Minimisation of Submodular Functions Using Gaussian Zeroth-Order Random Oracles

Amir Ali Farzin[1], Yuen-Man Pun[1], Philipp Braun[1], Tyler Summers[2], and Iman Shames[3]

***Abstract*— We consider the minimisation problem of submodular functions and investigate the application of a zeroth-order method to this problem. The method is based on exploiting a Gaussian smoothing random oracle to estimate the smoothed function gradient. We prove the convergence of the algorithm to a global ϵ-approximate solution in the offline case and show that the algorithm is Hannan-consistent in the online case with respect to static regret. Moreover, we show that the algorithm achieves $O(\sqrt{NP_N^*})$ dynamic regret, where N is the number of iterations and P_N^* is the path length. The complexity analysis and hyperparameter selection are presented for all the cases. The theoretical results are illustrated via numerical examples.**

I. INTRODUCTION

In this paper, we study a minimisation problem of the form

$$\min_{S\subseteq Q} f(S), \tag{1}$$

where the objective function $f\colon 2^Q \to \mathbb{R}$ is a set function over a ground set Q of n elements, where 2^Q denotes the power set of Q. Without loss of generality, we assume $Q = [n] = \{1, 2, \dots, n\}$. We assume a minimiser for f exists (which may not be unique) and consider computing an ϵ-approximate minimiser of f, i.e., $S \subseteq Q$ with $f(S) \leq f^* + \epsilon$, where $f^* \stackrel{\text{def}}{=} \min_{S\subseteq Q} f(S)$. More specifically, we focus on submodular function minimisation (SFM), which counts as a foundational problem in combinatorial optimisation. After the seminal work of [1], SFM found its place as a key tool in many areas. Its application can be seen in machine learning [2], economics [3], computer vision [4], and speech recognition [5]. For more information, we refer to the survey papers [6], [7]. In the seminal work [8], the authors showed that SFM problems can be solved in polynomial time. Since then, there has been extensive research and multiple advances in characterising SFM [9], [10], [11].

An important key in solving SFM is the fact that SFM reduces to minimising the Lovász extension [12] over $[0, 1]^n$. The Lovász extension enjoys desirable properties such as convexity and Lipschitz continuity, which can be leveraged to solve SFM. The Lovász extension is non-smooth and thus gradient methods can not be used to find its minimisers. Hence, leveraging the subgradient of the Lovász extension plays a key part in the developed of methods to solve both offline SFM [13], [11] and online SFM [14], [15], [16]. Since the Lovász extension is non-smooth, zeroth-order methods are a suitable choice to solve SFM.

Optimisation frameworks that rely only on the evaluations of the objective function (but not on its derivatives) are commonly called zeroth-order (ZO) or derivative-free methods [17]. Multiple ZO algorithms have been developed to solve various optimisation problems. The authors in [18], proposed and analysed a method based on ZO oracles leveraging Gaussian smoothing. This method has been extensively studied for different classes of functions in both offline [19], [20] and online settings [21]. In this work, we leverage a Gaussian smoothing framework to solve SFM in both offline and online settings. Leveraging this method, we do not need to calculate the subgradient of the Lovász extension. In this work, we first consider the offline SFM and show that the Gaussian smoothing ZO method, in the expectation sense, converges to an ϵ-approximate solution in $O(n^2\epsilon^{-2})$ function calls. Next, we study online SFM under both static and dynamic regrets (unlike the previous works on online SFM, which only consider the static regret). We show that, in the expectation sense, the algorithm is Hannan-consistent (see [14]) in terms of static regret, meaning that its average regret per iteration vanishes as the number of iterations grows. Moreover, it achieves a dynamic regret of $O(\sqrt{NP_N^*})$, where N is the number of iterations and P_N^* denotes the path length of the minimisers. It is known that the dynamic regret bound is dependent on the path length [22], [23].

Outline: The paper is organised as follows. Preliminaries are introduced in Section II. In Section III, the main framework is given, and convergence and complexity results related to the proposed framework are presented for offline and online settings. Illustrative examples are given in Section IV. Lastly, we conclude our paper and discuss potential future directions in Section V.

Notation: For $x, y \in \mathbb{R}^d$, $\langle x, y\rangle = x^\top y$ and let $\|\cdot\|$ be the Euclidean norm of its argument. The gradient of a differentiable function $f : \mathbb{R}^{n+m} \to \mathbb{R}$ is denoted by ∇f. Let $\{0, 1\}^n \subseteq \mathbb{N}^n$ denote the set of n-dimensional binary vectors, where each coordinate is either 0 or 1. There is a natural bijection between a vector $x \in \{0, 1\}^n$ and a subset $S \subseteq [n]$. For a set $S \subseteq [n]$, let $\chi_S \in \{0, 1\}^n$ denote its characteristic vector, defined by $\chi_S(i) = 1$ if $i \in S$ and $\chi_S(i) = 0$ otherwise. We will freely move between these two representations and, in particular, interchangeably use

$$f(S),\ S \subseteq [n] \quad \text{and} \quad f(\chi_S),\ \chi_S \in \{0, 1\}^n. \tag{2}$$

The function $F : D \to \mathbb{R}$ with $D \subseteq \mathbb{R}^n$ is Lipschitz if there

[1]School of Engineering, Australian National University and CIICADA Lab {amirali.farzin,philipp.braun, yuenman.pun}@anu.edu.au

[2] Department of Mechanical Engineering, University of Texas at Dallas tyler.summers@utdallas.edu

[3] Department of Electrical and Electronic Engineering, University of Melbourne and CIICADA Lab iman.shames@unimelb.edu.au

exists a *Lipschitz constant* $L_0 > 0$, such that

$$|F(x) - F(y)| \leq L_0 \|x - y\|, \quad \forall\, x, y \in D. \tag{3}$$

II. Preliminaries

In this section, we review the necessary definitions and background material of submodular functions, Lovász extension, and Gaussian smoothing.

A. Submodular Functions and the Lovász Extension

In this section, we give the background related to submodular functions and their Lovász Extensions.

Definition 1 (Submodular Functions [14]): A function $f : 2^{[n]} \to \mathbb{R}$ is called *submodular* if it exhibits diminishing marginal returns, i.e., for all $S \subseteq T \subseteq [n]$ and any $i \notin T$,

$$f(S \cup \{i\}) - f(S) \;\geq\; f(T \cup \{i\}) - f(T).$$

An equivalent characterisation is that for all $S, T \subseteq [n]$,

$$f(S) + f(T) \;\geq\; f(S \cap T) + f(S \cup T).$$

Let $f : 2^{[n]} \to \mathbb{R}$ be a submodular function defined on subsets of a ground set E. We assume that access to f is provided only through a value oracle. That is, given any set $S \subseteq [n]$, the oracle returns the value $f(S)$. The *Lovász extension* provides a continuous, convex extension of a submodular function $f : 2^{[n]} \to \mathbb{R}$ to a function $f^L : [0,1]^n \to \mathbb{R}$, where $[0,1]^n \subset \mathbb{R}^n$ is the unit hypercube.

