A LEVEL-SET METHOD FOR CONVEX OPTIMIZATION WITH A FEASIBLE SOLUTION PATH

QIHANG LIN*, SELVAPRABU NADARAJAH†, AND NEGAR SOHEILI‡

Abstract. Large-scale constrained convex optimization problems arise in several application domains. First-order methods are good candidates to tackle such problems due to their low iteration complexity and memory requirement. The level-set framework extends the applicability of first-order methods to tackle problems with complicated convex objectives and constraint sets. Current methods based on this framework either rely on the solution of challenging subproblems or do not guarantee a feasible solution, especially if the procedure is terminated before convergence. We develop a level-set method that finds an $\epsilon$-relative optimal and feasible solution to a constrained convex optimization problem with a fairly general objective function and set of constraints, maintains a feasible solution at each iteration, and only relies on calls to first-order oracles. We establish the iteration complexity of our approach, also accounting for the smoothness and strong convexity of the objective function and constraints when these properties hold. The dependence of our complexity on $\epsilon$ is similar to the analogous dependence in the unconstrained setting, which is not known to be true for level-set methods in the literature. Nevertheless, ensuring feasibility is not free. The iteration complexity of our method depends on a condition number, while existing level-set methods that do not guarantee feasibility can avoid such dependence.

Key words. constrained convex optimization, level-set technique, first-order methods, complexity analysis

AMS subject classifications. 90C25, 90C30, 90C52

1. Introduction. Large-scale constrained convex optimization problems arise in several business, science, and engineering applications. A commonly encountered form for such problems is

\begin{align}
(1) & \quad f^* := \min_{x \in \mathcal{X}} f(x) \\
(2) & \quad \text{s.t. } g_i(x) \leq 0, \quad i = 1, \ldots, m,
\end{align}

where $\mathcal{X} \subseteq \mathbb{R}^n$ is a closed and convex set, and $f$ and $g_i$, $i = 1, \ldots, m$, are convex real functions defined on $\mathcal{X}$. Given $\epsilon > 0$, a point $x^\epsilon$ is $\epsilon$-optimal if it satisfies $f(x^\epsilon) - f^* \leq \epsilon$ and $\epsilon$-feasible if it satisfies $\max_{i=1,\ldots,m} g_i(x^\epsilon) \leq \epsilon$. In addition, given $0 < \epsilon \leq 1$, and a feasible and suboptimal solution $e \in \mathcal{X}^*$, a point $x^\epsilon$ is $\epsilon$-relative optimal with respect to $e$ if it satisfies $(f(x^\epsilon) - f^*)/(f(e) - f^*) \leq \epsilon$.

Level-set techniques [2, 3, 11, 17, 27, 28] provide a flexible framework for solving the optimization problem (1)-(2). Specifically, this problem can be reformulated as a convex root finding problem $L^*(t) = 0$, where $L^*(t) = \min_{x \in \mathcal{X}} \max_{i=1,\ldots,m} \{f(x) - t; g_i(x), i = 1, \ldots, m\}$ (see Lemaréchal et al. [17]). Then a bundle method [16, 29, 14] or an accelerated gradient method [20, Section 2.3.5] can be applied to handle the minimization subproblem to estimate $L^*(t)$ and facilitate the root finding process. In general, these methods solve the subproblem using a nontrivial quadratic program that needs to be solved at each iteration.

An important variant of (1)-(2) arising in statistics, image processing, and signal processing occurs when $f(x)$ takes a simple form, such as a gauge function, and there is a single constraint with the structure $\rho(Ax - b) \leq \lambda$, where $\rho$ is a convex function, $A$ is a matrix, $b$ is a vector, and $\lambda$ is a scalar. In this setting, Van Den Berg and

* Tippie College of Business, University of Iowa, USA, qihang-lin@uiowa.edu
† College of Business Administration, University of Illinois at Chicago, USA, selvan@uic.edu
‡ College of Business Administration, University of Illinois at Chicago, USA, nazad@uic.edu

This manuscript is for review purposes only.
Friedlander [27, 28] and Aravkin et al. [2] use a level-set approach to swap the roles of the objective function and constraints in (1)-(2) to obtain the convex root finding problem \( v(t) = 0 \) with \( v(t) := \min_{x \in X, f(x) \leq t} \rho(Ax - b) - \lambda \). Harchaoui et al. [11] proposed a similar approach. The root finding problem arising in this approach is solved using inexact newton and secant methods in [27, 28, 2] and [2], respectively. Since \( f(x) \) is assumed to be simple, projection onto the feasible set \( \{ x \in \mathbb{R}^n | x \in X, f(x) \leq t \} \) is possible and the subproblem in the definition of \( v(t) \) can be solved by various first-order methods. Although the assumption on the objective function is important, the constraint structure is not limiting because (2) can be obtained by choosing \( A \) to be an identity matrix, \( b = 0 \), \( \lambda = 0 \), and \( \rho(x) = \max_{i=1,...,m} g_i(x) \).

The level-set techniques discussed above find an \( \epsilon \)-feasible and \( \epsilon \)-optimal solution to (1)-(2), that is, the terminal solution may not be feasible. This issue can be overcome when the infeasible terminal solution is also super-optimal and a strictly feasible solution to the problem exists. In this case, a radial-projection scheme [2, 25] can be applied to obtain a feasible and \( \epsilon \)-relative optimal solution. Super-optimality of the terminal solution holds if the constraint \( f(x) \leq t \) is satisfied at termination. This condition can be enforced when projections on to the feasible set with this constraint are computationally cheap, which typically requires \( f(x) \) to be simple (e.g. see [2]). Otherwise, such a guarantee does not exist.

Building on this rich literature, we develop a feasible level-set method that (i) employs a first-order oracle to solve subproblems and compute a feasible and \( \epsilon \)-relative optimal solution to (1)-(2) with a general objective function and set of constraints while also maintaining feasible intermediate solutions at each iteration (i.e., a feasible solution path), and (ii) avoids potentially costly projections. Our focus on maintaining feasibility is especially desirable in situations where level-set methods need to be terminated before convergence, for instance due to a time constraint. In this case, our level-set approach would return a feasible, albeit possibly suboptimal, solution that can be implemented whereas the solution from existing level-set approaches may not be implementable due to potentially large constraint violations.

Convex optimization problems where constraints model operating conditions that need to be satisfied by solutions for implementation arise in several applications, including network and nonlinear resource allocation, signal processing, and circuit design [7, 8, 23].

The radial sub-gradient method proposed by Renegar [25] and extended by Grimmer [10] also emphasizes feasibility. This method avoids projection and finds a feasible and \( \epsilon \)-relative optimal solution by performing a line search at each iteration. This line search, while easy to execute for some linear or conic programs, can be challenging for general convex programs (1)-(2). Moreover, it is unknown whether the iteration complexity of this approach can be improved by utilizing the potential smoothness and strong convexity of the objective function and constraints. In contrast, the feasible level-set approach in this paper avoids the need for line search, in addition to projection, and its iteration complexity leverages the strong convexity and smoothness of the objective and constraints when these properties are true.

Our work is indeed related to first-order methods, which are popular due to their low per-iteration cost and memory requirement, for example compared to interior point methods, for solving large scale convex optimization problems [12]. Most existing first-order methods find a feasible and \( \epsilon \)-optimal solution to a version of the optimization problem (1)-(2) with a simple feasible region (e.g. a box, simplex or ball) which makes projection on to the feasible set at each iteration inexpensive. Ensuring feasibility using projection becomes difficult for general constraints (2). Some sub-gradient methods, including the method by Nesterov [20, Section 3.2.4] and its
recent variants by Lan and Zhou [15] and Bayandina [4], can be applied to (1)-(2) without the need for projection mappings. However, these methods do not ensure feasibility of the feasible solution as we do, that is, they instead find an $\epsilon$-feasible and $\epsilon$-optimal solution to (1)-(2) at convergence.

