



AFRL-AFOSR-VA-TR-2021-0054

Information content of big data

Amin Karbasi
YALE UNIV NEW HAVEN CT
105 WALL ST
NEW HAVEN, CT,
US

06/21/2021
Final Technical Report

DISTRIBUTION A: Distribution approved for public release.
--

Air Force Research Laboratory
Air Force Office of Scientific Research
Arlington, Virginia 22203
Air Force Materiel Command

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) 21-06-2021		2. REPORT TYPE Final		3. DATES COVERED (From - To) 15 Feb 2018 - 14 Feb 2022	
4. TITLE AND SUBTITLE Information content of big data				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER FA9550-18-1-0160	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Amin Karbasi				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) YALE UNIV NEW HAVEN CT 105 WALL ST NEW HAVEN, CT US				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AF Office of Scientific Research 875 N. Randolph St. Room 3112 Arlington, VA 22203				10. SPONSOR/MONITOR'S ACRONYM(S) AFRL/AFOSR RTA2	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S) AFRL-AFOSR-VA-TR-2021-0054	
12. DISTRIBUTION/AVAILABILITY STATEMENT A Distribution Unlimited: PB Public Release					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT Every day, an enormous volume of data is generated at an unprecedented rate and modern decision-making feels like "swimming in a sea of sensors and drowning in data". One of the primary challenges in current data science research is to identify the most valuable information from a large data set. To this end we have investigated the applications of submodular maximization in fairness, privacy, recommender systems, and time series.					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			ERIK BLASCH
U	U	U	UU	73	19b. TELEPHONE NUMBER (Include area code) 426-7311

Standard Form 298 (Rev.8/98)
Prescribed by ANSI Std. Z39.18

AFOSR Final Report

Contract/Grant Title: Information Content of Big Data

Contract/Grant #: FA9550-18-1-0160

Reporting Period: 02/15/2018-02/15/2019

Author: Amin Karbasi

Abstract

Every day, an enormous volume of data is generated at an unprecedented rate and modern decision-making feels like “swimming in a sea of sensors and drowning in data”. One of the primary challenges in current data science research is to identify the most valuable information from a large data set. To this end we have investigated the applications of submodular maximization in fairness, privacy, recommender systems, and time series.

Main Accomplishments/Successes

A. Published 24 articles in top-tier machine learning and artificial intelligence venues

B. Three PhD students and a post-doc graduated

C. Received multiple awards/grants

- NSF CAREER
- ONR YIA
- Amazon Faculty Award

Research Contributions:

The research contributions include:

1. We published extensively in top-tier machine learning conferences including:
 - ICML
 - NIPS
 - AISTATS
 - AAAI
2. I gave three tutorials on the role of submodularity in artificial intelligence:
 - ISIT 2018 Tutorial on “Submodularity in Information and Data Science”
 - CVPR 2018 Tutorial on “Big Data Summarization: Algorithms and Applications”
 - ICML 2020 Tutorial on “Submodular Optimization: From Discrete to Continuous and Back”
3. I am writing two books
 - “Submodular Maximization: From Discrete to Continuous and Back”
 - “Projection-Free Optimization”

Transitions: (required for final report)

1. Methods to industry: Submodular optimization techniques are being heavily used in companies like Google and Amazon. I am currently visiting Google and consult them on the applications of such methods for large-scale machine learning methods.

2. Students to employment (trained under grant)

- Lin Chen => Post-doc at Berkeley
- Marko Mitrovic => Engineer at Google
- Mingrui Zhang
- Ehsan Kazemi => Engineer at Google

3. Techniques to AFRL (correspondence with AFRL researchers): I have given a few talks and also wrote a tutorial on submodular optimization methods along with a Julia package.

Collaborations:

Industry: Google & Amazon

Student Internships: Lin Chen, Marko Mitrovic

Awards:

1. Best paper: Facebook MAIN Award for the paper “State-Specific Individualized Functional Networks Form a Predictive Signature of Brain State” from Montreal Artificial Intelligence and Neuroscience Conference.

2. Recognition at school: Promoted to the level of associate professor.

3. External recognition: NSF CAREER Award

Publications:

1. Journal:

- “The Power of Subsampling in Submodular Optimization”. Mathematics of Operations Research, 2021.
- “Stochastic Conditional Gradient ++: (Non-)Convex Minimization and Continuous Submodular Maximization”. SIAM Journal on Optimization 2020.
- “Stochastic Conditional Gradient Methods: From Convex Minimization to Submodular Maximization”. Journal of Machine Learning Research (JMLR), 2020.

3. Conference

- One sample stochastic frank-wolfe (AISTATS 2020)
- Quantized frank-wolfe: Faster optimization, lower communication, and projection free (AISTATS 2020)
- Black box submodular maximization: Discrete and continuous settings (AISTATS 2020)
- More data can expand the generalization gap between adversarially robust and standard models (ICML 2020)
- Unconstrained submodular maximization with constant adaptive complexity (STOC 2019)
- Submodular streaming in all its glory: Tight approximation, minimum memory and low adaptive complexity (ICML 2019)

- Submodular maximization beyond non-negativity: Guarantees, fast algorithms, and applications (ICML 2019)
- Projection-free bandit convex optimization (AISTATS 2019)
- Stochastic Continuous Greedy++: When Upper and Lower Bounds Match (NeurIPS 2019)
- Online Continuous Submodular Maximization: From Full-Information to Bandit Feedback (NeurIPS 2019)
- Adaptive sequence submodularity (NeurIPS 2019)
- Do Less, Get More: Streaming Submodular Maximization with Subsampling (NIPS 2018)
- Decentralized submodular maximization: Bridging discrete and continuous settings (ICML 2018)
- Data summarization at scale: A two-stage submodular approach (ICML 2018)
- Weakly submodular maximization beyond cardinality constraints: Does randomization help greedy? (ICML 2018)
- Scalable deletion-robust submodular maximization: Data summarization with privacy and fairness constraints (ICML 2018)
- Projection-free online optimization with stochastic gradient: From convexity to submodularity (ICML 2018)
- Online continuous submodular maximization (AISTATS 2018)
- Comparison based learning from weak oracles (AISTATS 2018)
- Submodularity on Hypergraphs: From Sets to Sequences (AISTATS 2018)
- Conditional gradient method for stochastic submodular maximization: Closing the gap (AISTATS 2018)

Software:

- A personalized risk assessment tool: <https://hunala.yale.edu>
- All the optimization software is publicly at <http://iid.yale.edu/software/software.md/>

Invited Talks:

1. DOD meetings

2. Professional Societies

- Foundations of Data Science - Virtual Talk Series - UC Berkeley
- Foundations of Data Science Seminar, UT Austin
- SyE Department Seminar, Georgia Tech
- Foundations of Data Science Seminar, ETH Zurich

2. Students

- ICML 2019, 2020
- NeurIPS 2019, 2020
- AISTATS 2018, 2019, 2020

4. AFOSR PI Meeting

- AFOSR DDIP/DDDAS meeting 2018
- AFOSR DDIP/DDDAS meeting 2019
- AFOSR DDIP/DDDAS meeting 2020

Community Efforts:

1. Data Collects

<https://hunala.yale.edu>

3. Student events

- Yale Pathway to Science
- Julia Robinson Math Festival

References:

E. Blasch, S. Ravela, A. Aved (eds), *Handbook of Dynamic Data Driven Applications Systems*, Springer, 2018.

Changes in Research Objectives: NA

Change in AFOSR Program Manager: NA

Extensions Granted or Milestones Slipped: We requested a no cost extension

New discoveries, inventions, or patent disclosures: NA

Append 1 – 3 Published Papers to the PDF file before submitting

Stochastic Conditional Gradient Methods: From Convex Minimization to Submodular Maximization

Aryan Mokhtari

*Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX 78712*

MOKHTARI@AUSTIN.UTEXAS.EDU

Hamed Hassani

*Department of Electrical and Systems Engineering
University of Pennsylvania
Philadelphia, PA 19104*

HASSANI@SEAS.UPENN.EDU

Amin Karbasi

*Department of Electrical Engineering and Computer Science
Yale University
New Haven, CT 06520*

AMIN.KARBASI@YALE.EDU

Editor: Francis Bach

Abstract

This paper considers stochastic optimization problems for a large class of objective functions, including convex and continuous submodular. Stochastic proximal gradient methods have been widely used to solve such problems; however, their applicability remains limited when the problem dimension is large and the projection onto a convex set is computationally costly. Instead, stochastic conditional gradient algorithms are proposed as an alternative solution which rely on (i) Approximating gradients via a simple averaging technique requiring a single stochastic gradient evaluation per iteration; (ii) Solving a linear program to compute the descent/ascent direction. The gradient averaging technique reduces the noise of gradient approximations as time progresses, and replacing projection step in proximal methods by a linear program lowers the computational complexity of each iteration. We show that under convexity and smoothness assumptions, our proposed stochastic conditional gradient method converges to the optimal objective function value at a sublinear rate of $\mathcal{O}(1/t^{1/3})$. Further, for a monotone and continuous DR-submodular function and subject to a *general* convex body constraint, we prove that our proposed method achieves a $((1 - 1/e)\text{OPT} - \epsilon)$ guarantee (in expectation) with $\mathcal{O}(1/\epsilon^3)$ stochastic gradient computations. This guarantee matches the known hardness results and closes the gap between deterministic and stochastic continuous submodular maximization. Additionally, we achieve $((1/e)\text{OPT} - \epsilon)$ guarantee after operating on $\mathcal{O}(1/\epsilon^3)$ stochastic gradients for the case that the objective function is continuous DR-submodular but non-monotone and the constraint set is a down-closed convex body. By using stochastic continuous optimization as an interface, we also provide the first $(1 - 1/e)$ tight approximation guarantee for maximizing a *monotone but stochastic* submodular *set* function subject to a general matroid constraint and $(1/e)$ approximation guarantee for the non-monotone case.

Keywords: Stochastic optimization, conditional gradient methods, convex minimization, submodular maximization, gradient averaging, Frank-Wolfe algorithm, greedy algorithm

1. Introduction

Stochastic optimization arises in various applications including wireless communications (Ribeiro, 2010), learning theory (Vapnik, 2013), machine learning (Bottou, 2010), adaptive filters (Haykin, 2008), portfolio selection (Shapiro et al., 2009) to name a few. In this class of problems the goal is to optimize an objective function defined as an expectation over a set of random functions subject to a general convex constraint. In particular, consider an optimization variable $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^n$ and a random variable $\mathbf{Z} \in \mathcal{Z}$ that together determine the choice of a stochastic function $\tilde{F} : \mathcal{X} \times \mathcal{Z} \rightarrow \mathbb{R}$. The goal is to solve the program

$$\max_{\mathbf{x} \in \mathcal{C}} F(\mathbf{x}) := \max_{\mathbf{x} \in \mathcal{C}} \mathbb{E}_{\mathbf{z} \sim P} [\tilde{F}(\mathbf{x}, \mathbf{z})], \quad (1)$$

where $\mathcal{C}$ is a convex compact set, $\mathbf{z}$ is the realization of the random variable $\mathbf{Z}$ drawn from a distribution P , and $F(\mathbf{x})$ is the expected value of the random functions $\tilde{F}(\mathbf{x}, \mathbf{z})$ with respect to the random variable $\mathbf{Z}$. In this paper, our focus is on the cases where the function F is either concave or continuous submodular. The first case which considers the problem of maximizing a concave function is equivalent to *stochastic convex minimization*, and the second case in which the goal is to maximize a continuous submodular function is called *stochastic continuous submodular maximization*. Note that the main challenge here is to solve Problem (1) without computing the objective function $F(\mathbf{x})$ or its gradient $\nabla F(\mathbf{x})$ explicitly, since we assume that either the probability distribution P is unknown or the cost of computing the expectation is prohibitive.

In this regime, stochastic algorithms are the method of choice since they operate on computationally cheap stochastic estimates of the objective function gradients. Stochastic variants of the proximal gradient method are perhaps the most popular algorithms to tackle this category of problems both in convex minimization and submodular maximization. However, implementation of proximal methods requires projection onto a convex set at each iteration to enforce feasibility, which could be computationally expensive or intractable in many settings. To avoid the cost of projection, recourse to conditional gradient methods arises as a natural alternative. Unlike proximal algorithms, conditional gradient methods do not suffer from computationally costly projection steps and only require solving linear programs which can be implemented at a significantly lower complexity in many practical applications.

In deterministic convex minimization, in which the exact gradient information is given, conditional gradient method, a.k.a., Frank-Wolfe algorithm (Frank and Wolfe, 1956; Jaggi, 2013), succeeds in lowering the computational complexity of proximal algorithms due to its projection free property. However, in the stochastic regime, where only an estimate of the gradient is available, stochastic conditional gradient methods either diverge or underperform the proximal gradient methods. In particular, in stochastic convex minimization it is known and proven that stochastic conditional gradient methods may not converge to the optimal solution without an increasing batch size, whereas stochastic proximal gradient methods converge to the optimal solution at the sublinear rate of $\mathcal{O}(1/\sqrt{t})$. Hence, the possibility of designing a convergent stochastic conditional gradient method with a small batch size remains unanswered.

In deterministic continuous submodular maximization where objective function is submodular, (neither convex nor concave), a variant of the condition gradient method called,

continuous (Calinescu et al., 2011; Bian et al., 2017b) obtains a $(1 - 1/e)$ approximation guarantee for monotone functions, in contrast the best-known result for proximal gradient methods is a $(1/2)$ approximation guarantee (Hassani et al., 2017). However, in stochastic submodular maximization setting, stochastic variants of the continuous greedy algorithm with a small batch size fail to achieve a constant factor approximation (Hassani et al., 2017), whereas stochastic proximal gradient method recovers the $(1/2)$ approximation obtained by proximal gradient method in deterministic settings (Hassani et al., 2017). It is unknown if one can design a stochastic conditional gradient method that obtains a constant approximation guarantee, ideally $(1 - 1/e)$, for Problem (1).

In this paper, we introduce a stochastic conditional gradient method for solving the generic stochastic optimization Problem (1). The proposed method lowers the noise of gradient approximations through a simple gradient averaging technique which only requires a single stochastic gradient computation per iteration, i.e., the batch size can be as small as 1. The proposed stochastic conditional gradient method improves the best-known convergence guarantees for stochastic conditional gradient methods in both convex minimization and submodular maximization settings. A summary of our theoretical contributions follows.

- (i) In *stochastic convex minimization*, we propose a stochastic variant of the Frank-Wolfe algorithm which converges to the optimal objective function value at the sublinear rate of $\mathcal{O}(1/t^{1/3})$. In other words, the proposed SFW algorithm achieves an ϵ -suboptimal objective function value after $\mathcal{O}(1/\epsilon^3)$ stochastic gradient evaluations.
- (ii) In *stochastic continuous submodular maximization*, we propose a stochastic conditional gradient method, which can be interpreted as a stochastic variant of the continuous greedy algorithm, that obtains the first tight $(1 - 1/e)$ approximation guarantee, when the function is monotone. For the non-monotone case, the proposed method obtains a $(1/e)$ approximation guarantee. Moreover, for the more general case of γ -weakly DR-submodular monotone maximization the proposed stochastic conditional gradient method achieves a $(1 - e^{-\gamma})$ -approximation guarantee.
- (iii) In *stochastic discrete submodular maximization*, the proposed stochastic conditional gradient method achieves $(1 - 1/e)$ and $(1/e)$ approximation guarantees for monotone and non-monotone settings, respectively, by maximizing the multilinear extension of the stochastic discrete objective function. Further, if the objective function is monotone and has curvature c the proposed stochastic conditional gradient method achieves a $(1 - e^{-c})/c$ approximation guarantee.

We begin the paper by studying the related work on stochastic methods in convex minimization and submodular maximization (Section 2). Then, we proceed by stating stochastic convex minimization problem (Section 3). We introduce a stochastic conditional gradient method which can be interpreted as a stochastic variant of Frank-Wolfe (FW) algorithm for solving stochastic convex minimization problems (Section 3.1). The proposed **Stochastic Frank-Wolfe** method (SFW) differs from the vanilla FW method in replacing gradients by their stochastic approximations evaluated based on averaging over previously observed stochastic gradients. We further analyze the convergence properties of the proposed SFW method (Section 3.2). In particular, we show that the averaging technique in SFW lowers

the noise of gradient approximation (Lemma 1) and consequently the sequence of objective function values generated by SFW converges to the optimal objective function value at a sublinear rate of $\mathcal{O}(1/t^{1/3})$ in expectation (Theorem 3). We complete this result by proving that the sequence of objective function values almost surely converges to the optimal objective function value (Theorem 4).

We then focus on the application of the proposed stochastic conditional gradient method in continuous submodular maximization (Section 4). After defining the notions of submodularity (Section 4.1), we introduce the **Stochastic Continuous Greedy** (SCG) algorithm for solving continuous submodular maximization problems (Section 4.2). The proposed SCG algorithm achieves the optimal $(1 - 1/e)$ -approximation when the expected objective function is DR-submodular, monotone, and smooth (Theorem 7). To be more precise, the expected objective value of the iterates generated by SCG in expectation is not smaller $(1 - 1/e)OPT - \epsilon$ after $\mathcal{O}(1/\epsilon^3)$ stochastic gradient evaluations. Moreover, for the case that the expected function is not DR-submodular but γ weakly DR-submodular, the SCG algorithm obtains an $(1 - e^{-\gamma})$ -approximation guarantee (Theorem 8). We further extend our results to non-monotone setting by introducing the **Non-monotone Stochastic Continuous Greedy** (NMSCG) method. We show that under the assumptions that the expected function is only DR-submodular and smooth NMSCG reaches a $(1/e)$ -approximation guarantee (Theorem 9).

The continuous multilinear extension of discrete submodular functions implies that the results for stochastic continuous DR-submodular maximization can be extended to stochastic discrete submodular maximization. We formalize this connection by introducing the stochastic discrete submodular maximization problem and defining its continuous multilinear extension (Section 5). By leveraging this connection, one can relax the discrete problem to a stochastic continuous submodular maximization, use SCG to solve the relaxed continuous problem within a $(1 - 1/e - \epsilon)$ approximation to the optimum value (Theorem 12), and use a proper rounding scheme (such as the contention resolution method (Chekuri et al., 2014)) to obtain a feasible set whose value is a $(1 - 1/e - \epsilon)$ approximation to the optimum set in expectation. In summary, we show that SCG achieves an $((1 - 1/e)OPT - \epsilon)$ approximation for a generic discrete monotone stochastic submodular maximization problem after $\mathcal{O}(n^{3/2}/\epsilon^3)$ iterations where n is the size of the ground set (Corollary 13). We further prove a $1/e$ approximation guarantee for NMSCG when we maximize a *non-monotone* stochastic submodular set function (Theorem 15). Moreover, if the expected set function has curvature $c \in [0, 1]$, SCG reaches an $(1/c)(1 - e^{-c})$ approximation guarantee (Theorem 17). We finally close the paper by concluding remarks (Section 7).

Notation. Lowercase boldface $\mathbf{v}$ denotes a vector and uppercase boldface $\mathbf{A}$ denotes a matrix. We use $\|\mathbf{v}\|$ to denote the Euclidean norm of vector $\mathbf{v}$. The i -th element of the vector $\mathbf{v}$ is written as v_i and the element on the i -th row and j -th column of the matrix $\mathbf{A}$ is denoted by $\mathbf{A}_{i,j}$.

2. Related Work

In this section we overview the literature on conditional gradient methods in convex minimization as well as submodular maximization and compare our novel theoretical guarantees with the existing results.

Convex minimization. The problem of minimizing a stochastic convex function subject to a convex constraint has been tackled by many researchers and many approaches have been reported in the literature. Projected stochastic gradient descent (PSGD) and stochastic variants of Frank-Wolfe algorithm are among the most popular approaches. PSGD updates the iterates by descending through the negative direction of stochastic gradient with a proper stepsize and projecting the resulted point onto the feasible set (Robbins and Monro, 1951; Nemirovski and Yudin, 1978; Nemirovskii et al., 1983). It is known that in convex and smooth settings, the iterates of PSGD converge at a rate of $\mathcal{O}(1/\sqrt{t})$, where t is the number of iterations. Although stochastic gradient computation is inexpensive, the cost of projection step can be high (Fujishige and Isotani, 2011) or intractable (Collins et al., 2008). In such cases projection-free conditional gradient methods, a.k.a., Frank-Wolfe algorithm, are more practical (Frank and Wolfe, 1956; Jaggi, 2013). The online Frank-Wolfe algorithm proposed by Hazan and Kale (2012) requires $\mathcal{O}(1/\epsilon^4)$ stochastic gradient evaluations to reach ϵ suboptimal objective function value, i.e., $\mathbb{E}[f(\mathbf{x}) - OPT] \leq \epsilon$ under the assumption that the objective function is convex and has bounded gradients. The stochastic Frank-Wolfe studied by Hazan and Luo (2016) obtains an improved complexity of $\mathcal{O}(1/\epsilon^3)$ under the assumptions that the expected objective function is smooth (has Lipschitz gradients) and Lipschitz continuous (the gradients are bounded). More importantly, the stochastic Frank-Wolfe algorithm in (Hazan and Luo, 2016) requires an increasing batch size b as time progresses, i.e., $b = \mathcal{O}(t)$. In this paper, we propose a stochastic variant of conditional gradient method which achieves the complexity of $\mathcal{O}(1/\epsilon^3)$ under milder assumptions (only requires smoothness of the expected function) and operates with a fixed batch size, e.g., $b = 1$.

Submodular maximization. Maximizing a deterministic submodular set function has been extensively studied. The celebrated result of Nemhauser et al. (1978) shows that a greedy algorithm achieves a $(1 - 1/e)$ approximation guarantee for a monotone function subject to a cardinality constraint. It is also known that this result is tight under reasonable complexity-theoretic assumptions (Feige, 1998). Recently, variants of the greedy algorithm have been proposed to extend the above result to non-monotone and more general constraints (Feige et al., 2011; Buchbinder et al., 2015, 2014; Mirzasoleiman et al., 2016; Feldman et al., 2017). While discrete greedy algorithms are fast, they usually do not provide the tightest guarantees for many classes of feasibility constraints. This is why continuous relaxations of submodular functions, e.g., the multilinear extension, have gained a lot of interest (Vondrák, 2008; Calinescu et al., 2011; Chekuri et al., 2014; Feldman et al., 2011; Gharan and Vondrák, 2011; Sviridenko et al., 2015). In particular, it is known that the continuous greedy algorithm achieves a $(1 - 1/e)$ approximation guarantee for monotone submodular functions under a general matroid constraint (Calinescu et al., 2011). An improved $((1 - e^{-c})/c)$ -approximation guarantee can be obtained if the objective function has curvature c (Vondrák, 2010). Moreover, the best-known result for the case that the objective function has curvature c is shown by (Sviridenko et al., 2015) which is a $((1 - c)/e)$ -approximation guarantee. The problem of maximizing submodular functions also has been studied for the non-monotone case, and constant approximation guarantees have been established (Feldman et al., 2011; Buchbinder et al., 2015; Ene and Nguyen, 2016; Buchbinder and Feldman, 2016).