Definition 2 (Lovász Extension [14]): Let $x \in [0,1]^n$ and let $A_i \subset [n]$ ($i \in [n]$, $n \in \mathbb{N}$), be a chain of subsets $A_0 \subset A_1 \subset \cdots \subset A_p$ such that x can be expressed as a convex combination $x = \sum_{i=0}^{n} \lambda_i \chi_{A_i}$ where $\lambda_i > 0$ and $\sum_{i=0}^{n} \lambda_i = 1$. Then, the Lovász extension f^L at x is defined to be

$$f^L(x) = \sum_{i=0}^{p} \lambda_i f(A_i). \tag{4}$$

Remark 1: According to [14], the Lovász extension can alternatively be obtained by sampling a threshold $\tau \in [0,1]$ uniformly at random, and considering

$$f^L(x) = E_\tau[f(S_\tau)], \qquad S_\tau = \{i \in [n] : x(i) > \tau\}. \tag{5}$$

Remark 1, implies that for all sets $S \subseteq [n]$, it follows that $f^L(\chi_S) = f(S)$. We use the following properties of the Lovász extension function shown in [12], [24], [25], [26].

Lemma 1: Let $f : 2^{[n]} \to \mathbb{R}$ be a submodular function and f^L be its Lovász extension. Then, we have that (i) f^L is convex, (ii) f^L is piecewise linear, (iii) $\min_{S \subseteq [n]} f(x) = \min_{x \in \{0,1\}^n} f^L(x) = \min_{x \in [0,1]^n} f^L(x)$, and (iv) the set of minimisers of $f^L(x)$ in $[0,1]^n$ is the convex hull of the set of minimisers of $f^L(x)$ in $\{0,1\}^n$.

Also, in [11, Lemma 2.2], the authors show that we can go from an approximate minimiser of f^L to an approximate minimiser of f in $O(n)$ function calls. Thus, leveraging this fact and considering Lemma 1, we can consider solving the non-smooth convex minimisation problem of the form

$$\min_{x \in [0,1]^n} f^L(x), \tag{6}$$

instead of Problem 1. Since f^L is continuous and the domain is compact and nonempty, at least a minimiser for f^L exists, which may not be unique. Since f^L is non-smooth, ZO methods are good candidates to solve (6).

B. Gaussian smoothing

In this section, following [18], we define the Gaussian smoothed version of a continuous function $F : \mathbb{R}^n \to \mathbb{R}$, termed F_μ as below:

$$\begin{aligned} F_\mu(x) &\stackrel{\text{def}}{=} \frac{1}{\kappa} \int_E F(x + \mu u) e^{-\frac{1}{2}||u||^2} \mathrm{d}u, \\ \kappa &\stackrel{\text{def}}{=} \int_E e^{-\frac{1}{2}||u||^2} \mathrm{d}u = \frac{(2\pi)^{n/2}}{[\det B]^{\frac{1}{2}}}, \end{aligned} \tag{7}$$

where vector $u \in \mathbb{R}^n$ is sampled from $\mathcal{N}(0, B^{-1})$, $B \in \mathbb{R}^{n \times n}$ is positive definite, and $\mu \in \mathbb{R}_{>0}$ is the smoothing parameter. In [18], it is shown that whether F is differentiable or not, F_μ is always differentiable, and $\nabla F_\mu(x) = E_u[\frac{F(x+\mu u)}{\mu} u]$. Instead of this one-point gradient estimation, to reduce the variance of the estimator, we consider the two-point estimator and define the random oracle g_μ as

$$g_\mu(x) \stackrel{\text{def}}{=} \tfrac{1}{\mu}(F(x + \mu u) - F(x) B u), \tag{8}$$

where u and B are defined above.

III. Main Results

In this section, we investigate the application of a ZO algorithm in solving offline and online minimisation of submodular functions by considering their Lovász extensions. First, we show that we can use Gaussian smoothing on the Lovász extension $f^L(x)$. Note that it is not trivial to apply (8) with f^L as $x + \mu u$ can be outside of the unit hypercube. To use Gaussian smoothing, we need to extend the domain of f^L to $\mathbb{R}^n$ using [25, Definition 3.1].

Definition 3 (Extended domain Lovász Extension): Given a set-function $f : 2^{[n]} \to \mathbb{R}$ such that $f(\emptyset) = 0$, the Lovász extension $f^L : \mathbb{R}^n \to \mathbb{R}$ is defined as follows: For $x \in \mathbb{R}^n$, order the components in decreasing order $x(j_1) \geq \cdots \geq x(j_n)$, where $(j_1, \ldots, j_n)$ is a permutation. The Lovász extension is given by

$$f^L(x) = \sum_{k=1}^{n} x_{j_k} [f(\{j_1, \ldots, j_k\}) - f(\{j_1, \ldots, j_{k-1}\})] \tag{9}$$

Remark 2: According to [25, (3.3)], it holds that (5) and (9) coincide on the domain $[0,1]^n$ and thus the notation f^L can be used in Definition 2 and in Definition 3. From here, we use f^L to denote (9).

Letting $f^L_\mu = E_{u \sim \mathcal{N}(0,I)}[f^L(x + \mu u)]$, the two-point gradient estimator given in (8) with $F = f^L$ gives an unbiased estimation of $\nabla f^L_\mu(x)$ for solving (6). In the next section, we will analyse the convergence for the offline problem (6) and extend the results to the online case.

A. Offline Problem

In this section, first we introduce Algorithm 1. This framework has been studied for different minimisation problems in the literature [18], [19], [20]. In each iteration of Algorithm 1, first, to update our guess, we sample t_k number of directions from the standard Gaussian distribution and calculate the random oracles, with $F = f^L$, using the

Algorithm 1 ZO-GD

Input: $x_0 \in \mathcal{K},\ \{h_k\}_{k=0}^{N} \subset \mathbb{R}_{>0},\ \mu > 0,\ N, t_k \in \mathbb{N}$
for $k = 0, \dots, N$ **do**
  Sample $u_k^0, \dots, u_k^{t_k}$ from $\mathcal{N}(0, \mathbb{I})$
  Obtain $g_\mu^0(x_k), \dots, g_\mu^{t_k}(x_k)$ using $u_k^0, \dots, u_k^t$ and (8)
  $g_\mu(x_k) = \frac{1}{t_k} \sum_{i=0}^{t_k} g_\mu^i(x_k)$
  $\bar{x}_{k+1} = x_k - h_k g_\mu(x_k)$
  $x_{k+1} = \text{Proj}_{[0,1]^n}(\bar{x}_{k+1})$
end for
return x_k for all $k = 0, \dots, N$.

sampled directions and leverage the average of the oracles. It is easy to see that with t_k number of samples instead of one, still $E[g_\mu(x_k)] = \nabla f_\mu^L(x_k)$, while the variance of $g_\mu(x_k)$ will be reduced [27]. Then, Algorithm 1 performs a descent step using the calculated random oracle with h_k as the step size. At last, we project back to the unit hypercube. The projection operator $\text{Proj}_{[0,1]^n} : \mathbb{R}^n \to [0,1]^n$ is defined by

$$\text{Proj}_{[0,1]^n, i}(x) = \max\{0, \min\{1, x(i)\}\}, \quad i \in [n].$$

Considering Lemma 1, it can be concluded that f^L is continuous and piecewise affine. Thus, f^L is Lipschitz continuous with Lipschitz constant $L_0 \in \mathbb{R}_{>0}$ defined through the maximal slope of the piecewise affine functions. Following [14], if in each iteration of Algorithm 1 we choose a threshold $\tau \in [0,1]$ uniformly at random, we can define the set

$$S_k = \{i \in [n] : x_k(i) > \tau\}. \tag{10}$$

Next, we present the main theorem of this section.

