsilon^{1-2/d}}} \right\}.
\]

Overall, Algorithm 3 must find an $\epsilon$-optimal and $\epsilon$-feasible solution by calculating

\[
(m+1) \cdot K \cdot \max \left\{ \frac{3 \sqrt{\gamma_u \ln (m+1)} M G^{1/d}}{(B-\alpha) \epsilon^{1-1/d}}, \sqrt{\frac{3 \gamma_u L G^2/d}{(B-\alpha) \epsilon^{1-2/d}}} \right\} \cdot C_{call}
\]

generates of $f_0, f_1, \ldots, f_m$ in total. □

---

7 Conclusion

We develop first-order methods for convex constrained optimization problem that satisfies an error bound condition. We formulate the original convex constrained problem into a sequence of level-set subproblems and solve them by different processors in parallel. Depending if the problem is smooth or not, each processor applies a subgradient method or an accelerated gradient method with the exponential smoothing technique. The subroutine in each processor is restarted when the objective value is reduced by a factor. The restart in one processor is the followed by a sequence of updates of the level parameters and restarts in the subsquence processors. We derive the total complexity of finding an $\epsilon$-optimal and $\epsilon$-feasible solution.
Appendix

In this section, we present some technical lemmas and propositions with proofs.

In the next lemma, we give a lower bound for $r_j^{(l_j-1)} - r_j^{(l_j)}$, i.e., the change of level parameter after a sequential restart.

**Lemma 9** Suppose $(r_j^{(l_j)})_{j=0}^{K-1}$ are generated by $\text{SeqUpdate}(i, (x_j^{(l_j,0)})_{j=0}^{K-1}, r_i^{(l_i-1)})$ in Line 8 of Algorithm 3. We have $r_i^{(l_i-1)} - r_i^{(l_i)} = 0$ and, for $j \geq i + 1$,

$$r_j^{(l_j-1)} - r_j^{(l_j)} \geq (1 - \alpha)^{j-i} \alpha(1 - B)P(x_i^{(l_i-1)}, r_i^{(l_i-1)}).$$

**Proof** Line 2 of Algorithm 2 implies $r_i^{(l_i-1)} = r_i^{(l_i)}$. By Line 4 of Algorithm 2, we have

$$r_{i+1}^{(l_i+1-1)} - r_{i+1}^{(l_i+1)} = r_i^{(l_i-1)} + \alpha P(x_i^{(l_i-1,0)}, r_i^{(l_i-1)}) - r_i^{(l_i)} - \alpha P(x_i^{(l_i,0)}, r_i^{(l_i)}),$$

where the inequality holds because Line 4 and Line 5 of Algorithm 3. For $j \geq i + 2$, we have

$$r_j^{(l_j-1)} - r_j^{(l_j)} = r_j^{(l_j-1)} - r_j^{(l_j-1)} + \alpha P(x_j^{(l_j-1,0)}, r_j^{(l_j-1)}) - r_j^{(l_j-1)} - \alpha P(x_j^{(l_j-1,0)}, r_j^{(l_j-1)}),$$

where the second equality is because $x_j^{(l_j-1,0)} = x_j^{(l_j-1,0)}$ by Line 6 of Algorithm 3 and the inequality is because $P(x, r) - P(x, r') \geq r' - r$, which holds by the definition of $P(x; r)$ in (6). Applying inequality (33) recursively leads to our conclusion for $j \geq i + 2$. \qed

**Proposition 5** Suppose a critical sequential restart happens and, after that, no $x_j^{(l_j,0)}$ is $\epsilon$-optimal and $\epsilon$-feasible for any $i$. Let $k$ be the critical index right after this critical sequential start. Let $C_k$ be the number of effective sequential restarts from $\text{fom}_k$ before the next critical sequential restart. We have

$$C_k \leq \left[ \hat{K} \ln \left( \frac{1}{1 - \alpha} \right) + \ln \left( \frac{f^* - r_{\text{ini}}}{\alpha(1 - B)\epsilon} \right) + 2 \ln \left( \frac{P(x_{\text{ini}}; r_{\text{ini}})}{\epsilon} \right) \right] / \ln(1/B).$$

**Proof** Suppose the critical sequential restart in the assumption happens at iteration $T_0$ and the next critical sequential happens at iteration $T_1$. We prove (34) under two cases.