Ref.	setting	assumptions	batch	rate	complexity
Jaggi (2013)	det.	smooth	—	$\mathcal{O}(1/t)$	—
Hazan and Kale (2012)	stoch.	smooth, bounded grad.	$\mathcal{O}(t)$	$\mathcal{O}(1/t^{1/2})$	$\mathcal{O}(1/\epsilon^4)$
Hazan and Luo (2016)	stoch.	smooth, bounded grad.	$\mathcal{O}(t^2)$	$\mathcal{O}(1/t)$	$\mathcal{O}(1/\epsilon^3)$
This paper	stoch.	smooth, bounded var.	$\mathcal{O}(1)$	$\mathcal{O}(1/t^{1/3})$	$\mathcal{O}(1/\epsilon^3)$

Table 1: Convergence guarantees of conditional gradient (FW) methods for convex minimization

Ref.	setting	function	const.	utility	complexity
Chekuri et al. (2015)	det.	mon.smooth sub.	poly.	$(1 - 1/e)\text{OPT} - \epsilon$	$\mathcal{O}(1/\epsilon^2)$
Bian et al. (2017b)	det.	mon. DR-sub.	cvx-down	$(1 - 1/e)\text{OPT} - \epsilon$	$\mathcal{O}(1/\epsilon)$
Bian et al. (2017a)	det.	non-mon. DR-sub.	cvx-down	$(1/e)\text{OPT} - \epsilon$	$\mathcal{O}(1/\epsilon)$
Hassani et al. (2017)	det.	mon. DR-sub.	convex	$(1/2)\text{OPT} - \epsilon$	$\mathcal{O}(1/\epsilon)$
Hassani et al. (2017)	stoch.	mon. DR-sub.	convex	$(1/2)\text{OPT} - \epsilon$	$\mathcal{O}(1/\epsilon^2)$
Hassani et al. (2017)	stoch.	mon. weak DR-sub.	convex	$\frac{\gamma^2}{1+\gamma^2}\text{OPT} - \epsilon$	$\mathcal{O}(1/\epsilon^2)$
This paper	stoch.	mon. DR-sub.	convex	$(1 - 1/e)\text{OPT} - \epsilon$	$\mathcal{O}(1/\epsilon^3)$
This paper	stoch.	weak DR-sub.	convex	$(1 - e^{-\gamma})\text{OPT} - \epsilon$	$\mathcal{O}(1/\epsilon^3)$
This paper	stoch.	non-mon. DR-sub.	convex	$(1/e)\text{OPT} - \epsilon$	$\mathcal{O}(1/\epsilon^3)$

Table 2: Convergence guarantees for continuous DR-submodular function maximization

Continuous submodularity naturally arises in many learning applications such as robust budget allocation (Staib and Jegelka, 2017; Soma et al., 2014), online resource allocation (Eghbali and Fazel, 2016), learning assignments (Golovin et al., 2014), as well as Adwords for e-commerce and advertising (Devanur and Jain, 2012; Mehta et al., 2007). Maximizing a *deterministic* continuous submodular function dates back to the work of Wolsey (1982). More recently, Chekuri et al. (2015) proposed a multiplicative weight update algorithm that achieves a $(1 - 1/e - \epsilon)$ approximation guarantee after $\tilde{\mathcal{O}}(n/\epsilon^2)$ oracle calls to gradients of a monotone smooth submodular function F (i.e., twice differentiable DR-submodular) subject to a polytope constraint. A similar approximation factor can be obtained after $\mathcal{O}(n/\epsilon)$ oracle calls to gradients of F for monotone DR-submodular functions subject to a down-closed convex body using the continuous greedy method (Bian et al., 2017b). However, such results require exact computation of the gradients ∇F which is not feasible in Problem (14). An alternative approach is then to modify the current algorithms by replacing gradients $\nabla F(\mathbf{x}_t)$ by their stochastic estimates $\nabla \tilde{F}(\mathbf{x}_t, \mathbf{z}_t)$; however, this modification may lead to arbitrarily poor solutions as demonstrated in (Hassani et al., 2017). Another alternative is to estimate the gradient by averaging over a (large) mini-batch of samples at each iteration. While this approach can potentially reduce the noise variance, it increases the computational complexity of each iteration and is not favorable. The work by Hassani et al. (2017) is perhaps the first attempt to solve Problem (14) only by executing stochastic estimates of gradients (without using a large batch). They showed that the stochastic gradient ascent method achieves a $(1/2 - \epsilon)$ approximation guarantee after $\mathcal{O}(1/\epsilon^2)$ iterations. Although this work opens the door for maximizing stochastic continuous submodular functions using computa-

Ref.	setting	function	constraint	approx	method
Nemhauser and Wolsey (1981)	det.	mon. sub.	cardinality	$1 - 1/e$	disc. greedy
Nemhauser and Wolsey (1981)	det.	mon. sub.	matroid	$1/2$	disc. greedy
Calinescu et al. (2011)	det.	mon. sub.	matroid	$1 - 1/e$	con. greedy
Hassani et al. (2017)	stoch.	mon. sub.	matroid	$1/2$	SGA
This paper	stoch.	mon. sub.	matroid	$1 - 1/e$	SCG
This paper	stoch.	mon. sub.	matroid	$(1 - 1/e^c)/c$	SCG
This paper	stoch.	sub.	matroid	$1/e$	NMSCG

Table 3: Convergence guarantees for submodular set function maximization

tionally cheap stochastic gradients, it fails to achieve the optimal $(1 - 1/e)$ approximation. To close the gap, we propose in this paper **Stochastic Continuous Greedy** which outputs a solution with function value at least $((1 - 1/e)\text{OPT} - \epsilon)$ after $O(1/\epsilon^3)$ iterations. Notably, our result only requires the expected function F to be monotone and DR-submodular and the stochastic functions $\tilde{F}$ need not be monotone nor DR-submodular. Moreover, in contrast to the result in (Bian et al., 2017b), which holds for down-closed convex constraints, our result holds for any convex constraints. For non-monotone DR-submodular functions, we also propose the non-monotone stochastic continuous greedy (NMSCG) method that achieves a solution with function value at least $((1/e)\text{OPT} - \epsilon)$ after at most $O(1/\epsilon^3)$ iterations. Crucially, the feasible set in this case should be down-closed or otherwise no constant approximation guarantee is possible (Chekuri et al., 2014).

Our result also has important implications for the problem of maximizing a stochastic discrete submodular function subject to a matroid constraint. Since the proposed SCG method works in stochastic settings, we can relax the discrete objective function f to a continuous function F through the multi-linear extension (note that expectation is a linear operator). Then we can maximize F within a $(1 - 1/e - \epsilon)$ approximation to the optimum value by using only $O(1/\epsilon^3)$ oracle calls to the stochastic gradients of F when the functions are monotone. Finally, a proper rounding scheme (such as the contention resolution method (Chekuri et al., 2014)) results in a feasible set whose value is a $(1 - 1/e)$ approximation to the optimum set in expectation¹. Using the same procedure we can also prove a $(1/e)$ approximation guarantee in expectation for the non-monotone case. Additionally, when the set function f is monotone and has a curvature $c < 1$ – check (41) for the definition of the curvature – we show that the approximation factor can be improved from $(1 - 1/e)$ to $((1 - 1/e^c)/c)$.

3. Stochastic Convex Minimization

Many problems in machine learning can be reduced to the minimization of a convex objective function defined as an expectation over a set of random functions. In this section, our focus

1. For the ease of presentation, and in the discrete setting, we only present our results for the matroid constraint. However, our stochastic continuous algorithms can provide constant factor approximations for any constrained submodular maximization setting where an efficient and loss-less rounding scheme exists. It includes, for example, knapsack constraints among many others.

is on developing a stochastic variant of the conditional gradient method, a.k.a., Frank-Wolfe, which can be applied to solve general stochastic convex minimization problems. To make the notation consistent with other works on stochastic convex minimization, instead of solving Problem (1) for the case that the objective function F is concave, we assume that F is convex and intend to minimize it subject to a convex set $\mathcal{C}$. Therefore, the goal is to solve the program

$$\min_{\mathbf{x} \in \mathcal{C}} F(\mathbf{x}) := \min_{\mathbf{x} \in \mathcal{C}} \mathbb{E}_{\mathbf{z} \sim P} [\tilde{F}(\mathbf{x}, \mathbf{z})]. \quad (2)$$

A canonical subset of problems having this form is support vector machines, least mean squares, and logistic regression.

In this section we assume that only the expected (average) function F is convex and smooth, and the stochastic functions $\tilde{F}$ may not be convex nor smooth. Since the expected function F is convex, descent methods can be used to solve the program in (2). However, computing the gradients (or Hessians) of the function F precisely requires access to the distribution P , which may not be feasible in many applications. To be more specific, we are interested in settings where the distribution P is either unknown or evaluation of the expected value is computationally prohibitive. In this regime, stochastic gradient descent methods, which operate on stochastic approximations of the gradients, are the mostly used alternatives. In the following section, we aim to develop a stochastic variant of the Frank-Wolfe method which converges to an optimal solution of (2), while it only requires access to a single stochastic gradient $\nabla \tilde{F}(\mathbf{x}, \mathbf{z})$ at each iteration.

3.1. Stochastic Frank-Wolfe Method

In this section, we introduce a stochastic variant of the Frank-Wolfe method to solve Problem (2). Assume that at each iteration we have access to the stochastic gradient $\nabla \tilde{F}(\mathbf{x}, \mathbf{z})$ which is an unbiased estimate of the gradient $\nabla F(\mathbf{x})$. It is known that a naive stochastic implementation of Frank-Wolfe (replacing gradient $\nabla F(\mathbf{x})$ by $\nabla \tilde{F}(\mathbf{x}, \mathbf{z})$) might diverge, due to non-vanishing variance of gradient approximations. To resolve this issue, we introduce a stochastic version of the Frank-Wolfe algorithm which reduces the noise of gradient approximations via a common averaging technique in stochastic optimization (Ruszczyński, 1980, 2008; Yang et al., 2016; Mokhtari et al., 2017).

By letting $t \in \mathbf{N}$ be a discrete time index and ρ_t a given stepsize which approaches zero as t grows, the proposed biased gradient estimation $\mathbf{d}_t$ is defined by the recursion

$$\mathbf{d}_t = (1 - \rho_t)\mathbf{d}_{t-1} + \rho_t \nabla \tilde{F}(\mathbf{x}_t, \mathbf{z}_t), \quad (3)$$

where the initial vector is given as $\mathbf{d}_0 = \mathbf{0}$. We will show that the averaging technique in (3) reduces the noise of gradient approximation as time increases. More formally, the expected noise of the gradient estimation $\mathbb{E} [\|\mathbf{d}_t - \nabla F(\mathbf{x}_t)\|^2]$ approaches zero asymptotically. This property implies that the biased gradient estimate $\mathbf{d}_t$ is a better candidate for approximating the gradient $\nabla F(\mathbf{x}_t)$ comparing to the unbiased gradient estimate $\nabla \tilde{F}(\mathbf{x}_t, \mathbf{z}_t)$ that suffers from a high variance approximation. We therefore define the descent direction $\mathbf{v}_t$ of our proposed Stochastic Frank-Wolfe (SFW) method as the solution of the linear program

$$\mathbf{v}_t \in \operatorname{argmin}_{\mathbf{v} \in \mathcal{C}} \{\mathbf{d}_t^T \mathbf{v}\}. \quad (4)$$

Algorithm 1 Stochastic Frank-Wolfe (SFW)

Require: Stepsizes $\rho_t > 0$ and $\gamma_t > 0$. Initialize $\mathbf{d}_0 = \mathbf{0}$ and choose $\mathbf{x}_0 \in \mathcal{C}$

- 1: **for** $t = 1, 2, \dots$ **do**
 - 2: Update the gradient estimate $\mathbf{d}_t = (1 - \rho_t)\mathbf{d}_{t-1} + \rho_t \nabla \tilde{F}(\mathbf{x}_t, \mathbf{z}_t)$;
 - 3: Compute $\mathbf{v}_t \in \operatorname{argmin}_{\mathbf{v} \in \mathcal{C}} \{\mathbf{d}_t^T \mathbf{v}\}$;
 - 4: Compute the updated variable $\mathbf{x}_{t+1} = (1 - \gamma_{t+1})\mathbf{x}_t + \gamma_{t+1}\mathbf{v}_t$;
 - 5: **end for**
-

As in the traditional FW method, the updated variable $\mathbf{x}_{t+1}$ is a convex combination of $\mathbf{v}_t$ and the iterate $\mathbf{x}_t$

$$\mathbf{x}_{t+1} = (1 - \gamma_{t+1})\mathbf{x}_t + \gamma_{t+1}\mathbf{v}_t, \quad (5)$$

where γ_t is a proper positive stepsize. Note that each iteration of the proposed SFW method only requires a single stochastic gradient computation, unlike the methods in (Hazan and Luo, 2016; Reddi et al., 2016) which require increasing number of stochastic gradient evaluations as the number of iterations t grows.

The proposed SFW is summarized in Algorithm 1. The core steps are Steps 2-5 which follow the updates in (3)-(5). The initial variable $\mathbf{x}_0$ can be any feasible vector in the convex set $\mathcal{C}$ and the initial gradient estimate is set to be the null vector $\mathbf{d}_0 = \mathbf{0}$. The sequence of positive parameters ρ_t and γ_t should be diminishing at proper rates as we describe in the following convergence analysis section.

3.2. Convergence Analysis

In this section we study the convergence rate of the proposed SFW method for solving the constraint convex program in (2). To do so, we first assume that the following conditions hold.

Assumption 1 *The convex set $\mathcal{C}$ is bounded with diameter D , i.e., for all $\mathbf{x}, \mathbf{y} \in \mathcal{C}$ we can write*

$$\|\mathbf{x} - \mathbf{y}\| \leq D. \quad (6)$$

Assumption 2 *The expected function F is convex. Moreover, its gradients ∇F are L -Lipschitz continuous over the set $\mathcal{C}$, i.e., for all $\mathbf{x}, \mathbf{y} \in \mathcal{C}$*

$$\|\nabla F(\mathbf{x}) - \nabla F(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|. \quad (7)$$

Assumption 3 *The variance of the unbiased stochastic gradients $\nabla \tilde{F}(\mathbf{x}, \mathbf{z})$ is bounded above by σ^2 , i.e., for all random variables $\mathbf{z}$ and vectors $\mathbf{x} \in \mathcal{C}$ we can write*

$$\mathbb{E} \left[\|\nabla \tilde{F}(\mathbf{x}, \mathbf{z}) - \nabla F(\mathbf{x})\|^2 \right] \leq \sigma^2. \quad (8)$$

Assumption 1 is standard in constrained convex optimization and is implied by the fact that the set $\mathcal{C}$ is convex and compact. The condition in Assumption 2 ensures that the objective function F is smooth on the set $\mathcal{C}$. Note that here we only assume that the (average) function F has Lipschitz continuous gradients, and the stochastic gradients $\nabla \tilde{F}(\mathbf{x}, \mathbf{z})$ may not be Lipschitz continuous. Finally, the required condition in Assumption 3 is customary

in stochastic optimization and guarantees that the variance of stochastic gradients $\nabla \tilde{F}(\mathbf{x}, \mathbf{z})$ is bounded by a finite constant $\sigma^2 < \infty$.

To study the convergence rate of SFW, we first derive an upper bound on the error of gradient approximation $\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2$ in the following lemma.

Lemma 1 *Consider the proposed **Stochastic Frank-Wolfe (SFW)** method outlined in Algorithm 1. If the conditions in Assumptions 1-3 hold, the sequence of squared gradient errors $\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2$ satisfies*

$$\mathbb{E} [\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2 \mid \mathcal{F}_t] \leq \left(1 - \frac{\rho_t}{2}\right) \|\nabla F(\mathbf{x}_{t-1}) - \mathbf{d}_{t-1}\|^2 + \rho_t^2 \sigma^2 + \frac{2L^2 D^2 \gamma_t^2}{\rho_t}, \quad (9)$$

where $\mathcal{F}_t$ is a sigma-algebra measuring all sources of randomness up to step t .

Proof Check Appendix A. ■

The result in Lemma 1 shows that squared error of gradient approximation $\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2$ decreases in expectation at each iteration by the factor $(1 - \rho_t/2)$ if the remaining terms on the right hand side of (9) are negligible relative to the term $(1 - \rho_t/2)\|\nabla F(\mathbf{x}_{t-1}) - \mathbf{d}_{t-1}\|^2$. This condition can be satisfied, if the parameters ρ_t and γ_t are properly chosen. This observation verifies our intuition that the noise of the stochastic gradient approximation diminishes as the number of iterations increases.

In the following lemma, we derive an upper bound on the suboptimality $F(\mathbf{x}_{t+1}) - F(\mathbf{x}^*)$ which depends on the norm of gradient error $\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|$.

Lemma 2 *Consider the proposed **Stochastic Frank-Wolfe (SFW)** method outlined in Algorithm 1. If the conditions in Assumptions 1-3 are satisfied, the suboptimality $F(\mathbf{x}_{t+1}) - F(\mathbf{x}^*)$ satisfies*

$$F(\mathbf{x}_{t+1}) - F(\mathbf{x}^*) \leq (1 - \gamma_{t+1})(F(\mathbf{x}_t) - F(\mathbf{x}^*)) + \gamma_{t+1} D \|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\| + \frac{LD^2 \gamma_{t+1}^2}{2}. \quad (10)$$

Proof Check Appendix B. ■

Based on Lemma 2, the suboptimality $F(\mathbf{x}_{t+1}) - F(\mathbf{x}^*)$ approaches zero if the error of gradient approximation $\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|$ converges to zero sufficiently fast and the last term $(LD^2 \gamma_{t+1}^2)/2$ is summable, i.e., $\sum_{t=1}^{\infty} \gamma_t^2 < \infty$. In the special case of zero approximation error, which is equivalent to the case that we have access to the expected gradient $\nabla F(\mathbf{x}_t)$, by setting $\gamma_t = \mathcal{O}(1/t)$ it can be shown that the suboptimality $F(\mathbf{x}_t) - F(\mathbf{x}^*)$ converges to zero at the sublinear rate of $\mathcal{O}(1/t)$. Therefore, the result in Lemma 2 is consistent with the analysis of Frank-Wolfe method (Jaggi, 2013).

In the following theorem, by using the results in Lemmas 1 and 2, we establish an upper bound on the expected suboptimality $\mathbb{E}[F(\mathbf{x}_t) - F(\mathbf{x}^*)]$.

Theorem 3 *Consider the proposed **Stochastic Frank-Wolfe (SFW)** method outlined in Algorithm 1. Suppose the conditions in Assumptions 1-3 are satisfied. If we set $\gamma_t = 2/(t+8)$*

and $\rho_t = 4/(t+8)^{2/3}$, then the expected suboptimality $\mathbb{E}[F(\mathbf{x}_{t+1}) - F(\mathbf{x}^*)]$ is bounded above by

$$\mathbb{E}[F(\mathbf{x}_t) - F(\mathbf{x}^*)] \leq \frac{Q}{(t+9)^{1/3}}, \quad (11)$$

where the constant Q is given by

$$Q := \max \left\{ 9^{1/3}(F(\mathbf{x}_0) - F(\mathbf{x}^*)), \frac{LD^2}{2} + 2D \max \left\{ 3\|\nabla F(\mathbf{x}_0) - \mathbf{d}_0\|, (16\sigma^2 + 2L^2D^2)^{1/2} \right\} \right\}. \quad (12)$$

Proof Check Appendix C. ■

The result in Theorem 3 indicates that the expected suboptimality $\mathbb{E}[F(\mathbf{x}_t) - F(\mathbf{x}^*)]$ of the iterates generated by the SFW method converges to zero at least at a sublinear rate of $\mathcal{O}(1/t^{1/3})$. In other words, it shows that to achieve the expected suboptimality $\mathbb{E}[F(\mathbf{x}_t) - F(\mathbf{x}^*)] \leq \epsilon$, the number of required stochastic gradients (sample gradients) to reach this accuracy is $\mathcal{O}(1/\epsilon^3)$.

To complete the convergence analysis of SFW we also prove that the sequence of the objective function values $F(\mathbf{x}_t)$ converges to the optimal value $F(\mathbf{x}^*)$ almost surely. This result is formalized in the following theorem.

Theorem 4 *Consider the proposed **Stochastic Frank-Wolfe (SFW)** method outlined in Algorithm 1. Suppose that the conditions in Assumptions 1-3 are satisfied. If we choose ρ_t and γ_t such that (i) $\sum_{t=0}^{\infty} \rho_t = \infty$, (ii) $\sum_{t=0}^{\infty} \rho_t^2 < \infty$, (iii) $\sum_{t=0}^{\infty} \gamma_t = \infty$, and (iv) $\sum_{t=0}^{\infty} (\gamma_t^2/\rho_t) < \infty$, then the suboptimality $F(\mathbf{x}_t) - F(\mathbf{x}^*)$ converges to zero almost surely, i.e.,*

$$\lim_{t \rightarrow \infty} F(\mathbf{x}_t) - F(\mathbf{x}^*) \stackrel{\text{a.s.}}{=} 0. \quad (13)$$

Proof Check Appendix D. ■

Theorem 4 provides almost sure convergence of the sequence of objective function value $F(\mathbf{x}_t)$ to the optimal value $F(\mathbf{x}^*)$. In other words it shows that the sequence of the objective function values $F(\mathbf{x}_t)$ converges to $F(\mathbf{x}^*)$ with probability 1. Indeed, a valid set of choices for γ_t and ρ_t to satisfy the required conditions in Theorem 4 are $\gamma_t = \mathcal{O}(1/t)$ and $\rho_t = \mathcal{O}(1/t^{2/3})$.

4. Stochastic Continuous Submodular Maximization

In the previous section, we focused on the convex setting, but what if the objective function F is not convex? In this section, we consider a broad class of non-convex optimization problems that possess special combinatorial structures. More specifically, we focus on constrained maximization of stochastic continuous submodular functions that demonstrate diminishing returns, i.e., continuous DR-submodular functions,

$$\max_{\mathbf{x} \in \mathcal{C}} F(\mathbf{x}) \doteq \max_{\mathbf{x} \in \mathcal{C}} \mathbb{E}_{\mathbf{z} \sim P} [\tilde{F}(\mathbf{x}, \mathbf{z})]. \quad (14)$$

As before, the functions $\tilde{F} : \mathcal{X} \times \mathcal{Z} \rightarrow \mathbb{R}_+$ are stochastic where $\mathbf{x} \in \mathcal{X}$ is the optimization variable, $\mathbf{z} \in \mathcal{Z}$ is a realization of the random variable $\mathbf{Z}$ drawn from a distribution P , and $\mathcal{X} \subseteq \mathbb{R}_+^n$ is a compact set. In this section, we assume that $F(\mathbf{x}) \geq 0$ for all $\mathbf{x} \in \mathcal{X}$.

Our goal is to maximize the expected value of the random functions $\tilde{F}(\mathbf{x}, \mathbf{z})$ over the convex body $\mathcal{C} \subseteq \mathcal{X}$. Note that we *only assume* that $F(\mathbf{x})$ is DR-submodular, and *not* necessarily the stochastic functions $\tilde{F}(\mathbf{x}, \mathbf{z})$. We also consider situations where the distribution P is either unknown (e.g., when the objective is given as an implicit stochastic model) or the domain of the random variable $\mathbf{Z}$ is very large (e.g., when the objective is defined in terms of an empirical risk) which makes the cost of computing the expectation very high. In these regimes, stochastic optimization methods, which operate on computationally cheap estimates of gradients, arise as natural solutions. In fact, very recently, it was shown in (Hassani et al., 2017) that stochastic gradient methods achieve a $(1/2)$ approximation guarantee to Problem (14). In Section 3 of (Hassani et al., 2017), the authors also showed that if we simply substitute gradients by stochastic gradients in the update of conditional gradient methods (a.k.a., Frank-Wolfe), such as continuous greedy (Vondrák, 2008) or its close variant (Bian et al., 2017b), the resulted method can perform arbitrarily poorly in stochastic continuous submodular maximization settings. Our goal in this section is to design a stable stochastic variant of conditional gradient method to solve Problem (14) up to a constant factor.

4.1. Preliminaries

We begin by recalling the definition of a submodular set function: A function $f : 2^V \rightarrow \mathbb{R}_+$, defined on the ground set V , is called submodular if for all subsets $A, B \subseteq V$, we have

$$f(A) + f(B) \geq f(A \cap B) + f(A \cup B).$$

The notion of submodularity goes beyond the discrete domain (Wolsey, 1982; Vondrák, 2007; Bach, 2015). Consider a continuous function $F : \mathcal{X} \rightarrow \mathbb{R}_+$ where the set $\mathcal{X}$ is of the form $\mathcal{X} = \prod_{i=1}^n \mathcal{X}_i$ and each $\mathcal{X}_i$ is a compact subset of $\mathbb{R}_+$. We call the continuous function F submodular if for all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ we have

$$F(\mathbf{x}) + F(\mathbf{y}) \geq F(\mathbf{x} \vee \mathbf{y}) + F(\mathbf{x} \wedge \mathbf{y}), \quad (15)$$

where $\mathbf{x} \vee \mathbf{y} := \max(\mathbf{x}, \mathbf{y})$ (component-wise) and $\mathbf{x} \wedge \mathbf{y} := \min(\mathbf{x}, \mathbf{y})$ (component-wise). Further, a submodular function F is monotone (on the set $\mathcal{X}$) if

$$\mathbf{x} \leq \mathbf{y} \implies F(\mathbf{x}) \leq F(\mathbf{y}), \quad (16)$$

for all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$. Note that $\mathbf{x} \leq \mathbf{y}$ in (16) means that $x_i \leq y_i$ for all $i = 1, \dots, n$. Furthermore, a differentiable submodular function F is called *DR-submodular* (i.e., shows diminishing returns) if the gradients are antitone, namely, for all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ we have

$$\mathbf{x} \leq \mathbf{y} \implies \nabla F(\mathbf{x}) \geq \nabla F(\mathbf{y}). \quad (17)$$

See (Eghbali and Fazel, 2016) for related definitions and check (Bilmes and Bai, 2017) for more examples of DR-submodular functions. When the function F is twice differentiable,

submodularity implies that all cross-second-derivatives are non-positive (Bach, 2015), i.e.,

$$\text{for all } i \neq j, \text{ for all } \mathbf{x} \in \mathcal{X}, \quad \frac{\partial^2 F(\mathbf{x})}{\partial x_i \partial x_j} \leq 0, \quad (18)$$

and DR-submodularity implies that all second-derivatives are non-positive (Bian et al., 2017b), i.e.,

$$\text{for all } i, j, \text{ for all } \mathbf{x} \in \mathcal{X}, \quad \frac{\partial^2 F(\mathbf{x})}{\partial x_i \partial x_j} \leq 0. \quad (19)$$

4.2. Stochastic Continuous Greedy

We proceed to introduce, **Stochastic Continuous Greedy** (SCG), which is a stochastic variant of the continuous greedy method (Vondrák, 2008) to solve Problem (14). We only assume that the expected objective function F is monotone and DR-submodular and the stochastic functions $\tilde{F}(\mathbf{x}, \mathbf{z})$ may not be monotone nor submodular. Since the objective function F is monotone and DR-submodular, continuous greedy algorithm (Calinescu et al., 2011; Bian et al., 2017b) can be used in principle to solve Problem (14). Note that each update of continuous greedy requires computing the gradient of F , i.e., $\nabla F(\mathbf{x}) := \mathbb{E}[\nabla \tilde{F}(\mathbf{x}, \mathbf{z})]$. However, if we only have access to the (computationally cheap) stochastic gradients $\nabla \tilde{F}(\mathbf{x}, \mathbf{z})$, then the continuous greedy method will not be directly usable (Hassani et al., 2017). This limitation is due to the non-vanishing variance of gradient approximations. To resolve this issue, we use the gradient averaging technique in Section 3.1. As in SFW, we define the estimated gradient $\mathbf{d}_t$ by the recursion

$$\mathbf{d}_t = (1 - \rho_t)\mathbf{d}_{t-1} + \rho_t \nabla \tilde{F}(\mathbf{x}_t, \mathbf{z}_t), \quad (20)$$

where ρ_t is a positive stepsize and the initial vector is defined as $\mathbf{d}_0 = \mathbf{0}$. We therefore define the ascent direction $\mathbf{v}_t$ of our proposed SCG method as follows

$$\mathbf{v}_t \in \operatorname{argmax}_{\mathbf{v} \in \mathcal{C}} \{\mathbf{d}_t^T \mathbf{v}\}, \quad (21)$$

which is a linear objective maximization over the convex set $\mathcal{C}$. Indeed, if instead of the gradient estimate $\mathbf{d}_t$ we use the exact gradient $\nabla F(\mathbf{x}_t)$ for the updates in (21), the continuous greedy update will be recovered. Here, as in continuous greedy, the initial decision vector is the null vector, $\mathbf{x}_0 = \mathbf{0}$. Further, the stepsize for updating the iterates is equal to $1/T$, and the variable $\mathbf{x}_t$ is updated as

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \frac{1}{T} \mathbf{v}_t. \quad (22)$$

The stepsize $1/T$ and the initialization $\mathbf{x}_0 = \mathbf{0}$ ensure that after T iterations the variable $\mathbf{x}_T$ ends up in the convex set $\mathcal{C}$. We would like to highlight that the convex body $\mathcal{C}$ may not be down-closed or contain $\mathbf{0}$. Nonetheless, the final iterate $\mathbf{x}_T$ returned by SCG will be a feasible point in $\mathcal{C}$. The steps of the proposed SCG method are outlined in Algorithm 2. Note that the major difference between SFW in Algorithm 1 and SCG in Algorithm 2 is in Step 4 where the variable $\mathbf{x}_{t+1}$ is computed. In SFW, $\mathbf{x}_{t+1}$ is a convex combination of $\mathbf{x}_t$ and $\mathbf{v}_t$, while in SCG $\mathbf{x}_{t+1}$ is computed by moving from $\mathbf{x}_t$ towards the direction $\mathbf{v}_t$ with the stepsize $1/T$.