Theorem 1: Consider Algorithm 1 and let $f : 2^{[n]} \to \mathbb{R}$ be a submodular function and f^L be its Lovász extension with x^* as a minimiser. Let L_0 be the Lipschitz constant of f^L, $N \geq 0$ be the number of iterations, $t_k = t \geq 1$ be the number of samples, $r_0 = \|x_0 - x^*\|$, $\mu > 0$ be smoothing parameter, $\mathcal{U}_k = [u_0, u_1, , \cdots, u_k]$, $k \in [N]$. Moreover, let $\{x_k\}_{k \geq 0}$ be the sequences generated by Algorithm 1, Then, for any iteration N, with $h_k = h > 0$, we have

$$\frac{1}{N+1} \sum_{k=0}^{N} E_{\mathcal{U}_k}[f^L(x_k)] - f^L(x^*) \leq \frac{n}{2h(N+1)} + \frac{\mu}{2} L_0 n^{\frac{1}{2}} L_0^2 h (n+4)^2. \tag{11}$$

Moreover, let $\epsilon > 0$, $\mu \leq \frac{\epsilon}{2L_0 n^{\frac{1}{2}}}$ and

$$h = \frac{r_0}{(N+1)^{\frac{1}{2}} L_0 (n+4)}, \quad N \geq \left\lceil \frac{4 r_0^2 L_0^2 (n+4)^2}{\epsilon^2} - 1 \right\rceil.$$

Then, we have

$$E_{\mathcal{U}_{N-1}}[f^L(\hat{x}_N)] - f^L(x^*) \leq \epsilon, \tag{12}$$

where $\hat{x}_N \overset{\text{def}}{=} \arg\min_x [f^L(x) : x \in \{x_0, \dots, x_N\}]$, and

$$E_{\tau, \mathcal{U}_{N-1}}[f(\hat{S}_N)] - f(S^*) \leq \epsilon, \tag{13}$$

where S^* is a minimiser of f, τ is the threshold used in (10) and $\hat{S}_N \overset{\text{def}}{=} \underset{S \in \{S_0, \dots, S_N\}}{\arg\min} f(S)$.

Proof: The proof follows the proof of [18, Thm. 6]. Let $r_k = \|x_k - x^*\|$. Using the non-expansiveness property of the projection to convex sets, we have

$$\begin{aligned} r_{k+1}^2 \leq & \|\bar{x}_{k+1} - x^*\|^2 \leq \|x_k - x^* - h g_\mu(x_k)\|^2 \\ \leq & r_k^2 - 2h \langle g_\mu(x_k), x_k - x^* \rangle + h^2 \|g_\mu(x_k)\|^2. \end{aligned} \tag{14}$$

Taking the expectation of (14) with respect to u_k and considering the fact that similar to the proof of [18, Thm. 4.1], we can obtain an upper-bound for (8), we have

$$E_{u_k}[r_{k+1}^2] \leq r_k^2 - 2h \langle \nabla f_\mu^L(x_k), x_k - x^* \rangle + h^2 L_0^2 (n+4)^2.$$

Considering the convexity of f^L (which implies that f_μ^L is convex [18]) and [18, Thm. 1], we have

$$f^L(x_k) - f^L(x^*) \leq \frac{r_k^2 - E_{u_k}[r_{k+1}^2]}{2h} + \mu L_0 n^{\frac{1}{2}} + \frac{h L_0^2 (n+4)^2}{2}. \tag{15}$$

Considering the fact that $r_0^2 = \|x_0 - x^*\|^2 \leq n$, taking expectation with respect to $\mathcal{U}_{k-1}$ from (15), summing from $k = 0$ to $k = N$, and dividing by $N+1$, we arrive at (11). If we substitute the values in Theorem 1 for μ, h, and N in (11), we have

$$\frac{1}{N+1} \sum_{k=0}^{N} E_{\mathcal{U}_k}[f^L(x_k)] - f^L(x^*) \leq \epsilon.$$

We know that for $k \in [N]$, $f^L(x_k) - f(x^*) \geq 0$. Considering the fact that if the average of a positive sequence is less than or equal to ϵ, then the minimum is less than or equal to ϵ, i.e., we arrive at (12).

Considering Lemma 1, we know $f(S^*) = f^L(x^*)$ where x^* is a minimiser of the Lovász extension. From the projection step in Algorithm 1, we know that $x_k \in [0,1]^n$ for all $k \in [N]$. Thus, considering Definition 2, we have $f^L(x_k) = E_\tau[f(S_k)]$. Moreover, we know that τ and u_k are independent random variables. Considering these facts and taking expectation from $f(S_k) - f(S^*)$ with respect to τ, then taking expectation with respect to $\mathcal{U}_{k-1}$, summing from $k = 0$ to $k = N$, and dividing by $N+1$, and letting the hypothesis of Theorem 1 be satisfied, we have

$$\frac{1}{N+1} \sum_{k=0}^{N} E_{\tau, \mathcal{U}_k}[f(S_k)] - f(S^*) \leq \epsilon. \tag{16}$$

Similarly, we can pick the best guess and arrive at (13). ■

Theorem 1, shows that in the expectation sense, there exists an ϵ-optimal point for f^L in the sequence generated by Algorithm 1.

Remark 3: If instead of (8), we use the centeral difference or backward random oracle as $\hat{g}_\mu(x) = \frac{F(x+\mu u) - F(x-\mu u)}{2\mu} u$ or $\bar{g}_\mu(x) = \frac{F(x) - F(x-\mu u)}{\mu} u$, we can obtain the same bounds as in Theorem 1. This is because the upper bounds on the expectation of the square norm of all these oracles are the same for a Lipschitz continuous function.

B. Online Problem

In this section, we analyse the online minimisation of submodular functions using a ZO method. In the online submodular minimisation problem, over a sequence of iterations $k \in \mathbb{N}$,, an online decision maker repeatedly selects a subset $S_k \subseteq [n]$. After choosing S_k, the cost is determined by a submodular function $f_k : 2^{[n]} \to \mathbb{R}$, and the decision maker incurs the cost $f_k(S_k)$. We will analyse both the static regret and dynamic regret of Algorithm 1.

