Online Risk-Averse Submodular Maximization

Tasuku Soma[1,2] , **Yuichi Yoshida**[3]
[1] The University of Tokyo
[2] Massachusetts Institute of Technology
[3] National Institute of Informatics
tasuku_soma@mist.i.u-tokyo.ac.jp, tasuku@mit.edu, yyoshida@nii.ac.jp

Abstract

We present a polynomial-time online algorithm for maximizing the conditional value at risk (CVaR) of a monotone stochastic submodular function. Given T i.i.d. samples from an underlying distribution arriving online, our algorithm produces a sequence of solutions that converges to a $(1-1/e)$-approximate solution with a convergence rate of $O(T^{-1/4})$ for monotone continuous DR-submodular functions. Compared with previous offline algorithms, which require $\Omega(T)$ space, our online algorithm only requires $O(\sqrt{T})$ space. We extend our online algorithm to portfolio optimization for monotone submodular set functions under a matroid constraint. Experiments conducted on real-world datasets demonstrate that our algorithm can rapidly achieve CVaRs that are comparable to those obtained by existing offline algorithms.

1 Introduction

Submodular function maximization is a ubiquitous problem naturally arising in broad areas such as machine learning, social network analysis, economics, combinatorial optimization, and decision-making [Krause and Golovin, 2014; Buchbinder and Feldman, 2018]. Submodular maximization in these applications is *stochastic* in nature, i.e., the input data could be a sequence of samples drawn from some underlying distribution, or there could be some uncertainty in the environment. In this paper, we consider nonnegative monotone *continuous DR-submodular*[1] [Bian *et al.*, 2017] functions $F(\boldsymbol{x}; z)$ parameterized by a random variable z drawn from some distribution $\mathcal{D}$. The simplest approach for such stochastic submodular objectives is to maximize the expectation $\mathbf{E}_{z\sim\mathcal{D}}[F(\boldsymbol{x}; z)]$, which has been extensively studied [Karimi *et al.*, 2017; Hassani *et al.*, 2017; Mokhtari *et al.*, 2018; Karbasi *et al.*, 2019].

However, in real-world decision-making tasks in finance, robotics, and medicine, we sometimes must be *risk-averse*: We want to minimize the risk of suffering a considerably small gain rather than simply maximizing the expected gain [Mansini *et al.*, 2007; Yau *et al.*, 2011; Tamar *et al.*, 2015]. In medical applications, for example, we must avoid catastrophic events such as patient fatalities. In finance and robotics, all progress ceases when poor decisions cause bankruptcies or irreversible damage to robots.

Conditional value at risk (CVaR) is a popular objective for such risk-averse domains [Rockafellar and others, 2000; Krokhmal *et al.*, 2002]. Formally, given a parameter $\alpha \in [0, 1]$, the CVaR of a feasible solution $\boldsymbol{x} \in \mathbb{R}^n$ is defined as

$$\mathrm{CVaR}_{\alpha,\mathcal{D}}(\boldsymbol{x}) = \mathop{\mathbf{E}}_{z\sim\mathcal{D}}[F(\boldsymbol{x}; z) \mid F(\boldsymbol{x}; z) \leq \mathrm{VaR}_{\alpha,\mathcal{D}}(\boldsymbol{x})],$$

where $\mathrm{VaR}_{\alpha,\mathcal{D}}(X)$ is the α-quantile of the random variable $F(\boldsymbol{x}; z)$, i.e.,

$$\mathrm{VaR}_{\alpha,\mathcal{D}}(X) = \sup\left\{\tau \in \mathbb{R} : \Pr_{z\sim\mathcal{D}}(F(\boldsymbol{x}; z) \leq \tau) \leq \alpha\right\}.$$

Note that when $\alpha = 1$, the CVaR is simply the expected value of $F(\boldsymbol{x}; z)$. Since α is typically set to be 0.10 or 0.05 in practice, we assume that α is a fixed constant throughout this paper. When α is clear from the context, we omit it from the notations. CVaR can also be characterized by a variational formula:

$$\mathrm{CVaR}_{\mathcal{D}}(\boldsymbol{x}) = \max_{\tau\in[0,1]} \tau - \frac{1}{\alpha} \mathop{\mathbf{E}}_{z\sim\mathcal{D}} [\tau - F(\boldsymbol{x}; z)]_+,$$

where $[\cdot]_+ = \max\{\cdot, 0\}$ [Rockafellar and others, 2000].

[Maehara, 2015] initiated the study of maximizing the CVaR of stochastic submodular *set* functions. It was shown that the CVaR of a stochastic submodular set function is not necessarily submodular, and that it is impossible to compute a single set that attains any multiplicative approximation to the optimal CVaR. [Ohsaka and Yoshida, 2017] introduced a relaxed problem of finding a portfolio over sets rather than a single set, and devised the first CVaR maximization algorithm with an approximation guarantee for the influence maximization problem [Kempe *et al.*, 2003], a prominent example of discrete submodular maximization. [Wilder, 2018] further considered this approach and devised an algorithm called RASCAL for maximizing the CVaR of continuous DR-submodular functions subject to a down-closed convex set.

The algorithms mentioned above are *offline (batch) methods*, i.e., a set of samples drawn from the underlying distribution $\mathcal{D}$ is given as the input. However, because the number of

[1] See Section 2 for the definition of continuous DR-submodularity.

samples needed to accurately represent $\mathcal{D}$ can be exceedingly large, it is often inefficient or even impossible to store all the samples in memory. Further, when a new sample is observed, these offline algorithms must be rerun from scratch.

In the machine learning community, *online methods*, which can efficiently handle large volumes of data, have been extensively studied [Hazan, 2016]. Online methods read data in a streaming fashion, and update the current solution using a few data elements stored in memory. Further, online methods can update a solution at a low computational cost.

1.1 Our Contributions

In this work, we propose *online algorithms* for maximizing the CVaR of stochastic submodular objectives in continuous and discrete settings.

