

# UNIVERSAL CONDITIONAL GRADIENT SLIDING FOR CONVEX OPTIMIZATION\*

YUYUAN OUYANG<sup>†</sup> AND TREVOR SQUIRES<sup>‡</sup>

**Abstract.** In this paper, we present a first-order projection-free method, namely, the universal conditional gradient sliding (UCGS) method, for solving  $\varepsilon$ -approximate solutions to convex differentiable optimization problems. For objective functions with Hölder continuous gradients, we show that UCGS is able to terminate with  $\varepsilon$ -solutions with at most  $\mathcal{O}((M_\nu D_X^{1+\nu}/\varepsilon)^{2/(1+3\nu)})$  gradient evaluations and  $\mathcal{O}((M_\nu D_X^{1+\nu}/\varepsilon)^{4/(1+3\nu)})$  linear objective optimizations, where  $\nu \in (0, 1]$  and  $M_\nu > 0$  are the exponent and constant of the Hölder condition. Furthermore, UCGS is able to perform such computations without requiring any specific knowledge of the smoothness information  $\nu$  and  $M_\nu$ . In the weakly smooth case when  $\nu \in (0, 1)$ , both complexity results improve the current state-of-the-art  $\mathcal{O}((M_\nu D_X^{1+\nu}/\varepsilon)^{1/\nu})$  results [19, 8] on first-order projection-free method achieved by the conditional gradient method. Within the class of sliding-type algorithms followed from the work of [15, 13], to the best of our knowledge, this is the first time a sliding-type algorithm is able to improve not only the gradient complexity but also the overall complexity for computing an approximate solution. In the smooth case when  $\nu = 1$ , UCGS matches the state-of-the-art complexity result achieved by the conditional gradient sliding method [15], but adds more features allowing for practical implementation.

**Key words.** Convex optimization, first-order method, conditional gradient method, conditional gradient sliding, universal gradient method

**AMS subject classifications.** 90C25, 90C06, 49M37

**1. Introduction.** In this paper, we study first-order projection-free methods for computing  $\varepsilon$ -approximation solutions to convex optimization problems of form

$$(1.1) \quad f^* = \min_{x \in X} f(x).$$

Here  $X \subset \mathbb{R}^n$  is a high-dimensional compact convex set,  $f$  is a convex function, and our goal is to compute an  $\varepsilon$ -solution  $y \in X$  such that  $f(y) - f^* \leq \varepsilon$ . We make the following assumptions concerning the set  $X$  and the objective function  $f$ . Under the Euclidean norm  $\|\cdot\|$ , we assume that  $X$  is compact with diameter

$$(1.2) \quad D_X := \max_{x, y \in X} \|x - y\| < \infty$$

and that there exists Hölder exponent  $\nu \in (0, 1]$  and constant  $M_\nu > 0$  such that

$$(1.3) \quad f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{M_\nu}{1 + \nu} \|x - y\|^{1+\nu}, \quad \forall x, y \in X.$$

Our problem of interest covers both smooth ( $\nu = 1$ ) and weakly smooth ( $\nu \in (0, 1)$ ) convex optimization problems. Specifically, any convex differentiable function whose gradient is  $(\nu, M_\nu)$ -Hölder continuous, namely,

$$\|\nabla f(y) - \nabla f(x)\| \leq M_\nu \|y - x\|^\nu, \quad \forall x, y \in X$$

---

\*Submitted to the editors DATE.

**Funding:** The authors are partially supported by the Office of Naval Research grant N00014-20-1-2089.

<sup>†</sup>School of Mathematical and Statistical Sciences, Clemson University, Clemson, SC (yuyuan@clmson.edu).

<sup>‡</sup>School of Mathematical and Statistical Sciences, Clemson University, Clemson, SC (tsquire@clmson.edu).

satisfies (1.3).

Assuming that the projection subproblem  $\min_{x \in X} \|x - y\|^2$  can be solved exactly and efficiently for any  $y \in \mathbb{R}^n$ , first-order projection-based methods for solving the convex optimization problem in (1.1) has already been well studied in the literature. The classical iteration complexity theory [18] has established that the lower complexity bound on the number of gradient evaluations of  $\nabla f$  is

$$(1.4) \quad \mathcal{O}(M_\nu D_X^{1+\nu} / \varepsilon)^{2/(1+3\nu)}$$

for computing an  $\varepsilon$ -solution. Note that the above lower complexity bound becomes the widely known  $\mathcal{O}(\sqrt{M_1 D_X^2 / \varepsilon})$  lower complexity bounds for smooth convex problems (when  $\nu = 1$ ). For the smooth case, there have been a large number of literature that developed first-order methods whose performance matches the lower complexity bounds (see, e.g., the books/monographs [2, 1, 20, 14] and references with in). There also exist several first-order methods in the literature that are able to uniformly achieve the lower complexity bound (1.4) for any  $\nu \in [0, 1]$ , including for example the fast gradient method (FGM) developed in [22], the bundle-level type methods in [12], and the fast bundle-level method in [3] (for the case when  $X$  is either a Euclidean norm ball or  $\mathbb{R}^n$ ). Note that the methods in [22, 12, 3] are universal methods, in the sense that they do not require any knowledge on the values of  $\nu$  and  $M_\nu$  and are able to achieve the complexity (1.4) with the best possible  $\nu \in [0, 1]$  and  $M_\nu > 0$ . Such uniform property is appealing since it allows the methods in [22, 12, 3] to be applied to convex optimization problems without requiring any smoothness information, i.e., whether the problem is nonsmooth, smooth, or weakly smooth, while still achieve the lower complexity bound with respect to the best smoothness information.

However, it should be noted that we may not always be able to solve the projection subproblem  $\min_{x \in X} \|x - y\|^2$  exactly and efficiently. For example, if  $X$  is a general polyhedron, then computing the projection with high accuracy would be challenging when the dimension  $n$  is large. Recently, there has been studies on projection-free methods (see, e.g., [11, 10, 7]) that replaces the possibly difficult projection subproblem  $\min_{x \in X} \|x - y\|^2$  with the easier-to-solve linear objective subproblems  $\min_{x \in X} \langle c, x \rangle$ . Such methods can be traced back to [6, 16] and are known as the Frank-Wolfe or conditional gradient methods due to their origin. For the smooth case ( $\nu = 1$ ) of problem (1.1), it is shown in [11, 10, 7] that the number of gradient evaluation of  $\nabla f$  and linear objective optimization subproblems are upper bounded by  $\mathcal{O}(M_1 D_X^2 / \varepsilon)$ . In implementations the linear subproblems can also be solved approximately within certain accuracy while still maintain the same upper complexity bound. Here the number of linear objective optimization subproblems can not be improved; worse-case problem instances that requires solving at least such number of linear objective optimization subproblems has been shown in [11, 14]. For the general case when  $\nu \in (0, 1]$ , universal methods have been developed in [19, 8] that compute  $\varepsilon$ -solutions with at most  $\mathcal{O}((M_\nu D_X^{1+\nu} / \varepsilon)^{1/\nu})$  gradient evaluation of  $\nabla f$  and linear objective optimization subproblems.

Focusing on the number of gradient evaluations of  $\nabla f$  required by the aforementioned projection-free methods, we can observe a significant gap with the lower complexity bound in (1.4). For example, the number of gradient evaluations required by the universal methods in [19, 8] is upper bounded by  $\mathcal{O}(1/\varepsilon^3)$  when  $\nu = 1/3$ . This complexity is significantly worse than the lower complexity bound in (1.4) which is of order  $\mathcal{O}(1/\varepsilon)$  when  $\nu = 1/3$ . For the special smooth case (when  $\nu = 1$ ), the number of gradient evaluations required by the methods in [11, 10, 7] are upper bounded by  $\mathcal{O}(1/\varepsilon)$  and is significantly worse than the  $\mathcal{O}(1/\sqrt{\varepsilon})$  lower complexity bound in (1.4).

Recently, there has been a breakthrough on closing some of the gap in the gradient evaluations of  $\nabla f$  between the upper and lower complexity bounds. For the special smooth case (when  $\nu = 1$ ), a condition gradient sliding (CGS) method is proposed in [15] that is able to compute an  $\varepsilon$ -approximate solution of problem (1.1) with  $\mathcal{O}(\sqrt{M_1 D_X^2 / \varepsilon})$  gradient evaluations of  $\nabla f$  and  $\mathcal{O}(M_1 D_X^2 / \varepsilon)$  linear objective optimization subproblems. Here the number of gradient evaluations required by the CGS method matches that in the lower complexity bound in (1.4) (with  $\nu = 1$ ). Also, the number of linear objective subproblems required by the CGS method also matches the lower bound in [11, 14]. Therefore, CGS is the first method that reaches the performance limit of first-order projection-free methods for solving the special smooth case of problem (1.1). It should be noted that the CGS method in [15] requires the knowledge of the Lipschitz continuity constant  $M_1$  of the gradient  $\nabla f$  and does not have a termination criterion for verifying whether it has computed an  $\varepsilon$ -solution. A backtracking linesearch version of the CGS method is proposed recently in [17], which has the same computational complexity as in [15], while only requires an initial guess  $L_0 \leq \mathcal{O}(1)M_1$  and the diameter constant  $D_X$  for its computation. It is also able to terminate whenever it verifies the successful computation of an  $\varepsilon$ -solution.

None of the above literature on sliding-type algorithms discuss the weakly smooth case when  $\nu \in (0, 1)$  or the design of universal methods. In this paper, we propose to close the remaining gap in the gradient evaluations of  $\nabla f$  between its upper complexity bounds in projection-free methods and the lower complexity bounds in (1.4). Specifically, we propose a novel first-order projection-free method, namely the universal conditional gradient sliding (UCGS) method, that is able to compute an  $\varepsilon$ -solution of the problem (1.1) without requiring any knowledge of the smoothness information  $(\nu, M_\nu)$ . The framework of UCGS is built around that of the fast gradient [22] and conditional gradient sliding [15] methods. The contributions of this paper are summarized below.

First, in terms of gradient evaluations of  $\nabla f$ , the total number of evaluations required by the proposed UCGS method for computing an  $\varepsilon$ -solution is upper bounded uniformly by  $\mathcal{O}(M_\nu D_X^{1+\nu} / \varepsilon)^{2/(1+3\nu)}$  for any  $\nu \in (0, 1]$ . Such bound matches the lower complexity bound in (1.4). To the best of our knowledge, this is the first first-order projection-free method that is able to achieve such gradient evaluation complexity bound uniformly for smooth and weakly smooth convex optimization problems.

Second, the total number of linear objective subproblems required by the proposed UCGS method for computing an  $\varepsilon$ -solution is upper bounded uniformly by  $\mathcal{O}(M_\nu D_X^{1+\nu} / \varepsilon)^{4/(1+3\nu)}$  for any  $\nu \in (0, 1]$ . Comparing with the  $\mathcal{O}((M_\nu D_X^{1+\nu} / \varepsilon)^{1/\nu})$  result [19, 8] in the literature, the proposed UCGS method has the same complexity when  $\nu = 1$  and is significantly better for all  $\nu \in (0, 1)$ . For example, when  $\nu = 1/3$ , the UCGS method has significantly better complexity of  $\mathcal{O}(1/\varepsilon^2)$  comparing the  $\mathcal{O}(1/\varepsilon^3)$  result in [19, 8]. Within the class of sliding-type algorithms followed from the work of [15, 13], to the best of our knowledge, this is the first time a sliding-type algorithm is able to improve not only the gradient complexity but also the overall complexity for computing an approximate solution.

Third, the proposed UCGS method is able to achieve the aforementioned complexity bounds without any knowledge of the smooth information  $(\nu, M_\nu)$  of the objective function. Therefore, it is a universal method that is able to solve weakly smooth and smooth convex optimization problems with the best possible  $\nu \in (0, 1]$  and  $M_\nu > 0$ . Note that in the special smooth case when  $\nu = 1$ , the proposed UCGS method can be understood as an extension of the CGS method [15] with add features for practical implementation. In such case, it has the same complexity results as the CGS

method [15] and its backtracking linesearch edition [17] in terms of both gradient evaluations of  $\nabla f$  and linear objective subproblems. However, unlike the linesearch edition [17], by incorporating a different backtracking linesearch strategy with a novel parameter choice, UCGS no longer require any information on the continuity constant  $M_1$ . UCGS also allows that all linear objective optimization subproblems be solved approximately within certain accuracy while maintain the same complexity results.