Definition 4 (Static Regret): The static regret of the decision maker is defined as:

$$\text{Regret}_N := \sum_{k=0}^{N} f_k(S_k) - \min_{S\subseteq[n]} \sum_{k=0}^{N} f_k(S).$$

If the sets S_k are chosen by a randomised algorithm, the expected regret over the randomness in the algorithm is considered. An online algorithm to choose the sets S_k is said to be Hannan-consistent if it ensures that $E[\text{Regret}_N] \leq o(N)$. Here, as considered in [21], we assume that the submodular cost function can change between two function evaluations that are needed to calculate (8). Thus we consider working with a family of submodular functions $f_k : 2^{[n]} \to \mathbb{R}$ where $k \in \mathbb{N} \cup \{j + \frac{1}{2} | j \in \mathbb{N}\} = \{0, 1/2, 1, 3/2, \dots\}$. For simplicity, we denote $k + \frac{1}{2}$ by k^+ for all $k \in \mathbb{N}$. Also, we denote the Lovász extension of f_k by f_k^L. We know that each function f_k^L is Lipschitz with $L_{0,k}$ as the Lipschitz constant. Next, We need to alter the random oracle defined in (8) to adapt it to this setting. As mentioned after (7) and before (8), we know that $\nabla f_{\mu,k}^L(x) = E_u[\frac{f_k^L(x+\mu u)}{\mu} u]$ and $E[u] = 0$, thus we can define

$$g_{\mu,k^+}(x) = \tfrac{1}{\mu}(f_k^L(x + \mu u) - f_{k^+}^L(x))u \tag{17}$$

with $\nabla f_{\mu,k}^L(x) = E_u[g_{\mu,k^+}(x)]$. Thus, to solve this problem, we consider Algorithm 1 with below update step

$$\bar{x}_{k+1} = x_k - h_k g_{\mu,k^+}(x_k). \tag{18}$$

Theorem 2: For $k \in \{0, \dots, N\}$, let $f_k : 2^{[n]} \to \mathbb{R}$ be submodular functions and f_k^L be their corresponding Lovász extension with x^* as a minimiser of $\sum_{k=0}^N f_k^L(x)$. Let $\bar{L}_0 = \max_k L_{0,k}$ be the Lipschitz constant, $N \geq 0$ be the number of iterations, $t_k = t \geq 1$ be the number of samples, $\mu > 0$ be the smoothing parameter in (7) and $\mathcal{U}_k = [u_0, u_1, , \cdots, u_k]$. Moreover, let $\{x_k\}_{k\geq 0}$ be the sequences generated by Algorithm 1 with (17) and (18) as the update step in Lines 5 and 6. Then, for any iteration N, with $h_k = h = \frac{n^{\frac{1}{2}}}{(N+1)^{\frac{1}{2}} \bar{L}_0 (n+4)}$ and $\mu \leq \frac{1}{\bar{L}_0 n^{\frac{1}{2}} (N+1)^{\frac{1}{2}}}$, we have

$$\sum_{k=0}^{N} E_{\mathcal{U}_k}[f_k^L(x_k)] - \min_{x\in\mathcal{K}} \sum_{k=0}^{N} f_k^L(x) \leq (N+1)^{\frac{1}{2}} \Big(1 + n^{\frac{1}{2}}(n+4)\bar{L}_0\Big). \tag{19}$$

Moreover, we get

$$E_{\tau,\mathcal{U}_{N-1}}[\text{Regret}_N] = O(n^{\frac{3}{2}}\sqrt{N}), \tag{20}$$

where τ is the threshold defined in Algorithm 1

Proof: Similar to the steps of the proof of Theorem 1, letting $r_k = \|x_k - x^*\|$, we have

$$E_{u_k}[r_{k+1}^2] \leq r_k^2 - 2h\langle \nabla f_{\mu,k}^L(x_k), x_k - x^* \rangle + h^2 \bar{L}_0^2 (n+4)^2 \tag{21}$$

Considering the convexity of f_k^L, which implies that $f_{\mu,k}^L$ is convex, and [18, Thm. 1], we have

$$f_k^L(x_k) - f_k^L(x^*) \leq \frac{r_k^2 - E_{u_k}[r_{k+1}^2]}{2h} + \mu \bar{L}_0 n^{\frac{1}{2}} + \frac{h\bar{L}_0^2(n+4)^2}{2}. \tag{22}$$

Considering the fact that $\|x_0 - x^*\|^2 \leq n$, taking expectation with respect to $\mathcal{U}_{k-1}$ from (22), summing from $k = 0$ to $k = N$, and substituting the given values for μ, h, and N, we arrive at (19).

Considering Lemma 1, $\sum_{k=0}^N f_k(S^*) = \sum_{k=0}^N f_k^L(x^*)$ where S^* is the minimiser of $\sum_{k=0}^N f_k(S)$. Also, considering the projection step in Algorithm 1, we know that $x_k \in [0,1]^n$ for all $k \in \{0, \dots, N\}$. Thus, considering Definition 2, we have $f_k^L(x_k) = E_\tau[f_k(S_k)]$. Moreover, we know that τ and u_k are independent random variables. Considering these facts and taking expectation from $f_k(S_k) - f_k(S^*)$ with respect to τ, taking expectation with respect to $\mathcal{U}_{k-1}$, summing from $k = 0$ to $k = N$, and considering (19), we arrive at (20). ■

Theorem 2 implies that Algorithm 1 with (17) and (18) as the update step in Lines 5 and 6, respectively, is Hannan-consistent. We should note that if $f_k = f_{k^+}$, we can arrive at the same results as in Theorem 2. This is because we would still have $\nabla f_{\mu,k}^L(x) = E_u[g_{\mu,k^+}(x)]$. In the next remark, we discuss what happens if, the random oracle is defined as

$$\tilde{g}_{\mu,k^+}(x) = \frac{f_{k^+}^L(x + \mu u) - f_k^L(x)}{\mu} u \tag{23}$$

instead of (17).

Remark 4: If in the update step for the online setting, (23) is used instead of (17), then we get $\nabla f_{\mu,k^+}^L(x) = E_u[\tilde{g}_{\mu,k^+}(x)]$. In this case, $\nabla f_{\mu,k^+}^L(x)$ is not an unbiased estimator of $\nabla f_{\mu,k}^L(x)$ and an extra assumption is necessary. In particular, we assume that $|f_k^L(x) - f_{k^+}^L(x)| \leq V$ for all x and where $V \in \mathbb{R}_{\geq 0}$ denotes a constant. This means that the change of the functions' values during the sampling period in each iteration is upper-bounded. With this assumption, we get the estimate

$$\sum_{k=0}^{N} E_{\mathcal{U}_k}[f_k^L(x_k)] - \min_{x\in\mathcal{K}} \sum_{k=0}^{N} f_k^L(x) \leq (N+1)^{\frac{1}{2}} \Big(1 + n^{\frac{1}{2}}(n+4)\bar{L}_0\Big) + 2V(N+1), \tag{24}$$

and $E_{\tau,\mathcal{U}_{N-1}}[\text{Regret}_N] = A + B$, where $A = O(n^{\frac{3}{2}}\sqrt{N})$ and $B = O(NV)$.