**Case 1:** Between iterations $T_0$ and $T_1$, all effective sequential restarts are from $\text{fom}_k$.

In this case, between iterations $T_0$ and $T_1$, $r_k^{(l_k)}$ remains unchanged and $P(x_k^{(l_k,0)}, r_k^{(l_k)})$ will change only because a new $x_k^{(l_k,0)}$ is generated. Note that $x_k^{(l_k,0)}$ generated by a sequential restart from $\text{fom}_k$ must satisfy

$$P(x_k^{(l_k,0)}, r_k^{(l_k)}) \leq BP(x_k^{(l_k-1,0)}, r_k^{(l_k-1)}).$$

Because $\epsilon < P(x_k^{(l_k,0)}, r_k^{(l_k)}) \leq P(x_{\text{ini}}; r_{\text{ini}})$ between iterations $T_0$ and $T_1$, by the definition of $C_k$, we must have

$$\epsilon \leq B^{C_k} P(x_{\text{ini}}; r_{\text{ini}}),$$

which implies

$$C_k \leq \ln \left( \frac{P(x_{\text{ini}}; r_{\text{ini}})}{\epsilon} \right) / \ln(1/B)$$

so that (34) holds for Case 1.

**Case 2:** Between iterations $T_0$ and $T_1$, at least one sequential restart happens from $\text{fom}_i$ with $i < k$.

17
Suppose, between iterations $T_0$ and $T_1$, the first sequential restart from $fom_i$ with $i < k$ happens in iteration $T_{0.5}$ with $T_0 < T_{0.5} < T_1$. In other words, all of the effective sequential restarts between $T_0$ and $T_{0.5}$ are still from $fom_k$. By the analysis of Case 1, we know that

$$T_{0.5} - T_0 \leq \ln \left( \frac{P(x_{ini}, r_{ini})}{\epsilon} \right) / \ln(1/B).$$

(35)

Suppose, after the sequential restart in iteration $T_{0.5}$, the solutions and the level parameters in $fom_i$ and $fom_k$ are $(x_{i(l_i,0)}^t, r_{i(l_i)}^t)$ and $(x_{k(l_k,0)}^t, r_{k(l_k)}^t)$, respectively. By Lemma 9, we have

$$f^* - r_{k(l_k)}^t > r_{k(l_k-1)} - r_{k(l_k)}^t \geq (1 - \alpha)k^{-i-1}\alpha(1 - B)P(x_{i(l_i-1,0)}^t, r_{i(l_i-1)}^t) > (1 - \alpha)k^{-i-1}\alpha(1 - B)\epsilon,$$

where the last inequality is because $x_{i(l_i-1,0)}^t$ is not $\epsilon$-optimal and $\epsilon$-feasible. Because $r_{k(l_k)}^t$ will never increase, between iterations $T_{0.5}$ and $T_1$, we always have

$$f^* - r_{k(l_k)}^t > (1 - \alpha)k^{-i-1}\alpha(1 - B)\epsilon \geq (1 - \alpha)\hat{K}\alpha(1 - B)\epsilon,$$

(36)

where the last inequality is because the critical index $k$ is no more than $\hat{K}$ before an $\epsilon$-optimal and $\epsilon$-feasible is found (see Lemma 7).

We define

$$U_{k(l_k)} := \frac{P(x_{k(l_k,0)}^t, r_{k(l_k)}^t)}{f^* - r_{k(l_k)}^t}$$

and analyze the difference between $U_{k(l_k)}$ and $U_{k(l_k-1)}$. Note that $x_{k(l_k,0)}^t$ and $r_{k(l_k)}^t$ cannot be generated by a sequential restart from $fom_j$ with $j > k$. Suppose $x_{k(l_k,0)}^t$ and $r_{k(l_k)}^t$ are generated by a sequential restart from $fom_j$ with $j < k$. We have $x_{k(l_k,0)}^t = x_{k(l_k-1,0)}^t$ and

$$U_{k(l_k)} = \frac{P(x_{k(l_k,0)}^t, r_{k(l_k)}^t)}{f^* - r_{k(l_k)}^t} \leq \frac{P(x_{k(l_k-1,0)}^t, r_{k(l_k-1)}^t) + r_{k(l_k-1)} - r_{k(l_k)}^t}{f^* - r_{k(l_k-1)}^t} \leq \frac{P(x_{k(l_k-1,0)}^t, r_{k(l_k-1)}^t)}{f^* - r_{k(l_k-1)}^t} = U_{k(l_k-1)},$$