Algorithm 2 Stochastic Continuous Greedy (SCG)

Require: Stepsizes $\rho_t > 0$. Initialize $\mathbf{d}_0 = \mathbf{x}_0 = \mathbf{0}$

- 1: **for** $t = 1, 2, \dots, T$ **do**
 - 2: Compute $\mathbf{d}_t = (1 - \rho_t)\mathbf{d}_{t-1} + \rho_t \nabla \tilde{F}(\mathbf{x}_t, \mathbf{z}_t)$;
 - 3: Compute $\mathbf{v}_t \in \operatorname{argmax}_{\mathbf{v} \in \mathcal{C}} \{\mathbf{d}_t^T \mathbf{v}\}$;
 - 4: Update the variable $\mathbf{x}_{t+1} = \mathbf{x}_t + \frac{1}{T} \mathbf{v}_t$;
 - 5: **end for**
-

Remark 5 *Note that the update schemes for the convex case and submodular case are slightly different. In the convex case, we follow the update in (5), while in the submodular setting, we use the update in (22) to compute the new iterate. This modification is essential to achieve a $(1 - 1/e)$ -approximation guarantee for the monotone continuous submodular setting. To better highlight this point, note that if we use an update rule of the form (5) for the submodular setting it could lead to a $1/2$ -approximation. In other words, by starting from a feasible point and using the update rule in (5), SCG will eventually converge to a stationary point of the problem, when $\mathbf{v}_t$ is computed according to the conditional gradient update. Converging to a stationary point in the convex setting is indeed desirable as any first-order stationary point in the convex setting is an optimal solution. However, in the continuous submodular setting, converging to a stationary point might lead to a $1/2$ -approximation guarantee. In fact, in (Hassani et al., 2017) it has been shown that for continuous submodular functions the value of objective function for a local maximum could be at most $1/2$ of the optimal objective function value. We refer the reader to (Hassani et al., 2017) for the example that shows the existence of such local maximum. By starting from the origin $\mathbf{0}$ (which is not a stationary point) and following the update rule in (22), SCG finds a path in the feasible set that without converging to a stationary point leads to an approximation factor of $1 - 1/e$.*

We proceed to study the convergence properties of our proposed SCG method for solving Problem (14). To do so, we first assume that the following conditions hold.

Assumption 4 *The Euclidean norm of the elements in the constraint set $\mathcal{C}$ are uniformly bounded, i.e., for all $\mathbf{x} \in \mathcal{C}$ we can write*

$$\|\mathbf{x}\| \leq D. \quad (23)$$

Assumption 5 *The function F is DR-submodular and monotone. Further, its gradients are L -Lipschitz continuous over the set $\mathcal{X}$, i.e., for all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$*

$$\|\nabla F(\mathbf{x}) - \nabla F(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|. \quad (24)$$

Assumption 6 *The variance of the unbiased stochastic gradients $\nabla \tilde{F}(\mathbf{x}, \mathbf{z})$ is bounded above by σ^2 , i.e., for any vector $\mathbf{x} \in \mathcal{X}$ we can write*

$$\mathbb{E} \left[\|\nabla \tilde{F}(\mathbf{x}, \mathbf{z}) - \nabla F(\mathbf{x})\|^2 \right] \leq \sigma^2, \quad (25)$$

where the expectation is with respect to the randomness of $\mathbf{z} \sim P$.

Due to the initialization step of SCG (i.e., starting from $\mathbf{0}$) we need a bound on the furthest feasible solution from $\mathbf{0}$ that we can end up with; and such a bound is guaranteed by Assumption 4. The condition in Assumption 5 ensures that the objective function F is smooth. Note again that $\nabla \tilde{F}(\mathbf{x}, \mathbf{z})$ may or may not be Lipschitz continuous. Finally, the required condition in Assumption 6 guarantees that the variance of stochastic gradients $\nabla \tilde{F}(\mathbf{x}, \mathbf{z})$ is bounded by a finite constant $\sigma^2 < \infty$. Note that Assumptions 5-6 are stronger than Assumptions 2-3 since they ensure smoothness and bounded gradients for all points $\mathbf{x} \in \mathcal{X}$ and not only for the feasible points $\mathbf{x} \in \mathcal{C}$.

To study the convergence of SCG, we first derive an upper bound for the expected error of gradient approximation (i.e., $\mathbb{E}[\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2]$) in the following lemma.

Lemma 6 *Consider **Stochastic Continuous Greedy (SCG)** outlined in Algorithm 2. If Assumptions 4-6 are satisfied and $\rho_t = \frac{4}{(t+8)^{2/3}}$, then for $t = 0, \dots, T$ we have*

$$\mathbb{E}[\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2] \leq \frac{Q}{(t+9)^{2/3}}, \quad (26)$$

where $Q := \max\{5\|\nabla F(\mathbf{x}_0) - \mathbf{d}_0\|^2, 16\sigma^2 + 2L^2D^2\}$.

Proof Check Appendix E. ■

Let us now use the result of Lemma 6 to show that the sequence of iterates generated by SCG reaches a $(1 - 1/e)$ approximation guarantee for Problem (14).

Theorem 7 *Consider **Stochastic Continuous Greedy (SCG)** outlined in Algorithm 2. If Assumptions 4-6 are satisfied and $\rho_t = \frac{4}{(t+8)^{2/3}}$, then the expected objective function value for the iterates generated by SCG satisfies the inequality*

$$\mathbb{E}[F(\mathbf{x}_T)] \geq (1 - 1/e)OPT - \frac{15DQ^{1/2}}{T^{1/3}} - \frac{LD^2}{2T}, \quad (27)$$

where $OPT = \max_{\mathbf{x} \in \mathcal{C}} F(\mathbf{x})$ and $Q := \max\{5\|\nabla F(\mathbf{x}_0) - \mathbf{d}_0\|^2, 16\sigma^2 + 2L^2D^2\}$.

Proof Check Appendix F. ■

The result in Theorem 7 shows that the sequence of iterates generated by SCG, which only has access to a noisy unbiased estimate of the gradient at each iteration, is able to achieve the optimal approximation bound $(1 - 1/e)$, while the error term vanishes at a sublinear rate of $\mathcal{O}(T^{-1/3})$.

4.3. Weak Submodularity

In this section, we extend our results to a more general case where the expected objective function F is weakly-submodular. A continuous function F is γ -weakly DR-submodular if

$$\gamma = \inf_{\mathbf{x}, \mathbf{y} \in \mathcal{X}, \mathbf{x} \leq \mathbf{y}} \inf_{i \in [n]} \frac{[\nabla F(\mathbf{x})]_i}{[\nabla F(\mathbf{y})]_i}, \quad (28)$$

Algorithm 3 Non-monotone Stochastic Continuous Greedy (NMSCG)**Require:** Stepsizes $\rho_t > 0$. Initialize $\mathbf{d}_0 = \mathbf{x}_0 = \mathbf{0}$

- 1: **for** $t = 1, 2, \dots, T$ **do**
- 2: Compute $\mathbf{d}_t = (1 - \rho_t)\mathbf{d}_{t-1} + \rho_t \nabla \tilde{F}(\mathbf{x}_t, \mathbf{z}_t)$;
- 3: Compute $\mathbf{v}_t \in \operatorname{argmax}_{\mathbf{v} \in \mathcal{C}, \mathbf{v} \leq \bar{\mathbf{u}} - \mathbf{x}_t} \{\mathbf{d}_t^T \mathbf{v}\}$;
- 4: Update the variable $\mathbf{x}_{t+1} = \mathbf{x}_t + \frac{1}{T} \mathbf{v}_t$;
- 5: **end for**

where $[\mathbf{a}]_i$ denotes the i -th element of vector $\mathbf{a}$. In the following theorem, we prove that the proposed SCG method achieves a $(1 - e^{-\gamma})$ approximation guarantee when the expected function F is monotone and weakly DR-submodular with parameter γ .

Theorem 8 Consider *Stochastic Continuous Greedy (SCG)* outlined in Algorithm 2. If Assumptions 4-6 are satisfied and the function F is γ -weakly DR-submodular, then for $\rho_t = \frac{4}{(t+8)^{2/3}}$ the expected objective function value of the iterates generated by SCG satisfies the inequality

$$\mathbb{E}[F(\mathbf{x}_T)] \geq (1 - e^{-\gamma})OPT - \frac{15DQ^{1/2}}{T^{1/3}} - \frac{LD^2}{2T}, \quad (29)$$

where $OPT = \max_{\mathbf{x} \in \mathcal{C}} F(\mathbf{x})$ and $Q := \max\{5\|\nabla F(\mathbf{x}_0) - \mathbf{d}_0\|^2, 16\sigma^2 + 2L^2D^2\}$.

Proof Check Appendix G. ■

4.4. Non-monotone Continuous Submodular Maximization

In this section, we aim to extend the results for the proposed **Stochastic Continuous Greedy** algorithm to maximize *non-monotone* stochastic DR-submodular. The problem formulation of interest is similar to Problem (14) except the facts that the objective function $F : \mathcal{X} \rightarrow \mathbb{R}_+$ may not be monotone and the set $\mathcal{X}$ is a bounded box. To be more precise, we aim to solve the program

$$\max_{\mathbf{x} \in \mathcal{C}} F(\mathbf{x}) \doteq \max_{\mathbf{x} \in \mathcal{C}} \mathbb{E}_{\mathbf{z} \sim P}[\tilde{F}(\mathbf{x}, \mathbf{z})], \quad (30)$$

where $F : \mathcal{X} \rightarrow \mathbb{R}$ is continuous DR-submodular, $\mathcal{X} = \prod_{i=1}^n \mathcal{X}_i$, each $\mathcal{X}_i = [\underline{u}_i, \bar{u}_i]$ is a bounded interval, and the convex set $\mathcal{C}$ is a subset of $\mathcal{X} = [\underline{\mathbf{u}}, \bar{\mathbf{u}}]$, where $\underline{\mathbf{u}} = [\underline{u}_1; \dots; \underline{u}_n]$ and $\bar{\mathbf{u}} = [\bar{u}_1; \dots; \bar{u}_n]$. In this section, we propose the first stochastic conditional gradient method for solving the stochastic non-monotone maximization problem in (30). In this section, we further assume that the convex set $\mathcal{C}$ is down-closed and $\mathbf{0} \in \mathcal{C}$.

We introduce a variant of the **Stochastic Continuous Greedy** method that achieves a $(1/e)$ -approximation guarantee for Problem (30). The proposed **Non-monotone Stochastic Continuous Greedy (NMSCG)** method is inspired by the unified measured continuous greedy algorithm in (Feldman et al., 2011) and the Frank-Wolfe method in (Bian et al., 2017a) for non-monotone *deterministic* continuous submodular maximization. The steps

of NMSCG are summarized in Algorithm 3. The stochastic gradient update (Step 2) and the update of the variable (Step 4) are identical to the ones for SCG in Section 4.2. The main difference between NMSCG and SCG is in the computation of the ascent direction $\mathbf{v}_t$. In particular, in NMSCG the ascent direction vector $\mathbf{v}_t$ is obtained by solving the linear program

$$\mathbf{v}_t \in \underset{\mathbf{v} \in \mathcal{C}, \mathbf{v} \leq \bar{\mathbf{u}} - \mathbf{x}_t}{\operatorname{argmax}} \{ \mathbf{d}_t^T \mathbf{v} \}, \quad (31)$$

which differs from (21) by having the extra constraint $\mathbf{v} \leq \bar{\mathbf{u}} - \mathbf{x}_t$. This extra condition is added to ensure that the solution does not grow aggressively, since in non-monotone case dramatic growth of the solution may lead to poor performance. In NMSCG, the initial variable is $\mathbf{x}_0 = \mathbf{0}$, which is a legitimate initialization as we assume that the convex set $\mathcal{C}$ is down-closed. In the following theorem, we establish a $1/e$ - guarantee for NMSCG.

Theorem 9 *Consider Non-monotone Stochastic Continuous Greedy (NMSCG) outlined in Algorithm 3 with the averaging parameter $\rho_t = 4/(t+8)^{2/3}$. If Assumptions 4 and 6 hold and the gradients ∇F are L -Lipschitz continuous and the convex set $\mathcal{C}$ is down-closed, then the iterate $\mathbf{x}_T$ generated by NMSCG satisfies the inequality*

$$\mathbb{E}[F(\mathbf{x}_T)] \geq e^{-1} F(\mathbf{x}^*) - \frac{15DQ^{1/2}}{T^{1/3}} - \frac{LD^2}{2T}, \quad (32)$$

where $Q := \max\{5\|\nabla F(\mathbf{x}_0) - \mathbf{d}_0\|^2, 16\sigma^2 + 2L^2D^2\}$.

Proof See Section H. ■

The result in Theorem 9 states that the sequence of iterates generated by NMSCG achieves a $((1/e)OPT - \epsilon)$ approximation guarantee after $\mathcal{O}(1/\epsilon^3)$ stochastic gradient computations.

5. Stochastic Discrete Submodular Maximization

Even though submodularity has been mainly studied in discrete domains (Fujishige, 2005), many efficient methods for optimizing such submodular set functions rely on continuous relaxations either through a multi-linear extension (Vondrák, 2008) (for maximization) or Lovász extension (Lovász, 1983) (for minimization). In fact, Problem (14) has a discrete counterpart, recently considered in (Hassani et al., 2017; Karimi et al., 2017):

$$\max_{S \in \mathcal{I}} f(S) \doteq \max_{S \in \mathcal{I}} \mathbb{E}_{\mathbf{z} \sim P} [\tilde{f}(S, \mathbf{z})], \quad (33)$$

where the function $f : 2^V \rightarrow \mathbb{R}_+$ is submodular, the function $\tilde{f} : 2^V \times \mathcal{Z} \rightarrow \mathbb{R}_+$ is an arbitrary mapping from the product of 2^V and the domain of the random variable $\mathbf{Z}$ to $\mathbb{R}$, S is the optimization set variable defined over a ground set V , $\mathbf{z} \in \mathcal{Z}$ is the realization of a random variable $\mathbf{Z}$ drawn from the distribution P , and $\mathcal{I}$ is a general matroid constraint. Since P is unknown, problem (33) cannot be directly solved using the current state-of-the-art techniques. Instead, Hassani et al. (2017) showed that by lifting the problem to the

continuous domain (via multi-linear relaxation) and using stochastic gradient methods on a continuous relaxation to reach a solution that is within a factor $(1/2)$ of the optimum. Contemporarily, (Karimi et al., 2017) used a concave relaxation technique to provide a $(1 - 1/e)$ approximation for the class of submodular coverage functions. Our work closes the gap for maximizing the stochastic submodular set maximization, namely, Problem (33), by providing the first tight $(1 - 1/e)$ approximation guarantee for general monotone submodular set functions subject to a matroid constraint. For the non-monotone case, we obtain a $(1/e)$ approximation guarantee. We, further, show that a $((1 - e^{-c})/c)$ -approximation guarantee can be achieved when the function f has curvature c .

According to the results in Section 4, SCG achieves in expectation a $(1 - 1/e)$ -optimal solution for Problem (14) when the function F is monotone and DR-submodular, and NMSCG obtains $(1/e)$ -optimal solution for the non-monotone case. The focus of this section is on extending these results into the discrete domain and showing that SCG and NMSCG can be used to maximize a stochastic submodular *set* function f , namely Problem (33), through the multilinear extension of the function f . To be more precise, in lieu of solving the program in (33) one can solve the continuous optimization problem

$$\max_{\mathbf{x} \in \mathcal{C}} F(\mathbf{x}) = \max_{\mathbf{x} \in \mathcal{C}} \sum_{S \subseteq V} f(S) \prod_{i \in S} x_i \prod_{j \notin S} (1 - x_j), \quad (34)$$

where F is the multilinear extension of the function f and the convex set $\mathcal{C} = \text{conv}\{1_I : I \in \mathcal{I}\}$ is the matroid polytope (Calinescu et al., 2011) which is down-closed (note that in (34), x_i denotes the i -th element of the vector $\mathbf{x}$). The fractional solution of the program (34) can then be rounded into a feasible discrete solution without any loss (in expectation) in objective value by methods such as randomized PIPAGE ROUNDING (Calinescu et al., 2011). Note that randomized PIPAGE ROUNDING requires $O(n)$ computational complexity (Karimi et al., 2017) for the uniform matroid and $O(n^2)$ complexity for general matroids.

Indeed, the conventional continuous greedy algorithm is able to solve the program in (34); however, each iteration of the method is computationally costly due to gradient $\nabla F(\mathbf{x})$ evaluations. Instead, Feldman et al. (2011) and Calinescu et al. (2011) suggested approximating the gradient using a sufficient number of samples from f . This mechanism still requires access to the set function f multiple times at each iteration, and hence is not efficient for solving Problem (33). The idea is then to use a stochastic (unbiased) estimate for the gradient ∇F . In the following remark, we provide a method to compute an unbiased estimate of the gradient using n samples from $\tilde{f}(S_i, \mathbf{z})$, where $\mathbf{z} \sim P$ and S_i 's, $i = 1, \dots, n$, are carefully chosen sets.

Remark 10 (*Constructing an Unbiased Estimator of the Gradient in Multilinear Extensions*) Recall that $f(S) = \mathbb{E}_{\mathbf{z} \sim P}[\tilde{f}(S, \mathbf{z})]$. In terms of the multilinear extensions, we obtain $F(\mathbf{x}) = \mathbb{E}_{\mathbf{z} \sim P}[\tilde{F}(\mathbf{x}, \mathbf{z})]$, where F and $\tilde{F}$ denote the multilinear extension of f and $\tilde{f}$, respectively. So $\nabla \tilde{F}(\mathbf{x}, \mathbf{z})$ is an unbiased estimator of $\nabla F(\mathbf{x})$ when $\mathbf{z} \sim P$. Note that finding the gradient of $\tilde{F}$ may not be easy as it contains exponentially many terms. Instead, we can provide computationally cheap unbiased estimators for $\nabla \tilde{F}(\mathbf{x}, \mathbf{z})$. It can easily be shown that

$$\frac{\partial \tilde{F}}{\partial x_i} = \tilde{F}(\mathbf{x}, \mathbf{z}; x_i \leftarrow 1) - \tilde{F}(\mathbf{x}, \mathbf{z}; x_i \leftarrow 0). \quad (35)$$

where for example by $(\mathbf{x}; x_i \leftarrow 1)$ we mean a vector which has value 1 on its i -th coordinate and is equal to $\mathbf{x}$ elsewhere. To create an unbiased estimator for $\frac{\partial \tilde{F}}{\partial x_i}$ at a point $\mathbf{x}$ with realization $\mathbf{z}$ we can simply sample a set S by including each element in it independently with probability x_i and use $\tilde{f}(S \cup \{i\}, \mathbf{z}) - \tilde{f}(S \setminus \{i\}, \mathbf{z})$ as an unbiased estimator for the i -th partial derivative of $\tilde{F}$. We can sample one single set S and use the above trick for all the coordinates. This involves n function computations for $\tilde{f}$.

Indeed, the stochastic gradient ascent method proposed by Hassani et al. (2017) can be used to solve the multilinear extension problem in (34) using unbiased estimates of the gradient at each iteration. However, the stochastic gradient ascent method fails to achieve the optimal $(1 - 1/e)$ approximation. Further, the work of Karimi et al. (2017) achieves a $(1 - 1/e)$ approximation solution only when each $\tilde{f}(\cdot, \mathbf{z})$ is a coverage function. Here, we show that SCG achieves the first $(1 - 1/e)$ tight approximation guarantee for the discrete stochastic submodular Problem (33). More precisely, we show that SCG finds a solution for (34), with an expected function value that is at least $(1 - 1/e)\text{OPT} - \epsilon$, in $\mathcal{O}(1/\epsilon^3)$ iterations. To do so, we first show in the following lemma that the difference between any coordinates of gradients of two consecutive iterates generated by SCG, i.e., $\nabla_j F(\mathbf{x}_{t+1}) - \nabla_j F(\mathbf{x}_t)$ for $j \in \{1, \dots, n\}$, is bounded by $\|\mathbf{x}_{t+1} - \mathbf{x}_t\|$ multiplied by a factor which is independent of the problem dimension n .

Lemma 11 *Consider Stochastic Continuous Greedy (SCG) outlined in Algorithm 2 with iterates $\mathbf{x}_t$, and recall the definition of the multilinear extension function F in (34). If we define r as the rank of the matroid $\mathcal{I}$ and $m_f \triangleq \max_{i \in \{1, \dots, n\}} f(i)$, then the following*

$$|\nabla_j F(\mathbf{x}_{t+1}) - \nabla_j F(\mathbf{x}_t)| \leq m_f \sqrt{r} \|\mathbf{x}_{t+1} - \mathbf{x}_t\|, \quad (36)$$

holds for $j = 1, \dots, n$.

Proof See Appendix I. ■

The result in Lemma 11 states that in an *ascent direction of SCG*, the gradient is $m_f \sqrt{r}$ -Lipschitz continuous. Here, m_f is the maximum marginal value of the function f and r is the rank of the matroid.

Let us now explain how the variance of the stochastic gradients of F relates to the variance of the marginal values of f . Recall that the stochastic function $\tilde{F}$ is a multilinear extension of the stochastic set function $\tilde{f}$, and it can be shown that

$$\nabla_j \tilde{F}(\mathbf{x}, \mathbf{z}) = \tilde{F}(\mathbf{x}, \mathbf{z}; x_j = 1) - \tilde{F}(\mathbf{x}, \mathbf{z}; x_j = 0). \quad (37)$$

Hence, from submodularity we have $\nabla_j \tilde{F}(\mathbf{x}, \mathbf{z}) \leq \tilde{f}(\{j\}, \mathbf{z})$. Using this simple fact we can deduce that

$$\mathbb{E} \left[\|\nabla \tilde{F}(\mathbf{x}, \mathbf{z}) - \nabla F(\mathbf{x})\|^2 \right] \leq n \max_{j \in [n]} \mathbb{E} [\tilde{f}(\{j\}, \mathbf{z})^2]. \quad (38)$$

Using the result of Lemma 11, the expression in (38), and a coordinate-wise analysis, the bounds in Theorem 7 can be improved and specified for the case of multilinear extension maximization problem as we show in the following theorem.