Next, we consider tracking the minimum value of each of f_k. To this end, we need to define the dynamic regret.

Definition 5 (Dynamic Regret): The dynamic regret of the decision maker is defined as:

$$\text{D-Regret}_N := \sum_{k=0}^{N} f_k(S_k) - \sum_{k=0}^{N} \min_{S\subseteq[n]} f_k(S).$$

If the sets $S_k \subseteq 2^{[n]}$ are chosen by a randomised algorithm, the expected regret over the randomness in the algorithm is considered. For the analysis of the dynamic regret, following [22], [23], we need to define the total path length

$$P_N(u_0, \dots, u_N) = \sum_{i=1}^{N} \|u_i - u_{i-1}\|, \tag{25}$$

which allows us to present our main result for the dynamic regret, corresponding to the sequence generated by Algorithm 1 with (17) and (18) as the update steps in Lines 5 and 6.

Theorem 3: For $k \in \{0, \dots, N\}$, let $f_k : 2^{[n]} \to \mathbb{R}$ be submodular functions and f_k^L be their corresponding Lovász extension with minimiser x_k^*. Let $\bar{L}_0 = \max_k L_{0,k}$ be the common Lipschitz constant of f_k^L, $N \geq 0$ be the number of iterations, $t_k = t \geq 1$ be the number of samples, $\mu > 0$ be the smoothing parameter in (7) and $\mathcal{U}_k = [u_0, u_1, \cdots, u_k]$, $k \in [N]$. Moreover, let $\{x_k\}_{k\geq 0}$ be the sequences generated by Algorithm 1 with (17) and (18) as the update step in Lines 5 and 6. Then, for any iteration N, with $h_k = h = \frac{(n+3\sqrt{n}P_N^*)^{\frac{1}{2}}}{(N+1)^{\frac{1}{2}} L_0(n+4)}$ and $\mu \leq \frac{1}{L_0 n^{\frac{1}{2}} (N+1)^{\frac{1}{2}}}$, we have

$$\begin{aligned} &\sum_{k=0}^{N} E_{\mathcal{U}_k}[f_k^L(x_k)] - \sum_{k=0}^{N} \min_{x\in\mathcal{K}} f_k^L(x) \\ &\quad \leq (N+1)^{\frac{1}{2}} \left(1 + (n+4)\bar{L}_0(n+3\sqrt{n}P_N^*)^{\frac{1}{2}}\right), \end{aligned} \tag{26}$$

where $P_N^* = P_N(x_0^*, \dots, x_N^*)$. Moreover, it holds that

$$E_{\tau,\mathcal{U}_{N-1}}[\text{Regret}_N] = O(n^{\frac{3}{2}}\sqrt{NP_N^*}), \tag{27}$$

where τ is the threshold defined in Algorithm 1

Proof: Let $e_k = \|x_k - x_k^*\|$. Then, we have

$$\begin{aligned} e_{k+1} =& \|x_{k+1} - x_{k+1}^* + x_k^* - x_k^*\| \\ \leq& \|x_{k+1} - x_k^*\| + \|x_{k+1}^* - x_k^*\| \end{aligned}$$

and with $\|x_{k+1} - x_k^*\| \leq \sqrt{n}$, it holds that

$$e_{k+1}^2 \leq \|x_{k+1} - x_k^*\|^2 + \|x_{k+1}^* - x_k^*\|^2 + 2\sqrt{n}\|x_{k+1}^* - x_k^*\|. \tag{28}$$

Using the non-expansiveness property of projections on convex sets, we have

$$\begin{aligned} \|x_{k+1} - x_k^*\|^2 \leq& \|\bar{x}_{k+1} - x_k^*\|^2 \leq \|x_k - x_k^* - h g_{\mu,k^+}(x_k)\|^2 \\ =& e_k^2 - 2h\langle g_{\mu,k^+}(x_k), x_k - x_k^*\rangle + h^2\|g_{\mu,k^+}(x_k)\|^2. \end{aligned}$$

Then, considering (28), the estimate

$$\begin{aligned} e_{k+1}^2 \leq& e_k^2 - 2h\langle g_{\mu,k^+}(x_k), x_k - x^*\rangle + h^2\|g_{\mu,k^+}(x_k)\|^2 \\ &+ \|x_{k+1}^* - x_k^*\|^2 + 2\sqrt{n}\|x_{k+1}^* - x_k^*\| \end{aligned} \tag{29}$$

is obtained. Taking expectation of (29) with respect to u_k and considering using similar steps as in the proof of Theorem 1, we have

$$\begin{aligned} E_{u_k}[e_{k+1}^2] \leq& e_k^2 - 2h\langle \nabla f_{\mu,k}^L(x_k), x_k - x_k^*\rangle + h^2 L_0^2 (n+4)^2 \\ &+ \|x_{k+1}^* - x_k^*\|^2 + 2\sqrt{n}\|x_{k+1}^* - x_k^*\| \end{aligned}$$

With the convexity of f_k^L, which implies that $f_{\mu,k}^L$ is also convex, and [18, Thm 1], we have

$$\begin{aligned} f_k^L(x_k) - f_k^L(x_k^*) \leq& \frac{e_k^2 - E_{u_k}[e_{k+1}^2]}{2h} + \mu L_0 n^{\frac{1}{2}} \\ &+ \frac{hL_0^2(n+4)^2}{2} + \frac{\sqrt{n}}{2h}\|x_{k+1}^* - x_k^*\| + \frac{\sqrt{n}}{h}\|x_{k+1}^* - x_k^*\|. \end{aligned} \tag{30}$$

Considering the fact that $\|x_0 - x_0^*\|^2 \leq n$, taking the expectation with respect to $\mathcal{U}_{k-1}$ from (30), summing from $k = 0$ to $k = N$, and substituting the given values for μ and h, we arrive at (26).

From Lemma 1 we know that $f_k(S_k^*) = f_k^L(x_k^*)$ and where S_k^* is the minimiser of $f_k(S)$. Considering the projection step in Algorithm 1, we know that $x_k \in [0,1]^n$ for all k. Thus, from Remark 1, we have that $f_k^L(x_k) = E_\tau[f_k(S_k)]$. Now, recall that τ and u_k are independent random variables. Combining these facts and taking the expectation from $f_k(S_k) - f_k(S^*)$ with respect to τ, taking the expectation with respect to $\mathcal{U}_{k-1}$, summing from $k = 0$ to $k = N$, and considering (26), we arrive at (27). ∎

In the next section, we will demonstrate the theoretical findings through a numerical example.

IV. Numerical Example

In this section, we will illustrate the theoretical results given in Section III through three numerical examples[1].