Given the focus of our research, one naturally wonders whether there is an associated computational cost of ensuring feasibility in the context of level-set methods. Our analysis suggests that the answer depends on the perspective one takes. First, consider the dependence of the iteration complexity on a condition measure as a criterion. A nice property of the level-set method in [2] is that its iteration complexity is independent of a condition measure for finding an $\epsilon$-feasible and $\epsilon$-optimal solution. Instead, finding an $\epsilon$-relative optimal and feasible solution leads to iteration complexities with dependence on condition measures when (i) combining the level-set approach [2] and the transformation in [25], or (ii) using our feasible level-set approach. This suggests that a cost of ensuring feasibility is the presence of a condition measure in the iteration complexity. Second, suppose we consider the dependence of the number of iterations on $\epsilon$ as our assessment criterion. The complexity result of the level-set approach provided by [2] can be interpreted as the product of the iteration complexity of the oracle used to solve the subproblem and a $O(\log(1/\epsilon))$ factor. The dependence of our algorithm would be similar under a non-adaptive analysis akin to [2]. However, using a novel adaptive analysis we find that the iteration complexity of our feasible level-set approach can in fact eliminate the $O(\log(1/\epsilon))$ factor. In other words, our complexity is analogous to the known complexity of first-order methods for unconstrained convex optimization and it is unclear to us whether a similar adaptive analysis can be applied to the approach in [2]. Overall, in terms of the dependence on $\epsilon$ there appears to be no cost to ensuring feasibility.

The rest of this paper is organized as follows. Section 2 introduces the level-set formulation and our feasible level-set approach, also establishing its outer iteration complexity to compute an $\epsilon$-relative optimal and feasible solution to the optimization problem (1)-(2). Section 3 describes two first-order oracles that solve the subproblem of our level-set formulation when the functions $f$ and $g_i$, $i = 1, \ldots, m$, are smooth and non-smooth, and in each case accounts for the potential strong convexity of these functions. The inner iteration complexity of each oracle is also analyzed in this section. Section 4 presents an adaptive complexity analysis to establish the overall iteration complexity of using our feasible level-set approach.

2. Feasible Level-set Method. Our methodological developments rely on the assumption below.

**Assumption 1.** There exists a solution $e \in \mathcal{X}$ such that $\max_{i=1,\ldots,m} g_i(e) < 0$ and $f(e) > f^*$. We label such a solution $g$-strictly feasible.

Given $t \in \mathbb{R}$, we define

$$L^*(t) := \min_{x \in \mathcal{X}} L(t, x),$$

where $L(t, x) := \max\{f(x) - t; g_i(x), i = 1, \ldots, m\}$ for all $x \in \mathcal{X}$. Let $\partial L^*(t)$ denote the sub-differential of $L^*$ at $t$. Lemma 1 summarizes known properties of $L^*(t)$.

**Lemma 1** (Lemmas 2.3.4, 2.3.5, and 2.3.6 in [20]). It holds that

(a) $L^*(t)$ is non-increasing and convex in $t$;
(b) $L^*(t) \leq 0$, if $t \geq f^*$ and $L^*(t) > 0$, if $t < f^*$. Moreover, under Assumption 1, it holds that $L^*(t) < 0$, for any $t > f^*$. 

This manuscript is for review purposes only.
Given $\Delta \geq 0$, we have $L^*(t) - \Delta \leq L^*(t + \Delta) \leq L^*(t)$. Therefore, $-1 \leq \zeta_L \leq 0$ for all $\zeta_L \in \partial L^*(t)$ and $t \in \mathbb{R}$.

**Proof.** We only prove the second part of Item (b) as the proofs of other items are provided in [20]. In particular, we show $L^*(t) < 0$ for any $t > f^*$ under Assumption 1.

Let $x^*$ denote an optimal solution to (1)-(2). If $t > f^*$ we have $f(x^*) - t < 0$. Since $f$ is a continuous function, there exists $\delta > 0$ such that for any $x \in B_\delta(x^*) \cap X$ we have $f(x) - t < 0$, where $B_\delta(x^*)$ is a ball of radius $\delta$ centered at $x^*$. Take $\bar{x} \in B_\delta(x^*) \cap X$ such that $\bar{x} = \lambda x^* + (1 - \lambda)e$ for some $\lambda \in (0, 1)$, where $e$ is a solution that satisfies Assumption 1. By the convexity of the functions $g_i$, $i = 1, \ldots, m$, the feasibility of $x^*$, and the $g$-strict feasibility of $e$, we have $g_i(\bar{x}) \leq \lambda g_i(x^*) + (1 - \lambda)g_i(e) < 0$ for $i = 1, \ldots, m$. In addition, since $\bar{x} \in B_\delta(x^*)$, we have $f(\bar{x}) - t < 0$. Therefore, $L^*(t) \leq \max\{f(\bar{x}) - t, g_i(\bar{x}), i = 1, \ldots, m\} < 0$.

Lemma 1(b), specifically $L^*(t) < 0$ for any $t > f^*$, ensures that the solution $t^* = f^*$ is a unique root to $L^*(t) = 0$ and the solution $x^*(t^*)$ determining $L^*(t^*)$ is an optimal solution to (1)-(2). In this case, it is possible to find an $\epsilon$-relative optimal and feasible solution by closely approximating $t^*$ from above with a $t$ and then finding a point $\bar{x} \in X$ with $L(t, \bar{x}) < 0$. The latter condition ensures $\bar{x}$ is feasible. In order to obtain such a $\bar{t}$, we start with an initial value $t_0 > t^*$ and generate a decreasing sequence $t_0 \geq t_1 \geq t_2 \geq \cdots \geq t^*$, where $t_k$ is updated to $t_{k+1}$ based on a term obtained by solving (3) with $t = t_k$ up to a certain optimality gap. We present this approach in Algorithm 1 which utilizes an oracle $A$ defined as follows.

**Definition.** An algorithm is called an oracle $A$ if, given $t \geq f^*$, $x \in X$, $\alpha > 1$ and $\bar{\gamma} \geq L(t, x) - L^*(t)$, it finds a point $x_+ \in X$, such that $L^*(t) \geq \alpha L(t, x_+)$. 

**Algorithm 1** Feasible Level-Set Method

1. **Input:** $g$-Strictly feasible solution $e$, and parameters $\alpha > 1$ and $0 < \epsilon \leq 1$.
2. **Initialization:** Set $x_0 = e$ and $t_0 = f(x_0)$.
3. for $k = 0, 1, 2, \ldots$ do
4. Choose $\bar{\gamma}_k$ such that $\bar{\gamma}_k \geq L(t_k, x_k) - L^*(t_k)$.
5. Generate $x_{k+1} = A(t_k, x_k, \alpha, \bar{\gamma}_k)$ such that $L^*(t_k) \geq \alpha L(t_k, x_{k+1})$.
6. if $\alpha L(t_k, x_{k+1}) \geq L(t_0, x_1)e$ then
7. Terminate and return $x_{k+1}$.
8. end if
9. Set $t_{k+1} = t_k + \frac{1}{2} L(t_k, x_{k+1})$.
10. end for