This paper is organized as follows. In Section 2 we provide a generic description of algorithms for solving problem (1.1). We discuss the relation of our generic description to the previously mentioned algorithms and perform theoretical complexity analysis. In particular, we demonstrate the theoretical novelty of our paper on the improvement of both the gradient and linear objective optimization complexities over that of the conditional gradient method. In Section 3, we provide a practical version, namely UCGS, of the generic algorithm and prove that UCGS maintains the same complexity results. We report some preliminary numerical results in Section 4 and provide concluding remarks and potential future works in Section 5.

**2. Conditional gradient sliding method in the Hölder case.** In this section, we propose to analyze the conditional gradient (CG) and conditional gradient sliding (CGS) method in [15] through a generic description, which we call the generic universal gradient (GUG) method. GUG can be simply understood as a generic description of the CGS method and serves purely as a tool for our theoretical analysis of CGS in the Hölder case. We will show that a version of GUG achieves better gradient evaluation complexity than CG for solving problem (1.1). While CGS can already achieve better gradient evaluation than that of CG for problems with Lipschitz continuous gradients, our result covers a more general case of problems with Hölder continuous gradients. Moreover, we will also show a novel theoretical result that GUG can also achieve better complexity on linear objective optimizations than that of CG when the Hölder continuity exponent  $\nu \in (0, 1)$ . Such theoretical result is particularly interesting within the class of sliding-type algorithms followed from the works of [15, 13]. To the best of our knowledge, this is the first time a sliding-type algorithm is able to improve not only the gradient complexity but also the overall complexity for computing an approximate solution.

The iterations of GUG is described in Algorithm 2.1. Let us make a few remarks regarding Algorithm 2.1. First, in both Algorithms 2.1 and in the sequel, we refer to the operations between increments in  $t$  as an inner iteration and that of  $k$  as an outer iteration. To distinguish inner and outer iteration descriptions, we will use subscripts and superscripts to denote outer and inner iteration indices, respectively. Second, the relation (2.5) in the Approx-Subproblem procedure can be satisfied through different algorithms and allows both projection-based and projection-free implementations. For example, if we require  $\eta_k \equiv 0$ , then the Approx-Subproblem procedure solves a projection problem and the iterate  $x_k$  computed by the procedure is an optimal solution to the projection problem

$$(2.1) \quad \min_{x \in X} \phi_k(x) := \langle \nabla f(z_k), x \rangle + \frac{\beta_k}{2} \|x - x_{k-1}\|^2.$$

Consequently, GUG reduces to a version of Nesterov's accelerated gradient method (see, e.g., [21]). For our study, we will focus on a projection-free implementation of the Approx-Subproblem procedure, namely the conditional gradient method (CGM) procedure, described in Algorithm 2.2. Third, if  $\beta_k \equiv 0$ , then the subproblem (2.5) becomes a linear objection optimization and it takes exactly one inner iteration for

**Algorithm 2.1** Generic universal gradient (GUG) method

---

Start: Choose tolerance  $\varepsilon > 0$  and initial iteration  $x_0 \in X$ . Set  $y_0 = x_0$ .

**for**  $k = 1, 2, \dots, N$  **do**

$$(2.2) \quad \begin{aligned} z_k &= (1 - \gamma_k)y_{k-1} + \gamma_k x_{k-1} \\ x_k &= \text{Approx-Subproblem}(\nabla f(z_k), x_{k-1}, \beta_k, \eta_k) \end{aligned}$$

$$(2.3) \quad y_k = (1 - \gamma_k)y_{k-1} + \gamma_k x_k$$

**end for**

Output  $y_N$  as the approximate solution.

**procedure**  $u^+ = \text{APPROX-SUBPROBLEM}(g, u, \beta, \eta)$

Use any algorithm to compute an approximate solution  $u^+$  to problem

$$(2.4) \quad \min_{x \in X} \phi(x) := \langle g, x \rangle + \frac{\beta}{2} \|x - u\|^2$$

that satisfies

$$(2.5) \quad \max_{x \in X} \langle \nabla \phi(u^+), u^+ - x \rangle = \max_{x \in X} \langle g + \beta(u^+ - u), u^+ - x \rangle \leq \eta.$$

**end procedure**

---

**Algorithm 2.2** Conditional gradient method (CGM) procedure for solving (2.5)

---

**procedure**  $u^+ = \text{CGM}(g, u, \beta)$

Initialize  $u^0 = u$ .

**while**  $u^{t-1}$  does not satisfy (2.5) **do**

    Compute  $v^t$  such that

$$(2.6) \quad \max_{x \in X} \langle g + \beta(u^{t-1} - u), v^t - x \rangle \leq 0$$

    Set

$$u^t = (1 - \alpha^t)u^{t-1} + \alpha^t v^t$$

**end while**

Output  $u^+ = u^t$

**end procedure**

---

CGM to compute an optimal solution to this subproblem. Consequently, GUG reduces to CG. Note that by the description of  $v^t$  in (2.6),  $v^t$  is the optimal solution to the linear subproblem. If instead we allow the right hand side of (2.6) to be nonzero, then we can study practical implementation variants of CG that solve the linear objective optimization subproblem approximately (see, e.g., [7] and the references within). However, we will focus on theoretical analysis in this section; the approximate linear subproblem implementation will be discussed in next section. Finally, if the parameter  $\alpha^t$  in the CGM procedure is chosen as described in (3.12) later, then CGM is exactly CndG in [15], and GUG reduces to CGS. The key concept behind CGS, which distinguishes it from projection-based methods and CG, is that uses the CGM procedure with multiple inner iterations to compute an approximate solution to the projection problem (2.1). Instead of solving an optimal solution, CGS runs several inner iterations through the CGM procedure to compute an approximate solution  $x_k$  satisfying

$$\langle \nabla f(z_k) + \beta_k(x_k - x_{k-1}), x_k - x \rangle \leq \eta_k, \quad \forall x \in X,$$

where  $\beta_k > 0$ . By doing so, in the special smooth case (when  $\nu = 1$ ) of problem 1.1

CGS successfully reduces the total number of outer iterations, and hence skipping gradient evaluations, to  $\mathcal{O}(1/\sqrt{\varepsilon})$  from CG's  $\mathcal{O}(1/\varepsilon)$ , while not affecting the total number of linear objective optimizations. This feature of skipping gradient evaluations is termed ‘‘sliding’’ in [15] (see also [13]). An interesting discovery we will make in this section, is that the use of sliding also reduces the total number of inner iterations, and consequently linear objective optimizations when the Hölder exponent  $\nu \in (0, 1)$ . To the best of our knowledge, such discovery was not made previously in the literature of sliding-type algorithms.

We will now analyze the performance of Algorithm 2.1 under various parameter settings using the CGM procedures in Algorithm 2.2 to solve an approximate solution that satisfies the requirement (2.5) in GUG. We begin by building a recurrence relation on the outer iterates. Such recurrence provides us a tool for performing complexity analysis on GUG.

PROPOSITION 2.1. *Suppose that  $\gamma_k \in [0, 1]$  for all  $k$  in Algorithm 2.1. We have*

$$(2.7) \quad \begin{aligned} & f(y_k) - (1 - \gamma_k)f(y_{k-1}) - \gamma_k f(x) \\ & \leq \gamma_k \eta_k + \frac{\beta_k \gamma_k}{2} (\|x_{k-1} - x\|^2 - \|x_k - x\|^2) \\ & \quad - \frac{\beta_k \gamma_k}{2} \|x_k - x_{k-1}\|^2 + \frac{M_\nu \gamma_k^{1+\nu}}{1+\nu} \|x_k - x_{k-1}\|^{1+\nu}, \quad \forall k \geq 1, x \in X. \end{aligned}$$

Specially, if  $\nu \in (0, 1)$  and  $\beta_k > 0$  for all  $k$ , then

$$(2.8) \quad \begin{aligned} & f(y_k) - (1 - \gamma_k)f(y_{k-1}) - \gamma_k f(x) \\ & \leq \gamma_k \eta_k + \frac{\beta_k \gamma_k}{2} (\|x_{k-1} - x\|^2 - \|x_k - x\|^2) + \xi_k, \quad \forall k \geq 1, x \in X, \end{aligned}$$

where

$$(2.9) \quad \xi_k := \frac{1 - \nu}{2(1 + \nu)} M_\nu^{\frac{2}{1-\nu}} \left( \frac{\gamma_k}{\beta_k} \right)^{\frac{1+\nu}{1-\nu}}.$$

*Proof.* From the Hölder condition (1.3) and the convexity of  $f(x)$  we have

$$\begin{aligned} & f(y_k) - (1 - \gamma_k)f(y_{k-1}) - \gamma_k f(x) \\ & \leq f(z_k) + \langle \nabla f(z_k), y_k - z_k \rangle + \frac{M_\nu}{1 + \nu} \|y_k - z_k\|^{1+\nu} \\ & \quad - (1 - \gamma_k)(f(z_k) + \langle \nabla f(z_k), y_{k-1} - z_k \rangle) - \gamma_k(f(z_k) + \langle \nabla f(z_k), x - z_k \rangle) \\ & = \gamma_k \langle \nabla f(z_k), x_k - x \rangle + \frac{M_\nu \gamma_k^{1+\nu}}{1 + \nu} \|x_k - x_{k-1}\|^{1+\nu}. \end{aligned}$$

Here the last equality is from the definitions of  $z_k$  and  $y_k$  in (2.2) and (2.3) respectively. Noting that  $x_k$  is computed from the Approx-Subproblem procedure that satisfies (2.5), we have

$$\begin{aligned} & \langle \nabla f(z_k), x_k - x \rangle + \frac{\beta_k}{2} (\|x_k - x_{k-1}\|^2 + \|x_k - x\|^2 - \|x_{k-1} - x\|^2) \\ & = \langle \nabla f(z_k) + \beta_k(x_k - x_{k-1}), x_k - x \rangle \leq \eta_k, \quad \forall x \in X. \end{aligned}$$

Summarizing the above two relations we obtain (2.7). By Young's inequality (applied to the product of  $(\beta_k \gamma_k / (1 + \nu))^{(1+\nu)/2} \|x_k - x_{k-1}\|^{1+\nu}$  and  $M_\nu (\gamma_k / \beta_k)^{(1+\nu)/2} (1 + \nu)^{-(1-\nu)/2}$  with exponents  $2/(1 + \nu)$  and  $2/(1 - \nu)$  respectively) we conclude the next result (2.8) from (2.7).  $\square$

In the above proposition there is a recurrence relation concerning weights  $(1 - \gamma_k)$ . The following notation will be used in the sequel for analyzing the complexity of GUG:

$$(2.10) \quad \Gamma_k = \begin{cases} 1 & k = 1 \\ \Gamma_{k-1}(1 - \gamma_k) & k > 1. \end{cases}$$

We will use the following simple lemma for analyzing the sum of recurrent terms.

LEMMA 2.2. *Suppose that  $\{a_k\}, \{b_k\} \subset \mathbb{R}$  and  $\{\gamma_k\} \subset [0, 1]$  are sequences that satisfy  $\gamma_1 = 1$  and*

$$(2.11) \quad a_k \leq (1 - \gamma_k)a_{k-1} + \gamma_k b_k, \quad \forall k \geq 1.$$

Then we have

$$(2.12) \quad a_k \leq \Gamma_k \sum_{i=1}^k \frac{\gamma_i}{\Gamma_i} b_i, \quad \forall k \geq 1, \quad \text{where } \Gamma_k := \begin{cases} 1 & \text{when } k = 1 \\ \Gamma_{k-1}(1 - \gamma_k) & \text{when } k > 1. \end{cases}$$

*Proof.* Dividing both sides of (2.11) by  $\Gamma_k$  we obtain a series of inequalities with telescoping terms concerning sequence  $\{a_k/\Gamma_k\}$ . Summing up we obtain (2.12).  $\square$

We are now ready to derive results on the complexity of CG as a special case of GUG. Theorem 2.3 below is a known complexity result of CG for problems with Hölder continuous gradients [19, 8].