A. Semi-Supervised Clustering

In this section, we solve the problem of semi-supervised clustering on a two-moon dataset. We consider the cost function defined in [25, Sec. 6.5]. Given $p \in \mathbb{N}$ data points in a certain set V, we intend to select set $A \subset V$, which minimises the cost function.

$$Cost(A) = I(f_A, f_{V\setminus A}) - \sum_{k\in A} \log\ \eta_k - \sum_{k\in V\setminus A} \log(1-\eta_k).$$

Here, f_A and $f_{V\setminus A}$ are two Gaussian processes with zero mean and covariance matrices K_{AA} and $K_{V\setminus AV\setminus A}$,

$$\begin{aligned} I(f_A, f_{V\setminus A}) = \tfrac{1}{2}\big(&\text{logdet}(K_{AA}) + \text{logdet}(K_{V\setminus AV\setminus A}) \\ &- \text{logdet}(K_{VV})\big) \end{aligned}$$

is the mutual information between the Gaussian processes, and η_k is the probability of each element to be in set A. For more information on this problem, we refer the reader to [25, Sec. 6.5]. We consider each data point $x_i \in \mathbb{R}^2$ and sample each data point from the "two-moons" dataset [25]. We consider 50 points and 8 randomly chosen labelled points, where we impose $\eta_k \in \{0, 1\}$ for them and $\eta_k = \frac{1}{2}$ for the rest. We consider the Gaussian radial basis function kernel $k(x, y) = \exp(\frac{-\|x-y\|^2}{2\sigma^2})$ with $\sigma^2 = 0.05$ to obtain covariance matrices. We use Algorithm 1 to find a minimiser of the cost function and we define the parameters $\mu = 10^{-5}$, $h = 10^{-4}$, and $N = 4000$ for the simulation. Moreover, for comparison, we implement the subgradient

[1]The Python code is publicly available at https://github.com/amirali78frz/Minimisation_projects.git.

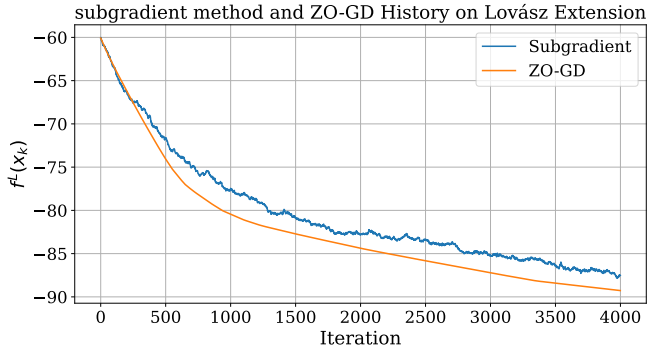

Fig. 1. Lovász Extension History Over the Sequence Generated by Algorithm 1 and the Subgradient Method for Offline Clustering

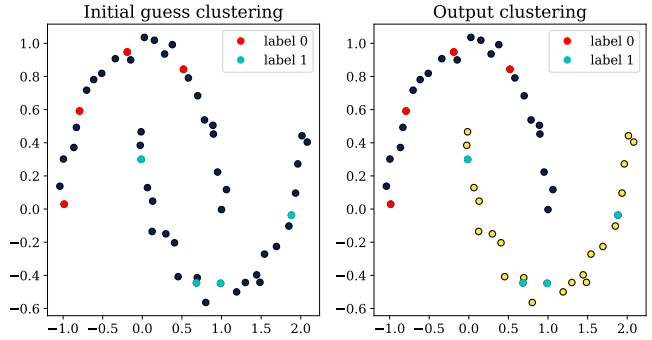

Fig. 2. Offline Clustering of the Two Moon Dataset Using the Sequence Generated by Algorithm 1

method described in [25, Sec. 10.8]. In Figure 1, we can see the Lovász extension value over the sequence generated by the subgradient method and Algorithm 1. We can see that the ZO method has a similar performance as the subgradient method in this scenario. Although we should note that the subgradient method is at least twice as fast as the ZO method. This is due to the fact that the calculation of the calculate the subgradient of the Lovász Extension requires n function queries and for the calculation of the random oracle (8), we need $2n$ function queries. Despite this drawback in terms of the function queries, in Figure 2, we see that Algorithm 1 successfully recovers the clustering from a random initial condition.

B. Approximating Lovász Extension

As discussed in the previous section, if we calculate the Lovász extension using (9), theoretically it can be seen that the subgradient method is at least twice as fast as Algorithm 1. In this section we investigate how to calculate the Lovász extension faster, i.e., in less than $2n$ queries of the submodular function. Here, we consider the cost function $\text{cost}(A) = \text{logdet}(K_{AA})$, where K_{AA} is defined in Section IV-A. We implement Algorithm 1 and the subgradient method, as a comparison. We implement both algorithms using the exact and low-rank approximations of the matrix K_{VV}. We calculate the lower-rank approximation using the Nyström method [28]. Moreover, to evaluate the Lovász

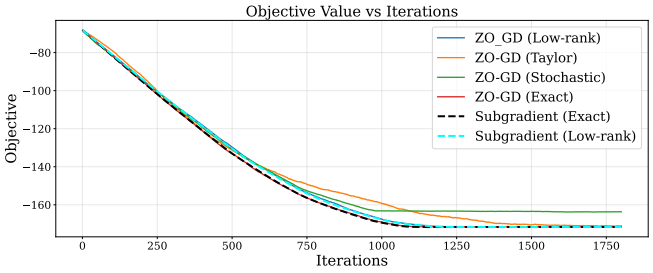

Fig. 3. Logdet Cost Function's Lovász Extension History Over Iterations

extension in Algorithm 1, we explore two different methods based on the stochastic Lovász extension and a surrogate Lovász extension obtained through the Taylor series approximation of $\text{logdet}(K_{AA})$. To reduce the computational cost of the Lovász extension, we adopt a stochastic sampling strategy. Instead of evaluating the function on all n components, we randomly select a subset of components of x according to a fixed sampling ratio $\rho \in (0, 1)$. For a random permutation π of $[n]$ and sorted weights $x_{\pi(1)} \geq \cdots \geq x_{\pi(n)}$, the Lovász extension is approximated by

$$\tilde{f}^L_{\text{stoch}}(x) = \sum_{i \in \mathcal{I}_\rho} \big(x_{\pi(i)} - x_{\pi(i+1)}\big)\, f\big(S_i\big),$$

where $\mathcal{I}_\rho \subset [n]$ is the sampled index set and $S_i = \{\pi(1), \ldots, \pi(i)\}$. This provides a noisy but cheaper estimate of the Lovász extension. In the Taylor surrogate method, we construct a smooth surrogate of $\log\det(K_{AA})$ using a second-order Taylor expansion around the identity. Given $w \in [0, 1]^n$, we define the weighted kernel $K_x = \text{diag}(\sqrt{x})\, K\, \text{diag}(\sqrt{x})$, and set $E = K_x - I$. Using the expansion

$$\log\det(I + E) \approx \text{tr}(E) - \tfrac{1}{2}\text{tr}(E^2),$$

the surrogate objective is

$$\tilde{f}^L_{\text{Taylor}}(x) = \text{tr}(K_x - I) - \tfrac{1}{2}\text{tr}\big((K_x - I)^2\big),$$

which avoids combinatorial evaluation and provides a differentiable approximation. We implement the mentioned methods with $N = 1800$ and step size $h = 10^{-4}$. The Lovász extension value versus number of iterations and wall-clock time is given in Figures 3 and 4. As can be seen, the performance of all the methods is similar in terms of finding the minimum value. Time-wise, we can see that using Taylor's surrogate method, the ZO method can achieve a similar runtime as the subgradient methods.