Continuous setting. Let $z_1, \dots, z_T$ be i.i.d. samples drawn from $\mathcal{D}$ arriving sequentially and $K \subseteq \mathbb{R}^n$ be a down-closed convex set. Our main result is a polynomial-time online algorithm, StochasticRASCAL, that finds $\boldsymbol{x} \in K$ such that

$$\mathbf{E}[\mathrm{CVaR}_{\mathcal{D}}(\boldsymbol{x})] \geq \left(1 - \frac{1}{e}\right) \mathrm{CVaR}_{\mathcal{D}}(\boldsymbol{x}^*) - O(T^{-1/4}),$$

for any $\boldsymbol{x}^* \in K$, where the expectation is taken over $z_1, \dots, z_T$ and the randomness of the algorithm. StochasticRASCAL only stores $O(\sqrt{T})$ data in memory, drawing a contrast to RASCAL, which store all T data points. Note that the approximation ratio of $1-1/e$ is optimal for any algorithm that performs polynomially many function value queries even if $\alpha = 1$ [Vondrák, 2013]. We also conduct several experiments on real-world datasets to show the practical efficiency of our algorithm. We demonstrate that our algorithm rapidly achieves CVaR comparable to that obtained by known offline methods.

Discrete setting. As an application of the above algorithm, we devise an online algorithm to create a portfolio that maximizes the CVaR of a discrete submodular function subject to a matroid constraint. Let $f(X; z) : 2^V \to [0, 1]$ be a monotone stochastic submodular set function on a ground set V and $\mathcal{I} \subseteq 2^V$ be a matroid. The goal is to find a portfolio $\mathcal{P}$ of feasible sets in $\mathcal{I}$ that maximizes

$$\mathrm{CVaR}_{\mathcal{D}}(\mathcal{P}) = \max_{\tau \in [0,1]} \tau - \frac{1}{\alpha} \mathop{\mathbf{E}}_{z \sim \mathcal{D}} [\tau - \mathop{\mathbf{E}}_{X \sim \mathcal{P}} f(X; z)]_+,$$

given i.i.d. samples from $\mathcal{D}$. We show that this problem can be reduced to online CVaR maximization of continuous DR-submodular functions on a matroid polytope, and devise a polynomial-time online approximation algorithm for creating a portfolio $\mathcal{P}$ such that

$$\mathbf{E}[\mathrm{CVaR}_{\mathcal{D}}(\mathcal{P})] \geq \left(1 - \frac{1}{e}\right) \mathrm{CVaR}_{\mathcal{D}}(\mathcal{P}^*) - O(T^{-1/10})$$

for any portfolio $\mathcal{P}^*$ over feasible sets. Note that this algorithm is the first online algorithm that converges to a $(1 - 1/e)$-approximation portfolio in the discrete setting, which generalizes the known offline algorithm [Wilder, 2018] to the i.i.d. setting.

1.2 Our Techniques

To analyze our online algorithms, we introduce a novel adversarial online learning problem, which we call *adversarial online submodular CVaR learning*. This online learning problem is described as follows. For $t = 1, \dots, T$, the learner chooses $\boldsymbol{x}_t \in K$ and $\tau_t \in [0, 1]$ possibly in a randomized manner. After $\boldsymbol{x}_t$ and τ_t are chosen, the adversary reveals a monotone continuous DR-submodular function $F_t : K \to [0, 1]$ to the learner. The goal of the learner is to minimize the approximate regret

$$\mathrm{regret}_{1-1/e}(T) = \left(1 - \frac{1}{e}\right) \sum_{t=1}^{T} H_t(\boldsymbol{x}^*, \tau^*) - \sum_{t=1}^{T} H_t(\boldsymbol{x}_t, \tau_t)$$

for arbitrary $\boldsymbol{x}^* \in K$ and $\tau^* \in [0, 1]$, where the function H_t is given by

$$H_t(\boldsymbol{x}, \tau) = \tau - \frac{1}{\alpha}[\tau - F_t(\boldsymbol{x})]_+.$$

We devise an efficient algorithm that achieves $O(T^{3/4})$ approximate regret in expectation. Further, we show that, given an online algorithm with a sublinear approximate regret, we can construct an online algorithm that achieves a $(1 - 1/e)$-approximation to CVaR maximization, whose convergence rate is $\mathbf{E}[\mathrm{regret}_{1-1/e}(T)]/T$. Combining these results, we obtain an online $(1-1/e)$-approximation algorithm for CVaR maximization with a convergence rate of $O(T^{-1/4})$.

We remark that adversarial online submodular CVaR learning may be of interest in its own right: Although the objective function H_t is neither monotone nor continuous DR-submodular in general, we can design an online algorithm with a sublinear $(1 - 1/e)$-regret by exploiting the underlying structure of H_t. As per our knowledge, an online algorithm for non-monotone and non-DR-submodular maximization does not exist in the literature.

1.3 Related Work

Several studies focused on CVaR optimization in the adversarial online settings and i.i.d. settings. [Tamar *et al.*, 2015] studied CVaR optimization over i.i.d. samples and analyzed stochastic gradient descent under the strong assumption that CVaR is continuously differentiable. Recently, [Cardoso and Xu, 2019] introduced the concept of the CVaR regret for convex loss functions and provided online algorithms for minimizing the CVaR regret under bandit feedback.

Online and stochastic optimization of submodular maximization have been extensively studied in [Streeter and Golovin, 2008; Streeter *et al.*, 2009; Golovin *et al.*, 2014; Karimi *et al.*, 2017; Hassani *et al.*, 2017; Mokhtari *et al.*, 2018; Chen *et al.*, 2018; Roughgarden and Wang, 2018; Soma, 2019; Karbasi *et al.*, 2019; Zhang *et al.*, 2019]. These studies optimize either the approximate regret or the expectation and do not consider CVaR.