THEOREM 2.3 (see also [19, 8]). *Suppose that we apply GUG in Algorithm 2.1 (with Algorithm 2.2 to solve Approx-Subproblem) with parameters  $\beta_k \equiv 0$ ,  $\eta_k \equiv 0$ ,  $\gamma_k = 2/(k+1)$  and  $\alpha^1 = 1$  in Algorithm 2.2. To compute an  $\varepsilon$ -solution to problem (1.1) with Hölder exponent  $\nu$  and constant  $M_\nu$ , GUG requires at most  $N_{grad}$  gradient evaluations and  $N_{lin}$  linear objective optimizations, in which*

$$(2.13) \quad N_{lin} = N_{grad} = \mathcal{O} \left( \left( \frac{M_\nu D_X^{1+\nu}}{\varepsilon} \right)^{\frac{1}{\nu}} \right).$$

*Proof.* Since  $\gamma_k = 2/(k+1)$ , by (2.10) we have  $\Gamma_k = 2/(k(k+1))$  and hence  $\gamma_k/\Gamma_k = k$ . Applying Proposition 2.1 and noting Lemma 2.2 with our parameter settings, we have for any  $x \in X$  that

$$f(y_N) - f(x) \leq \frac{2M_\nu}{N(N+1)(1+\nu)} \sum_{k=1}^N k \left( \frac{2}{k+1} \right)^\nu \|x_k - x_{k-1}\|^{1+\nu} \leq \mathcal{O} \left( \frac{M_\nu D_X^{1+\nu}}{N^\nu} \right).$$

Thus, in order to obtain an  $\varepsilon$ -solution, we require at most  $N_{grad}$  outer iterations. Moreover, noting that  $\alpha^1 = 1$  and  $\beta = 0$  in the CGM procedure, comparing (2.6) and (2.5) we observe that CGM will always terminate after one inner iteration. Therefore, the total number of gradient evaluations and linear optimizations must both be upper bounded by (2.13).  $\square$

As pointed in the remarks after the description of Algorithm 2.1, CG is a special case of GUG with  $\beta_k \equiv 0$ . Therefore, Theorem 2.3 above provides a complexity result for the CG algorithm applied to functions with Hölder continuity exponent  $\nu \in (0, 1]$ . One achieves similar results to Theorem 2.3 when choosing different  $\gamma_k$  (e.g.,  $\gamma_k = 1/k$ ; see, e.g., [19] for other choices of  $\gamma_k$ ). It should also be noted that the choice of  $\eta_k \equiv 0$  does not affect the above analysis; indeed, with  $\beta = 0$  and any  $\eta \geq 0$ ,

the CGM procedure will always terminate after one inner iteration. However, as we describe below, if  $\beta > 0$ , the choice of  $\eta$  will affect the number of inner iterations performed by the CGM procedure before termination. The proposition below is a known complexity result (see Theorem 2.2(c) in [15]) of CG for solving projection problems. For completeness, we will prove it later in the next section as an immediate consequence of Proposition 3.2.

PROPOSITION 2.4. *In the CGM procedure for computing an approximate solution to the projection problem (2.4), if we choose  $\alpha^t = 2/(t+1)$ , then*

$$\min_{j=0,\dots,t} \max_{x \in X} \langle \nabla \phi(u^j), u^j - x \rangle \leq \frac{6\beta D_X^2}{t}, \quad \forall t \geq 1.$$

Proposition 2.4 provides insight on the number of inner iterations required by the CGM procedure in Algorithm 2.2 to solve the projection problem 2.4 approximately. For example, if we set  $\eta_k \geq 6\beta_k D_X^2$ , then the CGM procedure always terminates after exactly one iteration. Noting that  $u^0 = u$  in CGM, we can observe that GUG reduces to CG not only when  $\beta_k \equiv 0$  (as stated previously in the remarks of GUG and after Theorem 2.3), but also when  $\beta_k > 0$  and  $\eta_k \geq 6\beta_k D_X^2$ . The latter observation is important for our analysis: as described in the following theorem, for problems with Hölder continuous exponent  $\nu \in (0, 1)$ , the latter observation will allow us to perform a simple analysis of CG that is different from the current literature [19, 8]. Such simple analysis leads to our interesting discovery that sliding could improve the complexity of linear objective optimizations.

THEOREM 2.5 (see also [19, 8]). *Assume in problem (1.1) that the Hölder exponent  $\nu \in (0, 1)$ . Suppose that we apply GUG in Algorithm 2.1 (with Algorithm 2.2 to solve Approx-Subproblem) with parameters  $\beta_k > 0$ ,  $\eta_k = 6\beta_k D_X^2$ , and  $\alpha^1 = 1$  in Algorithm 2.2. Then we have for any  $x \in X$  that*

$$(2.14) \quad \begin{aligned} & f(y_N) - f(x) \\ & \leq \Gamma_N \sum_{k=1}^N \frac{\beta_k \gamma_k}{2\Gamma_k} (12D_X^2 + \|x_{k-1} - x\|^2 - \|x_k - x\|^2) + \frac{1}{\Gamma_k} \frac{1-\nu}{2(1+\nu)} M_\nu^{\frac{2}{1-\nu}} \left( \frac{\gamma_k}{\beta_k} \right)^{\frac{1+\nu}{1-\nu}}. \end{aligned}$$

*Specially, if we set  $\beta_k = M_\nu \gamma_k^\nu / D_X^{1-\nu}$  and  $\gamma_k = 2/(k+1)$ , to compute an  $\varepsilon$ -solution to problem (1.1) with Hölder exponent  $\nu$  and constant  $M_\nu$ , GUG requires at most  $N_{grad}$  gradient evaluations and  $N_{lin}$  linear objective optimizations, in which  $N_{lin} = N_{grad} = \mathcal{O}\left((M_\nu D_X^{1+\nu} / \varepsilon)^{\frac{1}{\nu}}\right)$ .*

*Proof.* Applying Proposition 2.1 and noting Lemma 2.2, with our choice of  $\eta_k$  we have (2.14). Consequently,

$$(2.15) \quad f(y_N) - f(x) \leq \Gamma_N \sum_{k=1}^N \frac{7\beta_k \gamma_k D_X^2}{\Gamma_k} + \frac{1}{\Gamma_k} \frac{1-\nu}{2(1+\nu)} M_\nu^{\frac{2}{1-\nu}} \left( \frac{\gamma_k}{\beta_k} \right)^{\frac{1+\nu}{1-\nu}}, \quad \forall x \in X.$$

Substituting to (2.15) the values of  $\beta_k$ ,  $\gamma_k$ , and noting that  $\Gamma_k = 1/k$  from (2.10), we have

$$f(y_N) - f(x) \leq \mathcal{O}\left(\frac{M_\nu D_X^{1+\nu}}{N^2}\right) \sum_{k=1}^N \frac{k}{(k+1)^\nu} \leq \mathcal{O}\left(\frac{M_\nu D_X^{1+\nu}}{N^\nu}\right).$$



Thus, in order to obtain an  $\varepsilon$ -solution, we require at most  $N_{grad}$  outer iterations. Moreover, noting that  $\alpha^1 = 1$  and  $\eta = 6\beta D_X^2$  in the CGM procedure, by Proposition 2.4 we observe that the CGM procedure will always terminate after one inner iteration. Therefore the total number of linear optimizations is upper bounded by  $N_{lin} = N_{grad}$ .  $\square$

In the above theorem, we observe an imperfection by in the derivation from (2.14) and (2.15), although we obtain the same complexity result of CG as in Theorem 2.3. Specifically, due to the existence of the dominant term  $D_X^2$ , we can only simply bound the telescoping difference ( $\|x_{k-1} - x\|^2 - \|x_k - x\|^2$ ) by  $D_X^2$  in to obtain (2.15). As a consequence, even if we attempt to choose the best  $\beta_k = \mathcal{O}(M_\nu \gamma_k^\nu / D_X^{1-\nu})$  to minimize the right hand side of (2.15), the complexity result remains to be  $\mathcal{O}\left(\left(M_\nu D_X^{1+\nu} / \varepsilon\right)^{\frac{1}{\nu}}\right)$ . Noting that the imperfection we observe is due to the choice that  $\eta_k = 6\beta_k D_X^2$ , we may choose a smaller  $\eta_k$  setting to improve the complexity results, as stated in the proposition below.

**THEOREM 2.6.** *Assume in problem (1.1) that the Hölder exponent  $\nu \in (0, 1)$ . Suppose that we apply GUG in Algorithm 2.1 (with Algorithm 2.2 to solve Approx-Subproblem) with parameters*

$$\beta_k = \frac{M_\nu k^{\frac{1-3\nu}{2}}}{D_X^{1-\nu}}, \eta_k = \frac{6\beta_k D_X^2}{k}, \text{ and } \gamma_k = \frac{2}{k+1}.$$

*To compute an  $\varepsilon$ -solution to problem (1.1) with Hölder exponent  $\nu$  and constant  $M_\nu$ , GUG requires at most  $N_{grad}$  gradient evaluations and  $N_{lin}$  linear objective optimizations, in which*

$$N_{grad} = \mathcal{O}\left(\left(\frac{M_\nu D_X^{1+\nu}}{\varepsilon}\right)^{\frac{2}{1+3\nu}}\right) \text{ and } N_{lin} = \mathcal{O}\left(\left(\frac{M_\nu D_X^{1+\nu}}{\varepsilon}\right)^{\frac{4}{1+3\nu}}\right).$$

for any  $\nu \in [0, 1)$ .

*Proof.* Since  $\gamma_k = 2/(k+1)$ , we have  $\Gamma_k = 2/(k(k+1))$  and hence  $\gamma_k/\Gamma_k = k$ . Applying Proposition 2.1 and noting Lemma 2.2, with our choice of parameters

$$\begin{aligned} & f(y_N) - f(x) \\ & \leq \frac{2}{N(N+1)} \sum_{k=1}^N 6\beta_k D_X^2 + \frac{k\beta_k}{2} \left( \|x_{k-1} - x\|^2 - \|x_k - x\|^2 \right) + \frac{\xi_k}{\Gamma_k}, \quad \forall x \in X. \end{aligned}$$

Noting that  $k\beta_k$  is increasing, we have

$$\begin{aligned} & \sum_{k=1}^N k\beta_k \left( \|x_{k-1} - x\|^2 - \|x_k - x\|^2 \right) \\ & = \beta_1 \|x_0 - x\|^2 + \sum_{k=1}^N ((k+1)\beta_{k+1} - k\beta_k) \|x_k - x\|^2 - N\beta_N \|x_N - x\|^2 \\ & \leq \beta_1 D_X^2 + \sum_{k=1}^N ((k+1)\beta_{k+1} - k\beta_k) D_X^2 = N\beta_N D_X^2. \end{aligned}$$

Combining the above two relations and noting our choice of  $\beta_k$  and the description of

$\xi_k$  in (2.9) we have

$$(2.16) \quad f(y_N) - f(x) \leq \mathcal{O} \left( \frac{M_\nu D_X^{1+\nu}}{N^2} \right) \left[ \sum_{k=1}^N k^{\frac{1-3\nu}{2}} + N^{\frac{3-3\nu}{2}} + \sum_{k=1}^N \frac{k^{\frac{1+3\nu^2}{2(1-\nu)}}}{(k+1)^{\frac{2\nu}{1-\nu}}} \right] \\ \leq \mathcal{O} \left( \frac{M_\nu D_X^{1+\nu}}{N^{\frac{1+3\nu}{2}}} \right), \quad \forall x \in X.$$

Thus, to obtain an  $\varepsilon$ -solution, we need at most  $N_{grad}$  outer iterations, or equivalently, at most  $N_{grad}$  gradient evaluations. Also, from Proposition 2.4, if  $\eta_k = 6\beta_k D_X^2/k$ , then we will perform at most  $k$  inner iterations per outer iteration. Thus, the total number of inner iterations and consequently linear objective optimizations is upper bounded by

$$\sum_{k=1}^{N_{grad}} k \leq \mathcal{O}(N_{grad}^2) = \mathcal{O} \left( \left( \frac{M_\nu D_X^{1+\nu}}{\varepsilon} \right)^{\frac{4}{1+3\nu}} \right).$$

The proof is now complete.  $\square$

Note that by the choice of  $\eta_k$  and Proposition 2.4, Theorem 2.6 provides a complexity result for a version of GUG with the sliding feature for solving problem (1.1). Comparing Theorems 2.5 and 2.6, the key difference in the proofs is the additional  $1/k$  factor in  $\eta_k$  in Theorem 2.6. With the additional factor, the three terms at the right hand side of (2.16) are of the same order with respect to  $N$ , resolving the imperfection we noticed previously in (2.15) in the proof of Theorem 2.5. In doing so, we achieve the optimal lower complexity bound of gradient evaluations (1.4) for first-order methods. Interestingly, we can discover that number of linear optimizations required in Theorem 2.6 is also significantly reduced comparing with that in Theorem 2.5, since  $4/(1+3\nu) < 1/\nu$  for all  $\nu \in (0, 1)$ .