Theorem 12 Consider *Stochastic Continuous Greedy (SCG)* outlined in Algorithm 2. Recall the definition of the multilinear extension function F in (34) and the definitions of r and m_f in Lemma 11. Further, set the averaging parameter as $\rho_t = 4/(t+8)^{2/3}$. If Assumption 4 holds and the function f is monotone and submodular, then the iterate $\mathbf{x}_T$ generated by SCG satisfies the inequality

$$\mathbb{E}[F(\mathbf{x}_T)] \geq (1 - 1/e)OPT - \frac{15DK}{T^{1/3}} - \frac{m_f r D^2}{2T}, \quad (39)$$

where $K := \max\{3\|\nabla F(\mathbf{x}_0) - \mathbf{d}_0\|, 2\sqrt{n}\sqrt{\max_{j \in [n]} \mathbb{E}[\tilde{f}(\{j\}, \mathbf{z})^2]} + \sqrt{3r}m_f D\}$ and OPT is the optimal value of Problem (34).

Proof The proof is similar to the proof of Theorem 7. Check Appendix J. ■

The result of Theorem 12 indicates that the sequence of iterates generated by SCG achieves a $(1 - 1/e)OPT - \epsilon$ approximation guarantee. Note that the constants on the right hand side of (39) are independent of n , except K that is at most proportional to $\sqrt{n}$. As a result, we have the following guarantee for SCG in the case of multilinear functions.

Corollary 13 Consider *Stochastic Continuous Greedy (SCG)* outlined in Algorithm 2. Suppose the conditions in Theorem 12 are satisfied. Then, the sequence of iterates generated by SCG achieves a $(1 - 1/e)OPT - \epsilon$ solution after $\mathcal{O}(n^{3/2}/\epsilon^3)$ iterations. As a consequence, maximizing a stochastic Submodular set function with SCG requires $\mathcal{O}(n^{5/2}/\epsilon^3)$ evaluations of the function $\tilde{f}$ in order to reach a $(1 - 1/e)OPT - \epsilon$ solution.

Proof According to the result in Theorem 12, SCG reaches a $(1 - 1/e)OPT - \mathcal{O}(n^{1/2}/T^{1/3})$ solution after T iterations. Therefore, to achieve a $((1 - 1/e)OPT - \epsilon)$ approximation, $\mathcal{O}(n^{3/2}/\epsilon^3)$ iterations are required. Since each iteration requires access to an unbiased estimator of the gradient $\nabla F(\mathbf{x})$ and it can be computed by n samples from $\tilde{f}(S_i, \mathbf{z})$ (Remark 10), then the total number of calls to the function $\tilde{f}$ to reach a $(1 - 1/e)OPT - \epsilon$ solution is of order $\mathcal{O}(n^{5/2}/\epsilon^3)$ for the SCG method. ■

The result in Corollary 13 shows that after at most $\mathcal{O}(n^{5/2}/\epsilon^3)$ function evaluations of the stochastic set function $\tilde{f}$ the iterates generated by SCG achieves a continuous solution $\mathbf{x}_t$ with an objective function value that satisfies $\mathbb{E}[F(\mathbf{x}_t)] \geq (1 - 1/e)OPT - \epsilon$ where OPT is the optimal objective function value of Problem (34). Further, by using a lossless rounding scheme we can obtain a discrete set $S^\dagger$ such that $\mathbb{E}[f(S^\dagger)] \geq (1 - 1/e) \max_{S \in \mathcal{I}} f(S) - \epsilon$.

Remark 14 According to Corollary 13, to reach a $(1 - (1/e)OPT - \epsilon)$ solution the total number of calls to the function $\tilde{f}$ for our proposed SCG method is $\mathcal{O}(n^{5/2}/\epsilon^3)$. The result in [Badanidiyuru and Vondrak, 2013] shows that for the deterministic case where we have access to f (not $\tilde{f}$), after $\mathcal{O}(n^2/\epsilon^4)$ calls to f , one can find $1 - (1/e)OPT - \epsilon$ solution. Hence, if we apply both methods to the deterministic submodular case, the overall complexity of our method is slightly worse than the one in [Badanidiyuru and Vondrak, 2013] in terms of dependency on n , while the dependency of our result on ϵ is better than the one in

[Badanidiyuru and Vondrak, 2013]. We would like to highlight that the proposed method and its complexity bound in [Badanidiyuru and Vondrak, 2013] are for the deterministic setting, and if one modifies the method in [Badanidiyuru and Vondrak, 2013] to be applicable to the considered stochastic setting, its query complexity $\mathcal{O}(n^2/\epsilon^4)$ will further increase.

Indeed, by following similar steps we can extend the result for NMSCG to the discrete submodular maximization problem when the objective function is non-monotone and stochastic. We formally prove this claim in the following theorem.

Theorem 15 *Consider Non-monotone Stochastic Continuous Greedy (NMSCG) outlined in Algorithm 3. Recall the definition of the multilinear extension function F in (34) and the definitions of r and m_f in Lemma 11. Further, set the averaging parameter as $\rho_t = 4/(t+8)^{2/3}$. If Assumption 4 holds and the function f is non-monotone and submodular, then the iterate $\mathbf{x}_T$ generated by SCG satisfies the inequality*

$$\mathbb{E}[F(\mathbf{x}_T)] \geq (1/e)OPT - \frac{15DK}{T^{1/3}} - \frac{m_f r D^2}{2T}, \quad (40)$$

where $K := \max\{3\|\nabla F(\mathbf{x}_0) - \mathbf{d}_0\|, 2\sqrt{n}\sqrt{\max_{j \in [n]} \mathbb{E}[\tilde{f}(\{j\}, \mathbf{z})^2]} + \sqrt{3r}m_f D\}$ and OPT is the optimal value of Problem (34).

Proof The proof is similar to the proof of Theorem 9. Check Appendix K. ■

Corollary 16 *Consider Non-monotone Stochastic Continuous Greedy (NMSCG) outlined in Algorithm 3. Suppose the conditions in Theorem 15 are satisfied. Then, the sequence of iterates generated by NMSCG achieves a $(1/e)OPT - \epsilon$ solution after $\mathcal{O}(n^{3/2}/\epsilon^3)$ iterations. As a consequence, maximizing a stochastic Submodular set function with NMSCG requires $\mathcal{O}(n^{5/2}/\epsilon^3)$ evaluations of the function f in order to reach a $(1/e)OPT - \epsilon$ solution.*

5.1. Convergence Bounds Based on Curvature

For the continuous greedy method it has been shown that if the submodular function f has a curvature $c \in [0, 1]$ the algorithm reaches a $(1/c)(1 - e^{-c})$ approximation guarantee. In this section, we show that the same improvement can be established for SCG in the stochastic setting. To do so, we first formally define the curvature c of a monotone submodular function f as

$$c := 1 - \min_{S, j \notin S} \frac{f(S \cup \{j\}) - f(S)}{f(\{j\})}. \quad (41)$$

Indeed, smaller curvature value c leads to an easier submodular maximization problem, and, in this case, we should be able to achieve a tighter approximate solution. In the following theorem, we match this expectation and show that if $c < 1$ the bound in (27) can be improved.

Theorem 17 *Consider the proposed Stochastic Continuous Greedy (SCG) defined in (20)-(22). Further recall the definition of the function f curvature c in (41). If Assumption 4*

is satisfied and the function f is monotone and submodular, then the expected objective function value for the iterate $\mathbf{x}_T$ generated by SCG satisfies the inequality

$$\mathbb{E}[F(\mathbf{x}_T)] \geq \frac{1}{c}(1 - e^{-c})OPT - \frac{15DK}{T^{1/3}} - \frac{m_f r D^2}{2T}, \quad (42)$$

where $K := \max\{3\|\nabla F(\mathbf{x}_0) - \mathbf{d}_0\|, 2\sqrt{n}\sqrt{\max_{j \in [n]} \mathbb{E}[\tilde{f}(\{j\}, \mathbf{z})^2]} + \sqrt{3r}m_f D\}$.

Proof Check Appendix L. ■

Corollary 18 *Consider Stochastic Continuous Greedy (SCG) outlined in Algorithm 2. Suppose the conditions in Theorem 17 are satisfied. Then, the sequence of iterates generated by SCG achieves a $((1 - e^{-c})/c)OPT - \epsilon$ solution after $\mathcal{O}(n^{3/2}/\epsilon^3)$ iterations. As a consequence, maximizing a stochastic submodular set function with SCG requires $\mathcal{O}(n^{5/2}/\epsilon^3)$ function evaluations.*

6. Numerical Experiments

In this section, we compare the performances of the proposed stochastic conditional gradient method with state-of-the-art algorithms in both convex and submodular settings.

6.1. Convex Setting

We first compare the proposed SFW algorithm and mini-batch FW for a stochastic quadratic program of the form (2). Then, we compare their performances in solving a matrix completion problem. In this section, by mini-batch FW we refer to a variant of FW that simply replaces gradients by a mini-batch of stochastic gradients.

Quadratic Programming. Consider a positive definite matrix $\mathbf{A} \in \mathbb{S}_{++}^n$, a vector $\mathbf{b} \in \mathbb{R}^n$, a random variable $\mathbf{z} \in \mathbb{R}^n$, and the random diagonal matrix $\text{diag}(\mathbf{z}) \in \mathbb{R}^{n \times n}$ defined by $\mathbf{z}$. The function F is defined as

$$F(\mathbf{x}) = \mathbb{E}[\tilde{F}(\mathbf{x}, \mathbf{z})] = \mathbb{E}\left[\frac{1}{2}\mathbf{x}^T(\mathbf{A} + \text{diag}(\mathbf{z}))\mathbf{x} + (\mathbf{b} + \mathbf{z})^T \mathbf{x}\right]. \quad (43)$$

We assume that each element of $\mathbf{z}$ is sampled from a normal distribution $\mathcal{N}(0, \sigma)$. Therefore, the objective function can be simplified to $F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{b}^T \mathbf{x}$. Further, we assume that the set $\mathcal{C}$ is defined as $\mathcal{C} = \{\mathbf{x} \in \mathbb{R}^n \mid l \leq x_i \leq u\}$. Here, we assume that the distribution is unknown to the algorithm and at each iteration we only have access to the stochastic gradients $\nabla \tilde{F}(\mathbf{x}, \mathbf{z}) = (\mathbf{A} + \text{diag}(\mathbf{z}))\mathbf{x} + \mathbf{b} + \mathbf{z}$.

In our experiments, we set the dimension of the problem to $n = 5$ and the lower bound and upper bounds for the set $\mathcal{C}$ to $l = 10$ and $u = 100$. We construct $\mathbf{A}$ and $\mathbf{b}$ in such a way that the optimal solution of the unconstrained set, namely $-\mathbf{A}^{-1}\mathbf{b}$, does not belong to the set $\mathcal{C}$.

Figure 1 demonstrates the suboptimality gap $F(\mathbf{x}_T) - F(\mathbf{x}^*)$ for the iterates generated by the proposed SFW method (with batch size $b = 1$) as well as the naive stochastic

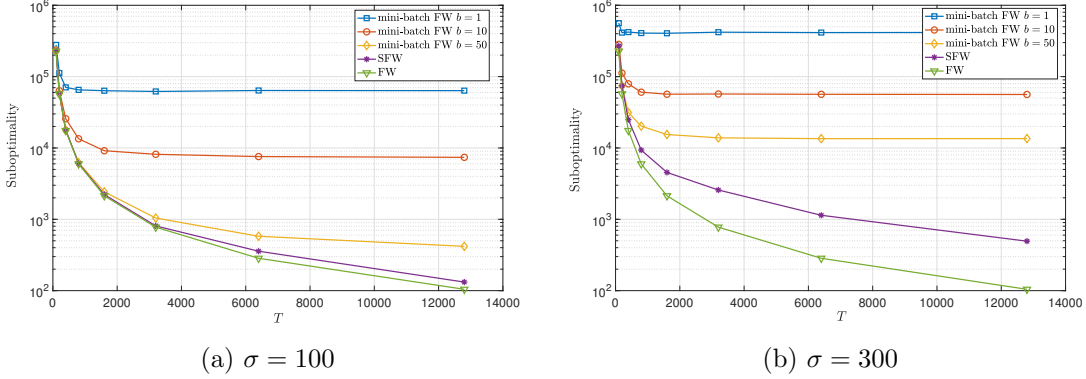


Figure 1: Comparison of the performances of (deterministic) FW, mini-batch FW ($b = 1$, $b = 10$, $b = 50$), and the proposed SFW method for the stochastic quadratic convex program defined in (43). The left and right plots correspond to the cases that the variance is $\sigma = 100$ and $\sigma = 300$, respectively. In both settings, the proposed SFW method that only uses a single stochastic gradient ($b = 1$) performs close to the full-batch (deterministic) FW algorithm, and it outperforms the mini-batch FW method with even larger batch sizes ($b = \{1, 10, 50\}$). This comparison is in terms of number of iterations. If we compare the algorithms in terms of number of evaluated stochastic gradients the gap between SFW and mini-batch FW algorithms with batch sizes larger than $b = 1$ will be even more significant as in this experiment SFW uses only a single stochastic gradient per iteration.

implementation of FW with batch sizes $b = \{1, 10, 50\}$ for the cases that the total number of iterations is $T = \{100, 200, 400, 800, 1600, 3200, 6400, 12800\}$. We further illustrates the performance of the (deterministic) FW as a benchmark. Indeed, to perform the update of FW we use the exact gradient $\mathbf{A}\mathbf{x} + \mathbf{b}$ at each iteration. Note that the optimal solution $\mathbf{x}^*$ and the optimal objective function value $F(\mathbf{x}^*)$ are pre-computed by solving the quadratic program $\min_{\mathbf{x} \in \mathcal{C}} \frac{1}{2}\mathbf{x}^T \mathbf{A}\mathbf{x} + \mathbf{b}^T \mathbf{x}$. The left and right plots correspond to the cases that $\sigma = 100$ and $\sigma = 300$, respectively. In the left plot, which corresponds to the case that $\sigma = 100$, we observe that our proposed SFW method performs similar to the FW algorithm, while it uses only a single noisy stochastic gradient per iteration. The vanilla mini-batch FW method with a single stochastic gradient evaluation ($b = 1$) performs poorly. By increasing the size of the batch, the performance of the mini-batch FW improves, but it still underperforms SFW. In the right plot, which corresponds to the case with a larger variance, naturally the gap between the deterministic FW method and the stochastic algorithms becomes more significant. In this case, we observe that mini-batch FW even with large batch size of $b = 100$ is significantly worse than the proposed SFW method that only uses a single stochastic gradient per iteration. It is worth mentioning that increasing the batch size in mini-batch FW accelerates convergence and improves convergence accuracy, however, the suboptimality saturates at some point. In contrast, SFW converges to the optimal objective function at a sublinear rate in both small and large variance cases, matching our theory.

Matrix Completion. In this experiment, we study the performance of our proposed SFW algorithm in solving a matrix completion problem which is a canonical application of conditional gradient (Frank-Wolfe) type methods. We focus on a special case of matrix completion in which matrices are assumed to be symmetric. In particular, consider a symmetric matrix $\mathbf{C} \in \mathbb{S}^n$, where we only have access to a subset of its indices indicated by $\mathcal{O}$. Note that as $\mathbf{C}$ is symmetric, if (i, j) is observed, i.e., $(i, j) \in \mathcal{O}$, then the pair (j, i) is also known, i.e., $(j, i) \in \mathcal{O}$. Our goal is to find a symmetric positive semidefinite matrix $\mathbf{X}$ such that its elements in the set $\mathcal{O}$ are close to the ones for $\mathbf{C}$, while its nuclear rank is smaller than a threshold. In other words, we focus on the optimization problem

$$\begin{aligned} \min f(\mathbf{X}) &:= \frac{1}{2} \sum_{(i,j) \in \mathcal{O}} \|\mathbf{X}_{ij} - \mathbf{C}_{ij}\|^2 \\ \text{s. t. } &\|\mathbf{X}\| \succeq \mathbf{0}, \quad \|\mathbf{X}\|_* \leq \alpha. \end{aligned} \quad (44)$$

In our simulations, we set the dimension to $n = 200$. We form the observation matrix as $\mathbf{C} = \hat{\mathbf{X}} + \mathbf{E}$. Here, $\hat{\mathbf{X}}$ is defined as $\hat{\mathbf{X}} = \mathbf{W}\mathbf{W}^T$ where $\mathbf{W} \in \mathbb{R}^{n \times r}$ has independent normal distributed entries, and $\mathbf{E}$ is defined as $\mathbf{E} = \frac{1}{10}(\mathbf{L} + \mathbf{L}^T)$ where $\mathbf{L} \in \mathbb{R}^{n \times n}$ has independent normal distributed entries. In our experiments, we set the rank to $r = 10$ and the bound on the nuclear norm to $\alpha = \|\hat{\mathbf{X}}\|_*$, where $\|\hat{\mathbf{X}}\|_*$ is the nuclear norm of the matrix $\hat{\mathbf{X}}$. For settings that $\hat{\mathbf{X}}$ is not known in advance, one might use different choices of α and pick the one that performs better. We further define the set of observed entries $\mathcal{O}$ by sampling the elements of the upper triangular part of $\mathbf{C}$ uniformly at random with probability 0.8. Therefore, the size of the set $\mathcal{O}$ is around $0.8 \times 200^2 = 32,000$. In the realization that we use the set $\mathcal{O}$ has 31,884 elements.

To solve (44), by using FW method, we need to solve the subproblem (Hazan et al., 2016, Chapter 7)

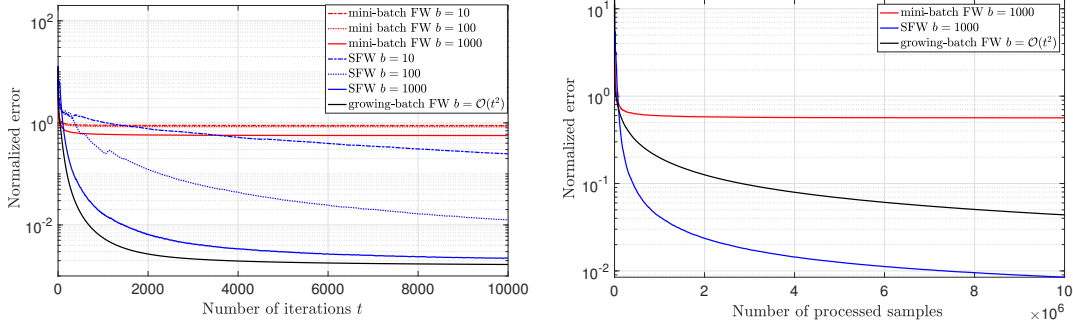
$$\begin{aligned} \min &\text{tr}(\nabla f(\mathbf{X}_t)^T \mathbf{V}) \\ \text{s. t. } &\|\mathbf{V}\| \succeq \mathbf{0}, \quad \|\mathbf{V}\|_* \leq \alpha. \end{aligned} \quad (45)$$

where the gradient $\nabla f(\mathbf{X}_k) \in \mathbb{R}^{n \times n}$ is defined as $\nabla f(\mathbf{X}_t)_{i,j} = \mathbf{X}_{ij} - \mathbf{C}_{ij}$ if $(i, j) \in \mathcal{O}$, and $\nabla f(\mathbf{X}_t)_{i,j} = 0$, otherwise. It can be shown that the solution to the subproblem (45) is given by

$$\mathbf{V}_t = \begin{cases} \alpha \mathbf{v}_n \mathbf{v}_n^T & \text{if } \lambda_n \geq 0, \\ \mathbf{0} & \text{if } \lambda_n < 0, \end{cases} \quad (46)$$

where λ_n is the smallest eigenvalue of the gradient $\nabla f(\mathbf{X}_t)$ and $\mathbf{v}_n$ is its corresponding eigenvector.

Indeed, evaluation of the gradient $\nabla f(\mathbf{X}_t)$ requires access to all observed elements in the set $\mathcal{O}$ which can be computationally costly. In such cases, one may use a subset of the set $\mathcal{O}$ as an unbiased estimate of the gradient. In our experiments, we consider (i) the mini-batch FW method that uses b elements of $\mathcal{O}$ to compute a stochastic approximation of $\nabla f(\mathbf{X}_t)$, (ii) the growing mini-batch FW method proposed by Hazan and Luo (2016) which uses a batch size of $b = \mathcal{O}(t^2)$ at step t , and (iii) the proposed SFW method that uses the average of stochastic gradients over time as suggested in (3).



(a) Comparison in terms of number of iterations (b) Comparison in terms of number of samples

Figure 2: Comparison of the normalized error $\sum_{(i,j) \in \mathcal{O}} \|\mathbf{X}_{ij} - \mathbf{C}_{ij}\|^2 / \sum_{(i,j) \in \mathcal{O}} \|\mathbf{C}_{ij}\|^2$ of the mini-batch FW algorithm, the growing mini-batch FW method proposed in (Hazan and Luo, 2016), and our proposed SFW method for solving the matrix completion problem defined in (44). For all the choices of batch sizes considered here ($b = \{10, 100, 1000\}$), the proposed SFW method outperforms the corresponding version of the mini-batch FW algorithm. As we observe, increasing the size of the mini-batch b improves the convergence speed of both algorithms, but the mini-batch FW algorithm with $b = 1000$ still underperforms our proposed method with $b = 10$. In terms of number of iterations, the growing-batch FW method outperforms our proposed SFW method by using growing batches of size $b = \mathcal{O}(t^2)$, e.g., at iteration $t = 10^4$ it uses 10^8 samples; however, in terms of number of samples processed, SFW has the best performance as illustrated in the right plot.

Figure 2 illustrates the convergence paths of mini-batch FW and SFW for batch sizes of $b = \{10, 100, 1000\}$ as well as the convergence path of growing-batch FW in terms of both number of iterations and number of samples processed. Here, the normalized error is defined as

$$\text{Normalized error} := \frac{\sum_{(i,j) \in \mathcal{O}} \|\mathbf{X}_{ij} - \mathbf{C}_{ij}\|^2}{\sum_{(i,j) \in \mathcal{O}} \|\mathbf{C}_{ij}\|^2}.$$

Note that the stepsize for all the three algorithms is $\gamma_t = \frac{1}{t+1}$ and the averaging parameter for our proposed SFW algorithm is $\rho_t = \frac{1}{(t+1)^{2/3}}$. As we observe in Figure 2a, even for a large batch size of $b = 1000$, the mini-batch FW algorithm cannot obtain a normalized error better than 0.55 after 10,000 iterations. On the other hand, SFW with a small batch size of $b = 10$ achieves an error of 0.25 after 10,000 iterations. Indeed, by increasing the batch size for SFW its performance becomes better. In particular, SFW with $b = 1000$ achieves a normalized error of 2.3×10^{-3} after 10,000 iterations. We would like to highlight that even for the case that we set $b = 1000$, we use less than 3.2% of the observed elements at each iteration – The number of observed elements is 31,884. However, the best performance in terms of number of iterations belongs to the growing-batch FW method that uses a batch size of $b = \mathcal{O}(t^2)$. This is not surprising as the convergence rate of the growing-batch FW algorithm is $\mathcal{O}(1/t)$, while the convergence rate of our proposed SFW is $\mathcal{O}(1/t^{1/3})$. However, the number of processed samples at each iteration by our method is

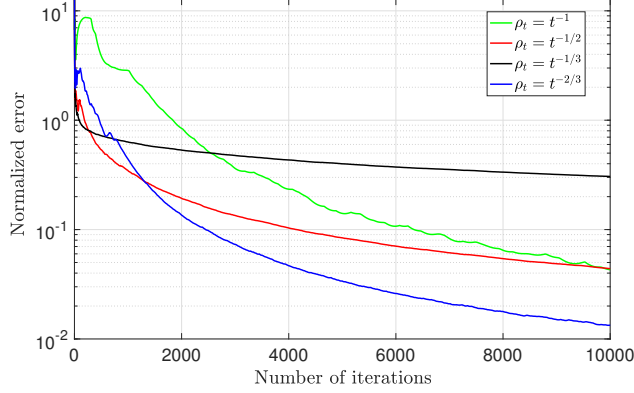


Figure 3: Comparison of the normalized error $\sum_{(i,j) \in \mathcal{O}} \|\mathbf{X}_{ij} - \mathbf{C}_{ij}\|^2 / \sum_{(i,j) \in \mathcal{O}} \|\mathbf{C}_{ij}\|^2$ of the proposed SFW method with different choices of the averaging parameter ρ_t for solving the matrix completion problem defined in (44). In this case we set the other parameters to $b = 100$ and $\gamma_t = 1/(t + 1)$. As suggested by our theory, the best performance belongs to the case that $\rho_t = \mathcal{O}(t^{-2/3})$.

much smaller than the one for growing-batch FW when t becomes large. To be more precise, after T iterations our method uses Tb samples, where b is a constant much smaller than T , while the growing-batch FW uses $\mathcal{O}(T^3)$ samples. Therefore, to have a better comparison between these algorithms we also compare their normalized errors versus the number of processed samples which is a more accurate measure for comparing the sample complexity of these algorithms. As we observe in Figure 2b, the proposed SFW method outperforms both mini-batch FW and growing-batch FW algorithms when we compare their normalized errors versus number of samples used. Note that in theory, both growing-batch FW and SFW may require processing $\mathcal{O}(1/\epsilon^3)$ samples to reach a suboptimality gap of $f(\mathbf{x}_t) - f^* \leq \epsilon$, but in practice we observe that the proposed SFW method outperforms growing-batch FW.