Another line of related work is *robust submodular maximization* [Krause *et al.*, 2008; Chen *et al.*, 2017; Anari *et al.*, 2019]. In robust submodular maximization, we maximize the minimum of N submodular functions, i.e., $\min_{i=1}^{N} f_i(X)$. Robust submodular maximization is the limit of CVaR maximization, where $\mathcal{D}$ is the uniform distribution over N values

and $\alpha \to 0$. Recently, [Staib *et al.*, 2019] proposed *distributionally robust submodular optimization*, which maximizes $\min_{\mathcal{D}\in P} \mathbf{E}_{z\sim\mathcal{D}} f(X; z)$ for an uncertainty set P of distributions. It is known that CVaR can be formulated in the distributionally robust framework [Shapiro *et al.*, 2014]. However, the algorithms proposed by [Staib *et al.*, 2019] require that P is a subset of the N-dimensional probability simplex; moreover, their time complexity depends on N. Our algorithms work even if N is infinite.

1.4 Organization of This Paper

This paper is organized as follows. Section 2 introduces the background of submodular optimization. Sections 3 and 4 describe our algorithms for continuous and discrete setting, respectively. Section 5 present experimental results using real-world dataset. The omitted analysis and the details of adversarial setting can be found in the full version.

2 Preliminaries

Throughout the paper, V denotes the ground set and n denotes the size of the ground set. For a set function $f : 2^V \to \mathbb{R}$, the *multilinear extension* $F : [0,1]^V \to \mathbb{R}$ is defined as $F(\boldsymbol{x}) = \sum_{S\subseteq V} f(S) \prod_{i\in S} x_i \prod_{i\notin S}(1 - x_i)$. For a matroid on V, the *base polytope* is the convex hull of bases of the matroid. It is well-known that the linear optimization on a base polytope can be solved by the greedy algorithm [Fujishige, 2005].

We denote the Euclidean norm and inner product by $\|\cdot\|$ and $\langle\cdot,\cdot\rangle$, respectively. The ℓ^p norm ($1 \le p \le \infty$) is denoted by $\|\cdot\|_p$. The Euclidean projection of $\boldsymbol{x}$ onto a set K is denoted by $\mathrm{proj}_K(\boldsymbol{x})$. A convex set $K \subseteq \mathbb{R}^n_{\ge 0}$ is said to be *down-closed* if $\boldsymbol{y} \in K$ and $\mathbf{0} \le \boldsymbol{x} \le \boldsymbol{y}$ imply $\boldsymbol{x} \in K$. A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be *L-Lipschitz (continuous)* for $L > 0$ if $|f(\boldsymbol{x}) - f(\boldsymbol{y})| \le L\|\boldsymbol{x} - \boldsymbol{y}\|$ for all $\boldsymbol{x}, \boldsymbol{y}$. We say that f is *β-smooth* for $\beta > 0$ if f is continuously differentiable and $\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| \le \beta\|\boldsymbol{x} - \boldsymbol{y}\|$. A smooth function $F : \mathbb{R}^n \to \mathbb{R}$ is said to be *continuous DR-submodular* [Bian *et al.*, 2017] if $\frac{\partial^2 F}{\partial x_i \partial x_j} \le 0$ for all i, j. The multilinear extension of a submodular function is known to be DR-submodular [Calinescu *et al.*, 2011]. The continuous DR-submodularity implies *up-concavity*: For a continuous DR-submodular function F, $\boldsymbol{x} \in \mathbb{R}^n$, and $\boldsymbol{d} \ge \mathbf{0}$, the univariate function $t \mapsto F(\boldsymbol{x} + t\boldsymbol{d})$ is concave.

The uniform distribution of a set K is denoted by $\mathrm{Unif}(K)$. The standard normal distribution is denoted by $N(\mathbf{0}, I)$.

3 CVaR Maximization of Continuous DR-submodular Functions

We present our online algorithm for CVaR maximization via i.i.d. samples. Let $F_t : \mathbb{R}^n \to [0,1]$ be the monotone continuous DR-submodular function corresponding to the t-th sample z_t, i.e.,

$$F_t(\boldsymbol{x}) = F(\boldsymbol{x}; z_t)$$

for $t = 1, \dots, T$. Similarly, define an auxiliary function H_t with respect to z_t by

$$H_t(\boldsymbol{x}, \tau) = \tau - \frac{1}{\alpha}[\tau - F_t(\boldsymbol{x})]_+ = \tau - \frac{1}{\alpha}[\tau - F(\boldsymbol{x}; z_t)]_+.$$

for $t = 1, \dots, T$. Let $K \subseteq \mathbb{R}^n_{\ge 0}$ be a down-closed convex set. Formally, we make the following very mild assumptions on F_t and K.

Assumption 1.

- For all t, F_t is L-Lipschitz and β-smooth, and $\|\nabla F_t\| \le G$.
- The diameter of K is bounded by D.
- We are given a linear optimization oracle over K.

When the underlying norm is the ℓ^p-norm with $p \ne 2$, we write G_p to emphasize it. For example, if F_t is the multilinear extension of a submodular set function $f_t : 2^V \to [0,1]$ and K is the base polytope of a rank-k matroid, we have $L = \beta = O(n)$, $G_\infty = 1$, $D = O(\sqrt{k})$.

Our algorithm borrows some ideas from an algorithm called RASCAL [Wilder, 2018]. First, we define a smoothed auxiliary function $\tilde{H}_t : [0,1]^V \times [0,1] \to [0,1]$ as

$$\tilde{H}_t(\boldsymbol{x}, \tau) = \frac{1}{u}\int_0^u \left(\tau + \xi - \frac{1}{\alpha}[\tau + \xi - F_t(\boldsymbol{x})]_+\right) d\xi,$$

where $u > 0$ is a smoothing parameter specified later. This smoothing guarantees that $\tilde{H}_t$ is differentiable for all $\boldsymbol{x}$ and has Lipschitz continuous gradients.