(37)

where the first inequality is by the definition of $P(x,r)$ and the second is because

$$P(x_{k(l_k-1,0)}^t, r_{k(l_k-1)}^t) \geq \frac{f^* - r_{k(l_k-1)}^t}{\alpha} \geq f^* - r_{k(l_k-1)}^t$$

and $r_{k(l_k-1)} - r_{k(l_k)}^t \geq 0$. Suppose $x_{k(l_k,0)}^t$ and $r_{k(l_k)}^t$ are generated by a sequential restart from $fom_k$. We must have $r_{k(l_k)}^t = r_{k(l_k-1)}^t$ and $P(x_{k(l_k,0)}^t, r_{k(l_k)}^t) \leq BP(x_{k(l_k-1,0)}^t, r_{k(l_k-1)}^t)$, which implies

$$U_{k(l_k)} = \frac{P(x_{k(l_k,0)}^t, r_{k(l_k)}^t)}{f^* - r_{k(l_k)}^t} \leq \frac{BP(x_{k(l_k-1,0)}^t, r_{k(l_k-1)}^t)}{f^* - r_{k(l_k-1)}^t} = BU_{k(l_k-1)}.$$

(38)

Moreover, between iterations $T_{0.5}$ and $T_1$, we have

$$U_{k(l_k)} = \frac{P(x_{k(l_k,0)}^t, r_{k(l_k)}^t)}{f^* - r_{k(l_k)}^t} > \frac{\epsilon}{f^* - r_{l_0}^t} = \frac{\epsilon}{f^* - r_{ini}},$$

(39)

where the inequality is because $x_{k(l_k,0)}^t$ is not $\epsilon$-optimal and $\epsilon$-feasible, and we also have

$$U_{k(l_k)} = \frac{P(x_{k(l_k,0)}^t, r_{k(l_k)}^t)}{f^* - r_{k(l_k)}^t} \leq \frac{P(x_{ini}, r_{ini})}{(1 - \alpha)\hat{K}\alpha(1 - B)\epsilon},$$

(40)

where the inequality is from (36). Let $C_k^T$ be the number of sequential restart from $fom_k$ between iteration $T_{0.5}$ and $T_1$. According to (37), (38), (39), and (40), the following inequality must hold between iterations $T_{0.5}$ and $T_1$,

$$\frac{\epsilon}{f^* - r_{ini}} \leq \frac{B C_k^T P(x_{ini}, r_{ini})}{(1 - \alpha)\hat{K}\alpha(1 - B)\epsilon}.$$
which implies
\[
C'_k \leq \left( \hat{K} \ln \left( \frac{1}{1-\alpha} \right) + \ln \left( \frac{f^* - r_{\text{ini}}}{\alpha(1-B)\epsilon} \right) + \ln \left( \frac{P(x_{\text{ini}}; r_{\text{ini}})}{\epsilon} \right) \right) / \ln(1/B).
\]
Since \(C_k \leq T_{0.5} - T_0 + C'_k\), combining the inequality above with (35) leads to (34) for Case 2.

\[
\square
\]

**Proposition 6** Suppose a critical sequential restart happens and, after that, no \(x^{(i,0)}\) is \(\epsilon\)-optimal and \(\epsilon\)-feasible for any \(i\). Let \(k'\) and \(k\) be the critical indexes before and after this critical sequential restart, respectively. Suppose an index \(i\) satisfies \(k' \leq i < k\). Let \(D_i\) be the total number of effective sequential restarts from \(f_{\text{om}}\), after this critical sequential restart until an \(\epsilon\)-optimal and \(\epsilon\)-feasible solution is found. We have
\[
D_i \leq \ln \left( \frac{2(f^* - r_{\text{ini}})}{\alpha\epsilon} \right) / \ln \left( 1 + \frac{\alpha}{2}(1-B)\theta \right) \quad \text{for} \quad i = 0, 1, \ldots, K - 1,
\]

**Proof** Let \(i\) and \(k\) be the indexes described in the proposition. Suppose the critical sequential restart in the assumption happens at iteration \(T_0\) and an \(\epsilon\)-feasible and \(\epsilon\)-optimal solution is found at iteration \(T_1\). The analysis below is conducted for the iterations between \(T_0\) and \(T_1\).