Given a $g$-strictly feasible solution $e$ and $\alpha > 1$, Algorithm 1 finds an $\epsilon$-relative optimal and feasible solution by calling an oracle $A$ with input tuple $(t_k, x_k, \alpha, \bar{\gamma}_k)$ at each iteration $k$. This oracle provides a lower bound, $\alpha L(t_k, x_{k+1}) \leq 0$, and an upper bound, $L(t_k, x_{k+1}) \leq 0$, for $L^*(t_k)$. We use the upper bound to update $t_k$ and the lower bound in the stopping criterion. In particular, Algorithm 1 terminates when $\alpha L(t_k, x_{k+1})$ is greater than or equal to $L(t_0, x_1)e$. We discuss possible choices for oracle $A$ in §3 and its input $\bar{\gamma}$ in Table 1. Theorem 2 shows that Algorithm 1 is a feasible level-set method, that is $x_k$ generated at each iteration $k$ is feasible, and terminates with an $\epsilon$-relative optimal solution. It also establishes its outer iteration complexity. This complexity depends on a condition measure

$$\beta := \frac{L^*(t_0)}{t_0 - t^*}.$$
for problem (1)-(2). The suboptimality of \( x_0 = e \) implies \( t_0 = f(x_0) > t^* \). By Lemma 1 we can show that \( \beta \in (0, 1] \). Larger values of \( \beta \) signify a better conditioned problem, as illustrated in Figure 1, because we iteratively use a close approximation of \( L^*(t) \) to approach \( t^* \) from \( t_0 \). Therefore, larger \( L^*(t) \) in absolute value intuitively suggests faster movement towards \( t^* \). In addition, the following inequalities can be easily verified from the convexity of \( L^*(t) \) and Lemma 1(c):

\[
-\beta = \frac{L^*(t_0)}{t_0 - t^*} \geq \frac{L^*(t)}{t - t^*} \geq -1, \quad \forall t \in (t^*, t_0].
\]

**Theorem 2.** Given \( \alpha > 1 \), \( 0 < \epsilon \leq 1 \), and a \( g \)-strictly feasible solution \( e \), Algorithm 1 maintains a feasible solution at each iteration and ensures

\[
t_{k+1} - t^* \leq \left( 1 - \frac{\beta}{2\alpha} \right) (t_k - t^*).
\]

In addition, it returns an \( \epsilon \)-relative optimal solution to (1)-(2) in at most

\[
K := \left\lceil \log_{1 - \frac{\alpha}{2\alpha}} \left( \frac{\alpha^2}{\beta \epsilon} \right) \right\rceil + 1
\]

outer iterations.

**Proof.** Notice that \( t_0 > t^* \) since \( t_0 = f(e) \). Suppose \( t^* \leq t_k \leq t_0 \) at iteration \( k \). At iteration \( k + 1 \) we have

\[
t_{k+1} - t^* = t_k - t^* + \frac{1}{2} L(t_k, x_{k+1})
\]

\[
\geq t_k - t^* + \frac{1}{2} L^*(t_k)
\]

\[
\geq t_k - t^* - \frac{1}{2} (t_k - t^*)
\]

\[
\geq 0,
\]
where the first equality is obtained by substituting for $t_{k+1}$ using the update equation for $t$ in Algorithm 1, the first inequality using the definition of $L^*(t_k)$, the second inequality from (4) with $t = t_k$, and the final inequality from $t_k - t^* \geq 0$. This implies that $t_k \geq t^*$ for all $k$. In addition, the condition $L^*(t_k) \geq \alpha L(t_k, x_{k+1})$ satisfied by the oracle $\mathcal{A}$, $\alpha > 1$, and Lemma 1(b) imply that $L(t_k, x_{k+1}) \leq L^*(t_k)/\alpha \leq 0$. It thus follows that $t_{k+1} \leq t_k$ and the $x_{k+1}$ maintained by Algorithm 1 is feasible for all $k$.

Next we establish (5) and the outer iteration complexity of Algorithm 1. Using the definition of $t_{k+1}$, $\alpha > 1$, and the condition $L^*(t_k) \geq \alpha L(t_k, x_{k+1})$ satisfied by oracle $\mathcal{A}$, we write

$$ t_{k+1} - t^* = t_k - t^* + \frac{1}{2} L(t_k, x_{k+1}) \leq t_k - t^* + \frac{L^*(t_k)}{2\alpha}. $$  

Using (4) with $t = t_k$ to substitute for $L^*(t_k)$ in (6) gives the required inequality (5).

Recursively using (5) starting from $k = 0$ gives

$$ t_k - t^* \leq \left(1 - \frac{\beta}{2\alpha}\right)^k (t_0 - t^*). $$

To obtain the terminal condition, we note that, for $k \geq K - 1$,

$$ -\alpha L(t_k, x_{k+1}) \leq -\alpha L^*(t_k) $$

$$ \leq \alpha (t_k - t^*) $$

$$ \leq \alpha \left(1 - \frac{\beta}{2\alpha}\right)^{K-1} (t_0 - t^*) $$

$$ \leq \frac{\epsilon}{\alpha} L^*(t_0) $$

$$ \leq -\epsilon L(t_0, x_1), $$

where the first inequality holds by the definition of $L^*(t)$; the second using (4) for $t = t_k$; the third using (7); the forth by applying the definitions of $K$ and $\beta$. The last inequality follows from the condition satisfied by the oracle at $t_0$ and $x_1$, that is, $\alpha L(t_0, x_1) \leq L^*(t_0)$. Hence, Algorithm 1 terminates after $K$ outer iterations.

Finally we proceed to show that the terminal solution is $\epsilon$-relative optimal. Observe that

$$ f(x_{k+1}) - f^* = f(x_{k+1}) - t^* \leq t_k - t^*, $$

where the equality is a consequence of the definition of $t^*$ and the inequality is obtained using $f(x_{k+1}) \leq t_k$, which holds because $f(x_{k+1}) - t_k \leq L(t_k, x_{k+1}) \leq 0$. Combining $t_k - t^* \leq (t_0 - t^*) L^*(t_k)/L^*(t_0)$ from (4) and (8), we obtain $f(x_{k+1}) - f^* \leq (t_0 - t^*)L^*(t_k)/L^*(t_0)$. Rearranging terms results in the inequality

$$ \frac{f(x_{k+1}) - f^*}{f(\epsilon) - f^*} \leq \frac{L^*(t_k)}{L^*(t_0)}, $$

where we use $t^* = f^*$ and $t_0 = f(\epsilon)$. Indeed, $L^*(t_k)/L^*(t_0) \leq \epsilon$ when the algorithm terminates. To verify this, the terminal condition and the property satisfied by oracle $\mathcal{A}$ indicate that $L^*(t_k) \geq \alpha L(t_k, x_{k+1}) \geq L(t_0, x_1) \epsilon \geq L^*(t_0) \epsilon$, which further implies

$$ \frac{L^*(t_k)}{L^*(t_0)} \leq \epsilon. $$

This manuscript is for review purposes only.
3. First-order oracles. In this section, we adapt first-order methods to be used as the oracle \( \mathcal{A} \) in Algorithm 1 to solve (3) under different properties of the convex functions \( f \) and \( g_i, i = 1, \ldots, m \). We adapt these methods in a manner that facilitates our adaptive complexity analysis in §4.

We assume functions \( f \) and \( g_i, i = 1, \ldots, m \), are \( \mu \)-convex for some \( \mu \geq 0 \). Namely, for \( h = f, g_1, \ldots, g_m \), we have

\[
(9) \quad h(x) \geq h(y) + \langle \zeta_h, x - y \rangle + \frac{\mu}{2} \|x - y\|^2
\]

for any \( x \) and \( y \) in \( \mathcal{X} \) and \( \zeta_h \in \partial h(y) \), where \( \partial h(y) \) denotes the set of all sub-gradients of \( h \) at \( y \in \mathcal{X} \) and \( \| \cdot \|_2 \) denotes the 2-norm. We call functions \( f \) and \( g_i, i = 1, \ldots, m \), convex if \( \mu = 0 \) and strongly convex if \( \mu > 0 \). Let

\[
D := \max_{x \in \mathcal{X}} \| x \|_2 \quad \text{and} \quad M := \max_{x \in \mathcal{X}} \{ \| \zeta_f \|_2; \| \zeta_{g_i} \|_2, i = 1, \ldots, m \},
\]

where \( \zeta_f \in \partial f(x) \) and \( \zeta_{g_i} \in \partial g_i(x) \), \( i = 1, \ldots, m \). When \( f \) and \( g_i, i = 1, \ldots, m \) are smooth, \( M := \max_{x \in \mathcal{X}} \max \{ \| \nabla f(x) \|_2; \| \nabla g_i(x) \|_2, i = 1, \ldots, m \} \).