Lemma 2 (Lemma 6 of [Wilder, 2018])**.**

1. *$|H_t(\boldsymbol{x}, \tau) - \tilde{H}_t(\boldsymbol{x}, \tau)| \le \frac{u(1+1/\alpha)}{2}$ for all $\boldsymbol{x}$ and τ.*
2. *If F_t is L-Lipschitz and β-smooth, and $\|\nabla F_t\| \le G$, then $\nabla_{\boldsymbol{x}}\tilde{H}_t$ is $\frac{1}{\alpha}(\beta + \frac{LG}{u})$-Lipschitz.*

Lemma 3 ([Wilder, 2018])**.** *The function $\max_\tau \tilde{H}_t(\cdot, \tau)$ is monotone and up-concave.*

3.1 StochasticRASCAL

We now formally describe our algorithm, StochasticRASCAL. We note that RASCAL runs the Frank-Wolfe algorithm on a function $\max_\tau \sum_{t=1}^T \tilde{H}_t(\cdot, \tau)$. Owing to the up-concavity and smoothness properties of this function, one can obtain $(1 - 1/e)$-approximation. However, in our online setting, we cannot evaluate this function because $\tilde{H}_t$ will be revealed online, and hence we cannot simply run RASCAL.

To overcome the issue above, first we split the T samples into *mini-batches* of length B, which we specify later. The key idea is to use the following objective function

$$\bar{H}_b(\boldsymbol{x}) = \max_\tau \frac{1}{B}\sum_{t=(b-1)B+1}^{bB} \tilde{H}_t(\boldsymbol{x}, \tau)$$

for each b-th mini-batch ($b = 1, \dots, T/B$). We can see that $\bar{H}_b(\boldsymbol{x})$ is monotone, up-concave, and can be evaluated only using samples in the b-th mini-batch. Then, we run a perturbed version of Frank-Wolfe algorithm [Golovin *et al.*, 2014; Bian *et al.*, 2017] on $\bar{H}_b$. More formally, we first initialize $\boldsymbol{x}^0_{b+1} = \mathbf{0}$ and for each $s = 0, \delta, 2\delta, \dots, 1 - \delta$, we perform the update

$$\boldsymbol{x}^{s+\delta}_b \leftarrow \boldsymbol{x}^s_b + \delta \boldsymbol{d}^s_b$$

Algorithm 1 StochasticRASCAL

Require: learning rates $\lambda > 0$, step size $\delta > 0$, perturbation distribution $\mathcal{D}_{\mathrm{FPL}}$, smoothing parameter $u > 0$, and mini-batch size $B > 0$

Initialize $\boldsymbol{x}_1 \in K$ arbitrary.
for $b = 1, \dots, T/B$ **do**
  Observe samples $z_{(b-1)B+1}, \dots, z_{bB}$ and store them in mini-batch $\mathcal{Z}_b$.
  /* continuous greedy */
  Let $\boldsymbol{x}_b^0 \leftarrow \boldsymbol{0}$.
  for $s = 0, \delta, 2\delta, \dots, 1-\delta$ **do**
    $\tau := \textsc{SmoothTau}(\boldsymbol{x}_b^s, u, \mathcal{Z}_b)$
    $\boldsymbol{g}_b^s := \textsc{SmoothGrad}(\boldsymbol{x}_b^s, \tau, u, \mathcal{Z}_b)$.
    Find a vertex $\boldsymbol{d}_{b+1}^s$ of K that maximizes $\left\langle \lambda \sum_{b'=1}^{b} \boldsymbol{g}_{b'}^s + \boldsymbol{r}_b^s, \boldsymbol{d} \right\rangle$ for $\boldsymbol{d} \in K$, where $\boldsymbol{r}_b^s \sim \mathcal{D}_{\mathrm{FPL}}$. // FPL
    $\boldsymbol{x}_b^{s+\delta} \leftarrow \boldsymbol{x}_b^s + \delta \boldsymbol{d}_b^s$
  end for
  Let $\boldsymbol{x}_b = \boldsymbol{x}_b^1$.
end for
return $\boldsymbol{x}_{b'}$ for b' chosen from $\{1, 2, \dots, T/B\}$ uniformly at random.

Algorithm 2 $\textsc{SmoothGrad}(\boldsymbol{x}, \tau, u, \mathcal{Z})$

Require: $\boldsymbol{x}$, τ, u, and mini-batch $\mathcal{Z}$

$I_z(\tau) := \max\{\min\{\frac{F(\boldsymbol{x};z)-\tau}{u}, 1\}$ for $z \in \mathcal{Z}$.
return $\sum_{z \in \mathcal{Z}} I_z(\tau) \nabla_{\boldsymbol{x}} F(\boldsymbol{x}, z)$

where δ is the step size, $\boldsymbol{d}_b^s$ is a solution to a perturbed linear optimization problem on K:

$$\boldsymbol{d}_b^s \in \operatorname*{argmax}_{\boldsymbol{d} \in K} \left\langle \lambda \sum_{b'=1}^{b} \nabla \bar{H}_{b'}(\boldsymbol{x}_{b'}^s) + \boldsymbol{r}_b^s, \boldsymbol{d} \right\rangle.$$

Here, $\boldsymbol{r}_b^s \sim \mathcal{D}_{\mathrm{FPL}}$ is a perturbation vector. This perturbation trick aims to stabilize the algorithm so that we can maximize the true objective $\max_\tau \sum_{t=1}^{T} \tilde{H}_t(\cdot, \tau)$ using only mini-batch objectives.

In each iteration of continuous greedy, we need the gradients $\nabla \bar{H}_b(\boldsymbol{x}_b^s)$, which in turn requires us to compute $\operatorname{argmax}_\tau \frac{1}{B} \sum_{t=(b-1)B+1}^{bB} \tilde{H}_t(\boldsymbol{x}_b^s, \tau)$. These gradients and the optimal τ can be computed by SmoothGrad and SmoothTau subroutines, respectively, which were proposed in [Wilder, 2018]; See Algorithms 2 and 3.