We proceed to study the effect of the averaging parameter ρ_t on the convergence of SFW. Our theoretical bound suggests that the best convergence guarantee is achieved when $\rho_t = \mathcal{O}(t^{-2/3})$. In this experiment we aim to check if this choice is reasonable relative to other possible sublinear rates. To do so, we compare the convergence paths of SFW with four different choices $\rho_t = \mathcal{O}(t^{-1/3})$, $\rho_t = \mathcal{O}(t^{-1/2})$, $\rho_t = \mathcal{O}(t^{-2/3})$, and $\rho_t = \mathcal{O}(t^{-1})$. As it can be observed in Figure 3, the best performance among these four choices belongs to $\rho_t = \mathcal{O}(t^{-2/3})$ used in our theoretical results. We would like to highlight that this experiment does not prove that $\rho_t = \mathcal{O}(t^{-2/3})$ is the optimal choice.

6.2. Submodular Setting

For the submodular setting, we consider a movie recommendation application (Stan et al., 2017) consisting of N users and n movies. Each user i has a user-specific utility function $f(\cdot, i)$ for evaluating sets of movies. The goal is to find a set of k movies such that in expectation over users' preferences it provides the highest utility, i.e., $\max_{|S| \leq k} f(S)$, where $f(S) \doteq \mathbb{E}_{i \sim P}[f(S, i)]$. This is an instance of the (discrete) stochastic submodular maximization problem in (33). For simplicity, we assume f has the form of an empirical objective

function, i.e. $f(S) = \frac{1}{N} \sum_{i=1}^N f(S, i)$. In other words, the distribution P is assumed to be uniform over the set of users. The continuous counterpart of this problem is to consider the the multilinear extension $F(\cdot, i)$ of any function $f(\cdot, i)$ and solve the problem in the continuous domain as follows. Let $F(\mathbf{x}) = \mathbb{E}_{i \sim \mathcal{D}}[F(\mathbf{x}, i)]$ for $x \in [0, 1]^n$ and define the constraint set $\mathcal{C} = \{\mathbf{x} \in [0, 1]^n : \sum_{i=1}^n x_i \leq k\}$. The discrete and continuous optimization formulations lead to the same optimal value (Calinescu et al., 2011):

$$\max_{S: |S| \leq k} f(S) = \max_{\mathbf{x} \in \mathcal{C}} F(\mathbf{x}).$$

Therefore, by running SCG we can find a solution in the continuous domain that is at least $1 - 1/e$ approximation to the optimal value. By rounding that fractional solution (for instance via randomized Pipage rounding (Calinescu et al., 2011)) we obtain a set whose utility is at least $1 - 1/e$ of the optimum solution set of size k . We note that randomized Pipage rounding does not need access to the value of f . We also remark that each iteration of SCG can be done very efficiently in $O(n)$ time (the linear program step reduces to finding the largest k elements of a vector of length n). Therefore, this approach easily scales to big data scenarios where the size of the data set N (e.g. number of users) or the number of items n (e.g. number of movies) are very large.

In our experiments, we consider the following baselines:

- (i) Stochastic Continuous Greedy (SCG): with $\rho_t = \frac{1}{2}t^{-2/3}$ and mini-batch size b . The details for computing an unbiased estimator for the gradient of F are given in Remark 10.
- (ii) Stochastic Gradient Ascent (SG/SGA) of (Hassani et al., 2017): with stepsize $\mu_t = c/\sqrt{t}$ and mini-batch size b .
- (iii) Frank-Wolfe (FW) variant of (Bian et al., 2017b; Calinescu et al., 2011): with parameter T for the total number of iterations and batch size b (we further let $\alpha = 1, \delta = 0$, see Algorithm 1 in (Bian et al., 2017b) or the continuous greedy method of (Calinescu et al., 2011) for more details).
- (iv) Batch-mode Greedy (Greedy): by running the vanilla greedy algorithm (in the discrete domain) in the following way. At each round of the algorithm (for selecting a new element), b random users are picked and the function f is estimated by the average over the b selected users.

To run the experiments we use the MovieLens data set. It consists of 1 million ratings (from 1 to 5) by $N = 6041$ users for $n = 4000$ movies. Let $r_{i,j}$ denote the rating of user i for movie j (if such a rating does not exist we assign $r_{i,j}$ to 0). In our experiments, we consider two well motivated objective functions. The first one is called “facility location” where the valuation function by user i is defined as $f(S, i) = \max_{j \in S} r_{i,j}$. In words, the way user i evaluates a set S is by picking the highest rated movie in S . Thus, the objective function is

$$f_{\text{fac}}(S) = \frac{1}{N} \sum_{i=1}^N \max_{j \in S} r_{i,j}.$$

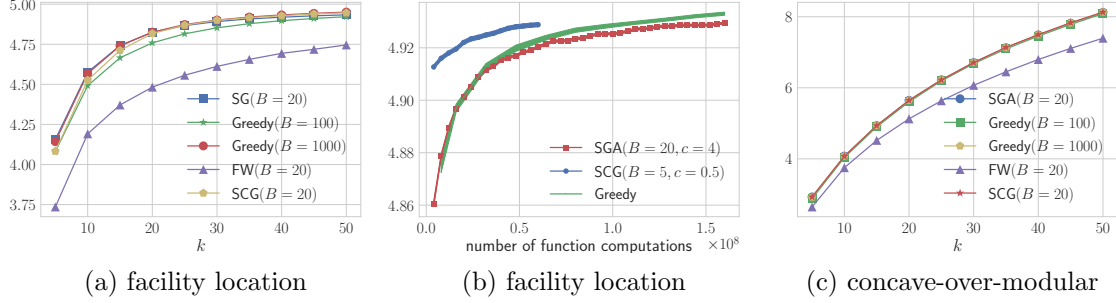


Figure 4: Comparison of the performances of SG/SGA, Greedy, FW, and SCG in a movie recommendation application. In all plots, y-axis represents the objective function value $f(S)$. Fig. 4a illustrates the performance of the algorithms in terms of the facility-location objective value w.r.t. the cardinality constraint size k after $T = 2000$ iterations. Fig. 4b compares the considered methods in terms of runtime (for a fixed $k = 40$) by illustrating the facility location objective function value vs. the number of (simple) function evaluations. Fig. 4c demonstrates the concave-over-modular objective function value vs. the size of the cardinality constraint k after running the algorithms for $T = 2000$ iterations.

In our second experiment, we consider a different user-specific valuation function which is a concave function composed with a modular function, i.e., $f(S, i) = (\sum_{j \in S} r_{i,j})^{1/2}$. Again, by considering the uniform distribution over the set of users, we obtain

$$f_{\text{con}}(S) = \frac{1}{N} \sum_{i=1}^N \left(\sum_{j \in S} r_{i,j} \right)^{1/2}.$$

Figure 4 depicts the performance of different algorithms for the two proposed objective functions. As Figures 4a and 4c show, the FW algorithm needs a higher mini-batch size to be comparable to SCG. Note that a smaller batch size leads to less computational effort (under the same value for b, T , the computational complexity of FW, SGA, SCG is almost the same). Figure 4b shows the performance of the algorithms with respect to the number of times the simple functions $f(\cdot, i)$ are evaluated. Note that the total number of (simple) function evaluations for SGA and SCG is nbT , where T is the number of iterations. Also, for Greedy the total number of evaluations is nkb . This further shows that SCG has a better computational complexity requirement w.r.t. SGA as well as the Greedy algorithm.

7. Conclusion

In this paper, we developed stochastic conditional gradient methods for solving convex minimization and submodular maximization problems. The main idea of the proposed methods in both domains was using a momentum term in the stochastic gradient approximation step to reduce the noise of the stochastic approximation. In particular, in the convex setting, we proposed a stochastic variant of the Frank-Wolfe algorithm called **Stochastic Frank-Wolfe** that achieves an ϵ -suboptimal objective function value after at

most $\mathcal{O}(1/\epsilon^3)$ iterations, if the expected objective function is smooth. The main advantage of the **Stochastic Frank-Wolfe** method (SFW) comparing to other stochastic conditional gradient methods for stochastic convex minimization is the required condition on the batch size. In particular, the batch size for SFW can be as small as 1, while the state-of-the-art stochastic conditional gradient methods require a growing batch size.

In the submodular setting, we provided the first tight approximation guarantee for maximizing a stochastic monotone DR-submodular function subject to a general convex body constraint. We developed **Stochastic Continuous Greedy** that achieves a $[(1 - 1/e)\text{OPT} - \epsilon]$ guarantee (in expectation) with $\mathcal{O}(1/\epsilon^3)$ stochastic gradient computations. We further extended our result to the non-monotone case and introduced **Non-monotone Stochastic Continuous Greedy** that obtains a $(1/e)$ approximation guarantee. We also demonstrated that our continuous algorithm can be used to provide the first $(1 - 1/e)$ tight approximation guarantee for maximizing a *monotone but stochastic* submodular *set* function subject to a general matroid constraint. Likewise, we provided the first $1/e$ approximation guarantee for maximizing a *non-monotone* stochastic submodular *set* function subject to a general matroid constraint. We believe that our results provide an important step towards unifying discrete and continuous submodular optimization in the stochastic setting.

Acknowledgments

This work was partially done while A. Mokhtari was visiting the Simons Institute for the Theory of Computing, and his work was partially supported by the DIMACS/Simons Collaboration on Bridging Continuous and Discrete Optimization through NSF grant CCF-1740425. The research of H. Hassani is supported by NSF HDR TRIPODS award 1934876, NSF award CPS-1837253, NSF award CIF-1910056, and NSF CAREER award CIF-1943064, and Air Force Office of Scientific Research Young Investigator Program (AFOSR-YIP) under award FA9550-20-1-0111. A. Karbasi is partially supported by NSF CAREER (IIS-1845032), ONR YIA (N00014-19-1-2406), and AFOSR YIP (FA9550-18-1-0160).

Appendix

A. Proof of Lemma 1

Use the definition $\mathbf{d}_t := (1 - \rho_t)\mathbf{d}_{t-1} + \rho_t \nabla \tilde{F}(\mathbf{x}_t, \mathbf{z}_t)$ to write $\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2$ as

$$\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2 = \|\nabla F(\mathbf{x}_t) - (1 - \rho_t)\mathbf{d}_{t-1} - \rho_t \nabla \tilde{F}(\mathbf{x}_t, \mathbf{z}_t)\|^2. \quad (47)$$

Add and subtract the term $(1 - \rho_t)\nabla F(\mathbf{x}_{t-1})$ to the right hand side of (47), regroup the terms and expand the squared term to obtain

$$\begin{aligned}
 & \|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2 \\
 &= \|\nabla F(\mathbf{x}_t) - (1 - \rho_t)\nabla F(\mathbf{x}_{t-1}) + (1 - \rho_t)\nabla F(\mathbf{x}_{t-1}) - (1 - \rho_t)\mathbf{d}_{t-1} - \rho_t\nabla\tilde{F}(\mathbf{x}_t, \mathbf{z}_t)\|^2 \\
 &= \|\rho_t(\nabla F(\mathbf{x}_t) - \nabla\tilde{F}(\mathbf{x}_t, \mathbf{z}_t)) + (1 - \rho_t)(\nabla F(\mathbf{x}_t) - \nabla F(\mathbf{x}_{t-1})) + (1 - \rho_t)(\nabla F(\mathbf{x}_{t-1}) - \mathbf{d}_{t-1})\|^2 \\
 &= \rho_t^2\|\nabla F(\mathbf{x}_t) - \nabla\tilde{F}(\mathbf{x}_t, \mathbf{z}_t)\|^2 + (1 - \rho_t)^2\|\nabla F(\mathbf{x}_t) - \nabla F(\mathbf{x}_{t-1})\|^2 \\
 &\quad + (1 - \rho_t)^2\|\nabla F(\mathbf{x}_{t-1}) - \mathbf{d}_{t-1}\|^2 \\
 &\quad + 2\rho_t(1 - \rho_t)(\nabla F(\mathbf{x}_t) - \nabla\tilde{F}(\mathbf{x}_t, \mathbf{z}_t))^T(\nabla F(\mathbf{x}_t) - \nabla F(\mathbf{x}_{t-1})) \\
 &\quad + 2\rho_t(1 - \rho_t)(\nabla F(\mathbf{x}_t) - \nabla\tilde{F}(\mathbf{x}_t, \mathbf{z}_t))^T(\nabla F(\mathbf{x}_{t-1}) - \mathbf{d}_{t-1}) \\
 &\quad + 2(1 - \rho_t)^2(\nabla F(\mathbf{x}_t) - \nabla F(\mathbf{x}_{t-1}))^T(\nabla F(\mathbf{x}_{t-1}) - \mathbf{d}_{t-1}). \tag{48}
 \end{aligned}$$

Compute the conditional expectation $\mathbb{E}[(\cdot) | \mathcal{F}_t]$ for both sides of (48), and use the fact that $\nabla\tilde{F}(\mathbf{x}_t, \mathbf{z}_t)$ is an unbiased estimator of the gradient $\nabla F(\mathbf{x}_t)$, i.e., $\mathbb{E}[\nabla\tilde{F}(\mathbf{x}_t, \mathbf{z}_t) | \mathcal{F}_t] = \nabla F(\mathbf{x}_t)$, to obtain

$$\begin{aligned}
 & \mathbb{E}[\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2 | \mathcal{F}_t] \\
 &= \rho_t^2\mathbb{E}[\|\nabla F(\mathbf{x}_t) - \nabla\tilde{F}(\mathbf{x}_t, \mathbf{z}_t)\|^2 | \mathcal{F}_t] + (1 - \rho_t)^2\|\nabla F(\mathbf{x}_t) - \nabla F(\mathbf{x}_{t-1})\|^2 \\
 &\quad + (1 - \rho_t)^2\|\nabla F(\mathbf{x}_{t-1}) - \mathbf{d}_{t-1}\|^2 + 2(1 - \rho_t)^2(\nabla F(\mathbf{x}_t) - \nabla F(\mathbf{x}_{t-1}))^T(\nabla F(\mathbf{x}_{t-1}) - \mathbf{d}_{t-1}). \tag{49}
 \end{aligned}$$

Use the condition in Assumption 3 to replace $\mathbb{E}[\|\nabla F(\mathbf{x}_t) - \nabla\tilde{F}(\mathbf{x}_t, \mathbf{z}_t)\|^2 | \mathcal{F}_t]$ by its upper bound σ^2 . Further, using Young's inequality to substitute the inner product $2\langle \nabla F(\mathbf{x}_t) - \nabla F(\mathbf{x}_{t-1}), \nabla F(\mathbf{x}_{t-1}) - \mathbf{d}_{t-1} \rangle$ by the upper bounded $\beta_t\|\nabla F(\mathbf{x}_{t-1}) - \mathbf{d}_{t-1}\|^2 + (1/\beta_t)\|\nabla F(\mathbf{x}_t) - \nabla F(\mathbf{x}_{t-1})\|^2$ where $\beta_t > 0$ is a free parameter. Applying these substitutions into (49) yields

$$\begin{aligned}
 \mathbb{E}[\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2 | \mathcal{F}_t] &\leq \rho_t^2\sigma^2 + (1 - \rho_t)^2(1 + \beta_t^{-1})\|\nabla F(\mathbf{x}_t) - \nabla F(\mathbf{x}_{t-1})\|^2 \\
 &\quad + (1 - \rho_t)^2(1 + \beta_t)\|\nabla F(\mathbf{x}_{t-1}) - \mathbf{d}_{t-1}\|^2. \tag{50}
 \end{aligned}$$

According to Assumption 2, the norm $\|\nabla F(\mathbf{x}_t) - \nabla F(\mathbf{x}_{t-1})\|$ is bounded above by $L\|\mathbf{x}_t - \mathbf{x}_{t-1}\|$. In addition, the condition in Assumption 4 implies that $L\|\mathbf{x}_t - \mathbf{x}_{t-1}\| = L\frac{1}{T}\|\mathbf{v}_t - \mathbf{x}_t\| \leq \frac{1}{T}LD$. Therefore, we can replace $\|\nabla F(\mathbf{x}_t) - \nabla F(\mathbf{x}_{t-1})\|$ in (50) by its upper bound $\frac{1}{T}LD$ and write

$$\begin{aligned}
 & \mathbb{E}[\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2 | \mathcal{F}_t] \\
 &\leq \rho_t^2\sigma^2 + \gamma_t^2(1 - \rho_t)^2(1 + \beta_t^{-1})L^2D^2 + (1 - \rho_t)^2(1 + \beta_t)\|\nabla F(\mathbf{x}_{t-1}) - \mathbf{d}_{t-1}\|^2. \tag{51}
 \end{aligned}$$

Since we assume that $\rho_t \leq 1$ we can replace all the terms $(1 - \rho_t)^2$ in (51) by $(1 - \rho_t)$. Applying this substitution into (51) and setting $\beta := \rho_t/2$ lead to the inequality

$$\begin{aligned}
 \mathbb{E}[\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2 | \mathcal{F}_t] &\leq \rho_t^2\sigma^2 + \gamma_t^2(1 - \rho_t)(1 + \frac{2}{\rho_t})L^2D^2 \\
 &\quad + (1 - \rho_t)(1 + \frac{\rho_t}{2})\|\nabla F(\mathbf{x}_{t-1}) - \mathbf{d}_{t-1}\|^2. \tag{52}
 \end{aligned}$$

Now using the inequalities $(1 - \rho_t)(1 + (2/\rho_t)) \leq (2/\rho_t)$ and $(1 - \rho_t)(1 + (\rho_t/2)) \leq (1 - \rho_t/2)$ we obtain

$$\mathbb{E} [\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2 \mid \mathcal{F}_t] \leq \rho_t^2 \sigma^2 + \frac{2L^2 D^2 \gamma_t^2}{\rho_t} + \left(1 - \frac{\rho_t}{2}\right) \|\nabla F(\mathbf{x}_{t-1}) - \mathbf{d}_{t-1}\|^2, \quad (53)$$

and the claim in (9) follows.

B. Proof of Lemma 2

Based on the L -smoothness of the expected function F we show that $F(\mathbf{x}_{t+1})$ is bounded above by

$$F(\mathbf{x}_{t+1}) \leq F(\mathbf{x}_t) + \nabla F(\mathbf{x}_t)^T (\mathbf{x}_{t+1} - \mathbf{x}_t) + \frac{L}{2} \|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2. \quad (54)$$

Replace the terms $\mathbf{x}_{t+1} - \mathbf{x}_t$ in (54) by $\gamma_{t+1}(\mathbf{v}_t - \mathbf{x}_t)$ and add and subtract the term $\gamma_{t+1} \mathbf{d}_t^T (\mathbf{v}_t - \mathbf{x}_t)$ to the right hand side of the resulted inequality to obtain

$$F(\mathbf{x}_{t+1}) \leq F(\mathbf{x}_t) + \gamma_{t+1}(\nabla F(\mathbf{x}_t) - \mathbf{d}_t)^T (\mathbf{v}_t - \mathbf{x}_t) + \gamma_{t+1} \mathbf{d}_t^T (\mathbf{v}_t - \mathbf{x}_t) + \frac{L\gamma_{t+1}^2}{2} \|\mathbf{v}_t - \mathbf{x}_t\|^2. \quad (55)$$

Since $\langle \mathbf{x}^*, \mathbf{d}_t \rangle \geq \min_{v \in \mathcal{C}} \{\langle \mathbf{v}, \mathbf{d}_t \rangle\} = \langle \mathbf{v}_t, \mathbf{d}_t \rangle$, we can replace the inner product $\langle \mathbf{v}, \mathbf{d}_t \rangle$ in (55) by its upper bound $\langle \mathbf{x}^*, \mathbf{d}_t \rangle$. Applying this substitution leads to

$$F(\mathbf{x}_{t+1}) \leq F(\mathbf{x}_t) + \gamma_{t+1}(\nabla F(\mathbf{x}_t) - \mathbf{d}_t)^T (\mathbf{v}_t - \mathbf{x}_t) + \gamma_{t+1} \mathbf{d}_t^T (\mathbf{x}^* - \mathbf{x}_t) + \frac{L\gamma_{t+1}^2}{2} \|\mathbf{v}_t - \mathbf{x}_t\|^2. \quad (56)$$

Add and subtract $\gamma_{t+1} \nabla F(\mathbf{x}_t)^T (\mathbf{x}^* - \mathbf{x}_t)$ to the right hand side of (56) and regroup the terms to obtain

$$\begin{aligned} F(\mathbf{x}_{t+1}) &\leq F(\mathbf{x}_t) + \gamma_{t+1}(\nabla F(\mathbf{x}_t) - \mathbf{d}_t)^T (\mathbf{v}_t - \mathbf{x}^*) + \gamma_{t+1} \nabla F(\mathbf{x}_t)^T (\mathbf{x}^* - \mathbf{x}_t) \\ &\quad + \frac{LD\gamma_{t+1}^2}{2} \|\mathbf{v}_t - \mathbf{x}_t\|^2. \end{aligned} \quad (57)$$

Using the Cauchy-Schwarz inequality, we can show that the inner product $(\nabla F(\mathbf{x}_t) - \mathbf{d}_t)^T (\mathbf{v}_t - \mathbf{x}^*)$ is bounded above by $\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\| \|\mathbf{v}_t - \mathbf{x}^*\|$. Moreover, the inner product $\nabla F(\mathbf{x}_t)^T (\mathbf{x}^* - \mathbf{x}_t)$ is upper bounded by $F(\mathbf{x}^*) - F(\mathbf{x}_t)$ due to the convexity of the function F . Applying these substitutions into (57) implies that

$$\begin{aligned} F(\mathbf{x}_{t+1}) &\leq F(\mathbf{x}_t) + \gamma_{t+1} \|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\| \|\mathbf{v}_t - \mathbf{x}^*\| - \gamma_{t+1} (F(\mathbf{x}_t) - F(\mathbf{x}^*)) + \frac{L\gamma_{t+1}^2}{2} \|\mathbf{v}_t - \mathbf{x}^*\|^2 \\ &\leq F(\mathbf{x}_t) + \gamma_{t+1} D \|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\| - \gamma_{t+1} (F(\mathbf{x}_t) - F(\mathbf{x}^*)) + \frac{LD^2 \gamma_{t+1}^2}{2}, \end{aligned} \quad (58)$$

where the second inequality holds since $\|\mathbf{v}_t - \mathbf{x}^*\| \leq D$ according to Assumption 1. Finally, subtract the optimal objective function value $F(\mathbf{x}^*)$ from both sides of (58) and regroup the terms to obtain

$$F(\mathbf{x}_{t+1}) - F(\mathbf{x}^*) \leq (1 - \gamma_{t+1})(F(\mathbf{x}_t) - F(\mathbf{x}^*)) + \gamma_{t+1} D \|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\| + \frac{LD^2 \gamma_{t+1}^2}{2}, \quad (59)$$

and the claim in (10) follows.

C. Proof of Theorem 3

Computing the expectation of both sides of (10) with respect to $\mathcal{F}_0$ yields

$$\mathbb{E}[F(\mathbf{x}_{t+1}) - F(\mathbf{x}^*)] \leq (1 - \gamma_{t+1})\mathbb{E}[(F(\mathbf{x}_t) - F(\mathbf{x}^*))] + \gamma_{t+1}D\mathbb{E}[\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|] + \frac{LD^2\gamma_{t+1}^2}{2} \quad (60)$$

Using Jensen's inequality, the expression $\mathbb{E}[\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|]$ can be substituted by its upper bound $\sqrt{\mathbb{E}[\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2]}$ to obtain

$$\begin{aligned} & \mathbb{E}[F(\mathbf{x}_{t+1}) - F(\mathbf{x}^*)] \\ & \leq (1 - \gamma_{t+1})\mathbb{E}[(F(\mathbf{x}_t) - F(\mathbf{x}^*))] + \gamma_{t+1}D\sqrt{\mathbb{E}[\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2]} + \frac{LD^2\gamma_{t+1}^2}{2}. \end{aligned} \quad (61)$$

Next, we study the rate that the sequence of expected gradient errors $\mathbb{E}[\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2]$ converges to zero. To do so, we first prove the following lemma which is an extension of Lemma 8 in (Mokhtari and Ribeiro, 2015).

Lemma 19 *Consider the scalars $b \geq 0$ and $c > 1$. Let ϕ_t be a sequence of real numbers satisfying*

$$\phi_t \leq \left(1 - \frac{c}{(t + t_0)^\alpha}\right) \phi_{t-1} + \frac{b}{(t + t_0)^{2\alpha}}, \quad (62)$$

for some $0 \leq \alpha \leq 1$ and $t_0 \geq 0$. Then, the sequence ϕ_t converges to zero at the following rate

$$\phi_t \leq \frac{Q}{(t + t_0 + 1)^\alpha}, \quad (63)$$

where $Q := \max\{\phi_0(t_0 + 1)^\alpha, b/(c - 1)\}$.