As it can been in Section IV-A, the cost function of the semi-supervised clustering example is a combination of logdet functions. Thus, we investigate the above approximations in solving the semi-supervised clustering problem. In Figure 5, we can see that from a random initial condition, we could successfully cluster the data using the subgradient method and using Algorithm 1 with exact, Taylor surrogate, and low rank queries of the Lovász extension. We should note that the subgradient method using lower rank approximation could not recover the true clusters for

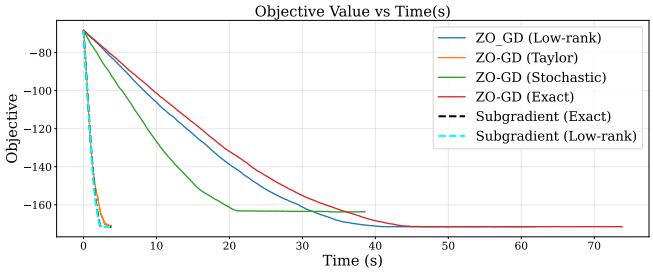

Fig. 4. Logdet Cost Function's Lovász Extension History Over Time

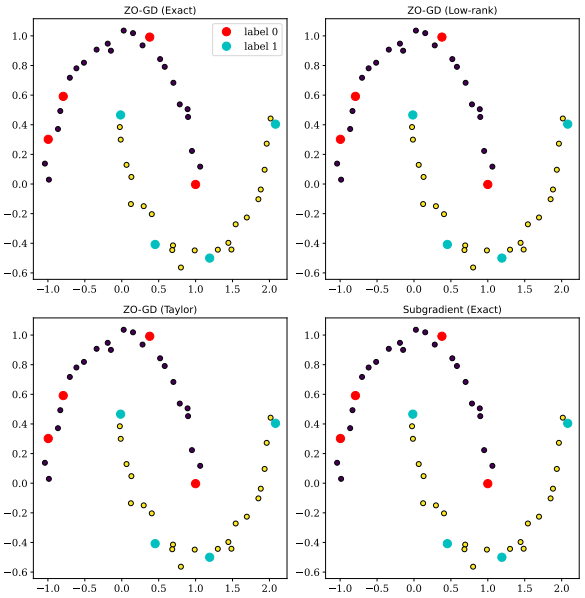

Fig. 5. Offline Clustering of the Two Moon Dataset Using the Subgradient Method and Algorithm 1 with Exact and Approximate Lovász Extension Queries

this particular example. Moreover, in Figure 6, we can see the Lovász extension value over the generated sequence by the mentioned methods versus. It can be seen that using the Taylor surrogate for querying the Lovász extension, is 90% faster compared to implementation of the exact Lovász extension.

C. Online Semi-Supervised Clustering

In this section, we solve the online semi-supervised clustering problem where each of the true clusters is moving with a different trajectory in each iteration. Thus, K_{VV}, defined in Section IV-A, changes constantly, leading to an online problem. In this section, we solve the problem using the subgradient method and Algorithm 1 with the Taylor surrogate and exact queries of the Lovász extension. The Lovász extension value versus iteration and time is given in Figures 7 and 8. Moreover, the final clustering is given in Figure 9 and the clustering and trajectory of the points over iterationss are given in Figure 10. As it can be seen, all three methods have successfully completed the online clustering task while using the Taylor surrogate instead of the exact Lovász extension can make Algorithm 1 up to 95% faster.

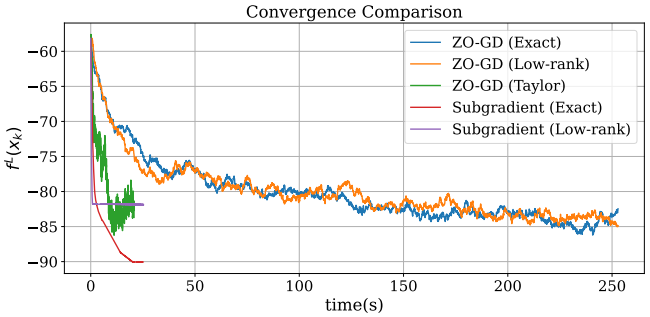

Fig. 6. Lovász Extension History Over the Sequence Generated by Subgradient Method and Algorithm 1 with Exact and Approximate Queries for Offline Clustering

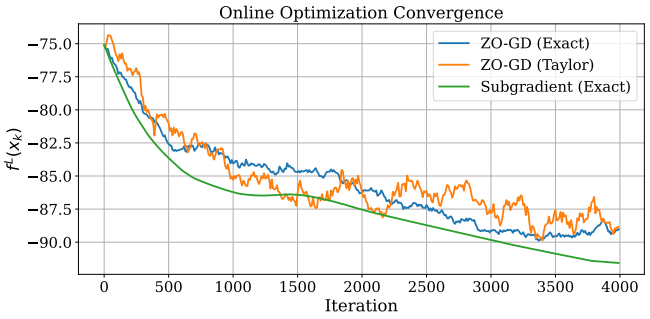

Fig. 7. Lovász Extension History Over iterations for Online Clustering

V. Conclusions and Future Work

In this study, we considered the submodular function minimisation problem in both offline and online settings. We considered a Gaussian smoothing ZO method and investigated its performance in solving the SFM problem. First, we considered the offline SFM and showed that the framework, in the expectation sense, converges to an ϵ-approximate solution in $O(n^2\epsilon^{-2})$ function calls. Then, we considered the online SFM under both static and dynamic regrets. We showed that, in the expectation sense, the algorithm is Hannan-consistent in terms of static regret, and achieves a dynamic regret of $O(\sqrt{NP_N^*})$. Numerical examples demonstrating the theoretical results were presented in the end. A possible future research direction is to investigate the use of different surrogates of Lovász extension in the implementation of ZO

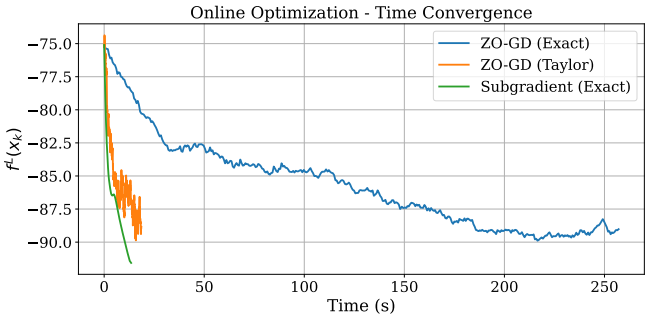

Fig. 8. Lovász Extension History Over Time for Online Clustering

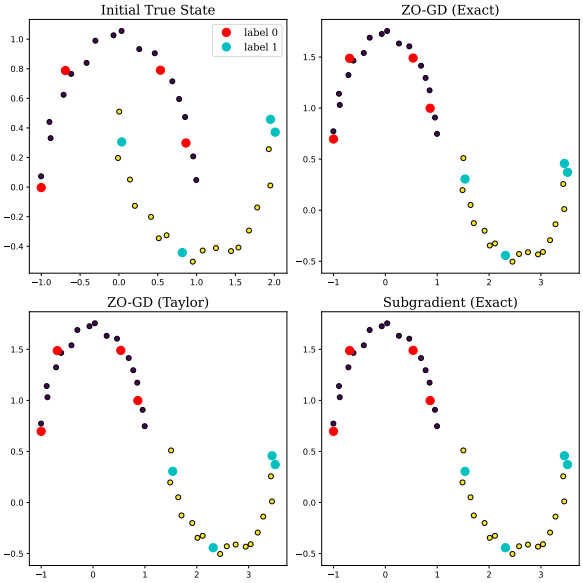

Fig. 9. Online Clustering of the Two Moon Dataset Using the Subgradient Method and Algorithm 1 with Exact and Approximate Lovász Extension Queries