Let us write $\boldsymbol{x}_b := \boldsymbol{x}_b^1$ for $b = 1, \dots, T/B$. The final output of StochasticRASCAL is $\boldsymbol{x}_{b'}$ for a random index b' chosen uniformly at random from $\{1, 2, \dots, T/B\}$. The pseudocode of StochasticRASCAL is presented in Algorithm 1.

3.2 Convergence Rate via Regret Bounds

Let us consider the convergence rate of StochasticRASCAL. The main challenge of the analysis is how to set the parameters used in the algorithm, i.e., learning rates λ, step size δ, perturbation distribution $\mathcal{D}_{\mathrm{FPL}}$, smoothing parameter

Algorithm 3 $\textsc{SmoothTau}(\boldsymbol{x}, u, \mathcal{Z})$

Require: $\boldsymbol{x}$, u, and mini-batch $\mathcal{Z}$

$\mathcal{B} := \{F(\boldsymbol{x}; z) \mid z \in \mathcal{Z}\} \cup \{F(\boldsymbol{x}; z) + u \mid z \in \mathcal{Z}\}$
Sort $\mathcal{B}$ in ascending order, obtaining $\mathcal{B} = \{b_1, \dots, b_{|\mathcal{B}|}\}$.
$i^* := \min\{i : \sum_{z \in \mathcal{Z}} I_z(b_i) < \alpha |\mathcal{Z}|\}$
$A := \{z \in \mathcal{Z} : b_{i^*-1} < F(\boldsymbol{x}; z) < b_{i^*}\}$
$C := \{z \in \mathcal{Z} : F(\boldsymbol{x}; z) \le b_{i^*-1}\}$
return the solution τ of the linear equation

$$\sum_{z \in \mathcal{Z}} \frac{F(\boldsymbol{x}; z) - \tau}{u} + |C| = \alpha |\mathcal{Z}|.$$

u, and mini-batch size B, to achieve the desired $O(T^{-1/4})$ convergence rate.

To this end, using tools from *online convex optimization*, we prove an approximate regret bound for a variant of StochasticRASCAL for adversarial online submodular CVaR learning (see Introduction for the definition).

Theorem 4 (informal)**.** *There exists an efficient online algorithm for adversarial online CVaR learning with*

$$\left(1 - \frac{1}{e}\right) \sum_{t=1}^{T} H_t(\boldsymbol{x}^*, \tau^*) - \mathbf{E}\left[\sum_{t=1}^{T} H_t(\boldsymbol{x}_t, \tau_t)\right] = O(T^{3/4})$$

for an arbitrary $\boldsymbol{x}^* \in K$ and $\tau^* \in [0, 1]$, where the big-O notation hides a factor polynomial in α, β, D, G, and n.

We then show that the above regret bound can be used to show a convergence rate of StochasticRASCAL. The technical detail of the adversarial setting and the proof of the following theorem is deferred to the full version.

Theorem 5. *Under Assumption 1,* StochasticRASCAL *outputs $\boldsymbol{x} \in K$ such that for any $\boldsymbol{x}^* \in K$,*

$$\mathbf{E}[\mathrm{CVaR}_{\mathcal{D}}(\boldsymbol{x})] \ge \left(1 - \frac{1}{e}\right) \mathrm{CVaR}_{\mathcal{D}}(\boldsymbol{x}^*) - O\left(\frac{\sqrt{C_\alpha G D} n^{1/8}}{\sqrt{\alpha}}\right) T^{-1/4},$$

where we set $B = \frac{\alpha C_\alpha \sqrt{T}}{DGn^{1/4}}$, $\delta = \frac{\alpha^2}{D^2\left((1+\alpha)GL\sqrt{T} + \alpha\beta T^{1/4}\right)}$, $\lambda = \frac{\alpha D n^{1/4} \sqrt{B/T}}{G}$, $u = \frac{T^{-1/4}}{(1+1/\alpha)}$, and $\mathcal{D}_{\mathrm{FPL}} = \mathrm{Unif}([0, 1]^n)$, and $C_\alpha := \max\{1, \frac{1}{\alpha} - 1\}$. Further, if K is an integral polytope contained in $\{\boldsymbol{x} \in [0, 1]^n : \sum_i x_i = k\}$, then

$$\mathbf{E}[\mathrm{CVaR}_{\mathcal{D}}(\boldsymbol{x})] \ge \left(1 - \frac{1}{e}\right) \mathrm{CVaR}_{\mathcal{D}}(\boldsymbol{x}^*) - O\left(\frac{\sqrt{C_\alpha G_\infty} k^{3/4} \log^{1/4} n}{\sqrt{\alpha}}\right) T^{-1/4}$$

for $B = \frac{\sqrt{2} C_\alpha \sqrt{T} \alpha}{2 G_\infty k^{3/2} \sqrt{\log(n)}}$, $\delta = \frac{\alpha^2}{D^2\left((1+\alpha)GL\sqrt{T} + \alpha\beta T^{1/4}\right)}$, $\lambda = \sqrt{\frac{B}{Tk}}$, $u = \frac{T^{-1/4}}{(1+1/\alpha)}$, and $\mathcal{D}_{\mathrm{FPL}} = N(\boldsymbol{0}, I)$.

To achieve $\mathbf{E}[\mathrm{CVaR}_{\mathcal{D}}(\boldsymbol{x})] \ge \left(1 - \frac{1}{e}\right) \mathrm{CVaR}_{\mathcal{D}}(\boldsymbol{x}^*) - \varepsilon$ for a desired error $\varepsilon > 0$, StochasticRASCAL requires

$O(\frac{D^2G^2\sqrt{n}}{\varepsilon^4})$ samples and $O(\frac{DGn^{1.25}}{\varepsilon^2})$ space, whereas RASCAL [Wilder, 2018] requires $O(\frac{n}{\varepsilon^2})$ samples and $O(\frac{n^2}{\varepsilon^2})$ space. Our algorithm runs in a smaller space when the parameters are of moderate size. For example, if $\beta = D = O(1)$ and $L = G = o(n^{1/4})$, the space complexity of STOCHASTICRASCAL is better than that of RASCAL.