We make Assumption 2 in the rest of the paper.

Assumption 2. \( M < +\infty \) and either (i) \( \mu > 0 \) or (ii) \( \mu = 0 \) and \( D < +\infty \).

Assuming \( M \) to be finite is a common assumption in the literature when developing first-order methods for solving smooth min-max problems (see e.g. [6, 24]) and non-smooth problems (see e.g. [9, 18, 22]). The assumption of finite \( D \) is present to obtain computable stopping criteria for the first-order oracles that we use.

3.1. Smooth convex functions. In this subsection, in addition to assumptions 1 and 2, we also assume that the functions in the objective and constrains of (1)-(2) are smooth, formally stated below.

Assumption 3. Functions \( f \) and \( g_i, i = 1, \ldots, m \), are \( S \)-smooth for some \( S \geq 0 \), namely, for \( h = f, g_1, \ldots, g_m \), \( h(x) \leq h(y) + \langle \nabla h(y), x - y \rangle + \frac{S}{2} \| x - y \|_2^2 \) for any \( x \) and \( y \) in \( \mathcal{X} \).

A first-order method achieves a fast linear convergence rate when the objective function is both smooth and strongly convex [20]. Despite this smoothness assumption, since the function \( L(t, x) \) is a maximum of functions modeling the objective and constraints, it may be non-smooth in \( x \). Nevertheless, we can approximate \( L(t, x) \) by a smooth and strongly convex function of \( x \) using a smoothing method [6, 21] and a standard regularization technique. Applying first-order methods to minimize this smooth approximation of \( L(t, x) \) leads to a lower iteration complexity than directly working with the original non-smooth function. The technique we utilize is a modified version of the smoothing and regularization reduction method by Allen-Zhu and Hazan [1]. However, since the formulation and the non-smooth structure of (3) are quite different from the problems considered in [1], we present below the reduction technique and conduct the complexity analysis under our framework. The exponentially smoothed and regularized version of \( L(t, x) \) is

\[
(10) \quad L_{\sigma, \lambda}(t, x) := \frac{1}{\sigma} \log \left( \exp(\sigma(f(x) - t)) + \sum_{i=1}^m \exp(\sigma g_i(x)) \right) + \frac{\lambda}{2} \| x \|_2^2,
\]

where \( \sigma > 0 \) and \( \lambda \geq 0 \) are smoothing and regularization parameters, respectively. Lemma 2 is based on related arguments in [6] and summarizes properties of \( L_{\sigma, \lambda}(t, x) \) and its “closeness” to \( L(t, x) \).
Suppose assumptions 2 and 3 hold. Given \( \sigma > 0 \) and \( \lambda \geq 0 \), the function \( L_{\sigma,\lambda}(t, x) \) is (i) \( (\sigma M^2 + S) \)-smooth and (ii) \( \lambda + \mu \)-convex. Moreover, we have

\[
0 \leq L_{\sigma,\lambda}(t, x) - L(t, x) \leq \frac{\log(m + 1)}{\sigma} + \frac{\lambda D^2}{2}.
\]

Here, we follow the convention that \( 0 + \infty = 0 \) so that \( \lambda D^2 = 0 \) if \( \lambda = 0 \) and \( D = +\infty \).

**Proof.** Item (i) and the relationship (11) are respectively minor modifications of Proposition 4.1 and Example 4.9 in [6]. Item (ii) directly follows from the \( \mu \)-convexity of \( f \), \( g_i \), \( i = 1, \ldots, m \) and \( \lambda \)-convexity of the function \( \frac{1}{2}\|x\|^2 \).

We solve the following version of (3) with \( L(t, x) \) replaced by \( L_{\sigma,\lambda}(t, x) \):

\[
\min_{x \in \mathcal{X}} L_{\sigma,\lambda}(t, x).
\]

Specifically, we apply a variant of Nesterov’s accelerated gradient method [5, 26] for a given \( t > t^* \) and a sequence of \( (\sigma, \lambda) \) with \( \sigma \) increasing to infinity and \( \lambda \) decreasing to zero. We refer to this gradient method as restarted smoothing gradient method (RSGM) and summarize its steps in Oracle 1. The choice of \( (\sigma, \lambda) \) and the number of iterations performed by RSGM depend on whether \( f \) and \( g_i \), \( i = 1, \ldots, m \) are strongly convex (\( \mu > 0 \)) or convex (\( \mu = 0 \)). Moreover, RSGM is re-started after \( N_i \) accelerated gradient method iterations with a smaller precision \( \gamma_i \) until this precision is below the required threshold of \( (1 - \alpha)L(t, x_i) \). Theorem 3 establishes that Oracle 1 returns a solution \( x_+ \in \mathcal{X} \) such that \( L^*(t) \geq \alpha L(t, x_+) \). Hence, RSGM can be used as an oracle \( \mathcal{A} \) (see Definition 1).

**Theorem 3.** Suppose assumptions 1, 2, and 3 hold. Given level \( t \in (t^*, t_0] \), solution \( x \in \mathcal{X} \), parameters \( \alpha > 1 \) and \( \bar{\gamma} \geq L(t, x) - L^*(t) \), Oracle 1 returns a solution \( x_+ \in \mathcal{X} \) that satisfies \( L^*(t) \geq \alpha L(t, x_+) \) in at most

\[
\left( \sqrt{\frac{40S}{\mu}} + 1 \right) \log_2 \left( \frac{\alpha \bar{\gamma}}{(1 - \alpha)L^*(t)} + 1 \right) + 3M \sqrt{\frac{160 \log(m + 1)}{\mu(1 - \alpha)L^*(t)}}
\]

iterations of the accelerated gradient method when \( \mu > 0 \) and

\[
12D \sqrt{\frac{10S\alpha}{(1 - \alpha)L^*(t)}} + \frac{4MD\alpha \sqrt{80 \log(m + 1)}}{(1 - \alpha)L^*(t)} + \log_2 \left( \frac{\alpha \bar{\gamma}}{(1 - \alpha)L^*(t)} + 1 \right)
\]

such iterations when \( \mu = 0 \) and \( D < +\infty \).

**Proof.** Let \( x_{\sigma,\lambda}(t) := \arg \min_{x \in \mathcal{X}} L_{\sigma,\lambda}(t, x) \) and \( L^*_{\sigma,\lambda}(t) := L_{\sigma,\lambda}(t, x^*_{\sigma,\lambda}(t)) \). We will first show by induction that at each iteration \( i = 0, 1, 2, \ldots \) of Oracle 1 the following inequality holds.

\[
L(t, x_i) - L^*(t) \leq \gamma_i.
\]

The validity of (12) for \( i = 0 \) holds because \( x_0 = x \), \( \gamma_0 = \bar{\gamma} \), and we know \( L(t, x) - L^*(t) \leq \bar{\gamma} \). Suppose (12) holds for iteration \( i \). We claim that it also holds for iteration \( i + 1 \) in both the strongly convex and convex cases. In fact, at iteration \( i + 1 \), we have

\[
L(t, x_{i+1}) - L^*(t) \leq L_{\sigma,\lambda_i}(t, x_{i+1}) - L^*_{\sigma,\lambda_i}(t) + \frac{\log(m + 1)}{\sigma_i} + \frac{\lambda_i D^2}{2}.
\]
Oracle 1 Restorted Smoothing Gradient Method: RSGM($t, x, \alpha, \tilde{\gamma}$)