It should be noted that we exclude the case  $\nu = 1$  case in Theorem 2.6 only for convenience of our analysis, since our focus in this section is mainly the theoretical analysis on improving the state-of-the-art complexity bounds [19, 8] when  $\nu \in (0, 1)$ . By slightly modifying the proof of Theorem 2.6 we can also achieve the same complexity results as the state-of-the-art in [15]. We will include the  $\nu = 1$  case in the convergence analysis of practical implementation in the next section.

We conclude this section with several comments regarding the implementation of GUG in Algorithm 2.1. Note that the sliding result shown in Theorem 2.6 requires a parameter choice  $\beta_k$  that assumes the knowledge of Hölder exponent  $\nu \in (0, 1)$  and constant  $M_\nu$ . Unfortunately, the knowledge of the best  $\nu$  and  $M_\nu$  for the performance of GUG may not be easily accessible in practice. Furthermore, the proposed Algorithm 2.1 has no termination criterion for verifying whether the current approximate solution  $y_k$  is an  $\varepsilon$ -solution. Lastly, there may exist problem instances in which a solution  $v^t$  to the linear subproblem (2.6) cannot be computed exactly and instead we can only compute an approximate solution. In the next section, we propose an algorithm called universal conditional gradient sliding (UCGS) that utilizes a backtracking linesearch scheme with an implementable stopping criterion to achieve better practical performance than Algorithm 2.1. We will also analyze its convergence under an approximate solution to linear subproblem (2.6).

**3. Practical universal conditional gradient sliding method.** In this section, we propose a practical universal conditional gradient sliding (UCGS) method

---

**Algorithm 3.1** Universal conditional gradient sliding (UCGS) method
 

---

Start: Choose tolerance  $\varepsilon > 0$  and initial iteration  $x_0 \in X$ . Set  $y_0 = x_0$ .

**for**  $k = 1, 2, \dots$ , **do**

Decide  $L_k > 0$  such that

$$(3.1) \quad f(y_k) \leq f(z_k) + \langle \nabla f(z_k), y_k - z_k \rangle + \frac{L_k}{2} \|y_k - z_k\|^2 + \frac{\varepsilon}{2} \gamma_k$$

where

$$(3.2) \quad \gamma_k := \begin{cases} 1, & k = 1 \\ \text{positive solution to } \Gamma_{k-1}(1 - \gamma_k) = \frac{L_k \gamma_k^2}{k}, & k > 1 \end{cases}$$

$$(3.3) \quad z_k := (1 - \gamma_k)y_{k-1} + \gamma_k x_{k-1}$$

$$(3.4) \quad x_k := \text{ACGM}(\nabla f(z_k), x_{k-1}, \beta_k, \eta_k)$$

$$(3.5) \quad y_k := (1 - \gamma_k)y_{k-1} + \gamma_k x_k$$

$$(3.6) \quad \Gamma_k := \frac{L_k \gamma_k^2}{k}.$$

Compute an approximate solution  $s_k$  to the problem

$$(3.7) \quad \min_{x \in X} \ell_k(x) := \Gamma_k \sum_{i=1}^k \frac{\gamma_i}{\Gamma_i} (f(z_i) + \langle \nabla f(z_i), x - z_i \rangle)$$

such that  $\ell_k(s_k) - \min_{x \in X} \ell_k(x) \leq \varepsilon_k$ . Terminate and output  $y_k$  as an approximate solution if

$$(3.8) \quad f(y_k) - \ell_k(s_k) + \varepsilon_k \leq \varepsilon.$$

**end for**

**procedure**  $u^+ = \text{ACGM}(g, u, \beta, \eta)$

Goal: Compute  $u^+$  such that  $\max_{x \in X} \langle \nabla \phi(u^+), u^+ - x \rangle \leq \eta$ , where

$$(3.9) \quad \phi(x) := \langle g, x \rangle + \frac{\beta}{2} \|x - u\|^2.$$

Start: Set  $u^0 = u$ .

**for**  $t = 1, 2, \dots$ , **do**

Compute a  $\delta^t$ -approximate solution  $v^t$  to the problem  $\min_{x \in X} \langle \nabla \phi(u^{t-1}), x \rangle$  such that

$$(3.10) \quad \langle g + \beta(u^{t-1} - u), v^t \rangle - \min_{x \in X} \langle g + \beta(u^{t-1} - u), x \rangle \leq \delta^t.$$

Terminate with  $u^+ := u^{t-1}$  if

$$(3.11) \quad \langle g + \beta(u^{t-1} - u), u^{t-1} - v^t \rangle + \delta^t \leq \eta.$$

Otherwise, compute  $u^t = (1 - \alpha^t)u^{t-1} + \alpha^t v^t$ .

**end for**

**end procedure**

---

that addresses the above issues of Algorithm 2.1. The proposed UCGS algorithm is described in Algorithm 3.1.

Let us make a few remarks regarding Algorithm 3.1. First, the approximate conditional gradient method (ACGM) procedure in Algorithm 3.1 is a generalization of the CGM procedure (Algorithm 2.2) discussed in the previous section. Specifically, whenever  $\delta^t \equiv 0$ , ACGM and CGM are equivalent. Note also that the parameter  $\alpha^t$  can be computed through an exact linesearch, namely,

$$(3.12) \quad \alpha^t := \min \left\{ 1, \frac{\langle g - \beta(u - u^{t-1}), u^{t-1} - v^t \rangle}{\beta \|v^t - u^{t-1}\|^2} \right\}.$$

It is easy to observe that the above  $\alpha^t$  is the optimal solution to the exact linesearch problem  $\min_{\alpha \in [0,1]} \phi((1-\alpha)u^{t-1} + \alpha v^t)$ . Second, if the objective function  $f(x)$  in problem (1.1) has Lipschitz continuous gradient (so  $\nu = 1$ ) with Lipschitz constant  $M_1$ , then UCGS can be understood as extension of CGS with added features for practical implementation. The new features include a backtracking linesearch strategy that computes adaptive estimates  $L_k$  for the Lipschitz constant  $M_1$ , the possibility of computing only approximate solutions to linear subproblems, and a termination criterion for verifying whether an approximate solution to problem (1.1) has been computed. Third, the choice of  $\gamma_k$  and  $\Gamma_k$  in (3.6) and (3.2) implies that

$$(3.13) \quad \Gamma_k = \begin{cases} 1, & k = 1 \\ \Gamma_{k-1}(1 - \gamma_k), & k > 1 \end{cases}.$$

Furthermore, it can be shown that for  $k > 1$ , the solution to (3.2) is given by

$$\gamma_k = \frac{2\sqrt{k\Gamma_{k-1}}}{\sqrt{4L_k + k\Gamma_{k-1}} + \sqrt{k\Gamma_{k-1}}}.$$

Observe that  $\gamma_k \in (0, 1)$ . Consequently, the recursively described approximate solution  $y_k$  is the convex combination of  $x_1, \dots, x_k$ . Also the point  $z_k$  for gradient evaluation is a convex combination of  $x_1, \dots, x_{k-1}$ . Such recursive description first appeared in Nesterov's seminal accelerated gradient algorithm (see, e.g., [21]) and is also used in the CGS algorithm [15] and the universal gradient algorithms studied in [22]. However, our choice of  $\gamma_k$  is novel and is different from the ones in [22, 15, 21]. In fact, to our knowledge, none of the settings of  $\gamma_k$  in [22, 21, 15] are suitable for CGS-type algorithms with adaptive  $L_k$ . In the only previous work [17] that successfully developed a linesearch scheme for CGS,  $\gamma_k$  needs to satisfy a more sophisticated cubic equation and  $L_k$  needs to be monotone increasing. As we will describe below, such monotonicity restriction on  $L_k$  is removed in our proposed UCGS method.

A few remarks on the practical implementation of Algorithm 3.1 are also in place. First, Algorithm 3.1 proposes that we find  $L_k > 0$  such that (3.1) is satisfied. The condition (3.1) originated from the framework of inexact oracle in [5] and is also used in [22]. We proposed to search for such  $L_k$  through a backtracking linesearch strategy. In particular, we initialize with any  $L_0 \in \mathbb{R}$  and choose  $L_1 = 2^i L_0$  where  $i$  is the smallest integer such that (3.1) is satisfied. At the start of the  $k$ -th outer iteration where  $k > 1$ , we set  $L_k = L_{k-1}/2$  and assess the validity of  $L_k$ . If it does not satisfy (3.1), we keep backtracking and replacing  $L_k$  to  $2L_k$  until (3.1) is satisfied. Through this backtracking linesearch strategy, we ensure that our choice of  $L_k$  is adaptive and that performance is independent of the choice of  $L_0$ . Previous literature [17] on backtracking linesearch strategy of CGS require monotonicity of  $L_k$  and may suffer from a poorly chosen  $L_0$ . Second, our termination criterion is based on (3.8). We can observe immediately that if the parameter  $\varepsilon_k \equiv 0$ , i.e.,  $s_k$  is the exact solution to problem (3.7), then when (3.8) is satisfied,  $y_k$  will be  $\varepsilon$ -approximation solution to problem (1.1). To see this, note from (3.13) that

$$(3.14) \quad \Gamma_k \sum_{i=1}^k \frac{\gamma_i}{\Gamma_i} = 1$$

and consequently

$$f(y_k) - f^* \leq f(y_k) - \min_{x \in X} \ell_k(x) = f(y_k) - \ell_k(s_k).$$

Such termination criterion also appeared in the previous literature (see, e.g., [22, 17]). For the case when  $\varepsilon_k > 0$ , we will show later in Theorem 3.7 that allowing approximate solution  $s_k$  with properly chosen accuracy  $\varepsilon_k$  will not affect the complexity results of UCGS.

We present convergence analysis for the UCGS algorithm proposed above, beginning with some results on the inner iteration complexity. The following lemma resembles a combination of the proofs of Theorem 2.2(c) in [15] and Theorem 5.2 in [7] on the analysis of conditional gradient method with approximate linear objective optimization subproblems for solving projection problems.