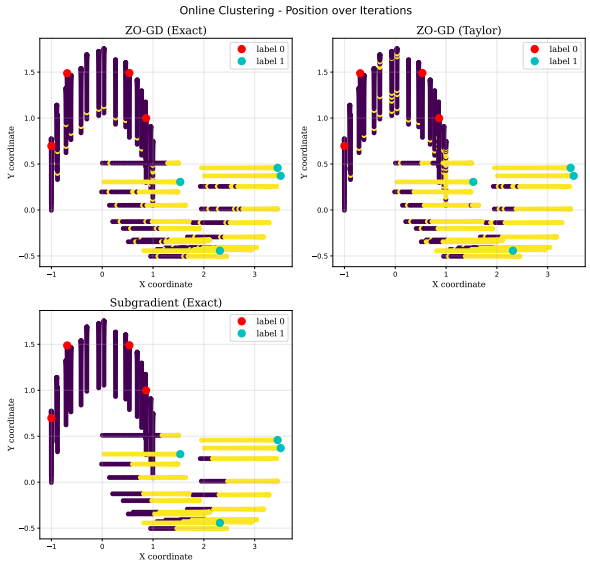

Fig. 10. Positions and Labels Over Iterations for Online Clustering

algorithms. Another interesting possible future direction is to study the performance of ZO methods in solving minimax problems by leveraging the submodularity structure.

REFERENCES

[1] J. Edmonds, "Submodular functions, matroids, and certain polyhedra," in *Combinatorial Structures and Their Application, pages 68–87*, pp. 11–26, Springer, 1970.

[2] A. Krause and V. Cevher, "Submodular dictionary selection for sparse representation," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 567–574, 2010.

[3] D. M. Topkis, *Supermodularity and complementarity*. Princeton university press, 1998.

[4] S. Jegelka, F. Bach, and S. Sra, "Reflection methods for user-friendly submodular optimization," *Advances in Neural Information Processing Systems*, vol. 26, 2013.

[5] H. Lin and J. A. Bilmes, "Optimal selection of limited vocabulary speech corpora.," in *INTERSPEECH*, pp. 1489–1492, 2011.

[6] S. Iwata, "Submodular function minimization," *Mathematical Programming*, vol. 112, no. 1, pp. 45–64, 2008.

[7] S. T. McCormick, "Submodular function minimization," *Handbooks in operations research and management science*, vol. 12, pp. 321–391, 2005.

[8] M. Grötschel, L. Lovász, and A. Schrijver, "The ellipsoid method and its consequences in combinatorial optimization," *Combinatorica*, vol. 1, no. 2, pp. 169–197, 1981.

[9] D. Chakrabarty, Y. T. Lee, A. Sidford, and S. C.-w. Wong, "Subquadratic submodular function minimization," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 1220–1231, 2017.

[10] D. Chakrabarty, P. Jain, and P. Kothari, "Provable submodular minimization using wolfe's algorithm," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[11] B. Axelrod, Y. P. Liu, and A. Sidford, "Near-optimal approximate discrete and continuous submodular function minimization," in *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 837–853, SIAM, 2020.

[12] L. Lovász, "Submodular functions and convexity," in *Mathematical Programming The State of the Art: Bonn 1982*, pp. 235–257, Springer, 1983.

[13] Y. Hamoudi, P. Rebentrost, A. Rosmanis, and M. Santha, "Quantum and classical algorithms for approximate submodular function minimization," *arXiv preprint arXiv:1907.05378*, 2019.

[14] E. Hazan and S. Kale, "Online submodular minimization," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 2903–2922, 2012.

[15] S. Jegelka and J. A. Bilmes, "Online submodular minimization for combinatorial structures.," in *ICML*, pp. 345–352, 2011.

[16] A. R. Cardoso and R. Cummings, "Differentially private online submodular minimization," in *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1650–1658, PMLR, 2019.

[17] C. Audet and W. Hare, "Introduction: tools and challenges in derivative-free and blackbox optimization," in *Derivative-Free and Blackbox Optimization*, pp. 3–14, Springer, 2017.

[18] Y. Nesterov and V. Spokoiny, "Random gradient-free minimization of convex functions," *Foundations of Computational Mathematics*, vol. 17, no. 2, pp. 527–566, 2017.

[19] A. A. Farzin and I. Shames, "Minimisation of polyak-łojasewicz functions using random zeroth-order oracles," in *2024 European Control Conference (ECC)*, pp. 3207–3212, IEEE, 2024.

[20] A. A. Farzin, Y.-M. Pun, and I. Shames, "Minimisation of quasar-convex functions using random zeroth-order oracles," *arXiv preprint arXiv:2505.02281*, 2025.

[21] I. Shames, D. Selvaratnam, and J. H. Manton, "Online optimization using zeroth order oracles," *IEEE Control Systems Letters*, vol. 4, no. 1, pp. 31–36, 2019.

[22] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proceedings of the 20th international conference on machine learning (icml-03)*, pp. 928–936, 2003.

[23] E. Hazan *et al.*, "Introduction to online convex optimization," *Foundations and Trends® in Optimization*, vol. 2, no. 3-4, pp. 157–325, 2016.

[24] S. Fujishige, *Submodular functions and optimization*, vol. 58. Elsevier, 2005.

[25] F. Bach *et al.*, "Learning with submodular functions: A convex optimization perspective," *Foundations and Trends® in machine learning*, vol. 6, no. 2-3, pp. 145–373, 2013.

[26] K. Ando, "K-submodular functions and convexity of their Lovász extension," *Discrete Applied Mathematics*, vol. 122, no. 1-3, pp. 1–12, 2002.

[27] A. A. Farzin, Y.-M. Pun, P. Braun, A. Lesage-Landry, Y. Diouane, and I. Shames, "Min-max optimisation for nonconvex-nonconcave functions using a random zeroth-order extragradient algorithm," *Transactions on Machine Learning Research*, 2025.

[28] C. Williams and M. Seeger, "Using the nyström method to speed up kernel machines," *Advances in neural information processing systems*, vol. 13, 2000.