1 **Input:** Level $t$, initial solution $x \in \mathcal{X}$, constant $\alpha > 1$, and initial precision $\tilde{\gamma} \geq L(t, x) - L^*(t)$.
2 Set $x_0 = x$ and $\gamma_0 = \tilde{\gamma}$.
3 for $i = 0, 1, 2, \ldots$ do
4 \hspace{1em} if $\gamma_i \leq (1 - \alpha)L(t, x_i)$ then
5 \hspace{2em} Terminate and return $x_i = x_i$.
6 \hspace{1em} end if
7 \hspace{1em} Set $\sigma_i = \left\{ \begin{array}{ll} \frac{4 \log(m + 1)}{\gamma_i} & \text{if } \mu > 0 \\
\frac{8 \log(m + 1)}{\gamma_i} & \text{if } \mu = 0 \end{array} \right.$ and $\lambda_i = \left\{ \begin{array}{ll} 0 & \text{if } \mu > 0 \\
\frac{\gamma_i}{4D^2} & \text{if } \mu = 0. \end{array} \right.$
8 Starting at $x_i$, apply the accelerated gradient method [26, Algorithm 1] on the optimization problem $\min_{x \in \mathcal{X}} L_{\sigma_i, \lambda_i}(t; x)$ for $N_i = \left\{ \begin{array}{ll} \max \left\{ \frac{40S}{\mu}, \frac{M \sqrt{160 \log(m + 1)}}{\sqrt{\mu \gamma_i}} \right\} & \text{if } \mu > 0 \\
\max \left\{ \frac{160SD^2}{\gamma_i}, \frac{4MD\sqrt{80 \log(m + 1)}}{\gamma_i} \right\} & \text{if } \mu = 0, \end{array} \right.$
9 \hspace{1em} Set $\gamma_{i+1} = \gamma_i/2$.
10 end for

The first inequality follows from Lemma 2; the second inequality from the $(\sigma_iM^2 + S)$-smoothness of $L_{\sigma_i, \lambda_i}(t, x)$ and the convergence property, i.e., $L_{\sigma_i, \lambda_i}(t, x_{i+1}) - L^*_{\sigma_i, \lambda_i}(t) \leq \frac{2(\sigma_iM^2 + S)\|x_i - x^*_{\sigma_i, \lambda_i}(t)\|^2}{N_i} + \frac{\log(m + 1)}{\sigma_i} + \frac{\lambda_iD^2}{2}$

$$\leq \frac{4(\sigma_iM^2 + S)(L_{\sigma_i, \lambda_i}(t, x_i) - L^*_{\sigma_i, \lambda_i}(t))}{(\mu + \lambda_i)N_i^2} + \frac{\log(m + 1)}{\sigma_i} + \frac{\lambda_iD^2}{2}$$

$$\leq \frac{5\gamma_i(\sigma_iM^2 + S)}{(\mu + \lambda_i)N_i^2} + \frac{\gamma_i}{4}$$

$$\leq \frac{\gamma_i}{8} + \frac{\gamma_i}{8} + \frac{\gamma_i}{4}$$

$$= \frac{\gamma_i}{2} = \gamma_{i+1}. \tag{318}$$

The manuscript is for review purposes only.
rest of the relationships follow from the inequalities $5\gamma_i \sigma_i M^2 / ((\mu + \lambda_i) N_i^2) \leq \gamma_i / 8$ and $5\gamma_i S / ((\mu + \lambda_i) N_i^2) \leq \gamma_i / 8$, which hold for both $\mu > 0$ and $\mu = 0$ according to the definitions of $\sigma_i$, $\lambda_i$, and $N_i$.

It can be easily verified that Oracle 1 terminates in at most $I$ iterations, where $I := \log_2 \left( \frac{(1-\alpha)}{L^*(t)} + 1 \right) \geq 0$. In fact, according to (12), for any $i \geq I$, we have

$$L(t, x_i) - L^*(t) \leq \frac{\gamma_i}{N_i} \leq \frac{\gamma_i}{2I} \leq \frac{1-\alpha}{\alpha} L^*(t),$$

which implies that $\alpha L(t, x_i) \leq L^*(t)$.

Hence $\gamma_i \leq \frac{\alpha}{2} \left( \alpha L(t, x_i) \right) = (1-\alpha)L(t, x_i)$ and Oracle 1 terminates.

We next analyze the total number of iterations across all calls of the accelerated gradient method [26, Algorithm 1] during the execution of Oracle 1.

Suppose $\mu > 0$. The number of iterations taken by Oracle 1 can be bounded as follows:

$$\sum_{i=0}^{I-1} N_i \leq \sum_{i=0}^{I-1} \sqrt{\frac{40S}{\mu}} + \sum_{i=0}^{I-1} M \sqrt{\frac{160\log(m+1)}{\mu \gamma_i}} + \sum_{i=0}^{I-1} \left( \frac{1}{2} \right)^{1/2}.$$

$$= \left( \sqrt{\frac{40S}{\mu}} + 1 \right) I + M \sqrt{\frac{160\log(m+1)}{\mu \gamma_0}} \sum_{i=0}^{I-1} \left( \frac{1}{\gamma_i} \right)^{1/2}$$

$$= \left( \sqrt{\frac{40S}{\mu}} + 1 \right) I + M \sqrt{\frac{160\log(m+1)}{\mu \gamma_0}} \sum_{i=0}^{I-1} 2^{i/2}$$

$$= \left( \sqrt{\frac{40S}{\mu}} + 1 \right) I + M \sqrt{\frac{160\log(m+1)}{\mu \gamma_0}} \left( \frac{1}{\gamma_0} \right)^{1/2}$$

$$\leq \left( \sqrt{\frac{40S}{\mu}} + 1 \right) \log_2 \left( \frac{\alpha \gamma_0}{(1-\alpha)L^*(t)} + 1 \right) + 3M \sqrt{\frac{160\alpha \log(m+1)}{\mu (1-\alpha)L^*(t)}}.$$

The first inequality above is obtained using the inequality $[a] \leq a + 1$ for $a \in \mathbb{R}_+$, and by bounding the maximum determining $N_i$ by a sum of its terms; the first equality from simplifying terms; the second equality from using the definition of $\gamma_i$; the third equality from $\gamma_0 = \gamma$ and the geometric sum formula $\sum_{i=0}^{I-1} 2^{i/2} = (1-2^{I/2})/(1-\sqrt{2})$; and the last inequality by substituting $I$, using the relationship $\sqrt{a+1} \leq \sqrt{a} + 1$ for $a \in \mathbb{R}_+$ to obtain the inequality

$$2^{I/2} - 1 = \left( \frac{\alpha \gamma_0}{(1-\alpha)L^*(t)} + 1 \right)^{1/2} - 1 \leq \left( \frac{\alpha \gamma_0}{(1-\alpha)L^*(t)} \right)^{1/2},$$

and bounding $1/(\sqrt{2} - 1)$ above by 3.

Suppose $\mu = 0$. The iteration complexity can be bounded using the following steps:

$$\sum_{i=0}^{I-1} N_i \leq \sum_{i=0}^{I-1} 4D \sqrt{\frac{10S}{\gamma_i}} + \sum_{i=0}^{I-1} M D \sqrt{\frac{80\log(m+1)}{\gamma_i}} + \sum_{i=0}^{I-1} \left( \frac{1}{2} \right)^{1/2}$$

$$= 4D \sqrt{\frac{10S}{\gamma_0}} \sum_{i=0}^{I-1} 2^{i/2} + 4MD \sqrt{\frac{80\log(m+1)}{\gamma_0}} \sum_{i=0}^{I-1} 2^i + I$$

$$= 4D \sqrt{\frac{10S}{\gamma_0}} \left( \frac{1}{1-\sqrt{2}} \right) + 4MD \sqrt{\frac{80\log(m+1)}{\gamma_0}} \left( \frac{1-2^I}{1-2} \right) + I.$$

This manuscript is for review purposes only.
\[
\leq 12D \sqrt{\frac{10S\alpha}{(1 - \alpha)L^*(t)}} + 4\alpha MD \sqrt{\frac{80\log(m + 1)}{(1 - \alpha)L^*(t)}} + \log_2 \left( \frac{\alpha\gamma}{(1 - \alpha)L^*(t)} + 1 \right)
\]

The first inequality follows from replacing the maximum in the definition of \( N_i \) by the sum of its terms and using \( \lceil a \rceil \leq a + 1 \) for \( a \in \mathbb{R}_+ \), the first and second equalities by rearranging terms and using the geometric sum formula, respectively, and the last inequality by substituting for \( I \) and using the inequality (13).