4 CVaR Maximization of Discrete Submodular Functions

We now present our online algorithm for a monotone submodular set function and a matroid constraint. Let $f_t : 2^V \to [0, 1]$ be a monotone submodular function corresponding to the t-th sample and F_t be its multilinear extension for $t = 1, \dots, T$.

The basic idea is to run STOCHASTICRASCAL on the multilinear extensions F_t and the matroid polytope K. However, we must address several technical obstacles. First, we must compare the output portfolio with the optimal *portfolio*; the error bound in the previous sections compared it with the optimal *solution*. To this end, we make multiple copies of variables so that we can approximate an optimal portfolio by a uniform distribution over a multiset of feasible solutions. More precisely, we define a continuous DR-submodular function $\bar{F}_t : K^r \to [0, 1]$ by

$$\bar{F}_t(\boldsymbol{x}^1, \dots, \boldsymbol{x}^r) = \frac{1}{r}\sum_{i=1}^{r} F_t(\boldsymbol{x}^i)$$

for some sufficiently large r. Then, we feed $\bar{F}_1, \dots, \bar{F}_T$ to STOCHASTICRASCAL. Suppose that we obtain $(\boldsymbol{x}_b^1, \dots, \boldsymbol{x}_b^r)$ at Line 12 for each mini-batch $b = 1, \dots, T/B$. Abusing the notation, let us denote $(\boldsymbol{x}_t^1, \dots, \boldsymbol{x}_t^r) := (\boldsymbol{x}_b^1, \dots, \boldsymbol{x}_b^r)$ when the t-th sample is in the b-th mini-batch.

Next, we need to convert $\boldsymbol{x}_t^1, \dots, \boldsymbol{x}_t^r$ to feasible sets without significantly deteriorating the values of the multilinear extensions. To this end, we independently apply *randomized swap rounding* [Chekuri *et al.*, 2010] q times to each $\boldsymbol{x}^i$ to obtain feasible sets $X_t^{i,1}, \dots, X_t^{i,q}$. Note that randomized swap rounding is oblivious rounding and independent from F_t. We can show that $\frac{1}{q}\sum_{j=1}^{q} f_t(X_t^{i,j})$ is close to $F_t(\boldsymbol{x}_t^i)$ by using a concentration inequality. Finally, after T rounds, we return the uniform portfolio over all $X_t^{i,q}$. The pseudocode is given in Algorithm 4. Carefully choosing r and q, we obtain the following theorem.

Algorithm 4 Online algorithm for maximizing a monotone submodular set function subject to a matroid constraint.

1: Run STOCHASTICRASCAL for $\bar{F}_1, \dots, \bar{F}_T$ and the matroid polytope K and let $(\boldsymbol{x}_t^1, \dots, \boldsymbol{x}_t^r)$ be the temporary solution at Line 12 in STOCHASTICRASCAL for $t = 1, \dots, T$.
2: $X_t^{i,j} \leftarrow$ RANDOMIZEDSWAPROUNDING$(\boldsymbol{x}_t^i)$ for $t = 1, \dots, T$, $i = 1, \dots, r$, and $j = 1, \dots, q$.
3: **return** Uniform portfolio $\bar{\mathcal{P}}$ over all $X_t^{i,j}$.

Theorem 6. *Algorithm 4 achieves*

$$\mathbf{E}[\mathrm{CVaR}(\bar{\mathcal{P}})] \geq \left(1 - \frac{1}{e}\right)\mathrm{CVaR}_{\mathcal{D}}(\mathcal{P}^*) - O(k^{3/4}\log^{1/4}(n)T^{-1/10})$$

for arbitrary portfolio $\mathcal{P}^*$, where we set $r = \tilde{O}(T^{1/5})$ and $q = \tilde{O}(T^{3/4})$ and the expectation is taken over $z_1, \dots, z_T$ and the randomness of the algorithm.

5 Experiments

In this section, we show our experimental results. In all the experiments, the parameter α of CVaR was set to 0.1. The experiments were conducted on a Linux server with Intel Xeon Gold 6242 (2.8GHz) and 384GB of main memory.

Problem Description. We conducted experiments on the sensor resource allocation problem, in which the goal is to rapidly detect a contagion spreading through a network using a limited budget [Bian *et al.*, 2017; Leskovec *et al.*, 2007; Soma and Yoshida, 2015].

Here, we follow the configuration of the experiments conducted in [Wilder, 2018]. Let $G = (V, E)$ be a graph on n vertices. A contagion starts at a random vertex and spreads over time according to some specific stochastic process. Let z_v be the time at which the contagion reaches v, and let $z_{\max} = \max_{v \in V : z_v < \infty} z_v$. If $z_v = \infty$ for some vertex $v \in V$, that is, the contagion does not reach v, we reassign $z_v = z_{\max}$, as described in [Wilder, 2018].