Proof We prove the claim in (63) by induction. First, note that $Q \geq \phi_0(t_0 + 1)^\alpha$ and therefore $\phi_0 \leq Q/(t_0 + 1)^\alpha$ and the base step of the induction holds true. Now assume that the condition in (63) holds for $t = k$, i.e.,

$$\phi_k \leq \frac{Q}{(k + 1 + t_0)^\alpha}. \quad (64)$$

The goal is to show that (63) also holds for $t = k + 1$. To do so, first set $t = k + 1$ in the expression in (62) to obtain

$$\phi_{k+1} \leq \left(1 - \frac{c}{(k + 1 + t_0)^\alpha}\right) \phi_k + \frac{b}{(k + 1 + t_0)^{2\alpha}}. \quad (65)$$

According to the definition of Q , we know that $b \leq Q(c - 1)$. Moreover, based on the induction hypothesis it holds that $\phi_k \leq \frac{Q}{(k+1+t_0)^\alpha}$. Using these inequalities and the expression in (65) we can write

$$\phi_{k+1} \leq \left(1 - \frac{c}{(k + 1 + t_0)^\alpha}\right) \frac{Q}{(k + 1 + t_0)^\alpha} + \frac{Q(c - 1)}{(k + 1 + t_0)^{2\alpha}}. \quad (66)$$

Pulling out $\frac{Q}{(k+1+t_0)^\alpha}$ as a common factor and simplifying and reordering terms it follows that (66) is equivalent to

$$\phi_{k+1} \leq Q \left(\frac{(k+1+t_0)^\alpha - 1}{(k+1+t_0)^{2\alpha}} \right). \quad (67)$$

Based on the inequality

$$((k+1+t_0)^\alpha - 1)((k+1+t_0)^\alpha + 1) < (k+1+t_0)^{2\alpha}, \quad (68)$$

the result in (67) implies that

$$\phi_{k+1} \leq \left(\frac{Q}{(k+1+t_0)^\alpha + 1} \right). \quad (69)$$

Since $(k+1+t_0)^\alpha + 1 \geq (k+1+t_0+1)^\alpha = (k+t_0+2)^\alpha$, the result in (69) implies that

$$\phi_{k+1} \leq \left(\frac{Q}{(k+2+t_0)^\alpha} \right), \quad (70)$$

and the induction step is complete. Therefore, the result in (63) holds for all $t \geq 0$. ■

Now using the result in Lemma 19 we can characterize the convergence of the sequence of expected errors $\mathbb{E} [\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2]$ to zero. To be more precise, compute the expectation of both sides of the result in (9) with respect to $\mathcal{F}_0$ and set $\gamma_t = 2/(t+8)$ and $\rho_t = 4/(t+8)^{2/3}$ to obtain

$$\mathbb{E} [\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2] \leq \left(1 - \frac{2}{(t+8)^{2/3}} \right) \mathbb{E} [\|\nabla F(\mathbf{x}_{t-1}) - \mathbf{d}_{t-1}\|^2] + \frac{16\sigma^2 + 2L^2D^2}{(t+8)^{4/3}}. \quad (71)$$

According to the result in Lemma 19, the inequality in (71) implies that

$$\mathbb{E} [\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2] \leq \frac{Q}{(t+9)^{2/3}}, \quad (72)$$

where $Q = \max\{5\|\nabla F(\mathbf{x}_0) - \mathbf{d}_0\|^2, 16\sigma^2 + 2L^2D^2\}$. This result is achieved by setting $\phi_t = \mathbb{E} [\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2]$, $\alpha = 2/3$, $b = 16\sigma^2 + 2L^2D^2$, $c = 2$, and $t_0 = 8$ in Lemma 19.

Now we proceed by replacing the term $\mathbb{E} [\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2]$ in (61) by its upper bound in (72) and γ_{t+1} by $2/(t+9)$ to write

$$\mathbb{E} [F(\mathbf{x}_{t+1}) - F(\mathbf{x}^*)] \leq \left(1 - \frac{2}{t+9} \right) \mathbb{E} [(F(\mathbf{x}_t) - F(\mathbf{x}^*))] + \frac{2D\sqrt{Q}}{(t+9)^{4/3}} + \frac{2LD^2}{(t+9)^2}. \quad (73)$$

Note that we can write $(t+9)^2 = (t+9)^{4/3}(t+9)^{2/3} \geq (t+9)^{4/3}9^{2/3} \geq 4(t+9)^{4/3}$. Therefore,

$$\mathbb{E} [F(\mathbf{x}_{t+1}) - F(\mathbf{x}^*)] \leq \left(1 - \frac{2}{t+9} \right) \mathbb{E} [(F(\mathbf{x}_t) - F(\mathbf{x}^*))] + \frac{2D\sqrt{Q} + LD^2/2}{(t+9)^{4/3}}. \quad (74)$$

Now we proceed to prove by induction for $t \geq 0$ that

$$\mathbb{E}[F(\mathbf{x}_t) - F(\mathbf{x}^*)] \leq \frac{Q'}{(t+9)^{1/3}} \quad (75)$$

where $Q' = \max\{9^{1/3}(F(\mathbf{x}_0) - F(\mathbf{x}^*)), 2D\sqrt{Q} + LD^2/2\}$. To do so, first note that $Q' \geq 9^{1/3}(F(\mathbf{x}_0) - F(\mathbf{x}^*))$, and, therefore, $F(\mathbf{x}_0) - F(\mathbf{x}^*) \leq Q'/9^{1/3}$. This leads to the base of induction for $t = 0$. Now assume that the inequality (75) holds for $t = k$, i.e., $\mathbb{E}[F(\mathbf{x}_k) - F(\mathbf{x}^*)] \leq \frac{Q'}{(k+9)^{1/3}}$ and we aim to show that it also holds for $t = k+1$.

To do so first set $t = k$ in (74) and replace $\mathbb{E}[F(\mathbf{x}_k) - F(\mathbf{x}^*)]$ by its upper bounds $\frac{Q'}{(k+9)^{1/3}}$ (as guaranteed by the hypothesis of the induction) to obtain

$$\mathbb{E}[F(\mathbf{x}_{k+1}) - F(\mathbf{x}^*)] \leq \left(1 - \frac{2}{k+9}\right) \frac{Q'}{(k+9)^{1/3}} + \frac{2D\sqrt{Q} + LD^2/2}{(k+9)^{4/3}}. \quad (76)$$

Now as in the proof of Lemma 19, replace $2D\sqrt{Q} + LD^2/2$ by Q' and simplify the terms to reach the inequality

$$\mathbb{E}[F(\mathbf{x}_{k+1}) - F(\mathbf{x}^*)] \leq Q' \left(\frac{k+8}{(k+9)^{4/3}} \right) \leq \frac{Q'}{(k+10)^{1/3}}, \quad (77)$$

and the induction is complete. Therefore, the inequality in (75) holds for all $t \geq 0$.

D. Proof of Theorem 4

To prove the claim in (13) we first show that the sum $\sum_{t=1}^{\infty} \rho_t \|\nabla F(\mathbf{x}_{t-1}) - \mathbf{d}_{t-1}\|^2$ is finite almost surely. To do so, we construct a supermartingale using the result in Lemma 1. Let's define the stochastic process ζ_t as

$$\zeta_t := \|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2 + 2L^2D^2 \sum_{u=t+1}^{\infty} \frac{\gamma_u^2}{\rho_u} + \sigma^2 \sum_{u=t+1}^{\infty} \rho_u^2. \quad (78)$$

Note that ζ_t is well defined because the sums on the right hand side of (78) are finite according to the hypotheses of Theorem 4. Further, define the stochastic process ξ_t as

$$\xi_t := \frac{\rho_{t+1}}{2} \|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2. \quad (79)$$

Considering the definitions of the sequences ζ_t and ξ_t and expression (9) in Lemma 1 we can write

$$\mathbb{E}[\zeta_t \mid \mathbf{x}_t] \leq \zeta_{t-1} - \xi_{t-1}. \quad (80)$$

Since the sequences ζ_t and ξ_t are nonnegative it follows from (80) that they satisfy the conditions of the supermartingale convergence theorem; see e.g. (Theorem E7.4 in (Solo and Kong, 1994)). Therefore, we can conclude that: (i) The sequence ζ_t converges almost surely to a limit. (ii) The sum $\sum_{t=0}^{\infty} \xi_t < \infty$ is almost surely finite. Hence, the second result implies that

$$\sum_{t=1}^{\infty} \rho_t \|\nabla F(\mathbf{x}_{t-1}) - \mathbf{d}_{t-1}\|^2 < \infty, \quad \text{a.s.} \quad (81)$$

Based on expression (10) in Lemma 2, we know that the suboptimality $F(\mathbf{x}_{t+1}) - F(\mathbf{x}^*)$ is upper bounded by

$$F(\mathbf{x}_{t+1}) - F(\mathbf{x}^*) \leq (1 - \gamma_{t+1})(F(\mathbf{x}_t) - F(\mathbf{x}^*)) + \gamma_{t+1}D\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\| + \frac{LD^2\gamma_{t+1}^2}{2}. \quad (82)$$

Further use Young's inequality to replace $\gamma_{t+1}\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|$ by the sum $\beta_{t+1}\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2 + \gamma_{t+1}^2/\beta_{t+1}$ where β_{t+1} is a free positive parameter. Set $\beta_{t+1} = \rho_{t+1}$ to obtain $\gamma_{t+1}\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\| \leq \rho_{t+1}\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2 + \gamma_{t+1}^2/\rho_{t+1}$. Applying this substitution into (82) implies that

$$F(\mathbf{x}_{t+1}) - F(\mathbf{x}^*) \leq (1 - \gamma_{t+1})(F(\mathbf{x}_t) - F(\mathbf{x}^*)) + \rho_{t+1}D\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2 + \frac{D\gamma_{t+1}^2}{\rho_{t+1}} + \frac{LD^2\gamma_{t+1}^2}{2}. \quad (83)$$

To conclude the almost sure convergence of the sequence $F(\mathbf{x}_t) - F(\mathbf{x}^*)$ to zero from the expression in (82) we first state the following Lemma from (Bertsekas and Tsitsiklis, 1996).

Lemma 20 *Let $\{X_t\}$, $\{Y_t\}$, and $\{Z_t\}$ be three sequences of numbers such that $Y_t \geq 0$ for all $t \geq 0$. Suppose that*

$$X_{t+1} \leq X_t - Y_t + Z_t, \quad \text{for } t = 0, 1, 2, \dots \quad (84)$$

and $\sum_{t=0}^{\infty} Z_t < \infty$. Then, either $X_t \rightarrow -\infty$ or else $\{X_t\}$ converges to a finite value and $\sum_{t=0}^{\infty} Y_t < \infty$.

From now on we focus on realizations that support $\sum_{t=1}^{\infty} \rho_t \|\nabla F(\mathbf{x}_{t-1}) - \mathbf{d}_{t-1}\|^2 < \infty$ which have probability 1, according to the result in (81).

Consider the outcome of Lemma 20 with the identifications $X_t = F(\mathbf{x}_t) - F(\mathbf{x}^*)$, $Y_t = \gamma_{t+1}(F(\mathbf{x}_t) - F(\mathbf{x}^*))$, and $Z_t = \rho_{t+1}D\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2 + \frac{D\gamma_{t+1}^2}{\rho_{t+1}} + \frac{LD^2\gamma_{t+1}^2}{2}$. Since the sequence $X_t = F(\mathbf{x}_t) - F(\mathbf{x}^*)$ is always non-negative, the first outcome of Lemma 20 is impossible and therefore we obtain that $F(\mathbf{x}_t) - F(\mathbf{x}^*)$ converges to a finite limit and

$$\sum_{t=0}^{\infty} \gamma_{t+1}F(\mathbf{x}_t) - F(\mathbf{x}^*) < \infty. \quad (85)$$

Recall that both of these results hold almost surely, since they are valid for the realization that $\sum_{t=1}^{\infty} \rho_t \|\nabla F(\mathbf{x}_{t-1}) - \mathbf{d}_{t-1}\|^2 < \infty$, which occur with probability 1 as shown in (81). The result in (85) implies that $\liminf_{t \rightarrow \infty} F(\mathbf{x}_t) - F(\mathbf{x}^*) = 0$ almost surely. Moreover, we know that the sequence $\{F(\mathbf{x}_t) - F(\mathbf{x}^*)\}$ almost surely converges to a finite limit. Combining these two observation we obtain that the finite limit is zero, and, therefore, $\lim_{t \rightarrow \infty} F(\mathbf{x}_t) - F(\mathbf{x}^*) = 0$ almost surely. Hence, the claim in (13) follows.

E. Proof of Lemma 6

By following the steps from (47) to (49) in the proof of Lemma 1 we can show that

$$\begin{aligned} & \mathbb{E} [\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2 \mid \mathcal{F}_t] \\ &= \rho_t^2 \mathbb{E} [\|\nabla F(\mathbf{x}_t) - \nabla \tilde{F}(\mathbf{x}_t, \mathbf{z}_t)\|^2 \mid \mathcal{F}_t] + (1 - \rho_t)^2 \|\nabla F(\mathbf{x}_{t-1}) - \mathbf{d}_{t-1}\|^2 \\ & \quad + (1 - \rho_t)^2 \|\nabla F(\mathbf{x}_t) - \nabla F(\mathbf{x}_{t-1})\|^2 + 2(1 - \rho_t)^2 \langle \nabla F(\mathbf{x}_t) - \nabla F(\mathbf{x}_{t-1}), \nabla F(\mathbf{x}_{t-1}) - \mathbf{d}_{t-1} \rangle. \end{aligned} \quad (86)$$

The term $\mathbb{E} [\|\nabla F(\mathbf{x}_t) - \nabla \tilde{F}(\mathbf{x}_t, \mathbf{z}_t)\|^2 \mid \mathcal{F}_t]$ can be bounded above by σ^2 according to Assumption 6. Based on Assumptions 4 and 5, we can also show that the squared norm $\|\nabla F(\mathbf{x}_t) - \nabla F(\mathbf{x}_{t-1})\|^2$ is upper bounded by $L^2 D^2 / T^2$. Moreover, the inner product $2\langle \nabla F(\mathbf{x}_t) - \nabla F(\mathbf{x}_{t-1}), \nabla F(\mathbf{x}_{t-1}) - \mathbf{d}_{t-1} \rangle$ can be upper bounded by $\beta_t \|\nabla F(\mathbf{x}_{t-1}) - \mathbf{d}_{t-1}\|^2 + (1/\beta_t) L^2 D^2 / T^2$ using Young's inequality (i.e., $2\langle \mathbf{a}, \mathbf{b} \rangle \leq \beta \|\mathbf{a}\|^2 + \|\mathbf{b}\|^2 / \beta$ for any $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ and $\beta > 0$) and the conditions in Assumptions 4 and 5, where $\beta_t > 0$ is a free scalar. Applying these substitutions into (49) leads to

$$\begin{aligned} & \mathbb{E} [\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2 \mid \mathcal{F}_t] \\ & \leq \rho_t^2 \sigma^2 + (1 - \rho_t)^2 \left(1 + \frac{1}{\beta_t}\right) \frac{L^2 D^2}{T^2} + (1 - \rho_t)^2 (1 + \beta_t) \|\nabla F(\mathbf{x}_{t-1}) - \mathbf{d}_{t-1}\|^2. \end{aligned} \quad (87)$$

Now by following the steps from (50) to (53) and computing the expected value with respect to $\mathcal{F}_0$ we obtain

$$\mathbb{E} [\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2] \leq \left(1 - \frac{\rho_t}{2}\right) \mathbb{E} [\|\nabla F(\mathbf{x}_{t-1}) - \mathbf{d}_{t-1}\|^2] + \rho_t^2 \sigma^2 + \frac{2L^2 D^2}{\rho_t T^2}. \quad (88)$$

Define $\phi_t := \mathbb{E} [\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2]$ and set $\rho_t = \frac{4}{(t+8)^{2/3}}$ to obtain

$$\phi_t \leq \left(1 - \frac{2}{(t+8)^{2/3}}\right) \phi_{t-1} + \frac{16\sigma^2}{(t+8)^{4/3}} + \frac{L^2 D^2 (t+8)^{2/3}}{2T^2}. \quad (89)$$

Now use the conditions $8 \leq T$ and $t \leq T$ to replace $1/T$ in (89) by its upper bound $2/(t+8)$. Applying this substitution leads to

$$\phi_t \leq \left(1 - \frac{2}{(t+8)^{2/3}}\right) \phi_{t-1} + \frac{16\sigma^2 + 2L^2 D^2}{(t+8)^{4/3}}. \quad (90)$$

Now using the result in Lemma 19, we obtain that

$$\phi_t \leq \frac{Q}{(t+9)^{2/3}}, \quad (91)$$

where $Q := \max\{5\phi_0, 16\sigma^2 + 2L^2 D^2\}$. Replacing ϕ_t by its definition $\mathbb{E} [\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2]$ yields (26).

F. Proof of Theorem 7

Let $\mathbf{x}^*$ be the global maximizer within the constraint set $\mathcal{C}$. Based on the smoothness of the function F with constant L we can write

$$\begin{aligned} F(\mathbf{x}_{t+1}) & \geq F(\mathbf{x}_t) + \langle \nabla F(\mathbf{x}_t), \mathbf{x}_{t+1} - \mathbf{x}_t \rangle - \frac{L}{2} \|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2 \\ & = F(\mathbf{x}_t) + \frac{1}{T} \langle \nabla F(\mathbf{x}_t), \mathbf{v}_t \rangle - \frac{L}{2T^2} \|\mathbf{v}_t\|^2, \end{aligned} \quad (92)$$

where the equality follows from the update in (22). Since $\mathbf{v}_t$ is in the set $\mathcal{C}$, it follows from Assumption 4 that the norm $\|\mathbf{v}_t\|^2$ is bounded above by D^2 . Apply this substitution and add and subtract the inner product $\langle \mathbf{d}_t, \mathbf{v}_t \rangle$ to the right hand side of (92) to obtain

$$\begin{aligned} F(\mathbf{x}_{t+1}) &\geq F(\mathbf{x}_t) + \frac{1}{T} \langle \mathbf{v}_t, \mathbf{d}_t \rangle + \frac{1}{T} \langle \mathbf{v}_t, \nabla F(\mathbf{x}_t) - \mathbf{d}_t \rangle - \frac{LD^2}{2T^2} \\ &\geq F(\mathbf{x}_t) + \frac{1}{T} \langle \mathbf{x}^*, \mathbf{d}_t \rangle + \frac{1}{T} \langle \mathbf{v}_t, \nabla F(\mathbf{x}_t) - \mathbf{d}_t \rangle - \frac{LD^2}{2T^2}. \end{aligned} \quad (93)$$

Note that the second inequality in (93) holds since based on (21) we can write

$$\langle \mathbf{x}^*, \mathbf{d}_t \rangle \leq \max_{\mathbf{v} \in \mathcal{C}} \{ \langle \mathbf{v}, \mathbf{d}_t \rangle \} = \langle \mathbf{v}_t, \mathbf{d}_t \rangle. \quad (94)$$

Now add and subtract the inner product $\langle \mathbf{x}^*, \nabla F(\mathbf{x}_t) \rangle / T$ to the RHS of (93) to get

$$F(\mathbf{x}_{t+1}) \geq F(\mathbf{x}_t) + \frac{1}{T} \langle \mathbf{x}^*, \nabla F(\mathbf{x}_t) \rangle + \frac{1}{T} \langle \mathbf{v}_t - \mathbf{x}^*, \nabla F(\mathbf{x}_t) - \mathbf{d}_t \rangle - \frac{LD^2}{2T^2}. \quad (95)$$

We further have $\langle \mathbf{x}^*, \nabla F(\mathbf{x}_t) \rangle \geq F(\mathbf{x}^*) - F(\mathbf{x}_t)$; this follows from monotonicity of F as well as concavity of F along positive directions; see, e.g., (Calinescu et al., 2011). Moreover, by Young's inequality we can show that the inner product $\langle \mathbf{v}_t - \mathbf{x}^*, \nabla F(\mathbf{x}_t) - \mathbf{d}_t \rangle$ is lower bounded by

$$\langle \mathbf{v}_t - \mathbf{x}^*, \nabla F(\mathbf{x}_t) - \mathbf{d}_t \rangle \geq -\frac{\beta_t}{2} \|\mathbf{v}_t - \mathbf{x}^*\|^2 - \frac{1}{2\beta_t} \|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2, \quad (96)$$

for any $\beta_t > 0$. By applying these substitutions into (95) we obtain

$$F(\mathbf{x}_{t+1}) \geq F(\mathbf{x}_t) + \frac{1}{T} (F(\mathbf{x}^*) - F(\mathbf{x}_t)) - \frac{LD^2}{2T^2} - \frac{1}{2T} \left(\beta_t \|\mathbf{v}_t - \mathbf{x}^*\|^2 + \frac{\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2}{\beta_t} \right). \quad (97)$$

Replace $\|\mathbf{v}_t - \mathbf{x}^*\|^2$ by its upper bound $4D^2$ and compute the expected value of (97) to write

$$\begin{aligned} \mathbb{E}[F(\mathbf{x}_{t+1})] &\geq \mathbb{E}[F(\mathbf{x}_t)] + \frac{1}{T} \mathbb{E}[F(\mathbf{x}^*) - F(\mathbf{x}_t)] - \frac{1}{2T} \left[4\beta_t D^2 + \frac{\mathbb{E}[\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2]}{\beta_t} \right] - \frac{LD^2}{2T^2}. \end{aligned} \quad (98)$$

Substitute $\mathbb{E}[\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2]$ by its upper bound $Q/((t+9)^{2/3})$ according to the result in (26). Further, set $\beta_t = (Q^{1/2})/(2D(t+9)^{1/3})$ and regroup the resulted expression to obtain

$$\mathbb{E}[F(\mathbf{x}^*) - F(\mathbf{x}_{t+1})] \leq \left(1 - \frac{1}{T}\right) \mathbb{E}[F(\mathbf{x}^*) - F(\mathbf{x}_t)] + \frac{2DQ^{1/2}}{(t+9)^{1/3}T} + \frac{LD^2}{2T^2}. \quad (99)$$

By applying the inequality in (99) recursively for $t = 0, \dots, T-1$ we obtain

$$\mathbb{E}[F(\mathbf{x}^*) - F(\mathbf{x}_T)] \leq \left(1 - \frac{1}{T}\right)^T (F(\mathbf{x}^*) - F(\mathbf{x}_0)) + \sum_{t=0}^{T-1} \frac{2DQ^{1/2}}{(t+9)^{1/3}T} + \sum_{t=0}^{T-1} \frac{LD^2}{2T^2}. \quad (100)$$

Note that we can write

$$\begin{aligned}
 \sum_{t=0}^{T-1} \frac{1}{(t+9)^{1/3}} &\leq \frac{1}{9^{1/3}} + \int_{t=0}^{T-1} \frac{1}{(t+9)^{1/3}} dt \\
 &= \frac{1}{9^{1/3}} + \frac{3}{2}(t+9)^{2/3} \Big|_{t=T-1} - \frac{3}{2}(t+9)^{2/3} \Big|_{t=0} \\
 &\leq \frac{3}{2}(T+8)^{2/3} \\
 &\leq \frac{15}{2}T^{2/3}
 \end{aligned} \tag{101}$$

where the last inequality holds since $(T+8)^{2/3} \leq 5T^{2/3}$ for any $T \geq 1$. By simplifying the terms on the right hand side (100) and using the inequality in (101) we can write

$$\mathbb{E}[F(\mathbf{x}^*) - F(\mathbf{x}_T)] \leq \frac{1}{e}(F(\mathbf{x}^*) - F(\mathbf{x}_0)) + \frac{15DQ^{1/2}}{T^{1/3}} + \frac{LD^2}{2T}. \tag{102}$$

Here, we use the fact that $F(\mathbf{x}_0) \geq 0$, and hence the expression in (102) can be simplified to

$$\mathbb{E}[F(\mathbf{x}_T)] \geq (1 - 1/e)F(\mathbf{x}^*) - \frac{15DQ^{1/2}}{T^{1/3}} - \frac{LD^2}{2T}, \tag{103}$$

and the claim in (27) follows.