### 3.2. Non-smooth functions.
In this subsection, we only enforce assumptions 1 and 2. In other words, the functions \( f \) and \( g_i \), \( i = 1, \ldots, m \) can be non-smooth. Under Assumption 2, we have \( \max_{x \in X} \| \zeta_{L,t}(x) \|_2 \leq M \) for any \( \zeta_{L,t}(x) \in \partial_x L(t, x) \) and \( t \in \mathbb{R} \), where \( \partial_x L(t, x) \) denotes the set of all sub-gradients of \( L(t, \cdot) \) at \( x \) with respect to \( x \).

As oracle \( A \), we choose the standard sub-gradient method with varying stepsizes from [19] and its variant in [13], respectively, when \( \mu = 0 \) and \( \mu > 0 \) to directly solve (3). We present this sub-gradient method for both these choices of \( \mu \) in Oracle 2. The iteration complexity of Oracle 2 is well-known and given in Proposition 1 without proof. The proof for \( \mu = 0 \) can be found in [19, §2.2] and and the one for \( \mu > 0 \) is given in [13, Section 3.2]. Here we only provide arguments to show that Oracle 2 will return a solution with the desired optimality gap required by Algorithm 1, which then qualifies this algorithm to be used as an oracle \( A \) (see Definition 1).

**Oracle 2 Sub-gradient Descent: SGD(\( t, x, \alpha \))**

1. **Input:** Level \( t \), initial solution \( x \in X \) and constant \( \alpha > 1 \)
2. Set \( x_0 = x \).
3. for \( i = 0, 1, 2, \ldots \) do
4. \[ \eta_i = \begin{cases} \frac{2}{(i + 2)\mu} & \text{if } \mu > 0, \\ \frac{\sqrt{2D}}{\sqrt{i + 1}M} & \text{if } \mu = 0, \end{cases} \]
5. and \( E_i := \begin{cases} \frac{M^2}{(i + 2)\mu} & \text{if } \mu > 0, \\ \frac{2\sqrt{2DM}}{\sqrt{i + 1}} & \text{if } \mu = 0. \end{cases} \)
6. Set \( \bar{x}_i = \begin{cases} \frac{2}{(i + 1)(i + 2)} \sum_{j=0}^{i} (j + 1)x_j & \text{if } \mu > 0, \\ \frac{1}{i + 1} \sum_{j=0}^{i} x_j & \text{if } \mu = 0. \end{cases} \)
7. Let \( x_{i+1} = \arg\min_{x \in X} \frac{1}{2} \| x - x_i + \eta_i \zeta_{L,t}(x_i) \|_2^2 \).
8. if \( E_i \leq (1 - \alpha)\alpha L(t, \bar{x}_i) \) then
9. Terminate and return \( x_+ = \bar{x}_i \).
10. end if
11. end for

**Proposition 1 ([13, 19]).** Given level \( t \in (t^*, t_0] \), solution \( x \in X \), and a parameter \( \alpha > 1 \), Oracle 2 returns a solution \( x_+ \in X \) that satisfies \( L^*(t) \geq \alpha L(t, x_+) \) in at

---

The original algorithms proposed in [13] and [19] are stochastic sub-gradient methods. Here, we apply these methods with deterministic sub-gradient.
most

\[ L(t, \bar{x}) - L^*(t) \leq \frac{2\sqrt{2}DM}{\sqrt{i + 1}} = E_i. \]

Hence, when \( i \geq \frac{8\alpha^2M^2D^2}{(\alpha - 1)L^*(t)} - 1 \), we have \( L(t, \bar{x}) - L^*(t) \leq E_i \leq \frac{1-\alpha}{\alpha} L^*(t) \), which implies \( L^*(t) \geq \alpha L(t, \bar{x}). \) Hence, \( E_i \leq \frac{1-\alpha}{\alpha} L^*(t) \leq (1 - \alpha)L(t, \bar{x}) \) which indicates that Oracle 2 terminates in no more than \( \frac{8\alpha^2M^2D^2}{(\alpha - 1)L^*(t)^2} \) iterations.

Now, suppose \( \mu > 0 \). Oracle 2, in this case, is the sub-gradient method in [13]. By Section 3.2 in [13], we have

\[ L(t, \bar{x}) - L^*(t) \leq \frac{M^2}{(i + 2)\mu}. \]

Therefore, when \( i \geq \frac{\alpha M^2}{\mu(\alpha - 1)L^*(t)} - 2 \), we have \( L(t, \bar{x}) - L^*(t) \leq E_i \leq \frac{1-\alpha}{\alpha} L^*(t) \), which implies \( L^*(t) \geq \alpha L(t, \bar{x}). \) In addition, the algorithm terminates after at most this many iterations because \( E_i \leq \frac{1-\alpha}{\alpha} (\alpha L(t, \bar{x})) \leq (1 - \alpha)L(t, \bar{x}). \)

This proposition indicates that the sub-gradient method in [19] and its variant in [13] qualify as oracles \( \mathcal{A} \) in Algorithm 1 when \( \mu = 0 \) and \( \mu > 0 \), respectively (see Definition 1). Since the iteration complexities (14) and (15) do not depend on the initial solution \( x \), the worst-case complexity does not restrict this choice of \( x \in X \). In addition, \( \bar{\gamma} \) is not needed for executing Oracle 2. Therefore, we omit it as an input to this algorithm even though the generic call to the oracle \( \mathcal{A} \) in Algorithm 1 includes \( \bar{\gamma} \).

4. Overall iteration complexity. We present an adaptive complexity analysis to establish the overall complexity of Algorithm 1 when using the oracles \( \mathcal{A} \) discussed in §3. We present two lemmas that are needed to show our main complexity result in Theorem 4.

**Lemma 3.** Suppose Algorithm 1 terminates at iteration \( K \). Then for any \( 1 \leq k \leq K \), we have

\[ L^*(t_k) \leq \frac{L^*(t_0)\epsilon}{2\alpha^2}. \]

**Proof.** We first use the relationship \( t_k = t_{k-1} + \frac{1}{2} L(t_{k-1}, x_k) \) to show \( L^*(t_k) \leq \frac{1}{2} L^*(t_{k-1}) \) for any \( k \geq 1 \). Defining \( x_k^* := \arg\min_{x \in X} L(t_k, x) \), we proceed as follows:

\[ L^*(t_k) \leq \max\{ f(x_k^* - t_{k-1} - \frac{1}{2} L(t_{k-1}, x_k); g_i(x_k^* - t_{k-1}), i = 1, \ldots, m \} \]
\[\begin{align*}
&= \max \{ f(x^*_{k-1}) - t_{k-1}; g_i(x^*_{k-1}) + \frac{1}{2} L(t_{k-1}, x_k), i = 1, \ldots, m \} \\
&\leq \frac{1}{2} L(t_{k-1}, x_k) \\
&\leq L^*(t_{k-1}) - \frac{1}{2} L(t_{k-1}, x_k)
\end{align*}\]

where the second inequality holds since \( L(t_{k-1}, x_k) \leq \frac{1}{\alpha} L^*(t_{k-1}) \leq 0 \) and the third inequality follows from the inequality \( L^*(t_{k-1}) \leq L(t_{k-1}, x_k) \). Since Algorithm 1 has not terminated at iteration \( k - 1 \) for \( 1 \leq k \leq K \), we have

\[ L^*(t_0) \epsilon \geq \alpha L(t_0, x_1) \epsilon > \alpha^2 L(t_{k-1}, x_k) \geq \alpha^2 L^*(t_{k-1}), \]

where the first inequality holds by the condition satisfied by oracle \( \mathcal{A} \), the second by the stopping criterion of Algorithm 1 not being satisfied, and the third by the definition of \( L^*(t_{k-1}) \). The inequality (16) then follows by combining (17) and (18).