LEMMA 3.1. *Suppose that  $\{\lambda^t\} \in [0, 1]$  is any predetermined sequence satisfying  $\lambda^1 = 1$ . In the ACGM procedure, if  $\alpha^t$  is chosen such that*

$$(3.15) \quad \phi(u^t) \leq \phi((1 - \lambda^t)u^{t-1} + \lambda^t v^t), \quad \forall t \geq 1,$$

then we have

$$\begin{aligned} \sum_{j=2}^t \frac{\lambda^j}{\Lambda^j} \max_{x \in X} \langle \nabla \phi(u^{j-1}), u^{j-1} - x \rangle &\leq \left[ \delta^1 + \sum_{j=2}^t \frac{\lambda^j}{\Lambda^j} \left( \delta^j + \Lambda^{j-1} \sum_{i=1}^{j-1} \frac{\lambda^i}{\Lambda^i} \delta^i \right) \right] \\ &\quad + \frac{\beta D_X^2}{2} \left[ 1 + \sum_{j=2}^t \frac{\lambda^j}{\Lambda^j} \left( \lambda^j + \Lambda^{j-1} \sum_{i=1}^{j-1} \frac{(\lambda^i)^2}{\Lambda^i} \right) \right] \end{aligned}$$

for all  $t \geq 2$ , where

$$(3.16) \quad \Lambda^t := \begin{cases} 1 & \text{when } t = 1 \\ \Lambda^{t-1}(1 - \lambda^t) & \text{when } t > 1. \end{cases}$$

*Proof.* Observing that the function  $\phi(x)$  in (3.9) is a strongly convex function with Lipschitz continuous (with constant  $\beta$ ) gradient, using the assumption (3.15), and noting the definition of approximate solution  $v^t$  in (3.10), we have

$$\begin{aligned} &\phi(u^t) - (1 - \lambda^t)\phi(u^{t-1}) - \lambda^t(\phi(u^{t-1}) + \langle \nabla \phi(u^{t-1}), x - u^{t-1} \rangle) \\ &\leq \phi((1 - \lambda^t)u^{t-1} + \lambda^t v^t) - \phi(u^{t-1}) - \lambda^t \langle \nabla \phi(u^{t-1}), x - u^{t-1} \rangle \\ (3.17) \quad &\leq \lambda^t \langle \nabla \phi(u^{t-1}), v^t - u^{t-1} \rangle + \frac{\beta(\lambda^t)^2}{2} \|v^t - u^{t-1}\|^2 - \lambda^t \langle \nabla \phi(u^{t-1}), x - u^{t-1} \rangle \\ &= \lambda^t \langle \nabla \phi(u^{t-1}), v^t - x \rangle + \frac{\beta(\lambda^t)^2}{2} \|v^t - u^{t-1}\|^2 \\ &\leq \lambda^t \delta^t + \frac{\beta D_X^2 (\lambda^t)^2}{2}, \quad \forall x \in X, t \geq 1. \end{aligned}$$

Defining  $x^* := \operatorname{argmin}_{x \in X} \phi(x)$ , from the above relation we have for any  $t \geq 2$  that

$$\begin{aligned}
& \sum_{j=2}^t \frac{\lambda^j}{\Lambda^j} \max_{x \in X} \langle \nabla \phi(u^{j-1}), u^{j-1} - x \rangle \\
& \leq \sum_{j=2}^t \frac{1}{\Lambda^j} [\phi(u^{j-1}) - \phi(x^*)] - \frac{1}{\Lambda^j} [\phi(u^j) - \phi(x^*)] + \frac{\lambda^j}{\Lambda^j} \delta^j + \frac{\beta D_X^2 (\lambda^j)^2}{2\Lambda^j} \\
(3.18) \quad & = [\phi(u^1) - \phi(x^*)] - \frac{1}{\Lambda^t} [\phi(u^t) - \phi(x^*)] \\
& \quad + \sum_{j=2}^t \frac{\lambda^j}{\Lambda^j} [\phi(u^{j-1}) - \phi(x^*)] + \frac{\lambda^j}{\Lambda^j} \delta^j + \frac{\beta D_X^2 (\lambda^j)^2}{2\Lambda^j} \\
& \leq [\phi(u^1) - \phi(x^*)] + \sum_{j=2}^t \frac{\lambda^j}{\Lambda^j} [\phi(u^{j-1}) - \phi(x^*)] + \frac{\lambda^j}{\Lambda^j} \delta^j + \frac{\beta D_X^2 (\lambda^j)^2}{2\Lambda^j}.
\end{aligned}$$

Here in the equality we use the following observations from the definition of  $\Lambda^t$  in (3.16):  $1/\Lambda^1 = 1$  and  $1/\Lambda^j = 1/\Lambda^{j-1} + \lambda^j/\Lambda^j$  for all  $j \geq 2$ ,

To finish the proof it suffices to bound  $\phi(u^{j-1}) - \phi(x^*)$  for any  $j \geq 2$ . Observing that  $\phi(x)$  in (3.9) is strongly convex and quadratic with

$$\frac{\beta}{2} \|x - u^{t-1}\|^2 = \phi(x) - (\phi(u^{t-1}) + \langle \nabla \phi(u^{t-1}), x - u^{t-1} \rangle), \quad \forall x \in X, t \geq 1,$$

we have from (3.17) (with  $x = x^*$ ) that

$$[\phi(u^t) - \phi(x^*)] - (1 - \lambda^t)[\phi(u^{t-1}) - \phi(x^*)] \leq \lambda^t \delta^t + \frac{\beta D_X^2 (\lambda^t)^2}{2} - \frac{\beta \lambda^t}{2} \|x^* - u^{t-1}\|^2.$$

Applying Lemma 2.2 to the above recurrence relation and ignoring negative terms at the right hand side, we have

$$\phi(u^t) - \phi(x^*) \leq \Lambda^t \sum_{i=1}^t \frac{\lambda^i}{\Lambda^i} \delta^i + \frac{\beta D_X^2 (\lambda^i)^2}{2\Lambda^i}, \quad \forall t \geq 1.$$

We conclude the lemma immediately by applying the above bound to (3.18) and rearranging terms.  $\square$

The complexity result of the above lemma depends on a predetermined sequence  $\{\lambda^t\}$ . In the proposition below, we provide a complexity result from an example choice of  $\{\lambda^t\}$ .

PROPOSITION 3.2. *In the ACGM procedure, at termination we have*

$$(3.19) \quad \max_{x \in X} \langle \nabla \phi(u^+), u^+ - x \rangle \leq \eta.$$

Moreover, if  $\delta^t = \sigma \beta D_X^2 / t$  for certain  $\sigma \geq 0$  and  $\alpha^t$  is chosen such that

$$(3.20) \quad \phi(u^t) \leq \phi\left(\frac{t-1}{t+1} u^{t-1} + \frac{2}{t+1} v^t\right),$$

then we have for any  $t \geq 1$  that

$$\begin{aligned}
(3.21) \quad \min_{j=1, \dots, t+1} \langle \nabla \phi(u^{j-1}), u^{j-1} - v^j \rangle & \leq \min_{j=1, \dots, t+1} \max_{x \in X} \langle \nabla \phi(u^{j-1}), u^{j-1} - x \rangle \\
& \leq \frac{6(\sigma+1)\beta D_X^2}{t}.
\end{aligned}$$

Specially, it takes at most

$$(3.22) \quad T := 1 + \left\lceil \frac{(\sigma + 6)\beta D_X^2}{\eta} \right\rceil$$

iterations for the ACGM procedure to terminate.

*Proof.* From the definition of the approximate solution  $v^t$  in (3.10), if the termination criterion in (3.11) of the ACGM procedure is satisfied, then the output  $u^+ = u^{t-1}$  satisfies

$$\begin{aligned} \max_{x \in X} \langle \nabla \phi(u^{t-1}), u^{t-1} - x \rangle &= \max_{x \in X} \langle \nabla \phi(u^{t-1}), u^{t-1} - v^t \rangle + \langle \nabla \phi(u^{t-1}), v^t - x \rangle \\ &\leq (\eta - \delta^t) + \delta^t = \eta. \end{aligned}$$

Therefore (3.19) holds. To conclude the proposition it suffices to estimate the rate of convergence of  $\max_{x \in X} \langle \nabla \phi(u^{t-1}), u^{t-1} - x \rangle$ . To analyze the rate, let us choose  $\lambda^t = 2/(t+1)$  and apply Lemma 3.1. Then  $\Lambda^t = 2/(t(t+1))$  and

$$\begin{aligned} &\sum_{j=2}^t j \cdot \max_{x \in X} \langle \phi(u^{j-1}), u^{j-1} - x \rangle \\ &\leq \sigma \beta D_X^2 \left[ 1 + \sum_{j=2}^t \left( 1 + \frac{2}{j-1} \sum_{i=1}^{j-1} 1 \right) \right] + \frac{\beta D_X^2}{2} \left[ 1 + \sum_{j=2}^t \left( \frac{2j}{j+1} + \frac{2}{j-1} \sum_{i=1}^{j-1} \frac{2i}{i+1} \right) \right] \\ &< \sigma \beta D_X^2 (3t-2) + \frac{\beta D_X^2}{2} (6t-5), \quad \forall t \geq 2. \end{aligned}$$

Noting that  $\sum_{j=2}^t j = (t+2)(t-1)/2$ , we have

$$\begin{aligned} \min_{j=2, \dots, t} \max_{x \in X} \langle \nabla \phi(u^{j-1}), u^{j-1} - x \rangle &\leq \frac{2}{(t+2)(t-1)} \sum_{j=2}^t j \cdot \max_{x \in X} \langle \phi(u^{j-1}), u^{j-1} - x \rangle \\ &< \frac{6(\sigma+1)\beta D_X^2}{t-1}, \quad \forall t \geq 2. \end{aligned}$$

Using the above result and observing that

$$\begin{aligned} \min_{j=1, \dots, t+1} \langle \nabla \phi(u^{j-1}), u^{j-1} - v^j \rangle &\leq \min_{j=1, \dots, t+1} \max_{x \in X} \langle \nabla \phi(u^{j-1}), u^{j-1} - x \rangle \\ &\leq \min_{j=2, \dots, t+1} \max_{x \in X} \langle \nabla \phi(u^{j-1}), u^{j-1} - x \rangle, \quad \forall t \geq 1 \end{aligned}$$

we conclude (3.21). Moreover, from (3.21) and noting the choice of  $\delta^t$ , the termination criterion (3.11) holds whenever

$$\frac{6(\sigma+1)\beta D_X^2}{t-1} + \frac{\sigma \beta D_X^2}{t} \leq \eta.$$

Noting the definition of  $T$  (3.22), the above condition clearly holds for all  $t \geq T$ .  $\square$

In the above proposition,  $\sigma \geq 0$  in the definition of  $\delta^t$  is a parameter related to the accuracy of approximately solving linear objective optimization subproblems.

Note that there may also exist other possible choice of  $\delta^t$ . For example, similar complexity result can be derived by choosing  $\delta^t = \sigma\eta$ . The benefit of our proposed choice  $\delta^t = \sigma\beta D_X^2/t$  from the perspective of practical implementation is that it allows adaptive error of the approximate solution  $v^t$  to the linear subproblems and larger error can be admissible when  $t$  is small.

As a side note, recalling that ACGM procedure reduces to CGM procedure in Algorithm 2.2, we can observe that Proposition 2.4 in the previous section is a direct consequence of the above result:

*Proof of Proposition 2.4.* Noting that the CGM procedure described in Algorithm 2.2 is equivalent to the ACGM procedure with  $\delta^t \equiv 0$ , applying Proposition 3.2 above with  $\alpha^t = 2/(t+1)$ , we conclude the proposition immediately from (3.21).  $\square$

From the above two proofs, it is clear that Proposition 3.2 is different from Proposition 2.4 in the previous section, since it shows us that we can instead compute an approximate solution to (3.10) and proceed with the convergence analysis. We will eventually utilize Proposition 3.2 to establish an upper bound on the number of inner iterations that Algorithm 3.1 requires to compute an  $\varepsilon$ -solution. We now continue onto the outer iteration analysis, starting with a few results that establish the relation between our computed  $L_k$  in the linesearch scheme and the underlying Hölder exponent  $\nu$  and constant  $M_\nu$  in (1.3). We will use the following lemma that appeared in [22].

LEMMA 3.3. *For any  $\delta > 0$  and any  $L$  such that*

$$L \geq \left( \frac{1-\nu}{1+\nu} \cdot \frac{1}{\tau} \right)^{\frac{1-\nu}{1+\nu}} M_\nu^{\frac{2}{1+\nu}},$$

where  $\nu$  and  $M_\nu$  are the Hölder continuity exponent and constant in (1.3), we have

$$(3.23) \quad f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2} \|y - x\|^2 + \frac{\tau}{2}, \quad \forall x, y \in X.$$

*Proof.* See Lemma 1 of [22].  $\square$

Note that for  $\nu = 1$ , the term  $\left( \frac{1-\nu}{1+\nu} \right)^{\frac{1-\nu}{1+\nu}}$  can be handled using a continuity argument  $\lim_{\nu \rightarrow 1} \left( \frac{1-\nu}{1+\nu} \right)^{\frac{1-\nu}{1+\nu}} = 1$ . We state an immediate corollary of the above lemma below.

COROLLARY 3.4. *Any  $L_k > 0$  chosen by Algorithm 3.1 according to (3.1) must also satisfy*

$$L_k \leq 2 \left( \frac{1-\nu}{1+\nu} \cdot \frac{1}{\varepsilon\gamma_k} \right)^{\frac{1-\nu}{1+\nu}} M_\nu^{\frac{2}{1+\nu}}$$

*Proof.* Suppose that  $L_k$  does not satisfy (3.23). Then applying Proposition 3.6 with  $\tau = \varepsilon\gamma_k$  implies that  $L_k/2$  satisfies (3.1), contradicting the fact that  $L_k$  was chosen at step  $k$  following the proposed backtracking linesearch implementation (see the remark on practical implementation after the description of Algorithm 3.1).  $\square$

The above result is an immediate consequence of the backtracking linesearch strategy we use to find a suitable  $L_k$  that satisfies (3.1). Based on the above result, we can estimate a bound of  $L_k\gamma_k^2$  in the proposition below. Recalling that  $\Gamma_k = L_k\gamma_k^2/k$  from (3.6) in Algorithm 3.1, the following lemma provides also a bound of  $\Gamma_k$  that is important for the outer iteration complexity analysis.