We define a quantity
\[
V_{i}^{(l,i)} := f^* - r_i^{(l,i)} - \frac{\alpha}{2}P(x_i^{(l,i)}, r_i^{(l,i)})
\]
and analyze the difference between \(V_{i}^{(l,i)}\) and \(V_{r_i^{(l,i)}}^{(l,i-1)}\). Note that \(x^{(l,i)}\) and \(r_i^{(l,i)}\) cannot be generated by a sequential restart from \(f_{\text{om}}\), with \(j > i\). Suppose \(x_i^{(l,i)}\) and \(r_i^{(l,i)}\) are generated by an effective sequential restart from \(f_{\text{om}}\), with \(j < i\). We have
\[
V_i^{(l,i)} = f^* - r_i^{(l,i)} - \frac{\alpha}{2}P(x_i^{(l,i)}, r_i^{(l,i)}) > f^* - r_i^{(l,i-1)} - \frac{\alpha}{2}P(x_i^{(l,i-1)}, r_i^{(l,i)}) = V_i^{(l,i-1)},
\]
where the inequality is because \(r_i^{(l,i-1)} > r_i^{(l,i)}\), which holds by Lemma 9, and that fact that \(f^* - r - \frac{\alpha}{2}P(x, r)\) is strictly decreasing in \(r\) for any \(x \in \mathcal{X}\) by the definition of \(P(x; r)\). Suppose \(x_i^{(l,i)}\) and \(r_i^{(l,i)}\) are generated by an effective sequential restart from \(f_{\text{om}}\). We must have \(r_i^{(l,i)} = r_i^{(l,i-1)}\) (Lemma 9) and
\[
P(x_i^{(l,i)}, r_i^{(l,i)}) \leq BP(x_i^{(l,i-1)}, r_i^{(l,i)}) \quad \text{according to Line 4-9 of Algorithm 3},
\]
which implies
\[
V_i^{(l,i)} = f^* - r_i^{(l,i)} - \frac{\alpha}{2}P(x_i^{(l,i)}, r_i^{(l,i)}) \geq f^* - r_i^{(l,i-1)} - \frac{\alpha}{2}BP(x_i^{(l,i-1), r_i^{(l,i-1)}}) = V_i^{(l,i-1)} + \frac{\alpha}{2}(1-B)P(x_i^{(l,i-1)}, r_i^{(l,i-1)}).
\]
Since \(r_i^{(l,i)} < r_i^{(l,k)} < f^*\) and \(\theta \leq \frac{H(r_i^{(l,i)})}{f^* - r_i^{(l,i)}} \leq \frac{P(x_i^{(l,i-1)}, r_i^{(l,i-1)})}{f^* - r_i^{(l,i-1)}}\) by Lemma 3, (34) further implies
\[
V_i^{(l,i)} \geq V_i^{(l,i-1)} + \frac{\alpha}{2}(1-B)\theta(f^* - r_i^{(l,i-1)}),
\]
where the second inequality is because \(P(x_i^{(l,i-1)}, r_i^{(l,i-1)}) > H(r_i^{(l,i-1)}) > 0\).

Given that \(i < k\), we have \(\alpha P(x_i^{(l,i)}, r_i^{(l,i)}) < f^* - r_i^{(l,i)}\) after \(T_0\) so that
\[
V_i^{(l,i)} = f^* - r_i^{(l,i)} - \frac{\alpha}{2}P(x_i^{(l,i)}, r_i^{(l,i)}) \geq f^* - r_i^{(l,i)} - \frac{\alpha}{2}P(x_i^{(l,i)}, r_i^{(l,i)}) = \frac{\alpha\epsilon}{2},
\]
where the last inequality is because \(r_i^{(l,i)} < f^*\) and \(x_i^{(l,i)}\) is not \(\epsilon\)-feasible and \(\epsilon\)-optimal (Lemma 2) before \(T_1\). On the other hand, we also have
\[
V_i^{(l,i)} < f^* - r_i^{(l,i)} \leq f^* - r_i^{(l,i)} = f^* - r_{\text{ini}} \quad \text{(46)}
\]
because $P(x_k^{(l,0)}, r_i^{(l)}) \geq H(r_i^{(l)}) > 0$ and $r_0^{(l)} = r_{ini}$ for any $l$ (Lemma 5). By the definition of $D_i$, (42), (44), (45), and (46), we have

$$f^* - r_{ini} \geq \left(1 + \frac{\alpha}{2} (1 - B) \theta\right)^{D_i} \frac{\alpha \epsilon}{2}.$$  

(47)

Taking logarithmic transformation on both sides and organizing terms lead to (41).