**Lemma 4.** Suppose Algorithm 1 terminates at iteration \( K \). The following hold:

1. \( \sum_{k=0}^{K} \frac{1}{L^*(t_k)} \leq \frac{4\alpha^3}{-\beta^2 L^*(t_0) \epsilon} \);
2. \( \sum_{k=0}^{K} \frac{1}{\sqrt{L^*(t_k)}} \leq \frac{4\sqrt{2}\alpha^2}{\beta^{1.5} \sqrt{L^*(t_0) \epsilon}} \);
3. \( \sum_{k=0}^{K} \frac{1}{(L^*(t_k))^2} \leq \frac{16\alpha^5}{3\beta^3 (L^*(t_0) \epsilon)^2} \).

**Proof.** We only prove Item 1. Items 2 and 3 can be derived following analogous steps. We have

\[ \sum_{k=0}^{K} \frac{1}{L^*(t_k)} \leq \sum_{k=0}^{K} \frac{1}{\beta(t_k - t^*)} \]

\[ \leq \sum_{k=0}^{K} \frac{(1 - \frac{\beta}{2\alpha})^{K-k}}{\beta(t_k - t^*)} \]

\[ \leq \frac{1}{-\beta L^*(t_K)} \sum_{k=0}^{K} \left(1 - \frac{\beta}{2\alpha}\right)^{K-k} \]

\[ \leq \frac{2\alpha^2}{-\beta L^*(t_0) \epsilon} \sum_{k=0}^{K} \left(1 - \frac{\beta}{2\alpha}\right)^{K-k} \]

\[ \leq \frac{2\alpha^2}{-\beta L^*(t_0) \epsilon} \left[1 - (1 - \frac{\beta}{2\alpha})^{K+1}\right] \]

\[ \leq \frac{4\alpha^3}{-\beta^2 L^*(t_0) \epsilon}, \]

where the first inequality follows by (4); the second by (5); the third by (4); the fourth using (16) with \( k = K \); the first equality by the geometric sum; and the fifth inequality by dropping negative terms.

This manuscript is for review purposes only.
Table 1: Choices of the oracle $\mathcal{A}$ and $\bar{\gamma}_k$ in Algorithm 1

<table>
<thead>
<tr>
<th>Case</th>
<th>Oracle</th>
<th>Choice of $\bar{\gamma}_k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smooth (Asms 1, 2, &amp; 3)</td>
<td>Strongly convex $(\mu &gt; 0)$</td>
<td>1 $\bar{\gamma} \geq -L^*(t_0)$ if $k = 0$ $-\alpha L(t_{k-1},x_k)$ if $k \geq 1$</td>
</tr>
<tr>
<td>Convex $(\mu = 0 &amp; D &lt; +\infty)$</td>
<td>1</td>
<td>$2MD$</td>
</tr>
<tr>
<td>Non-smooth (Asms 1 &amp; 2)</td>
<td>Strongly convex $(\mu &gt; 0)$</td>
<td>2</td>
</tr>
<tr>
<td>Convex $(\mu = 0 &amp; D &lt; +\infty)$</td>
<td>2</td>
<td>N/A</td>
</tr>
</tbody>
</table>

We summarize the choice of oracle $\mathcal{A}$ and $\bar{\gamma}_k$ in each iteration of Algorithm 1 in Table 1 (asms abbreviates assumptions) and characterize the overall iteration complexity (measured by the total number of gradient iterations) of this algorithm in Theorem 4. The dependence on $\epsilon$ in our big-Oh complexity for solving the constrained convex optimization problem (1)-(2) and the known analogue of this complexity for the unconstrained version of this problem are interestingly the same. However, unlike the unconstrained case, the complexity terms in Theorem 4 depend on the condition measure $\beta$.

**Theorem 4.** Suppose the oracle $\mathcal{A}$ and $\bar{\gamma}_k$ in Algorithm 1 are chosen according to Table 1. Given $0 < \epsilon \leq 1$, $\alpha > 1$, and a $g$-strictly feasible solution $e$, Algorithm 1 returns an $\epsilon$-relative optimal and feasible solution to (1)-(2) in at most

\begin{align*}
&\mathcal{O}\left(\frac{1}{\beta^{1.5}} \sqrt{-L^*(t_0)} \epsilon\right) + \log(1- \frac{\alpha}{\beta^2}) + \log(1- \frac{1}{\beta^2}) + \log(1- \frac{1}{\beta L^*(t_0)}) + \log(1- \frac{1}{\beta^2 L^*(t_0)}) + \log(1- \frac{1}{\beta^3 L^*(t_0)^2}) \right), \\
&\mathcal{O}\left(\frac{1}{\beta^{2} L^*(t_0)} \epsilon\right) + \mathcal{O}\left(\frac{1}{\beta^{2} L^*(t_0)} \epsilon\right), \quad \text{and} \quad \mathcal{O}\left(\frac{1}{\beta^{3} L^*(t_0)^2} \epsilon\right) \right),
\end{align*}

gradient iterations, respectively, when the functions $f$ and $g_i$, $i = 1, \ldots, m$ are smooth strongly convex, smooth convex, non-smooth strongly convex, and non-smooth convex.

Notice that we have only presented the terms $\epsilon$, $\beta$, and $L^*(t_0)$ in our big-Oh complexity. The exact numbers of iterations are given explicitly in the proof.

**Remark 1.** Based on Table 1, when the functions $f$, $g_i$, $i = 1, \ldots, m$, are smooth and $\mu > 0$, the input $\bar{\gamma}$ used in the first call of Oracle 1 should be bounded below by $-L^*(t_0)$. We do not specify the choice of such $\bar{\gamma}_0$ in our complexity bound (see (23)). However, one simple choice of $\bar{\gamma}_0$, in this case, could be $\frac{1}{2\mu} \|\nabla f(e)\|_2^2$ since

\begin{align*}
-L^*(t_0) \leq & -\min_{x \in X} \max \left\{ \langle \nabla f(e), x - e \rangle + \frac{\mu}{2} \|x - e\|^2, g_i(x), i = 1, \ldots, m \right\} \\
& \leq -\min_{x \in X} \left\{ \langle \nabla f(e), x - e \rangle + \frac{\mu}{2} \|x - e\|^2 \right\} \\
& \leq \frac{1}{2\mu} \|\nabla f(e)\|^2,
\end{align*}

where the first inequality follows from (9) and the third holds since $x = e - \frac{1}{\mu} \nabla f(e)$ solves $\min_{x \in \mathbb{R}^n} \left\{ \langle \nabla f(e), x - e \rangle + \frac{\mu}{2} \|x - e\|^2 \right\}$. 

This manuscript is for review purposes only.
Proof. [Smooth strongly convex case] We first show that we can guarantee $\bar{\gamma}_k \geq \gamma_0 \geq -L^*(t_0)$ and $\bar{\gamma}_k = -\alpha L(t_k, x_k)$ for $k \geq 1$. Hence $\bar{\gamma}_k$ can be used as an input to Oracle 1.

When $k = 0$, we have $L(t_0, x_0) = 0$ and hence $L(t_0, x_0) - L^*(t_0) = -L^*(t_0) \leq \bar{\gamma}_0$.