LEMMA 3.5. *Let  $L_k > 0$  be chosen by (3.1) of Algorithm 3.1 at step  $k$ , then*

$$L_k \gamma_k^2 \leq \frac{C_\nu M_\nu^{\frac{2}{1+\nu}}}{k^{\frac{1+3\nu}{1+\nu}} \varepsilon^{\frac{1-\nu}{1+\nu}}}$$

where

$$(3.24) \quad C_\nu := \left( \frac{1+2\nu}{1+3\nu} \right)^{\frac{1+3\nu}{1+\nu}} \left( \frac{1-\nu}{1+\nu} \right)^{\frac{1-\nu}{1+\nu}} 2^{\frac{4+10\nu}{1+\nu}}$$

is a constant depending only on  $\nu$ .

*Proof.* The case when  $k = 1$  is immediate from Corollary 3.4. Therefore, throughout the proof we will assume that  $k \geq 2$ . Since we set  $\Gamma_k = L_k \gamma_k^2 / k$  in Algorithm 3.1, we can prove this proposition by bounding  $\Gamma_k$ . Set  $s := (1 + \nu)/(1 + 3\nu)$ . Since  $\nu \in [0, 1]$ , we have  $s \in [1/2, 1]$ . We will study the quantity  $1/\Gamma_k^s - 1/\Gamma_{k-1}^s$ , which can be rewritten as

$$\begin{aligned} \frac{1}{\Gamma_k^s} - \frac{1}{\Gamma_{k-1}^s} &= \frac{\left( \frac{1}{\Gamma_k^s} - \frac{1}{\Gamma_{k-1}^s} \right) \left( \frac{1}{\Gamma_k^{1-s}} + \frac{1}{\Gamma_{k-1}^{1-s}} \right)}{\frac{1}{\Gamma_k^{1-s}} + \frac{1}{\Gamma_{k-1}^{1-s}}} \\ &= \frac{\frac{1}{\Gamma_k} - \frac{1}{\Gamma_{k-1}} - \frac{1}{\Gamma_k} \left( \frac{\Gamma_k}{\Gamma_{k-1}} \right)^s + \frac{1}{\Gamma_{k-1}} \left( \frac{\Gamma_{k-1}}{\Gamma_k} \right)^s}{\frac{1}{\Gamma_k^{1-s}} + \frac{1}{\Gamma_{k-1}^{1-s}}}. \end{aligned}$$

Here, noting from the relation of  $\Gamma_k$  and  $\Gamma_{k-1}$  in (3.13) that  $\Gamma_k \leq \Gamma_{k-1}$  and recalling that  $s \in [1/2, 1]$ , we can make two observations. First, we have  $\Gamma_k^{2s-1} \leq \Gamma_{k-1}^{2s-1}$ , and hence

$$-\frac{1}{\Gamma_k} \left( \frac{\Gamma_k}{\Gamma_{k-1}} \right)^s + \frac{1}{\Gamma_{k-1}} \left( \frac{\Gamma_{k-1}}{\Gamma_k} \right)^s \geq 0.$$

Second, we have  $\Gamma_{k-1}^{1-s} \leq \Gamma_k^{1-s}$ , and hence

$$\frac{1}{\Gamma_k^{1-s}} + \frac{1}{\Gamma_{k-1}^{1-s}} \leq \frac{2}{\Gamma_{k-1}^{1-s}}.$$

Combining the above two observations and recalling that  $s = (1 + \nu)/(1 + 3\nu)$  and the relations concerning  $\Gamma_k$  and  $\Gamma_{k-1}$  in (3.13), we have that

$$\frac{1}{\Gamma_k^s} - \frac{1}{\Gamma_{k-1}^s} \geq \frac{\frac{1}{\Gamma_k} - \frac{1}{\Gamma_{k-1}}}{\frac{2}{\Gamma_k^{1-s}}} = \frac{\frac{\gamma_k}{\Gamma_k}}{\frac{2}{\Gamma_k^{1-s}}} = \frac{\gamma_k}{2} \Gamma_k^{-\frac{1+\nu}{1+3\nu}}.$$

We can further bound the last expression in the above relation. Indeed, recalling that  $\Gamma_k = L_k \gamma_k^2 / k$  and applying Corollary 3.4, we have the inequality

$$\frac{\gamma_k^2}{k \Gamma_k} = \frac{1}{L_k} \geq \frac{1}{2M_\nu^{\frac{1+\nu}{1+\nu}}} \left( \frac{1+\nu}{1-\nu} \cdot \varepsilon \gamma_k \right)^{\frac{1-\nu}{1+\nu}}.$$

In the above recall that we can use a continuity argument for the  $\nu = 1$  case since  $(1 - \nu)^{-(1-\nu)} \rightarrow 1$  as  $\nu \rightarrow 1$ . Rearranging terms in the above relation, we have

$$\gamma_k \Gamma_k^{-\frac{1+\nu}{1+3\nu}} \geq \left( \frac{1+\nu}{1-\nu} \right)^{\frac{1-\nu}{1+3\nu}} \frac{\varepsilon^{\frac{1-\nu}{1+3\nu}} k^{\frac{1+\nu}{1+3\nu}}}{2^{\frac{1+\nu}{1+3\nu}} M_\nu^{\frac{2}{1+3\nu}}}.$$

Applying the above bound to (11), it follows that

$$\frac{1}{\Gamma_k^s} - \frac{1}{\Gamma_{k-1}^s} \geq \left( \frac{1+\nu}{1-\nu} \right)^{\frac{1-\nu}{1+3\nu}} \frac{\varepsilon^{\frac{1-\nu}{1+3\nu}} k^{\frac{1+\nu}{1+3\nu}}}{2^{\frac{2+4\nu}{1+3\nu}} M_\nu^{\frac{2}{1+3\nu}}}.$$

Summing the above from  $i = 2$  to  $k$  and using the fact that

$$\sum_{i=2}^k i^{\frac{1+\nu}{1+3\nu}} \geq \int_1^k u^{\frac{1+\nu}{1+3\nu}} du = \frac{1+3\nu}{2+4\nu} \cdot \left( k^{\frac{2+4\nu}{1+3\nu}} - 1 \right) \geq \frac{1+3\nu}{4+8\nu} k^{\frac{2+4\nu}{1+3\nu}}, \quad \forall k \geq 2$$

and the definition of  $C_\nu$  in (3.24), we obtain

$$\frac{1}{\Gamma_k^s} \geq \frac{1}{\Gamma_k^s} - \frac{1}{\Gamma_1^s} \geq \left( \frac{1+\nu}{1-\nu} \right)^{\frac{1-\nu}{1+3\nu}} \frac{\varepsilon^{\frac{1-\nu}{1+3\nu}}}{2^{\frac{4+10\nu}{1+3\nu}} M_\nu^{\frac{2}{1+3\nu}}} \frac{1+3\nu}{1+2\nu} k^{\frac{2+4\nu}{1+3\nu}}$$

Recalling that  $s = (1+\nu)/(1+3\nu)$  and  $\Gamma_k = L_k \gamma_k^2/k$ , we conclude the proposition immediately from the above result.  $\square$

It should be noted that the technique utilized in Lemma 3.5 is similar to that of the proof surrounding equation (4.4) in [22]. However, note that the choice of parameter  $\gamma_k$  in UCGS is different from the one in [22]. Therefore, the proof in [22] needs to be adapted to the above proof. With the help of Lemma 3.5, we are now ready to prove our primary convergence properties on the proposed UCGS algorithm. We start with the following proposition that resembles the outer iteration analysis in Proposition 2.1 of the previous section.

**PROPOSITION 3.6.** *Suppose that the parameters in Algorithm 3.1 satisfy  $\beta_k \geq L_k \gamma_k$  for all  $k$ . Then for any  $x \in X$ ,*

$$f(y_k) - \ell_k(x) \leq \frac{\varepsilon}{2} + \Gamma_k \sum_{i=1}^k \frac{\gamma_i \beta_i}{2\Gamma_i} \left( \|x - x_{i-1}\|^2 - \|x - x_i\|^2 \right)^2 + \Gamma_k \sum_{i=1}^k \frac{\gamma_i \eta_i}{\Gamma_i}.$$

*Proof.* Fix any  $x \in X$ . From the definitions of  $\ell_k(x)$  and  $y_k$  in (3.1) and (3.5) respectively, we have

$$\begin{aligned} \frac{1}{\Gamma_k} \ell_k(x) &= \sum_{i=1}^k \frac{1}{\Gamma_i} (\gamma_i f(z_i) + \gamma_i \langle \nabla f(z_i), x - x_i \rangle + \langle \nabla f(z_i), \gamma_i (x_i - z_i) \rangle) \\ &= \sum_{i=1}^k \frac{1}{\Gamma_i} (f(z_i) + \langle \nabla f(z_i), y_i - z_i \rangle) + \frac{\gamma_i}{\Gamma_i} \langle \nabla f(z_i), x - x_i \rangle \\ &\quad - \frac{1-\gamma_i}{\Gamma_i} (f(z_i) + \langle \nabla f(z_i), y_{i-1} - z_i \rangle) \end{aligned}$$

We will now bound three terms in the above relation. First, by convexity of  $f$ ,

$$-(f(z_i) + \langle \nabla f(z_i), y_{i-1} - z_i \rangle) \geq -f(y_{i-1}).$$

Second, by our choice of  $L_k$  in (3.1) and the definitions of  $y_k$  and  $z_k$  in (3.5) and (3.3) respectively, we have

$$\begin{aligned} f(z_i) + \langle \nabla f(z_i), y_i - z_i \rangle &\geq f(y_i) - \frac{L_i}{2} \|y_i - z_i\|^2 - \frac{\varepsilon}{2} \gamma_i \\ &= f(y_i) - \frac{L_i \gamma_i^2}{2} \|x_i - x_{i-1}\|^2 - \frac{\varepsilon}{2} \gamma_i. \end{aligned}$$

Lastly, using the result (3.19) in Lemma 3.1 and noting the definition of  $\phi(x)$  in (3.9), we obtain the following result during the termination of the ACGM procedure in computing  $x_i$ :

$$\begin{aligned} \langle \nabla f(z_i), x - x_i \rangle &\geq \beta_i \langle x_i - x_{i-1}, x_i - x \rangle - \eta_i \\ &= -\frac{\beta_i}{2} \left( \|x - x_{i-1}\|^2 - \|x_i - x_{i-1}\|^2 - \|x - x_i\|^2 \right) - \eta_i \\ &\geq -\frac{\beta_i}{2} \left( \|x - x_{i-1}\|^2 - \|x - x_i\|^2 \right) - \eta_i + \frac{L_i^2}{2} \|x_i - x_{i-1}\|^2. \end{aligned}$$

In the last inequality above we use our assumption that  $\beta_k \geq L_k \gamma_k$  for all  $k$ . Based on the above three observations and rearranging terms we obtain that

$$\begin{aligned} \frac{1}{\Gamma_k} \ell_k(x) &\geq \sum_{i=1}^k \frac{1}{\Gamma_i} \left( f(y_i) - (1 - \gamma_i) f(y_{i-1}) - \frac{\gamma_i \beta_i}{2} (\|x - x_{i-1}\|^2 - \|x - x_i\|^2) \right) \\ &\quad - \frac{1}{\Gamma_i} \left( \frac{\varepsilon}{2} \gamma_i + \gamma_i \eta_i \right) \\ &= \frac{f(y_k)}{\Gamma_k} - \sum_{i=1}^k \frac{\gamma_i \beta_i}{2 \Gamma_i} (\|x - x_{i-1}\|^2 - \|x - x_i\|^2) - \sum_{i=1}^k \frac{\gamma_i \eta_i}{\Gamma_i} - \frac{\varepsilon}{2 \Gamma_k}. \end{aligned}$$

Here in the last equality we use the relations (3.13) and (3.14) and the fact that  $\gamma_1 = 1$  in its definition (3.2). We conclude the result by multiplying by  $\Gamma_k$  and rearranging terms.  $\square$

With the help of Propositions 3.2, Lemma 3.5, and Proposition 3.6, we are ready to present the complexity results of UCGS in the following theorem.