G. Proof of Theorem 8

Following the steps of the proof of Theorem 7 we can derive the inequality

$$F(\mathbf{x}_{t+1}) \geq F(\mathbf{x}_t) + \frac{1}{T}\langle \mathbf{x}^*, \nabla F(\mathbf{x}_t) \rangle + \frac{1}{T}\langle \mathbf{v}_t - \mathbf{x}^*, \nabla F(\mathbf{x}_t) - \mathbf{d}_t \rangle - \frac{LD^2}{2T^2}. \tag{104}$$

Using the definition of weak DR-submodularity and monotonicity of F we can show that $\langle \mathbf{x}^*, \nabla F(\mathbf{x}_t) \rangle \geq \gamma(F(\mathbf{x}^*) - F(\mathbf{x}_t))$. Further, based on Young's inequality the inner product $\langle \mathbf{v}_t - \mathbf{x}^*, \nabla F(\mathbf{x}_t) - \mathbf{d}_t \rangle$ is lower bounded by $-(\beta_t/2)\|\mathbf{v}_t - \mathbf{x}^*\|^2 - (1/2\beta_t)\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2$ for any $\beta_t > 0$. Applying these substitutions into (104) leads to

$$F(\mathbf{x}_{t+1}) \geq F(\mathbf{x}_t) + \frac{\gamma}{T}(F(\mathbf{x}^*) - F(\mathbf{x}_t)) - \frac{LD^2}{2T^2} - \frac{1}{2T} \left(\beta_t \|\mathbf{v}_t - \mathbf{x}^*\|^2 + \frac{\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2}{\beta_t} \right). \tag{105}$$

Substitute $\|\mathbf{v}_t - \mathbf{x}^*\|^2$ by its upper bound $4D^2$ and compute the expected value of (105) to write

$$\begin{aligned}
 &\mathbb{E}[F(\mathbf{x}_{t+1})] \\
 &\geq \mathbb{E}[F(\mathbf{x}_t)] + \frac{\gamma}{T}\mathbb{E}[F(\mathbf{x}^*) - F(\mathbf{x}_t)] - \frac{1}{2T} \left[4\beta_t D^2 + \frac{\mathbb{E}[\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2]}{\beta_t} \right] - \frac{LD^2}{2T^2}.
 \end{aligned} \tag{106}$$

Substitute $\mathbb{E} [\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2]$ by its upper bound $Q/((t+9)^{2/3})$ according to the result in (26). Further, set $\beta_t = (Q^{1/2})/(2D(t+9)^{1/3})$ and regroup the resulted expression to obtain

$$\mathbb{E} [F(\mathbf{x}^*) - F(\mathbf{x}_{t+1})] \leq \left(1 - \frac{\gamma}{T}\right) \mathbb{E} [F(\mathbf{x}^*) - F(\mathbf{x}_t)] + \frac{2DQ^{1/2}}{(t+9)^{1/3}T} + \frac{LD^2}{2T^2}. \quad (107)$$

By applying the inequality in (107) recursively for $t = 0, \dots, T-1$ we obtain

$$\mathbb{E} [F(\mathbf{x}^*) - F(\mathbf{x}_T)] \leq \left(1 - \frac{\gamma}{T}\right)^T (F(\mathbf{x}^*) - F(\mathbf{x}_0)) + \sum_{t=0}^{T-1} \frac{2DQ^{1/2}}{(t+9)^{1/3}T} + \sum_{t=0}^{T-1} \frac{LD^2}{2T^2}. \quad (108)$$

Simplify the terms on the right hand side (108) and use (101) to obtain

$$\mathbb{E} [F(\mathbf{x}^*) - F(\mathbf{x}_T)] \leq e^{-\gamma} (F(\mathbf{x}^*) - F(\mathbf{x}_0)) + \frac{15DQ^{1/2}}{T^{1/3}} + \frac{LD^2}{2T}. \quad (109)$$

Here, we use the fact that $F(\mathbf{x}_0) \geq 0$, and hence the expression in (109) can be simplified to

$$\mathbb{E} [F(\mathbf{x}_T)] \geq (1 - e^{-\gamma})F(\mathbf{x}^*) - \frac{15DQ^{1/2}}{T^{1/3}} - \frac{LD^2}{2T}, \quad (110)$$

and the claim in (29) follows.

H. Proof of Theorem 9

Using the update of the NMSCG method we can write

$$\begin{aligned} \mathbf{x}_{i,t+1} &= \mathbf{x}_{i,t} + \frac{1}{T} \mathbf{v}_{i,t} \\ &\leq \mathbf{x}_{i,t} + \frac{1}{T} (\bar{\mathbf{u}}_i - \mathbf{x}_{i,t}) \\ &\leq \left(1 - \frac{1}{T}\right) \mathbf{x}_{i,t} + \frac{1}{T} \bar{\mathbf{u}}_i \\ &= \left(1 - \frac{1}{T}\right)^t \mathbf{x}_{i,0} + \frac{1}{T} \bar{\mathbf{u}}_i \sum_{j=0}^t \left(1 - \frac{1}{T}\right)^j \\ &= \bar{\mathbf{u}}_i \left(1 - \left(1 - \frac{1}{T}\right)^{t+1}\right) \end{aligned} \quad (111)$$

where the first inequality follows by the condition $\mathbf{v}_{i,t} \leq \bar{\mathbf{u}}_i - \mathbf{x}_{i,t}$. The result in (111) implies that $\mathbf{x}_t \leq \bar{\mathbf{u}}(1 - (1 - 1/T)^t)$. Therefore, according to Lemma 3 in (Bian et al., 2017a) which is a generalized version of Lemma 7 in (Chekuri et al., 2015) it follows that

$$F(\mathbf{x}_t \vee \mathbf{x}^*) \geq (1 - (1/T))^t F(\mathbf{x}^*). \quad (112)$$

Using this result and the Taylor's expansion of the objective function F we can write

$$\begin{aligned} F(\mathbf{x}_{t+1}) - F(\mathbf{x}_t) &\geq \langle \nabla F(\mathbf{x}_t), \mathbf{x}_{t+1} - \mathbf{x}_t \rangle - \frac{L}{2} \|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2 \\ &= \frac{1}{T} \langle \nabla F(\mathbf{x}_t), \mathbf{v}_t \rangle - \frac{L}{2T^2} \|\mathbf{v}_t\|^2, \end{aligned} \quad (113)$$

where the equality follows by the update of NMSCG. Add and subtract $\langle \mathbf{d}_t, \mathbf{v}_t \rangle$ to the right hand side of (113) to obtain

$$\begin{aligned} F(\mathbf{x}_{t+1}) - F(\mathbf{x}_t) &\geq \frac{1}{T} \langle \mathbf{d}_t, \mathbf{v}_t \rangle + \frac{1}{T} \langle \nabla F(\mathbf{x}_t) - \mathbf{d}_t, \mathbf{v}_t \rangle - \frac{L}{2T^2} \|\mathbf{v}_t\|^2 \\ &\geq \frac{1}{T} \langle \mathbf{d}_t, \mathbf{x}_t \vee \mathbf{x}^* - \mathbf{x}_t \rangle + \frac{1}{T} \langle \nabla F(\mathbf{x}_t) - \mathbf{d}_t, \mathbf{v}_t \rangle - \frac{L}{2T^2} \|\mathbf{v}_t\|^2, \end{aligned} \quad (114)$$

where the second inequality is valid since $\{\mathbf{d}_t^T \mathbf{v}_t\} \geq \{\mathbf{d}_t^T \mathbf{y}\}$ for all $\mathbf{y} \leq \bar{\mathbf{u}} - \mathbf{x}_t$ and we know that $\mathbf{x}_t \vee \mathbf{x}^* - \mathbf{x}_t \leq \bar{\mathbf{u}} - \mathbf{x}_t$. Now add and subtract $(1/T) \langle \nabla F(\mathbf{x}_t), \mathbf{x}_t \vee \mathbf{x}^* - \mathbf{x}_t \rangle$ to the right hand side of (114) to obtain

$$\begin{aligned} &F(\mathbf{x}_{t+1}) - F(\mathbf{x}_t) \\ &\geq \frac{1}{T} \langle \nabla F(\mathbf{x}_t), \mathbf{x}_t \vee \mathbf{x}^* - \mathbf{x}_t \rangle + \frac{1}{T} \langle \mathbf{d}_t - \nabla F(\mathbf{x}_t), \mathbf{x}_t \vee \mathbf{x}^* - \mathbf{x}_t \rangle \\ &\quad + \frac{1}{T} \langle \nabla F(\mathbf{x}_t) - \mathbf{d}_t, \mathbf{v}_t \rangle - \frac{L}{2T^2} \|\mathbf{v}_t\|^2 \\ &= \frac{1}{T} \langle \nabla F(\mathbf{x}_t), \mathbf{x}_t \vee \mathbf{x}^* - \mathbf{x}_t \rangle + \frac{1}{T} \langle \nabla F(\mathbf{x}_t) - \mathbf{d}_t, \mathbf{v}_t + \mathbf{x}_t - \mathbf{x}_t \vee \mathbf{x}^* \rangle - \frac{L}{2T^2} \|\mathbf{v}_t\|^2 \\ &\geq \frac{1}{T} \langle \nabla F(\mathbf{x}_t), \mathbf{x}_t \vee \mathbf{x}^* - \mathbf{x}_t \rangle - \frac{2D}{T} \|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\| - \frac{L}{2T^2} \|\mathbf{v}_t\|^2. \end{aligned} \quad (115)$$

The last inequality holds since the inner product $\langle \nabla F(\mathbf{x}_t) - \mathbf{d}_t, \mathbf{v}_t + \mathbf{x}_t - \mathbf{x}_t \vee \mathbf{x}^* \rangle$ can be upper bounded by $\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\| \|\mathbf{v}_t + \mathbf{x}_t - \mathbf{x}_t \vee \mathbf{x}^*\|$ using the Cauchy-Schwartz inequality and further we can upper bound the norm $\|\mathbf{v}_t + \mathbf{x}_t - \mathbf{x}_t \vee \mathbf{x}^*\|$ by $2D$ since $\|\mathbf{v}_t + \mathbf{x}_t - \mathbf{x}_t \vee \mathbf{x}^*\| \leq \|\mathbf{v}_t\| + \|\mathbf{x}_t - \mathbf{x}_t \vee \mathbf{x}^*\|$ and both $\mathbf{x}_t - \mathbf{x}_t \vee \mathbf{x}^*$ and $\mathbf{v}_t$ belong to the set $\mathcal{C}$. This holds due to the assumption that $\mathcal{C}$ is down-closed.

Now replace $\|\mathbf{v}_t\|^2$ by its upper bound D^2 and use the concavity of the function F in positive directions to write

$$F(\mathbf{x}_{t+1}) - F(\mathbf{x}_t) \geq \frac{1}{T} (F(\mathbf{x}_t \vee \mathbf{x}^*) - F(\mathbf{x}_t)) - \frac{2D}{T} \|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\| - \frac{LD^2}{2T^2}. \quad (116)$$

Now use the expression in (112) to obtain

$$F(\mathbf{x}_{t+1}) - F(\mathbf{x}_t) \geq \frac{1}{T} \left[\left(1 - \frac{1}{T}\right)^t F(\mathbf{x}^*) - F(\mathbf{x}_t) \right] - \frac{2D}{T} \|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\| - \frac{LD^2}{2T^2}. \quad (117)$$

Computing the expectation of both sides and replacing $\mathbb{E}[\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|]$ by its upper bound $\sqrt{Q}/((t+9)^{1/3})$ according to the result in (26) lead to

$$\mathbb{E}[F(\mathbf{x}_{t+1})] \geq \left(1 - \frac{1}{T}\right) \mathbb{E}[F(\mathbf{x}_t)] + \frac{1}{T} \left(1 - \frac{1}{T}\right)^t F(\mathbf{x}^*) - \frac{2D\sqrt{Q}}{T(t+9)^{1/3}} - \frac{LD^2}{2T^2}. \quad (118)$$

By applying the inequality in (118) recursively for $t = 0, \dots, T - 1$ we obtain

$$\begin{aligned}
 & \mathbb{E}[F(\mathbf{x}_T)] \\
 & \geq \left(1 - \frac{1}{T}\right)^T F(\mathbf{x}_0) + \sum_{i=0}^{T-1} \left(\frac{1}{T} \left(1 - \frac{1}{T}\right)^i F(\mathbf{x}^*) - \frac{2D\sqrt{Q}}{T(i+9)^{1/3}} - \frac{LD^2}{2T^2} \right) \left(1 - \frac{1}{T}\right)^{T-1-i} \\
 & = \left(1 - \frac{1}{T}\right)^T F(\mathbf{x}_0) + \left(1 - \frac{1}{T}\right)^{T-1} F(\mathbf{x}^*) - \sum_{i=0}^{T-1} \left(\frac{2D\sqrt{Q}}{T(i+9)^{1/3}} + \frac{LD^2}{2T^2} \right) \left(1 - \frac{1}{T}\right)^{T-1-i} \\
 & \geq \left(1 - \frac{1}{T}\right)^T F(\mathbf{x}_0) + \left(1 - \frac{1}{T}\right)^{T-1} F(\mathbf{x}^*) - \sum_{i=0}^{T-1} \left(\frac{2D\sqrt{Q}}{T(i+9)^{1/3}} + \frac{LD^2}{2T^2} \right), \tag{119}
 \end{aligned}$$

where the second inequality holds since $(1 - 1/T)$ is strictly smaller than 1. Replacing the sum $\sum_{i=0}^{T-1} \frac{1}{(i+9)^{1/3}}$ in (119) by its upper bound in (101) leads to

$$\mathbb{E}[F(\mathbf{x}_T)] \geq \left(1 - \frac{1}{T}\right)^T F(\mathbf{x}_0) + \left(1 - \frac{1}{T}\right)^{T-1} F(\mathbf{x}^*) - \frac{15D\sqrt{Q}}{T^{1/3}} - \frac{LD^2}{2T} \tag{120}$$

Use the fact that $F(\mathbf{x}_0) \geq 0$ and the inequality $(1 - \frac{1}{T})^{T-1} \geq e^{-1}$ to obtain

$$\mathbb{E}[F(\mathbf{x}_T)] \geq e^{-1} F(\mathbf{x}^*) - \frac{15D\sqrt{Q}}{T^{1/3}} - \frac{LD^2}{2T}, \tag{121}$$

and the claim in (32) follows.

I. Proof of Lemma 11

Based on the mean value theorem, we can write

$$\nabla F(\mathbf{x}_t + \frac{1}{T}\mathbf{v}_t) - \nabla F(\mathbf{x}_T) = \frac{1}{T}\mathbf{H}(\tilde{\mathbf{x}}_t)\mathbf{v}_t, \tag{122}$$

where $\tilde{\mathbf{x}}_t$ is a convex combination of $\mathbf{x}_t$ and $\mathbf{x}_t + \frac{1}{T}\mathbf{v}_t$ and $\mathbf{H}(\tilde{\mathbf{x}}_t) := \nabla^2 F(\tilde{\mathbf{x}}_t)$. This expression shows that the difference between the coordinates of the vectors $\nabla F(\mathbf{x}_t + \frac{1}{T}\mathbf{v}_t)$ and $\nabla F(\mathbf{x}_t)$ can be written as

$$\nabla_j F(\mathbf{x}_t + \frac{1}{T}\mathbf{v}_t) - \nabla_j F(\mathbf{x}_t) = \frac{1}{T} \sum_{i=1}^n H_{j,i}(\tilde{\mathbf{x}}_t) v_{i,t}, \tag{123}$$

where $v_{i,t}$ is the i -th element of the vector $\mathbf{v}_t$ and $H_{j,i}$ denotes the component in the j -th row and i -th column of the matrix $\mathbf{H}$. Hence, the norm of the difference $|\nabla_j F(\mathbf{x}_t + \frac{1}{T}\mathbf{v}_t) - \nabla_j F(\mathbf{x}_t)|$ is bounded above by

$$|\nabla_j F(\mathbf{x}_t + \frac{1}{T}\mathbf{v}_t) - \nabla_j F(\mathbf{x}_t)| \leq \frac{1}{T} \left| \sum_{i=1}^n H_{j,i}(\tilde{\mathbf{x}}_t) v_{i,t} \right|. \tag{124}$$

Note here that the elements of the matrix $\mathbf{H}(\tilde{\mathbf{x}}_t)$ are less than the maximum marginal value (i.e. $\max_{i,j} |H_{i,j}(\tilde{\mathbf{x}}_t)| \leq \max_{i \in \{1, \dots, n\}} f(i) \triangleq m_f$). We thus get

$$|\nabla_j F(\mathbf{x}_t + \frac{1}{T} \mathbf{v}_t) - \nabla_j F(\mathbf{x}_t)| \leq \frac{m_f}{T} \sum_{i=1}^n |v_{i,t}|. \quad (125)$$

Note that at each round t of the algorithm, we have to pick a vector $\mathbf{v}_t \in \mathcal{C}$ s.t. the inner product $\langle \mathbf{v}_t, \mathbf{d}_t \rangle$ is maximized. Hence, without loss of generality we can assume that the vector $\mathbf{v}_t$ is one of the extreme points of $\mathcal{C}$, i.e. it is of the form $\mathbf{1}_I$ for some $I \in \mathcal{I}$ (note that we can easily force integer vectors). Therefore by noticing that $\mathbf{v}_t$ is an integer vector with at most r ones, we have

$$|\nabla_j F(\mathbf{x}_t + \frac{1}{T} \mathbf{v}_t) - \nabla_j F(\mathbf{x}_t)| \leq \frac{m_f \sqrt{r}}{T} \sqrt{\sum_{i=1}^n |v_{i,t}|^2}, \quad (126)$$

which yields the claim in (36).

J. Proof of Theorem 12

According to the Taylor's expansion of the function F near the point $\mathbf{x}_t$ we can write

$$\begin{aligned} F(\mathbf{x}_{t+1}) &= F(\mathbf{x}_t) + \langle \nabla F(\mathbf{x}_t), \mathbf{x}_{t+1} - \mathbf{x}_t \rangle + \frac{1}{2} \langle \mathbf{x}_{t+1} - \mathbf{x}_t, \mathbf{H}(\tilde{\mathbf{x}}_t)(\mathbf{x}_{t+1} - \mathbf{x}_t) \rangle \\ &= F(\mathbf{x}_t) + \frac{1}{T} \langle \nabla F(\mathbf{x}_t), \mathbf{v}_t \rangle + \frac{1}{2T^2} \langle \mathbf{v}_t, \mathbf{H}(\tilde{\mathbf{x}}_t) \mathbf{v}_t \rangle, \end{aligned} \quad (127)$$

where $\tilde{\mathbf{x}}_t$ is a convex combination of $\mathbf{x}_t$ and $\mathbf{x}_t + \frac{1}{T} \mathbf{v}_t$ and $\mathbf{H}(\tilde{\mathbf{x}}_t) := \nabla^2 F(\tilde{\mathbf{x}}_t)$. Note that based on the inequality $\max_{i,j} |H_{i,j}(\tilde{\mathbf{x}}_t)| \leq \max_{i \in \{1, \dots, n\}} f(i) \triangleq m_f$, we can lower bound H_{ij} by $-m_f$. Therefore,

$$\begin{aligned} \langle \mathbf{v}_t, \mathbf{H}(\tilde{\mathbf{x}}_t) \mathbf{v}_t \rangle &= \sum_{j=1}^n \sum_{i=1}^n v_{i,t} v_{j,t} H_{ij}(\tilde{\mathbf{x}}_t) \\ &\geq -m_f \sum_{j=1}^n \sum_{i=1}^n v_{i,t} v_{j,t} = -m_f \left(\sum_{i=1}^n v_{i,t} \right)^2 = -m_f r \|\mathbf{v}_t\|^2, \end{aligned} \quad (128)$$

where the last equality is because $\mathbf{v}_t$ is a vector with r ones and $n - r$ zeros (see the explanation in the proof of Lemma 11). Replace the expression $\langle \mathbf{v}_t, \mathbf{H}(\tilde{\mathbf{x}}_t) \mathbf{v}_t \rangle$ in (127) by its lower bound in (128) to obtain

$$F(\mathbf{x}_{t+1}) \geq F(\mathbf{x}_t) + \frac{1}{T} \langle \nabla F(\mathbf{x}_t), \mathbf{v}_t \rangle - \frac{m_f r}{2T^2} \|\mathbf{v}_t\|^2. \quad (129)$$

In the following lemma we derive a variant of the result in Lemma 6 for the multilinear extension setting.

Lemma 21 *Consider Stochastic Continuous Greedy (SCG) outlined in Algorithm 2, and recall the definitions of the function F in (34), the rank r , and $m_f \triangleq \max_{i \in \{1, \dots, n\}} f(i)$. If we set $\rho_t = \frac{4}{(t+8)^{2/3}}$, then for $t = 0, \dots, T$ and for $j = 1, \dots, n$ it holds*

$$\mathbb{E} [\|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|^2] \leq \frac{Q}{(t+9)^{2/3}}, \quad (130)$$

where $Q := \max\{5\|\nabla F(\mathbf{x}_0) - \mathbf{d}_0\|^2, 16\sigma^2 + 3m_f^2 r D^2\}$.

Proof The proof is similar to the proof of Lemma 6. The main difference is to write the analysis for the j -th coordinate and replace L by $m_f \sqrt{r}$ as shown in Lemma 11. Then using the proof techniques in Lemma 6 the claim in Lemma 21 follows. $\blacksquare$

The rest of the proof is identical to the proof of Theorem 7, by following the steps from (92) to (103) and considering the bound in (130) we obtain

$$\mathbb{E} [F(\mathbf{x}_T)] \geq (1 - 1/e)F(\mathbf{x}^*) - \frac{2DQ^{1/2}}{T^{1/3}} - \frac{m_f r D^2}{2T}, \quad (131)$$

where $Q := \max\{5\|\nabla F(\mathbf{x}_0) - \mathbf{d}_0\|^2, 16\sigma^2 + 3m_f^2 r D^2\}$. Therefore, the claim in Theorem 12 follows by replacing σ^2 by $n \max_{j \in [n]} \mathbb{E}[\tilde{f}(\{j\}, \mathbf{z})^2]$ as shown in (38).

K. Proof of Theorem 15

Following the steps of the proof of Theorem 12 from (127) to (129) we obtain

$$F(\mathbf{x}_{t+1}) \geq F(\mathbf{x}_t) + \frac{1}{T} \langle \nabla F(\mathbf{x}_t), \mathbf{v}_t \rangle - \frac{m_f r}{2T^2} \|\mathbf{v}_t\|^2. \quad (132)$$

Then, by following the steps from (112) to (117) we obtain

$$F(\mathbf{x}_{t+1}) - F(\mathbf{x}_t) \geq \frac{1}{T} \left[\left(1 - \frac{1}{T}\right)^t F(\mathbf{x}^*) - F(\mathbf{x}_t) \right] - \frac{2D}{T} \|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\| - \frac{m_f r D^2}{2T^2}. \quad (133)$$

Note that the result in Lemma 21 holds for both monotone and non-monotone functions F . Therefore, by computing the expected value of both sides of (117) we obtain Then, by following the steps from (112) to (117) we obtain

$$\mathbb{E} [F(\mathbf{x}_{t+1})] - \mathbb{E} [F(\mathbf{x}_t)] \geq \frac{1}{T} \left[\left(1 - \frac{1}{T}\right)^t F(\mathbf{x}^*) - \mathbb{E} [F(\mathbf{x}_t)] \right] - \frac{2D}{T} \frac{\sqrt{Q}}{(t+9)^{1/3}} - \frac{m_f r D^2}{2T^2}. \quad (134)$$

Now by following the steps from (118) to (121) the claim in (40) follows.

L. Proof of Theorem 17

According to the result in Lemma 3 of (Hassani et al., 2017), it can be shown that when the function F has a curvature c then

$$\max_{\mathbf{v}} \mathbf{v}^T \nabla F(\mathbf{x}_t) \geq F(\mathbf{x}^*) - cF(\mathbf{x}_t), \quad (135)$$

Using this inequality we can write

$$\begin{aligned} \mathbf{v}_t^T \mathbf{d}_t &= \max_{\mathbf{v}} \{\mathbf{v}^T \mathbf{d}_t\} \\ &= \max_{\mathbf{v}} \{\mathbf{v}^T \nabla F(\mathbf{x}_t) + \mathbf{v}^T (\nabla F(\mathbf{x}_t) - \mathbf{d}_t)\} \\ &\geq \max_{\mathbf{v}} \{\mathbf{v}^T \nabla F(\mathbf{x}_t)\} - \max_{\mathbf{v}} \{\mathbf{v}^T (\nabla F(\mathbf{x}_t) - \mathbf{d}_t)\} \\ &\geq F(\mathbf{x}^*) - cF(\mathbf{x}_t) - D \|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\|. \end{aligned} \quad (136)$$

Therefore, using the result in (129) we can write

$$\begin{aligned} F(\mathbf{x}_{t+1}) &\geq F(\mathbf{x}_t) + \frac{1}{T} \langle \mathbf{v}_t, \nabla F(\mathbf{x}_t) \rangle - \frac{m_f r}{2T^2} \|\mathbf{v}_t\|^2 \\ &= F(\mathbf{x}_t) + \frac{1}{T} \langle \mathbf{v}_t, \mathbf{d}_t \rangle + \frac{1}{T} \langle \mathbf{v}_t, \nabla F(\mathbf{x}_t) - \mathbf{d}_t \rangle - \frac{L}{2T^2} \|\mathbf{v}_t\|^2 \\ &\geq F(\mathbf{x}_t) + \frac{1}{T} (F(\mathbf{x}^*) - cF(\mathbf{x}_t)) - \frac{2D}{T} \|\nabla F(\mathbf{x}_t) - \mathbf{d}_t\| - \frac{m_f r D^2}{2T^2}. \end{aligned} \quad (137)$$

Compute the expected value of both sides and use the result of Lemma 21 to obtain

$$\mathbb{E}[F(\mathbf{x}_{t+1})] \geq \mathbb{E}[F(\mathbf{x}_t)] + \frac{1}{T} (F(\mathbf{x}^*) - c\mathbb{E}[F(\mathbf{x}_t)]) - \frac{2D\sqrt{Q}}{T(t+9)^{1/3}} - \frac{m_f r D^2}{2T^2}. \quad (138)$$

Subtract $F^* := F(\mathbf{x}^*)$ from both sides and regroup the terms to obtain

$$\mathbb{E}[F(\mathbf{x}_{t+1})] - F^* \geq (1 - \frac{c}{T})(\mathbb{E}[F(\mathbf{x}_t)] - F^*) + \frac{1-c}{T} F^* - \frac{2D\sqrt{Q}}{T(t+9)^{1/3}} - \frac{m_f r D^2}{2T^2}. \quad (139)$$

Now applying the expression in (139) for $t = 0, \dots, T-1$ recursively yields

$$\begin{aligned} &\mathbb{E}[F(\mathbf{x}_T)] - F^* \\ &\geq (1 - \frac{c}{T})^T (F(\mathbf{x}_0) - F^*) + \sum_{i=0}^{T-1} \left[\frac{1-c}{T} F^* - \frac{2D\sqrt{Q}}{T(i+9)^{1/3}} - \frac{m_f r D^2}{2T^2} \right] (1 - \frac{c}{T})^{T-1-i} \\ &\geq (1 - \frac{c}{T})^T (F(\mathbf{x}_0) - F^*) + \frac{1-c}{T} F^* \sum_{i=0}^{T-1} (1 - \frac{c}{T})^{T-1-i} - \sum_{i=0}^{T-1} \left[\frac{2D\sqrt{Q}}{T(i+9)^{1/3}} + \frac{m_f r D^2}{2T^2} \right] \\ &\geq (1 - \frac{c}{T})^T (F(\mathbf{x}_0) - F^*) + \frac{1-c}{c} \left(1 - (1 - \frac{c}{T})^T \right) F^* - \frac{15D\sqrt{Q}}{T^{1/3}} - \frac{m_f r D^2}{2T} \\ &\geq -(1 - \frac{c}{T})^T F^* + \frac{1-c}{c} \left(1 - (1 - \frac{c}{T})^T \right) F^* - \frac{15D\sqrt{Q}}{T^{1/3}} - \frac{m_f r D^2}{2T} \\ &= \left(\frac{1-c}{c} - \frac{1}{c} (1 - \frac{c}{T})^T \right) F^* - \frac{15D\sqrt{Q}}{T^{1/3}} - \frac{m_f r D^2}{2T}, \end{aligned} \quad (140)$$

where in the second inequality we use the fact that $(1 - c/T) \leq 1$, in the third inequality we use the result in (101), and in the fourth inequality we use the assumption that $F(\mathbf{x}_0) \geq 0$. Therefore, by regrouping the terms we obtain that

$$\begin{aligned} \mathbb{E}[F(\mathbf{x}_T)] &\geq \frac{1}{c} \left(1 - \left(1 - \frac{c}{T}\right)^T\right) F^* - \frac{15D\sqrt{Q}}{T^{1/3}} - \frac{m_f r D^2}{2T} \\ &\geq \frac{1}{c} (1 - e^{-c}) F^* - \frac{15D\sqrt{Q}}{T^{1/3}} - \frac{m_f r D^2}{2T}, \end{aligned} \quad (141)$$

and the claim in (42) follows.