References

1. Aravkin, A.Y., Burke, J.V., Druytsvatsky, D., Friedlander, M.P., Roy, S.: Level-set methods for convex optimization. Mathematical Programming 174(1-2), 359–390 (2019)
2. Auslender, A., Crouzeix, J.P.: Global regularity theorems. Mathematics of Operations Research 13(2), 243–253 (1988)
3. Bayandina, A., Dvurechensky, P., Gasnikov, A., Stonyakin, F., Titov, A.: Mirror descent and convex optimization problems with non-smooth inequality constraints. In: Large-Scale and Distributed Optimization, pp. 181–213. Springer (2018)
4. Beck, A., Teboulle, M.: Smoothing and first order methods: A unified framework. SIAM Journal on Optimization 22(2), 557–580 (2012)
5. Bolte, J., Nguyen, T.P., Peyrouquet, J., Suter, B.W.: From error bounds to the complexity of first-order descent methods for convex functions. Mathematical Programming 165(2), 471–507 (2017)
6. Fercoq, O., Alacaoglu, A., Necoara, I., Cevher, V.: Almost surely constrained convex optimization. arXiv preprint arXiv:1902.00126 (2019)
7. Fercoq, O., Qu, Z.: Adaptive restart of accelerated gradient methods under local quadratic growth condition. IMA Journal of Numerical Analysis 39(4), 2069–2095 (2019)
8. Freund, R.M., Lu, H.: New computational guarantees for solving convex optimization problems with first order methods, via a function growth condition measure. Mathematical Programming 170(2), 445–477 (2018)
9. Gilpin, A., Pena, J., Sandholm, T.: First-order algorithm with $o(\ln(1/\epsilon))$ convergence for $\epsilon$-equilibrium in two-person zero-sum games. Mathematical Programming 133(1-2), 279–298 (2012)
10. Grimmer, B.: Radial subgradient method. SIAM Journal on Optimization 28(1), 459–469 (2018)
11. Grimmer, B.: General convergence rates follow from specialized rates assuming growth bounds. arXiv preprint arXiv:1905.06275 (2019)
12. Hoffman, A.J.: On approximate solutions of systems of linear inequalities. Journal of Research of the National Bureau of Standards 49(4) (1952)
13. Iouditski, A., Nesterov, Y.: Primal-dual subgradient methods for minimizing uniformly convex functions. arXiv preprint arXiv:1401.1792 (2014)
14. Johnstone, P.R., Moulin, P.: Faster subgradient methods for functions with hölderian growth. Mathematical Programming 180(1), 417–450 (2020)
15. Lan, G., Zhou, Z.: Algorithms for stochastic optimization with expectation constraints. arXiv preprint arXiv:1604.03887 (2016)
16. Lin, Q., Ma, R., Yang, T.: Level-set methods for finite-sum constrained convex optimization. In: International Conference on Machine Learning, pp. 3112–3121 (2018)
17. Lin, Q., Nadarajah, S., Soheili, N.: A level-set method for convex optimization with a feasible solution path. SIAM Journal on Optimization 28(4), 3290–3311 (2018)
18. Lin, Q., Nadarajah, S., Soheili, N., Yang, T.: A data efficient and feasible level set method for stochastic convex optimization with expectation constraints. arXiv pp. arXiv–1908 (2019)
19. Liu, M., Yang, T.: Adaptive accelerated gradient converging method under $h^{1/2} \{0\}$ Hölderian error bound condition. In: Advances in Neural Information Processing Systems, pp. 73–81 (2014)
20. Luo, X.D., Luo, Z.Q.: Extension of hoffman’s error bound to polynomial systems. SIAM Journal on Optimization 4(2), 383–392 (1994)
21. Luo, Z.Q., Pang, J.S.: Error bounds for analytic systems and their applications. Mathematical Programming 67(1-3), 1–28 (1994)
22. Luo, Z.Q., Sturm, J.F.: Error bounds for quadratic systems. In: High performance optimization, pp. 383–404. Springer (2000)
23. Luo, Z.Q., Tseng, P.: Error bounds and convergence analysis of feasible descent methods: a general approach. Annals of Operations Research 46(1), 157–176 (1993)
24. Mangasarian, O.L.: A condition number for differentiable convex inequalities. Mathematics of Operations Research 10(2), 175–179 (1985)