Take $k \geq 1$. We show that

$$L(t_k, x_k) = \max\{f(x_k) - t_{k-1} - \frac{1}{2}L(t_{k-1}, x_k) + g_i(x_k), i = 1, \ldots, m\} \geq L(t_{k-1}, x_k) - \frac{1}{2}L(t_{k-1}, x_k)$$

where the first and the second inequalities hold because $L(t_{k-1}, x_k) \leq \frac{1}{\alpha}L^*(t_{k-1}) \leq 0$.

Furthermore, since $L^*(t)$ is a decreasing function in $t$ we have $L^*(t_{k-1}) \leq L^*(t_k)$ since $t_k \leq t_{k-1}$. Combining this with (19) yields $L(t_k, x_k) - L^*(t_k) \leq -L^*(t_{k-1}) \leq -\alpha L(t_{k-1}, x_k) = \bar{\gamma}_k$.

We next characterize the iteration complexity of oracle $A$ in each iteration of Algorithm 1 using Theorem 3. By virtue of this theorem, the total number of gradient iterations performed by Algorithm 1 is at most

$$\sum_{k=0}^{K} \left[ \left( \frac{40S}{\mu} + 1 \right) \log_2 \left( \frac{\alpha \bar{\gamma}_k}{(1-\alpha)L^*(t_k)} + 1 \right) + 3M \sqrt{\frac{160\alpha \log(m+1)}{\mu(1-\alpha)L^*(t_k)}} \right]$$

$$+ 3M \sqrt{\frac{160\alpha \log(m+1)}{\mu(1-\alpha)}} \sum_{k=0}^{K} \frac{1}{\sqrt{-L^*(t_k)}}$$

where we have replaced $\bar{\gamma}_k$ by $-\alpha L(t_{k-1}, x_k)$ for $k > 0$, and used the inequality $(1-\alpha)L^*(t_k) \leq (1-\alpha)L^*(t_{k-1}) \leq \alpha(1-\alpha)L(t_{k-1}, x_k)$. The first sum in (20) can be bounded as follows.

$$\sum_{k=1}^{K} \log_2 \left( \frac{(\alpha - 2\alpha^2)L(t_{k-1}, x_k)}{(1-\alpha)L^*(t_k)} \right)$$

$$= \log_2 \left( \prod_{k=1}^{K} \frac{(\alpha - 2\alpha^2)L(t_{k-1}, x_k)}{(1-\alpha)L^*(t_k)} \right)$$

$$= \log_2 \left( \left( \frac{2\alpha^2 - \alpha}{\alpha - 1} \right)^K \cdot \frac{L(t_0, x_1)}{L^*(t_K)} \prod_{k=1}^{K-1} \frac{L(t_k, x_{k+1})}{L^*(t_k)} \right)$$

$$= \log_2 \left( \left( \frac{2\alpha^2 - \alpha}{\alpha - 1} \right)^K \cdot \frac{L(t_0, x_1)}{L^*(t_K)} \right)$$

$$= \left[ K \log_2 \left( \frac{2\alpha^2 - \alpha}{\alpha - 1} \right) + \log_2 \left( \frac{L(t_0, x_1)}{L^*(t_K)} \right) \right]$$
where the first inequality holds because \( L(t_k, x_{k+1})/L^*(t_k) \leq 1 \) for all \( k = 0, 1, \ldots, K \), and the second inequality follows from setting \( K \) equal to \( \lceil \log_{(1-\frac{a}{\beta\epsilon})^{-1}} \left( \frac{\alpha^2}{\beta\epsilon} \right) + 1 \rceil \), using (16) for \( k = K \), and the inequality \( L^*(t_0) \leq L(t_0, x_1) \leq 0 \).

The second sum in (20) can be upper bounded using Item 2 of Lemma 4. Specifically,

\[
\sum_{k=0}^{K} 3M \sqrt{\frac{160a \log(m+1)}{(1-\alpha)\mu L^*(t_k)}} \leq \frac{12M\alpha^2}{\beta^{1.5}} \sqrt{\frac{320a \log(m+1)}{\mu(1-\alpha)L^*(t_0)\epsilon}}.
\]

Adding (21) and (22), we obtain the following upper bound on the total number of iterations (20) required by Algorithm 1:

\[
(\sqrt{\frac{40S}{\mu}} + 1) \left[ \left( \log_{(1-\frac{a}{\beta\epsilon})^{-1}} \left( \frac{\alpha^2}{\beta\epsilon} \right) + 1 \right) \log_2 \left( \frac{2\alpha^2 - \alpha}{\alpha - 1} \right) + \log_2 \left( \frac{2\alpha^2}{\epsilon} \right) \right] + \log_2 \left( \frac{\alpha\gamma_0}{(1-\alpha)L^*(t_0)} + 1 \right) + \frac{12M\alpha^2}{\beta^{1.5}} \left( \frac{320a \log(m+1)}{\mu(1-\alpha)L^*(t_0)\epsilon} \right)^{1/2}.
\]

[Smooth convex case] Notice that \( \bar{\gamma}_k = 2MD \) can be used as an input to Oracle 1 at every iteration \( k \geq 1 \) since

\[
L(t_k, x_k) - L^*(t_k) \leq - \langle \zeta_{L,t_k}(x_k), x_k^* - x_k \rangle \leq \|\zeta_{L,t_k}(x_k)\|_2 \|x_k^* - x_k\|_2 \leq 2MD,
\]

where the first inequality follows from the convexity of \( L(t, x) \) in \( x \), the second using the Cauchy-Schwarz inequality, and the last using the definition of \( M \) and \( D \). By theorems 2 and 3, the total number of iterations required by Algorithm 1 is equal to

\[
\sum_{k=0}^{K} \left[ 12D \sqrt{\frac{10S}{(1-\alpha)L^*(t_k)}} + \frac{4MD\alpha\sqrt{80\log(m+1)}}{(1-\alpha)L^*(t_k)} + \log_2 \left( \frac{\alpha\bar{\gamma}_k}{(1-\alpha)L^*(t_k)} + 1 \right) \right].
\]

An upper bound on this quantity is

\[
\sum_{k=0}^{K} \left[ 12D \sqrt{\frac{10S}{(1-\alpha)L^*(t_k)}} + \frac{4MD\alpha\sqrt{80\log(m+1)}}{(1-\alpha)L^*(t_k)} + \sqrt{\frac{2MD\alpha}{(1-\alpha)L^*(t_k)}} \right],
\]

which is obtained by replacing \( \bar{\gamma}_k = 2MD \) for all \( k \) and using the inequality \( \log(a) \leq \frac{a - 1}{a} \) for \( a > 1 \). Next using Lemma 4, we obtain the required upper bound on the total number of iterations taken by Algorithm 1 as

\[
\frac{48D\alpha^2}{\beta^{1.5}} \left( \frac{20S\alpha}{(1-\alpha)L^*(t_0)\epsilon} \right)^{1/2} + \frac{16MD\alpha^4}{\beta^2(1-\alpha)L^*(t_0)\epsilon}\left(80\log(m+1)\right)^{1/2} + \frac{8\alpha^2}{\beta^{1.5}} \left( \frac{MD\alpha}{(1-\alpha)L^*(t_0)\epsilon} \right)^{1/2}.
\]
[Non-smooth strongly convex case] By Lemma 4 and Proposition 1, the total number of iterations required by Algorithm 1 can be bounded above in a straightforward manner:
\[
\sum_{k=0}^{K} \frac{\alpha M^2}{\mu(1-\alpha)L^*(t_k)} \leq \frac{4M^2\alpha^4}{\mu^2(1-\alpha)L^*(t_0)\epsilon}.
\]

[Non-smooth convex case] In a similar fashion, by Lemma 4 and Proposition 1, the total number of iterations required by Algorithm 1 is upper bounded by
\[
\sum_{k=0}^{K} \frac{8\alpha^2M^2D^2}{((1-\alpha)L^*(t_k))^2} \leq \frac{43M^2D^2\alpha^7}{\beta^4((1-\alpha)L^*(t_0)\epsilon)^2}.
\]

REFERENCES