**THEOREM 3.7.** *Suppose that we apply UCGS described in Algorithm 3.1 with parameters*

$$(3.25) \quad \beta_k = L_k \gamma_k, \quad \eta_k = \frac{L_k \gamma_k D_X^2}{k}, \quad \text{and} \quad \varepsilon_k = \frac{\sigma L_k \gamma_k D_X^2}{2},$$

and  $\alpha^t$  in (3.12) and  $\delta^t = \sigma \beta D_X^2 / t$  in the ACGM procedure, where  $\sigma \geq 0$  is a parameter related to the accuracy of approximately solving linear objective optimization subproblems. Then Algorithm 3.1 terminates with an  $\varepsilon$ -solution after at most  $N_{grad}$  gradient evaluations and  $N_{lin}$  linear objective optimizations, where

$$\begin{aligned} N_{grad} &:= \left\lceil 16 \left( \frac{(3 + \sigma)^{\frac{1+\nu}{2}} M_\nu D_X^{1+\nu}}{\varepsilon} \right)^{\frac{2}{1+3\nu}} \right\rceil \quad \text{and} \\ N_{lin} &:= \left\lceil \left( \frac{7}{2} \sigma + 3 \right) N_{grad}^2 + \left( \frac{7}{2} \sigma + 6 \right) N_{grad} \right\rceil. \end{aligned}$$

*Proof.* From the definition of  $s_k$  in Algorithm 3.1, we have that if the termination criterion of UCGS in (3.8) holds, then  $y_k$  is an  $\varepsilon$ -solution to problem (1.1). Let us evaluate the number of gradient evaluations, or equivalently, the number of outer iterations of UCGS in order to compute an  $\varepsilon$ -solution  $y_k$ . Applying Proposition 3.6 with our choice of parameters we have

$$(3.26) \quad f(y_k) - \ell_k(x) = \frac{\varepsilon}{2} + \frac{\Gamma_k}{2} \sum_{i=1}^k i (\|x - x_{i-1}\|^2 - \|x - x_i\|^2) + \Gamma_k k D_X^2, \quad \forall x \in X.$$

The second term above can be further simplified by noting from the compactness of  $X$  and the definition of the diameter  $D_X$  in (1.2). Indeed, we have

$$\begin{aligned}
& \sum_{i=1}^k i \left( \|x - x_{i-1}\|^2 - \|x - x_i\|^2 \right) \\
(3.27) \quad &= \|x - x_0\|^2 + \sum_{i=2}^k (i - (i-1)) \|x - x_{i-1}\|^2 - k \|x - x_k\|^2 \\
&\leq D_X^2 + \sum_{i=2}^k D_X^2 = k D_X^2, \quad \forall x \in X.
\end{aligned}$$

Thus, we may continue by applying (3.27) to (3.26) with  $x = s_k$  to conclude that

$$f(y_k) - \ell_k(s_k) \leq \frac{\varepsilon}{2} + \frac{\Gamma_k}{2} k D_X^2 + \Gamma_k k D_X^2 = \frac{\varepsilon}{2} + \frac{3L_k \gamma_k^2}{2} D_X^2.$$

Here we make use of the description of  $\Gamma_k$  in (3.6) for the last equality. In view of the above result and the value of parameter  $\varepsilon_k$  in (3.25),  $y_k$  satisfies the termination criterion (3.8) of UCGS and hence becomes an  $\varepsilon$ -solution whenever  $k$  satisfies the relation  $(3 + \sigma)L_k \gamma_k^2 D_X^2 \leq \varepsilon$ . Applying Lemma 3.5, it follows that such relation holds whenever

$$\frac{(3 + \sigma)C_\nu D_X^2 M_\nu^{\frac{2}{1+\nu}}}{k^{\frac{1+3\nu}{1+\nu}} \varepsilon^{\frac{1-\nu}{1+\nu}}} \leq \varepsilon, \quad \text{i.e., } k \geq C_\nu^{\frac{1+\nu}{1+3\nu}} \left( \frac{(3 + \sigma)^{\frac{1+\nu}{2}} M_\nu D_X^{1+\nu}}{\varepsilon} \right)^{\frac{2}{1+3\nu}}.$$

Noting that  $C_\nu$  defined in (3.24) is a constant that depends only on  $\nu \in [0, 1]$  and observing that  $C_\nu^{\frac{1+\nu}{1+3\nu}} \leq 16$  for all  $\nu \in [0, 1]$ , we conclude that whenever  $k \geq N_{\text{grad}}$ ,  $y_k$  is an  $\varepsilon$ -solution. Therefore, UCGS requires at most  $N_{\text{grad}}$  gradient evaluations of  $\nabla f$  to compute an  $\varepsilon$ -solution.

It suffices to compute the number of linear objective optimizations that UCGS requires for computing an  $\varepsilon$ -solution. This is equivalent to estimating the total number of inner iterations that UCGS requires. Let us estimate the maximal number of inner iterations required before the termination criterion (3.11) is satisfied. Recall from the remark after (3.12) that  $\alpha^t$  is the best linesearch parameter and hence satisfies assumption (3.20) in Proposition 3.2. Applying Proposition 3.2 and noting the definition of approximate solution  $v^t$  in (3.10), we have that the maximal number of linear objective optimizations performed at the  $k$ -th call to the ACGM procedure is at most

$$T_k := 1 + \left\lceil \frac{(\sigma + 6)\beta_k D_X^2}{\eta_k} \right\rceil = 1 + \lceil (\sigma + 6)k \rceil.$$

Adding one linear objective optimization problem in (3.7) in each other iteration concerning the termination criterion of UCGS, we conclude that the total number of linear objective optimizations for UCGS to compute an  $\varepsilon$ -solution is bounded above by

$$\sum_{k=1}^{N_{\text{grad}}} (T_k + 1) \leq \sum_{k=1}^{N_{\text{grad}}} 3 + (\sigma + 6)k = \left( \frac{7}{2}\sigma + 3 \right) N_{\text{grad}}^2 + \left( \frac{7}{2}\sigma + 6 \right) N_{\text{grad}}. \quad \square$$

We conclude this section with a few remarks on the above complexity results of UCGS. First, we note that UCGS is similar to FGM in [22] in the sense that the number of gradient evaluations generalizes the accelerated gradient descent method in [21]. From Theorem 3.7, number of gradient evaluations required by UCGS to compute an approximate solution is  $\mathcal{O}((M_\nu D_X^{1+\nu}/\varepsilon)^{\frac{2}{1+3\nu}})$ . In the smooth case when  $\nu = 1$ , this becomes  $\mathcal{O}(\sqrt{M_1 D_X^2/\varepsilon})$  which matches the complexities of gradient evaluations in [22, 21]. Second, unlike FGM that requires exact solutions to projection subproblems, we have a bound on the number of linear objective optimizations required to solve the projection subproblem (3.4). From this perspective, UCGS is a generalization of CGS in [15] as a universal method that covers not only the smooth case (when  $\nu = 1$ ) but also the weakly smooth case (when  $\nu \in (0, 1)$ ), without requiring any knowledge of Hölder exponent  $\nu$  and constant  $M_\nu$ . Indeed, when  $\nu = 1$  our complexity on the number of linear objective optimizations is on the order of  $\mathcal{O}(M_1 D_X^2/\varepsilon)$ , which matches the that of CGS in [15]. Third, the number of linear objective optimizations and gradient evaluations when CG is applied to (1.1) was shown to be  $\mathcal{O}\left((M_\nu D^{1+\nu}/\varepsilon)^{\frac{1}{\nu}}\right)$  in [19, 8]. In view of Theorem 3.7, UCGS benefits from sliding and only requires  $\mathcal{O}\left((M_\nu D^{1+\nu}/\varepsilon)^{\frac{2}{1+3\nu}}\right)$  gradient evaluations and  $\mathcal{O}\left((M_\nu D^{1+\nu}/\varepsilon)^{\frac{4}{1+3\nu}}\right)$  linear objective optimizations which are both improvements over the results in [19, 8] whenever  $\nu \in (0, 1]$ . Fourth, our proposed UCGS method is not only a universal method generalization of the CGS in [15]. Indeed, there are more features added for practical implementation: it has an implementable exit criterion and allows for an approximate solution to (3.10). Note that such added features of the UCGS does not affect its theoretical complexity. Finally, we use the same accuracy constant  $\sigma$  for approximately solving the linear subproblems in setting the parameters  $\varepsilon_k$  and  $\delta^t$ . It is easy to change the proof if we use different accuracy constants for  $\varepsilon_k$  and  $\delta^t$ .

**4. Numerical results.** Our goal in this section is to present preliminary results from our numerical experiments. We will compare the performance of our proposed UCGS algorithm with that of the CG method in [19] in two numerical experiments described below. The experiments are performed using MATLAB R2018b.

In the first experiment, we consider the problem

$$\min_{x \in \text{conv}(V)} f(x) := \|Ax - b\|_2$$

with  $V = \{v_1, \dots, v_p\} \subseteq \mathbb{R}^n$ ,  $\text{conv}(V) := \{x \in \mathbb{R}^n : \exists \lambda \in \Delta_p \text{ s.t. } x = \sum_{j=1}^p \lambda_j v_j\}$ , and  $\Delta_p := \{\lambda \in \mathbb{R}^p : \sum_{i=1}^p \lambda_i = 1, \lambda_i \geq 0\}$  is the standard simplex. In this experiment, we generated vectors  $v_i$  uniformly in  $[0, 1]^n$ . The matrix  $A \in \mathbb{R}^{m \times n}$  is a Gaussian randomly generated sparse matrix with density  $d$ . For this experiment, we fix the number of vectors in the set  $V$  to be  $p = 500$  and set  $m = 2n$ . The linear objective optimization subproblem is a linear program over the standard simplex and can be computed easily.

For our second experiment, we solve the problem

$$\min_{X \in \text{Spe}_n} f(X) := \sum_{i=1}^m \|X - A_i\|_2$$

where  $\text{Spe}_n := \{X \in \mathbb{R}^{n \times n} : \text{tr}(X) = 1, X \succeq 0\}$  and  $A_i \in \text{Spe}_n$  for each  $i = 1, \dots, m$ . The matrices  $A_i$  are obtained by randomly generating a  $n \times n$  matrix

whose entries follow uniform  $[0, 1]$  distributions and then projecting it into  $\text{Spe}_n$ . The linear objective optimization problem over  $\text{Spe}_n$  is equivalent to a smallest eigenvalue problem, which will be solved by MATLAB's `eigs()` function. Note that a solution to the smallest eigenvalue problem will not be exact, and therefore we benefit from being able to solve the linear subproblems approximately.

In our experiments, UCGS will terminate whenever an  $\varepsilon$ -solution with tolerance  $\varepsilon = 10^{-3}$  is computed. We will terminate CG if its computational time exceeds twice the amount that UCGS spent before termination. Note that both models in the experiments have nonsmooth objective functions, but are still differentiable at many feasible points. Therefore, they may benefit from a universal method for  $\nu \in (0, 1]$ .