The decision maker has a budget B (e.g., energy) to spend on sensing resources. Let x_v represent the amount of energy allocated to the sensor at a vertex v. When contagion reaches v at time z_v, the sensor detects it with probability $1 - (1 - p)^{x_v}$, where $p \in [0, 1]$ is the probability that detects the contagion per unit of energy. The objective F on vectors $\boldsymbol{x} = (x_v)_{v \in V}$ and $\boldsymbol{z} = (z_v)_{v \in V}$ is the expected amount of detection time that is saved by the sensor placements:

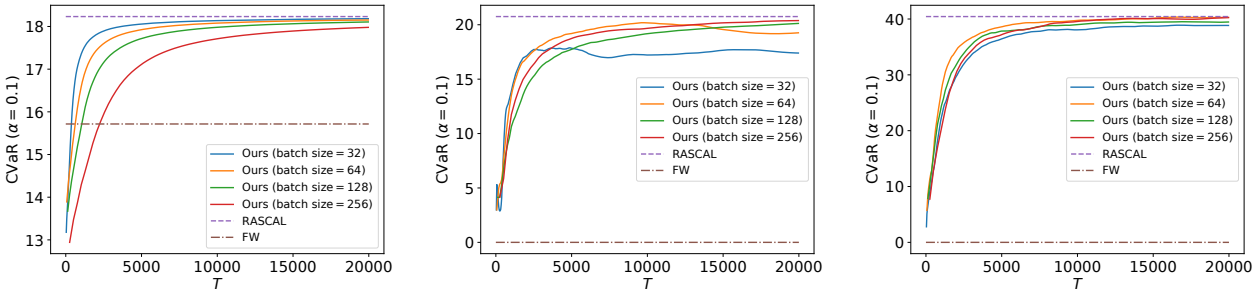
$$F(\boldsymbol{x}; \boldsymbol{z}) = z_{\max} - \sum_{i=1}^{n} z_{v_i}(1 - (1 - p)^{x_{v_i}}) \prod_{j<i} (1 - p)^{x_{v_j}},$$

where the vertices are ordered so that $z_{v_1} \leq z_{v_2} \leq \cdots \leq z_{v_n}$. It is known that the function F is DR-submodular [Bian *et al.*, 2017].

Datasets. We consider two sensing models and generated three datasets. In all of them, the source vertex s is chosen uniformly at random.

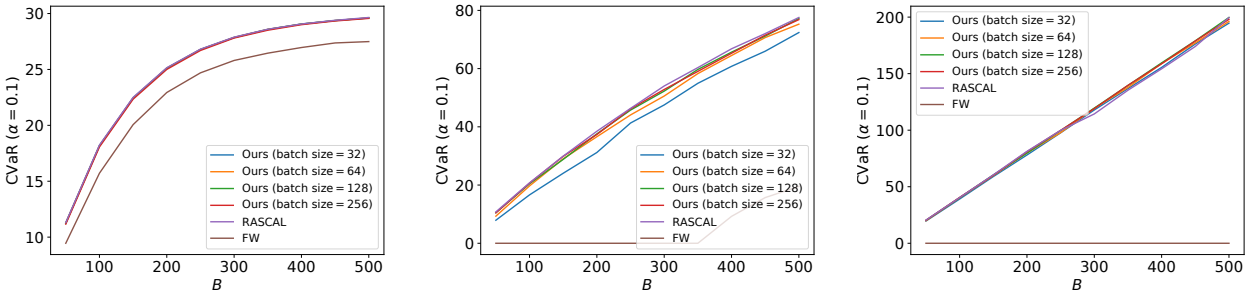
The first model is the continuous time independent cascade model (CTIC). In this model, each edge $uv \in E$ has propagation time ρ_{uv} drawn from an exponential distribution with mean λ. The contagion starts at the source vertex s, i.e., $z_s = 0$, and we iteratively set $z_v = \min_{u \in N(v)} z_u + \rho_{uv}$, where $N(v)$ is the set of neighbors of v. Note that z_v is the first time that the contagion reaches v from its neighbor. We generated datasets using two real-world networks[2]: NetScience, a collaboration network of 1,461 network scientists, and EuroRoad, a network of 1,174 European cities and

[2] http://konect.cc



(a) NetScience (b) EuroRoad (c) BWSN

Figure 1: CVaR and the number of samples T



(a) NetScience (b) EuroRoad (c) BWSN

Figure 2: CVaR and budget B

the roads between them. For both networks, we set $\lambda = 5$ and $p = 0.01$, and we generated 1,000 scenarios.

The second model, known as the Battle of Water Sensor Networks (BWSN) [Ostfeld and others, 2008], involves contamination detection in a water network. BWSM simulates the spread of contamination through a 126-vertex water network consisting of junctions, tanks, pumps, and the links between them, and the z_v values are provided by a simulator. We set $p = 0.001$ and generated 1,000 scenarios.

Methods. We compared our method against two offline algorithms, RASCAL [Wilder, 2018] and the Frank–Wolfe (FW) algorithm [Bian *et al.*, 2017]. We note that the latter algorithm is designed to maximize the expectation of a DR-submodular function instead of its CVaR. We run those offline methods on the generated 1,000 scenarios for each dataset. As our method is an online algorithm, we run our method on 20,000 samples in an online manner, where each sample was uniformly drawn from the set of generated scenarios.

Results. Figure 1 shows how the CVaR changes as T increases. For each dataset, as long as the batch size is not excessively small, the CVaR attained by our method approaches to that attained by RASCAL. FW algorithm showed significantly lower performance because it is not designed to maximize CVaR.

Figure 2 shows how the CVaR changes as the budget B increases. For our method, we plotted the CVaR after processing $10,000$ samples. We can again confirm that the CVaR attained by our method is close to that attained by RASCAL.

Acknowledgments

T.S. is supported by JST, ERATO, Grant Number JPMJER1903, Japan. Y.Y. is supported in part by JSPS KAKENHI Grant Number 18H05291 and 20H05965.

References

[Anari *et al.*, 2019] N. Anari, N. Haghtalab, S. Naor, S. Pokutta, M. Singh, and A. Torrico. Structured robust submodular maximization: Offline and online algorithms. In *AISTATS*, volume 89, 2019.

[Bian *et al.*, 2017] A. Bian, B. Mirzasoleiman, J. Buhmann, and A. Krause. Guaranteed non-convex optimization: Submodular maximization over continuous domains. In *AISTATS*, pages 111–120, 2017.

[Buchbinder and Feldman, 2018] N. Buchbinder and M. Feldman. Submodular functions maximization problems. In *Handbook of Approximation Algorithms and Metaheuristics*, 2018.