27. Necoara, I., Nesterov, Y., Glineur, F.: Linear convergence of first order methods for non-strongly convex optimization. Mathematical Programming 175(1-2), 69–107 (2019)
28. Nesterov, Y.: Smooth minimization of non-smooth functions. Mathematical programming 103(1), 127–152 (2005)
29. Nesterov, Y.: Gradient methods for minimizing composite functions. Mathematical Programming 140(1), 125–161 (2013)
30. Nesterov, Y.: Lectures on convex optimization, vol. 137. Springer (2018)
31. Renegar, J.: "efficient" subgradient methods for general convex optimization. SIAM Journal on Optimization 26(4), 2649–2676 (2016)
32. Renegar, J., Grimmer, B.: A simple nearly-optimal restart scheme for speeding-up first order methods. arXiv preprint arXiv:1803.00151 (2018)
33. Robinson, S.M.: An application of error bounds for convex programming in a linear space. SIAM Journal on Control 13(2), 271–273 (1975)
34. Roulet, V., d’Aspremont, A.: Sharpness, restart and acceleration. In: Advances in Neural Information Processing Systems, pp. 1119–1129 (2017)
35. Wei, X., Yu, H., Ling, Q., Neely, M.: Solving non-smooth constrained programs with lower complexity than $O(1/\epsilon)$: A primal-dual homotopy smoothing approach. In: Advances in Neural Information Processing Systems, pp. 3995–4005 (2018)
36. Xu, Y.: First-order methods for constrained convex programming based on linearized augmented lagrangian function. arXiv preprint arXiv:1711.08020 (2017)
37. Xu, Y.: Global convergence rates of augmented lagrangian methods for constrained convex programming. arXiv preprint arXiv:1711.05812 10 (2017)
38. Xu, Y.: Primal-dual stochastic gradient method for convex programs with many functional constraints. arXiv preprint arXiv:1802.02724 (2018)
39. Xu, Y., Lin, Q., Yang, T.: Adaptive svrg methods under error bound conditions with unknown growth parameter. In: Advances in Neural Information Processing Systems, pp. 3277–3287 (2017)
40. Xu, Y., Lin, Q., Yang, T.: Stochastic convex optimization: Faster local growth implies faster global convergence. In: International Conference on Machine Learning, pp. 3821–3830 (2017)
41. Xu, Y., Liu, M., Lin, Q., Yang, T.: Admm without a fixed penalty parameter: Faster convergence with new adaptive penalization. In: Advances in Neural Information Processing Systems, pp. 1267–1277 (2017)
42. Xu, Y., Yan, Y., Lin, Q., Yang, T.: Homotopy smoothing for non-smooth problems with lower complexity than $o(1/\epsilon)$. In: Advances In Neural Information Processing Systems, pp. 1208–1216 (2016)
43. Xu, Y., Yang, T.: Frank-wolfe method is automatically adaptive to error bound condition. arXiv preprint arXiv:1810.04765 (2018)
44. Yan, Y., Xu, Y., Lin, Q., Zhang, L., Yang, T.: Stochastic primal-dual algorithms with faster convergence than $o(1/\sqrt{T})$ for problems without bilinear structure. arXiv preprint arXiv:1904.10112 (2019)
45. Yang, T., Lin, Q.: Rsg: Beating subgradient method without smoothness and strong convexity. The Journal of Machine Learning Research 19(1), 236–268 (2018)
46. Yang, T., Lin, Q., Zhang, L.: A richer theory of convex constrained optimization with reduced projections and improved rates. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70, pp. 3901–3910. JMLR. org (2017)
47. Yu, H., Neely, M., Wei, X.: Online convex optimization with stochastic constraints. In: Advances in Neural Information Processing Systems, pp. 1428–1438 (2017)
48. Zhang, H.: New analysis of linear convergence of gradient-type methods via unifying error bound conditions. Mathematical Programming 180(1), 371–416 (2020)
49. Zhang, H., Dai, Y.H., Guo, L., Peng, W.: Proximal-like incremental aggregated gradient method with linear convergence under bregman distance growth conditions. Mathematics of Operations Research (2019)