$n$	$d$	UCGS				CG		
		GE	LO	Time	Error	Iter	Time	Error
2500	0.2	66	2690	6.71	$9.945e-4$	572	13.42	$9.7086e1$
2500	0.4	60	3679	9.08	$9.976e-4$	524	18.17	$1.404e2$
2500	0.6	62	245	2.64	$9.678e-4$	146	5.29	$5.598e2$
2500	0.8	57	3176	8.45	$9.768e-4$	399	16.93	$2.400e2$
5000	0.2	71	286	7.13	$9.882e-4$	178	14.32	$6.037e2$
5000	0.4	42	52	4.89	$9.585e-4$	84	9.81	$1.689e3$
5000	0.6	68	4564	36.14	$9.727e-4$	483	72.40	$3.527e2$
5000	0.8	67	419	12.91	$9.815e-4$	161	25.94	$1.165e3$
10000	0.2	85	12269	150.51	$9.96e-4$	915	301.21	$2.449e2$
10000	0.4	69	12614	157.39	$9.916e-4$	636	315.27	$4.734e2$
10000	0.6	70	16063	205.87	$9.821e-4$	653	412.14	$5.423e2$
10000	0.8	69	12707	180.65	$9.862e-4$	473	361.73	$8.162e2$

TABLE 1

*Minimizing over a convex hull. Here, we report the gradient evaluations (outer iterations) and linear objective optimization (inner iterations) for UCGS as well as the error that it terminated with. For CG, we allow it to run for twice the amount of time that UCGS took. We then report the number of iterations and whether terminating objective value was better than that of UCGS.*

$n$	$m$	UCGS				CG		
		GE	LO	Time	Error	Iter	Time	Error
50	50	1354	8493	9.87	$9.992e-4$	6908	19.74	$6.073e-3$
50	100	1767	11138	13.09	$9.994e-4$	7038	26.19	$1.172e-2$
50	200	2425	15173	25.39	$9.995e-4$	8273	50.79	$2.271e-2$
100	50	1836	13056	159.61	$9.980e-4$	11648	319.25	$3.225e-3$
100	100	2347	16816	216.59	$9.990e-4$	13372	433.20	$5.634e-3$
100	200	3296	23836	310.16	$9.984e-4$	16053	620.36	$9.892e-3$
200	50	1722	33673	470.71	$9.989e-4$	15966	941.43	$3.308e-3$
200	100	2314	46323	730.69	$9.994e-4$	17033	1461.42	$6.870e-3$
200	200	3154	64511	1086.42	$9.992e-4$	19762	2172.85	$1.015e-2$

TABLE 2

*Minimizing over standard spectrahedron. Here, we report the gradient evaluations (outer iterations) and linear objective optimization (inner iterations) for UCGS as well as the error that it terminated with. For CG, we allow it to run for twice the amount of time that UCGS took. We then report the number of iterations and whether terminating objective value was better than that of UCGS.*

The results from the numerical experiments are documented in Tables 1 and 2.

Column 1 indicates the sizes  $n$ <sup>1</sup> whereas the second column represents either the density of  $A$  or the value of  $m$  for experiments 1 and 2 respectively. Columns 3 and 4 denote the number of outer iterations, i.e. gradient evaluations (GE), and inner iterations, i.e. linear objective optimization (LO), respectively that UCGS performed before terminating with the desired tolerance. Columns 5 and 6 present the time (in seconds) used and error upon termination of UCGS. For CG, we report the total number of iterations (Iter) performed, the computational time (in seconds) required and the final error in Columns 7, 8, and 9. Note that if the time of CG is twice that of UCGS, then the error is not expected to be below our specified tolerance.

Let us make a few comments regarding the results in Tables 1 and 2. For the convex hull experiment in Table 1, we see that the excessive number of gradient evaluations of CG prevents it from being competitive. The gradient of our objective function requires a matrix multiplication of increasingly dense matrices. As these densities tend to 1, the gradient evaluations become more computationally expensive, and CG cannot report as good of a solution as UCGS with even in twice the allotted time, because it requires much more gradient evaluations to compute an approximate solution. We also note the necessity of a projection-free algorithm for this feasible set since the projection onto the convex hull requires the solving of a quadratic program. For any moderately sized  $n$ , this quadratic program is computationally infeasible to solve. For example, one iteration of FGM in [22] applied to the problem instance with  $n = 2500$  and  $d = 0.2$  takes at least 20 seconds, which is three times as long as UCGS took to converge.

The second experiment over the standard spectrahedron removes the previous difficulty of computing the gradient. In experiment 2, the cost of the gradient evaluation is almost negligible. However, we still see in Table 2 that UCGS outperforms CG. In this case, the superior linear objective optimization complexity of UCGS can be seen by noting that CG performs 1 linear objective optimization per iteration. Thus, even with a comparable amount of linear objective optimizations, CG can still not match the complexity of UCGS. This directly highlights the differences in the linear objective optimization complexity mentioned previously. We also observe the effectiveness of the implementable stopping criterion which enabled us to terminate when an  $\varepsilon$ -solution was achieved.

**5. Concluding remarks.** In this paper, we present a novel projection-free method, namely the universal conditional gradient sliding (UCGS) method, for convex differentiable optimization with Hölder continuous gradients. We show that UCGS is a generalization of other conditional gradient type methods in terms of gradient evaluations and linear objective optimizations and at the same time has a more practical implementation by requiring less problem dependent parameters. Specifically, for an objective function whose gradient is Hölder continuous with exponent  $\nu \in (0, 1]$  and constant  $M_\nu > 0$ , we prove that UCGS is able to terminate and output an  $\varepsilon$ -solution with at most  $\mathcal{O}((M_\nu D_X^{1+\nu}/\varepsilon)^{2/(1+3\nu)})$  gradient evaluations and  $\mathcal{O}((M_\nu D_X^{1+\nu}/\varepsilon)^{4/(1+3\nu)})$  linear objective optimizations. Moreover, it is able to perform the computation without requiring any specific knowledge of the Hölder exponent  $\nu$  and constant  $M_\nu$ . UCGS improves the state-of-the-art complexity results (achieved by the conditional gradient method [19, 8]) of first-order projection-free methods when  $\nu \in (0, 1)$ . It also matches the state-of-the-art complexity results when  $\nu = 1$  (achieved by the conditional gradient sliding method [15]) and adds more features

---

<sup>1</sup>Note that the length of the vectors are  $n$  and  $n^2$  in the first and second experiments respectively.

allowing for practical implementation.

The results of this paper can be further generalized. First, if there exists a prox-function  $x \mapsto V(u, x)$  defined over the feasible set  $X$  that is strongly convex and has Lipschitz continuous gradients with respect to a general norm  $\|\cdot\|$  (see, e.g., [9, 4] for discussions on such prox-functions), then it can be shown that the UCGS can be generalized to a non-Euclidean version that still achieves  $\mathcal{O}(1/\varepsilon^{2/(1+3\nu)})$  gradient evaluations and  $\mathcal{O}(1/\varepsilon^{4/(1+3\nu)})$  linear objective optimizations. The constants in these complexities will depend on the smoothness information  $\nu$  and  $M_\nu$  with respect to the general norm  $\|\cdot\|$ , the diameter of  $X$  in terms of  $\max_{x,y \in X} V(x, y)$ , and the strongly convex constant of the prox-function and the Lipschitz constant of its gradient. Second, while we focus on convex differentiable optimization, the results in this paper can also be extended to non-differentiable cases. Indeed, note that a convex function that is Lipschitz continuous with constant  $2M_0$  also satisfies the Hölder condition we describe in (1.3) with exponent  $\nu = 0$  and constant  $M_0$ . Also, our proof throughout this paper can be easily extended by replacing gradients  $\nabla f$  to subgradients  $f'$ . Consequently, Theorem 3.7 shows that UCGS computes an  $\varepsilon$ -solution with  $\mathcal{O}((M_0 D_X/\varepsilon)^2)$  subgradient evaluations and  $\mathcal{O}((M_0 D_X/\varepsilon)^4)$  linear objective optimizations.

We conclude this paper by discussing some potential future work. First, similar to the conditional gradient sliding method [15], our proposed UCGS requires some information on the diameter  $D_X$  of the feasible set  $X$ . It is interesting to study whether this diameter can be estimated adaptively by backtracking search, similar to the line-search strategy we use to estimate the smoothness information  $L_k$ . Second, while we mention in the above generalization of UCGS that our results can be extended to convex non-differentiable optimization, if we focus solely on non-differentiable cases, there exists interesting complexity results in the literature. Specifically, in [23] it is shown that in the non-differentiable case it is possible to achieve  $\mathcal{O}((M_0 D_X/\varepsilon)^2)$  complexity for both subgradient evaluations and linear objective optimizations whenever the objective function is Lipschitz continuous. The method in [23] can only be applied to nonsmooth cases when  $\nu = 0$ , and is unknown whether it can be generalized to a universal method that uniformly computes approximate solutions for nonsmooth, weakly smooth, and smooth convex optimization problems. It is a potential future work to study whether the technique in [23] can improve our developed linear objective optimization complexity of the Hölder continuous gradient case when  $\nu \in (0, 1]$ , and whether a universal method for all cases  $\nu \in [0, 1]$  can be developed.

#### REFERENCES

- [1] A. BECK, *First-order methods in optimization*, SIAM, 2017.
- [2] S. BUBECK, *Convex optimization: Algorithms and complexity*, Foundations and Trends® in Machine Learning, 8 (2015), pp. 231–357.
- [3] Y. CHEN, G. LAN, Y. OUYANG, AND W. ZHANG, *Fast bundle-level methods for unconstrained and ball-constrained convex optimization*, Computational Optimization and Applications, 73 (2019), pp. 159–199.
- [4] C. D. DANG AND G. LAN, *On the convergence properties of non-Euclidean extragradient methods for variational inequalities with generalized monotone operators*, Computational Optimization and applications, 60 (2015), pp. 277–310.
- [5] O. DEVOLDER, F. GLINEUR, AND Y. NESTEROV, *First-order methods of smooth convex optimization with inexact oracle*, Mathematical Programming, 146 (2014), pp. 37–75.
- [6] M. FRANK AND P. WOLFE, *An algorithm for quadratic programming*, Naval research logistics quarterly, 3 (1956), pp. 95–110.
- [7] R. M. FREUND AND P. GRIGAS, *New analysis and results for the Frank–Wolfe method*, Math-



- emational Programming, 155 (2016), pp. 199–230.
- [8] S. GHADIMI, *Conditional gradient type methods for composite nonlinear and stochastic optimization*, Mathematical Programming, 173 (2019), pp. 431–464.
  - [9] S. GHADIMI AND G. LAN, *Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization I: A generic algorithmic framework*, SIAM Journal on Optimization, 22 (2012), pp. 1469–1492.
  - [10] Z. HARCHAOUI, A. JUDITSKY, AND A. NEMIROVSKI, *Conditional gradient algorithms for norm-regularized smooth convex optimization*, Mathematical Programming, 152 (2015), pp. 75–112.
  - [11] M. JAGGI, *Revisiting Frank-Wolfe: Projection-free sparse convex optimization.*, in ICML (1), 2013, pp. 427–435.
  - [12] G. LAN, *Bundle-level type methods uniformly optimal for smooth and nonsmooth convex optimization*, Mathematical Programming, 149 (2015), pp. 1–45.
  - [13] G. LAN, *Gradient sliding for composite optimization*, Mathematical Programming, 159 (2016), pp. 201–235.
  - [14] G. LAN, *First-order and Stochastic Optimization Methods for Machine Learning*, Springer, 2020.
  - [15] G. LAN AND Y. ZHOU, *Conditional gradient sliding for convex optimization*, SIAM Journal on Optimization, 26 (2016), pp. 1379–1409.
  - [16] E. S. LEVITIN AND B. T. POLYAK, *Constrained minimization methods*, USSR Computational mathematics and mathematical physics, 6 (1966), pp. 1–50.
  - [17] H. NAZARI AND Y. OUYANG, *Backtracking linesearch for conditional gradient sliding*, arXiv preprint arXiv:2006.05272, (2020).
  - [18] A. NEMIROVSKI AND D. YUDIN, *Problem complexity and method efficiency in optimization*, Wiley-Interscience Series in Discrete Mathematics, John Wiley, XV, 1983.
  - [19] Y. NESTEROV, *Complexity bounds for primal-dual methods minimizing the model of objective function*, Mathematical Programming, 171 (2018), pp. 311–330.
  - [20] Y. NESTEROV, *Lectures on Convex Optimization*, vol. 137, Springer, 2018.
  - [21] Y. E. NESTEROV, *Introductory Lectures on Convex Optimization: A Basic Course*, Kluwer Academic Publishers, Massachusetts, 2004.
  - [22] Y. E. NESTEROV, *Universal gradient methods for convex optimization problems*, Mathematical Programming, 152 (2015), pp. 381–404.
  - [23] K. K. THEKUMPARAMPIL, P. JAIN, P. NETRAPALLI, AND S. OH, *Projection efficient subgradient method and optimal nonsmooth frank-wolfe method*, arXiv preprint arXiv:2010.01848, (2020).