[Calinescu *et al.*, 2011] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM J. Comput.*, 40(6):1740–1766, 2011.

[Cardoso and Xu, 2019] A. R. Cardoso and H. Xu. Risk-averse stochastic convex bandit. In *AISTATS*, pages 39–47, 2019.

[Chekuri *et al.*, 2010] C. Chekuri, J. Vondrák, and R. Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. In *FOCS*, pages 575–584, 2010.

[Chen *et al.*, 2017] R. Chen, B. Lucier, Y. Singer, and V. Syrgkanis. Robust optimization for non-convex objectives. In *NIPS*, pages 4705–4714, 2017.

[Chen *et al.*, 2018] L. Chen, H. Hassani, and A. Karbasi. Online continuous submodular maximization. In *AISTATS*, pages 1896–1905, 2018.

[Fujishige, 2005] S. Fujishige. *Submodular Functions and Optimization*. Elsevier, 2nd edition, 2005.

[Golovin *et al.*, 2014] D. Golovin, A. Krause, and M. Streeter. Online submodular maximization under a matroid constraint with application to learning assignments. *arXiv*, 2014.

[Hassani *et al.*, 2017] H. Hassani, M. Soltanolkotabi, and A. Karbasi. Gradient methods for submodular maximization. In *NIPS*, pages 5841–5851, 2017.

[Hazan, 2016] E. Hazan. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

[Karbasi *et al.*, 2019] A. Karbasi, H. Hassani, A. Mokhtari, and Z. Shen. Stochastic continuous greedy++: When upper and lower bounds match. In *NeurIPS*, pages 13087–13097, 2019.

[Karimi *et al.*, 2017] M. R. Karimi, M. Lucic, H. Hassani, and A. Krause. Stochastic submodular maximization: The case of coverage functions. In *NIPS*, 2017.

[Kempe *et al.*, 2003] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *KDD*, pages 137–146, 2003.

[Krause and Golovin, 2014] A. Krause and D. Golovin. Submodular function maximization. In *Tractability: Practical Approaches to Hard Problems*, pages 71–104. Cambridge University Press, 2014.

[Krause *et al.*, 2008] A. Krause, B. McMahan, C. Guestrin, and A. Gupta. Robust submodular observation selection. *JMLR*, 9:2761–2801, 2008.

[Krokhmal *et al.*, 2002] P. Krokhmal, J. Palmquist, and S. Uryasev. Portfolio optimization with conditional value-at-risk objective and constraints. *Journal of Risk*, 4:43–68, 2002.

[Leskovec *et al.*, 2007] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N Glance. Cost-effective outbreak detection in networks. In *KDD*, pages 420–429, 2007.

[Maehara, 2015] T. Maehara. Risk averse submodular utility maximization. *Operations Research Letters*, 43(5):526 – 529, 2015.

[Mansini *et al.*, 2007] R. Mansini, W. Ogryczak, and G. Speranza. Conditional value at risk and related linear programming models for portfolio optimization. *Annals of Operations Research*, 152(1):227–256, 2007.

[Mokhtari *et al.*, 2018] A. Mokhtari, H. Hassani, and A. Karbasi. Conditional gradient method for stochastic submodular maximization: Closing the gap. In *AISTATS*, pages 1886–1895, 2018.

[Ohsaka and Yoshida, 2017] N. Ohsaka and Y. Yoshida. Portfolio optimization for influence spread. In *WWW*, pages 977–985, 2017.

[Ostfeld and others, 2008] A. Ostfeld et al. The battle of the water sensor networks (bwsn): A design challenge for engineers and algorithms. *J. Water Resour. Plan. Manag.*, 134(6):556–568, 2008.

[Rockafellar and others, 2000] T. Rockafellar et al. Optimization of conditional value-at-risk. *Journal of Risk*, 2:21–42, 2000.

[Roughgarden and Wang, 2018] T. Roughgarden and J. R. Wang. An optimal algorithm for online unconstrained submodular maximization. In *COLT*, pages 1307–1325, 2018.

[Shapiro *et al.*, 2014] A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on stochastic programming: modeling and theory*. SIAM, 2014.

[Soma and Yoshida, 2015] T. Soma and Y. Yoshida. A generalization of submodular cover via the diminishing return property on the integer lattice. In *NIPS*, pages 847–855, 2015.

[Soma, 2019] T. Soma. No-regret algorithms for online k-submodular maximization. In *AISTATS*, pages 1205–1214, 2019.

[Staib *et al.*, 2019] M. Staib, B. Wilder, and S. Jegelka. Distributionally robust submodular maximization. In *AISTATS*, volume 89, pages 506–516, 2019.

[Streeter and Golovin, 2008] M. Streeter and D. Golovin. An online algorithm for maximizing submodular functions. In *NIPS*, pages 1577–1584, 2008.

[Streeter *et al.*, 2009] M. Streeter, D. Golovin, and A. Krause. Online learning of assignments. In *NIPS*, pages 1794–1802, 2009.

[Tamar *et al.*, 2015] A. Tamar, Y. Glassner, and S. Mannor. Optimizing the CVaR via sampling. In *AAAI*, 2015.

[Vondrák, 2013] J. Vondrák. Symmetry and approximability of submodular maximization problems. *SIAM J. Comput.*, 42(1):265–304, 2013.

[Wilder, 2018] B. Wilder. Risk-sensitive submodular optimization. In *AAAI*, pages 6451–6458, 2018.

[Yau *et al.*, 2011] S. Yau, R. H. Kwon, J. S. Rogers, and D. Wu. Financial and operational decisions in the electricity sector: Contract portfolio optimization with the conditional value-at-risk criterion. *Int. J. Prod. Econ*, 134(1):67–77, 2011.

[Zhang *et al.*, 2019] M. Zhang, L. Chen, H. Hassani, and A. Karbasi. Online continuous submodular maximization: From full-information to bandit feedback. In *NeurIPS*, volume 32, pages 9210–9221, 2019.