References

- Francis Bach. Submodular functions: from discrete to continuous domains. *arXiv preprint arXiv:1511.00394*, 2015.
- Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-dynamic programming*, volume 3 of *Optimization and neural computation series*. Athena Scientific, 1996.
- Andrew An Bian, Kfir Yehuda Levy, Andreas Krause, and Joachim M. Buhmann. Non-monotone continuous dr-submodular maximization: Structure and algorithms. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 486–496, 2017a.
- Andrew An Bian, Baharan Mirzasoleiman, Joachim M. Buhmann, and Andreas Krause. Guaranteed non-convex optimization: Submodular maximization over continuous domains. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, pages 111–120, 2017b.
- Jeffrey Bilmes and Wenruo Bai. Deep submodular functions. *arXiv preprint arXiv:1701.08939*, 2017.
- Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT’2010*, pages 177–186. Springer, 2010.
- Niv Buchbinder and Moran Feldman. Constrained submodular maximization via a non-symmetric technique. *arXiv preprint arXiv:1611.03253*, 2016.
- Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. Submodular maximization with cardinality constraints. In *SODA*, pages 1433–1452, 2014.
- Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. A tight linear time $(1/2)$ -approximation for unconstrained submodular maximization. *SIAM Journal on Computing*, 44(5):1384–1402, 2015.
- Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.

- C. Chekuri, T. S. Jayram, and J. Vondrak. On multiplicative weight updates for concave and submodular function maximization. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 201–210. ACM, 2015.
- Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. *SIAM Journal on Computing*, 43(6):1831–1879, 2014.
- Michael Collins, Amir Globerson, Terry Koo, Xavier Carreras, and Peter L Bartlett. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *Journal of Machine Learning Research*, 9(Aug):1775–1822, 2008.
- Nikhil R. Devanur and Kamal Jain. Online matching with concave returns. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 137–144, 2012.
- Reza Eghbali and Maryam Fazel. Designing smoothing functions for improved worst-case competitive ratio in online optimization. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3279–3287, 2016.
- Alina Ene and Huy L. Nguyen. Constrained submodular maximization: Beyond $1/e$. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 248–257, 2016.
- Uriel Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.
- Uriel Feige, Vahab S Mirrokni, and Jan Vondrak. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011.
- Moran Feldman, Joseph Naor, and Roy Schwartz. A unified continuous greedy algorithm for submodular maximization. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 570–579, 2011.
- Moran Feldman, Christopher Harshaw, and Amin Karbasi. Greed is good: Near-optimal submodular maximization via greedy optimization. In *Proceedings of the 30th Conference on Learning Theory, COLT 2017, Amsterdam, The Netherlands, 7-10 July 2017*, pages 758–784, 2017.
- Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Research Logistics (NRL)*, 3(1-2):95–110, 1956.
- Satoru Fujishige. *Submodular functions and optimization*, volume 58. Elsevier, 2005.
- Satoru Fujishige and Shiguelo Isotani. A submodular function minimization algorithm based on the minimum-norm base. *Pacific Journal of Optimization*, 7(1):3–17, 2011.

- Shayan Oveis Gharan and Jan Vondrák. Submodular maximization by simulated annealing. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1098–1116. SIAM, 2011.
- Daniel Golovin, Andreas Krause, and Matthew Streeter. Online submodular maximization under a matroid constraint with application to learning assignments. *arXiv preprint arXiv:1407.1082*, 2014.
- Hamed Hassani, Mahdi Soltanolkotabi, and Amin Karbasi. Gradient methods for submodular maximization. *arXiv preprint arXiv:1708.03949*, 2017.
- Simon S Haykin. *Adaptive filter theory*. Pearson Education India, 2008.
- Elad Hazan and Satyen Kale. Projection-free online learning. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, pages 1843–1850, 2012.
- Elad Hazan and Haipeng Luo. Variance-reduced and projection-free stochastic optimization. In *Proceedings of the 33rd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 1263–1271, 2016.
- Elad Hazan et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.
- Martin Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 427–435, 2013.
- Mohammad Karimi, Mario Lucic, Hamed Hassani, and Andreas Krause. Stochastic submodular maximization: The case of coverage functions. In *Advances in Neural Information Processing Systems*, 2017.
- László Lovász. Submodular functions and convexity. In *Mathematical Programming The State of the Art*, pages 235–257. Springer, 1983.
- Aranyak Mehta, Amin Saberi, Umesh V. Vazirani, and Vijay V. Vazirani. Adwords and generalized online matching. *Journal of the ACM*, 54(5), 2007.
- Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, and Amin Karbasi. Fast constrained submodular maximization: Personalized data summarization. In *Proceedings of the 33rd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 1358–1367, 2016.
- Aryan Mokhtari and Alejandro Ribeiro. Global convergence of online limited memory BFGS. *Journal of Machine Learning Research*, 16(1):3151–3181, 2015.
- Aryan Mokhtari, Alec Koppel, Gesualdo Scutari, and Alejandro Ribeiro. Large-scale non-convex stochastic optimization by doubly stochastic successive convex approximation. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 4701–4705. IEEE, 2017.

- G. L. Nemhauser and L. A. Wolsey. Maximizing submodular set functions: formulations and analysis of algorithms. *Studies on Graphs and Discrete Programming, volume 11 of Annals of Discrete Mathematics*, 1981.
- George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming*, 14(1): 265–294, 1978.
- Arkadi Nemirovski and D Yudin. On Cezari’s convergence of the steepest descent method for approximating saddle point of convex-concave functions. In *Soviet Math. Dokl*, volume 19, 1978.
- Arkadii Nemirovskii, David Borisovich Yudin, and Edgar Ronald Dawson. Problem complexity and method efficiency in optimization. 1983.
- S. J. Reddi, S. Sra, B. Póczos, and A. Smola. Stochastic frank-wolfe methods for nonconvex optimization. *arXiv preprint arXiv:1607.08254*, 2016.
- Alejandro Ribeiro. Ergodic stochastic optimization algorithms for wireless communication and networking. *IEEE Transactions on Signal Processing*, 58(12):6369–6386, 2010.
- Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.
- Andrzej Ruszczyński. Feasible direction methods for stochastic programming problems. *Mathematical Programming*, 19(1):220–229, 1980.
- Andrzej Ruszczyński. A merit function approach to the subgradient method with averaging. *Optimisation Methods and Software*, 23(1):161–172, 2008.
- Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczyński. *Lectures on stochastic programming: modeling and theory*. SIAM, 2009.
- Victor Solo and Xuan Kong. *Adaptive signal processing algorithms: stability and performance*. Prentice-Hall, Inc., 1994.
- Tasuku Soma, Naonori Kakimura, Kazuhiro Inaba, and Ken-ichi Kawarabayashi. Optimal budget allocation: Theoretical guarantee and efficient algorithm. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 351–359, 2014.
- Matthew Staib and Stefanie Jegelka. Robust budget allocation via continuous submodular functions. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 3230–3240, 2017.
- Serban Stan, Morteza Zadimoghaddam, Andreas Krause, and Amin Karbasi. Probabilistic submodular maximization in sub-linear time. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 3241–3250, 2017.

- Maxim Sviridenko, Jan Vondrák, and Justin Ward. Optimal approximation for submodular and supermodular optimization with bounded curvature. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1134–1148, 2015.
- Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.
- Jan Vondrák. Submodularity in combinatorial optimization. 2007.
- Jan Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 67–74, 2008.
- Jan Vondrák. Submodularity and curvature : The optimal algorithm (combinatorial optimization and discrete algorithms). *RIMS Kokyuroku Bessatsu*, B23:253–266, 2010.
- Laurence A Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982.
- Yang Yang, Gesualdo Scutari, Daniel P. Palomar, and Marius Pesavento. A parallel decomposition method for nonconvex stochastic multi-agent optimization problems. *IEEE Transactions on Signal Processing*, 64(11):2949–2964, 2016.

Quantized Frank-Wolfe: Faster Optimization, Lower Communication, and Projection Free

Mingrui Zhang¹ Lin Chen¹ Aryan Mokhtari² Hamed Hassani³ Amin Karbasi¹
¹Yale University ²University of Texas at Austin ³University of Pennsylvania

Abstract

How can we efficiently mitigate the overhead of gradient communications in distributed optimization? This problem is at the heart of training scalable machine learning models and has been mainly studied in the unconstrained setting. In this paper, we propose **Quantized Frank-Wolfe** (QFW), the first projection-free and communication-efficient algorithm for solving *constrained* optimization problems at scale. We consider both convex and non-convex objective functions, expressed as a finite-sum or more generally a stochastic optimization problem, and provide strong theoretical guarantees on the convergence rate of QFW. This is accomplished by proposing novel quantization schemes that efficiently compress gradients while controlling the noise variance introduced during this process. Finally, we empirically validate the efficiency of QFW in terms of communication and the quality of returned solution against natural baselines.

1 Introduction

The Frank-Wolfe (FW) method [Frank and Wolfe, 1956], also known as conditional gradient, has recently received considerable attention in the machine learning community, as a projection free algorithm for various *constrained* convex [Jaggi, 2013, Garber and Hazan, 2014, Lacoste-Julien and Jaggi, 2015, Garber and Hazan, 2015, Hazan and Luo, 2016, Mokhtari et al., 2018a] and non-convex [Lacoste-Julien, 2016, Reddi et al., 2016, Mokhtari et al., 2018b, Zhang et al., 2019b, Hassani et al., 2019] optimization problems. In order to apply the FW method to large-scale prob-

lems (*e.g.*, training deep neural networks [Ravi et al., 2018, Schramowski et al., 2018, Berrada et al., 2018], RBMs [Ping et al., 2016]) parallelization is unavoidable. To this end, distributed FW variants have been proposed for specific problems, *e.g.*, online learning [Zhang et al., 2017], learning low-rank matrices [Zheng et al., 2018], and optimization under block-separable constraint sets [Wang et al., 2016]. A significant performance bottleneck of distributed optimization methods is the cost of communicating gradients, typically handled by using a parameter-server framework. Intuitively, if each worker in the distributed system transmits the entire gradient, then at least d floating-point numbers are communicated for each worker, where d is the dimension of the problem. This communication cost can be a huge burden on the performance of parallel optimization algorithms [Chilimbi et al., 2014, Seide et al., 2014, Strom, 2015]. To circumvent this drawback, communication-efficient parallel algorithms have received significant attention. One major approach is to quantize the gradients while maintaining sufficient information [De Sa et al., 2015, Abadi et al., 2016, Wen et al., 2017]. For *unconstrained* optimization, when projection is not required for implementing Stochastic Gradient Descent (SGD), several communication-efficient distributed methods have been proposed, including QSGD [Alistarh et al., 2017], SIGN-SGD [Bernstein et al., 2018], and Sparsified-SGD [Stich et al., 2018].

In the constrained setting, and in particular for distributed FW methods, the communication-efficient versions were only studied for specific problems such as sparse learning [Bellet et al., 2015, Lafond et al., 2016]. In this paper, however, we develop **Quantized Frank-Wolfe** (QFW), a general communication-efficient distributed FW for both convex and non-convex objective functions. We study the performance of QFW in two widely recognized settings: 1) stochastic optimization and 2) finite-sum optimization.

To be more specific, let $\mathcal{K} \subseteq \mathbb{R}^d$ be the constraint set. In *constrained stochastic optimization* the goal is to

Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108. Copyright 2020 by the author(s).

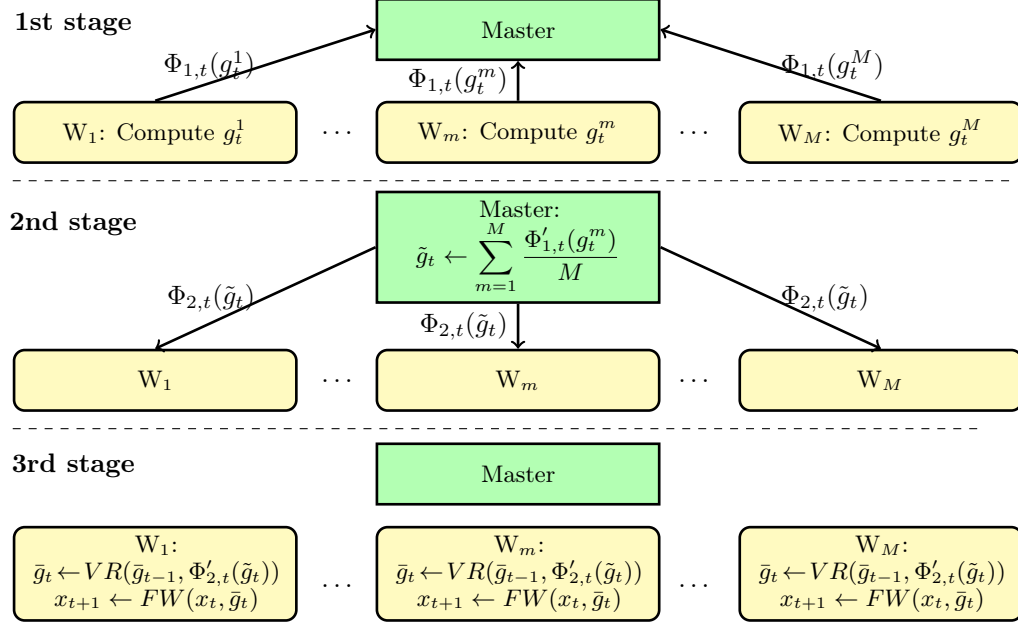


Figure 1: Stages of our general Quantized Frank-Wolfe scheme at time t . In the first stage, each worker m computes its local gradient information g_t^m and sends the quantized version $\Phi_{1,t}(g_t^m)$ to the master node. In the second stage, master computes the average of decoded received signals $\Phi_{1,t}(g_t^m)$, *i.e.*, $\tilde{g}_t \leftarrow (1/M) \sum_{m=1}^M \Phi'_{1,t}(g_t^m)$ and then sends its quantized version $\Phi_{2,t}(\tilde{g}_t)$ to the workers. Note that the two quantization schemes $\Phi_{1,t}, \Phi_{2,t}$ depend on t and can be different from each other. In the third stage, workers use the decoded gradient information computed by all workers $\Phi_{2,t}(\tilde{g}_t)$ and their previous gradient estimation $\tilde{g}_{t-1}$ to update their new gradient estimation $\tilde{g}_t$ via a variance reduction (VR) scheme. Once the variance reduced gradient approximation $\tilde{g}_t$ is evaluated, workers compute the new variable x_{t+1} by following the update of Frank-Wolfe (FW).

solve

$$\min_{x \in \mathcal{K}} f(x) := \min_{x \in \mathcal{K}} \mathbb{E}_{z \sim P} [\tilde{f}(x, z)], \quad (1)$$

where $x \in \mathbb{R}^d$ is the optimization variable, $z \in \mathbb{R}^q$ is a random variable drawn from a probability distribution P , which determines the choice of a stochastic function $\tilde{f} : \mathbb{R}^d \times \mathbb{R}^q \rightarrow \mathbb{R}$. For *constrained finite-sum optimization*, we further assume that P is a uniform distribution over $[N] = \{1, 2, \dots, N\}$ and the goal is to solve a special case of Problem (1), namely,

$$\min_{x \in \mathcal{K}} f(x) := \min_{x \in \mathcal{K}} \frac{1}{N} \sum_{j=1}^N f_j(x). \quad (2)$$

In parallel settings, we suppose that we have a computing system consisting of a master node and M workers, and each worker maintains a local copy of x . At every iteration of the stochastic case, each worker has access to independent stochastic gradients of f ; whereas in the finite-sum case, we assume $N = Mn$, thus the objective function can be decomposed as $f(x) = \frac{1}{Mn} \sum_{m \in [M], j \in [n]} f_{m,j}(x)$, and each worker m has access to the exact gradients of n component functions $f_{m,j}(x)$ for all $j \in [n]$.

This way the task of computing gradients is divided among the workers. The master node aggregates local gradients from the workers, and sends the aggregated gradients back to them so that each worker can update the model (*i.e.*, their own iterate) locally. Thus, by transmitting quantized gradients, we can reduce the communication complexity (*i.e.*, number of transmitted bits) significantly. The workflow diagram of the proposed Quantized Frank-Wolfe scheme is summarized in Figure 1. We should highlight that there is a trade-off between gradient quantization and information flow. Intuitively, more intensive quantization reduces the communication cost, but also loses more information, which may decelerate the convergence rate.

Our contributions: In this paper, we propose a novel distributed projection-free framework that handles quantization for constrained convex and non-convex optimization problems in finite-sum and stochastic cases. It is well-known that unlike projected gradient-based methods, FW methods may diverge when fed with stochastic gradient [Hazan and Luo, 2016, Mokhtari et al., 2018a]. Indeed, a similar issue arises in a distributed setting where nodes exchange *quantized gradients* which are noisy estimates of the gradients. By in-

Table 1: SFO/IFO Complexity per worker in different settings (M is the number of workers).

Setting	Function	SFO/IFO Complexity
Finite-sum	Convex	$\mathcal{O}\left(\frac{N \ln(1/\epsilon) + 1/\epsilon^2}{M}\right)$
Finite-sum	Non-convex	$\mathcal{O}\left(\frac{\sqrt{N}}{\epsilon^2 \sqrt{M}}\right)$
Stochastic	Convex	$\mathcal{O}\left(\frac{1}{M \epsilon^2}\right)$
Stochastic	Non-convex	$\mathcal{O}\left(\frac{1}{\epsilon^3 \sqrt{M}}\right)$

incorporating appropriate variance reduction techniques, we show that with quantized gradients, we can obtain a provably convergent method which preserves the convergence rates of the state-of-the-art vanilla centralized methods in all the considered cases [Zhang et al., 2019b, Shen et al., 2019b, Hassani et al., 2019, Yurtsever et al., 2019]. We believe our work presents the first quantized, distributed, and projection-free method. Our theoretical results for **Quantized Frank-Wolfe** (QFW) are summarized in Table 1, where the SFO complexity is the required number of stochastic gradients in stochastic case, and the IFO complexity is the number of exact gradients for component functions in finite-sum case. For the convex case, the complexity indicates the number of gradients to achieve an ϵ -suboptimal solution; while in the non-convex case, it refers to the number of gradients to find a first-order ϵ -stationary point. We note that since the M workers compute the gradients simultaneously, the time to obtain gradients is proportional to the SFO/IFO complexity *per worker*. So we report the SFO/IFO complexity per worker, as in many other works on parallel optimization (e.g., Sign-SGD [Bernstein et al., 2018]). The results in Table 1 show that more workers can decrease the SFO/IFO complexity per worker effectively, and thus accelerate the optimization procedure. **All the proofs in this paper are provided in the appendix.**

2 Gradient Quantization Schemes

In most distributed optimization algorithms, the task of computing gradients is divided among the workers, and the master node uses parts of gradients at the workers to update the model (iterate) directly or sends the aggregated gradients to the worker so that each of them can update the model (iterate) locally. Therefore, the information that workers need to send to the master is the elements of the objective function gradient. Thus, by transmitting quan-

tized gradients, we can reduce the communication bits effectively. In this section, we introduce a quantization scheme called **s-Partition Encoding Scheme** and explain how this scheme reduces the overall cost of exchanging gradients. Consider the gradient vector $g \in \mathbb{R}^d$ and let g_i be the i -th coordinate of the gradient. The **s-Partition Encoding Scheme** encodes g_i into an element from the set $\{\pm 1, \pm \frac{s-1}{s}, \dots, \pm \frac{1}{s}, 0\}$ in a random way. To do so, we first compute the ratio $|g_i|/\|g\|_\infty$ and find the indicator $l_i \in \{0, 1, \dots, s-1\}$ such that $|g_i|/\|g\|_\infty \in [l_i/s, (l_i+1)/s]$. Then we define the random variable b_i as

$$b_i = \begin{cases} l_i/s, & \text{w.p. } 1 - \frac{|g_i|}{\|g\|_\infty} s + l_i, \\ (l_i+1)/s, & \text{w.p. } \frac{|g_i|}{\|g\|_\infty} s - l_i. \end{cases} \quad (3)$$

Finally, instead of transmitting g_i , we send $\text{sgn}(g_i) \cdot b_i$, alongside the norm $\|g\|_\infty$. It can be verified that $\mathbb{E}[b_i|g] = |g_i|/\|g\|_\infty$. So we define the corresponding decoding scheme as $\phi'(g_i) = \text{sgn}(g_i)b_i\|g\|_\infty$ to ensure that $\phi'(g_i)$ is an unbiased estimator of g_i . We note that the encoding/decoding schemes in Figure 1 are denoted as capital Φ/Φ' , indicating that they can be any general schemes. The proposed **s-Partition Encoding Scheme** is denoted by ϕ/ϕ' . We also note that this quantization scheme is similar to the Stochastic Quantization method in [Alistarh et al., 2017], except that we use ℓ_∞ -norm while they adopt the ℓ_2 -norm. In the **s-Partition Encoding Scheme**, for each coordinate i , we need 1 bit to transmit $\text{sgn}(g_i)$. Moreover, since $b_i \in \{0, 1/s, \dots, (s-1)/s, 1\}$, we need $z = \log_2(s+1)$ bits to send b_i . Finally, we need 32 bits to transmit $\|g\|_\infty$. Hence, the total number of communicated bits is $32 + d(z+1)$. Here, by “bits” we mean the number of 0’s and 1’s transmitted.

One major advantage of the **s-Partition Encoding Scheme** is that by tuning the partition parameter s or the corresponding assigned bits z , we can smoothly control the trade-off between gradient quantization and information loss, which helps distributed algorithms to attain their best performance. We proceed to characterize the variance of the **s-Partition Encoding Scheme**.

Lemma 1 *The variance of s-Partition Encoding Scheme ϕ for any $g \in \mathbb{R}^d$ is bounded by*

$$\text{Var}[\phi'(g)|g] \leq \frac{d}{s^2} \|g\|_\infty^2. \quad (4)$$

Lemma 1 demonstrates the trade-off between the error of quantization and the communication cost for **s-Partition Encoding Scheme**. In a nutshell, for larger choices of s , the variance is smaller, which in turn results in higher communication cost. If we set $s = 1$, we obtain the **Sign Encoding Scheme**, which requires communicating the encoded scalars $\text{sgn}(g_i)b_i \in \{\pm 1, 0\}$

and the norm $\|g\|_\infty$. Since $z = \log_2(s+1) = 1$, the overall communicated bits for each worker are $32 + 2d$ per round. We characterize its variance in Lemma 2.

Lemma 2 *The variance of Sign Encoding Scheme is given by*

$$\text{Var}[\phi'(g)|g] = \|g\|_1 \|g\|_\infty - \|g\|_2^2. \quad (5)$$

Remark 1 *For the probability distribution of the random variable b_i , instead of $\|g\|_\infty$, we can use other norms $\|g\|_p$ (where $p \geq 1$). But it can be verified that the ℓ_∞ -norm leads to the smallest variance for Sign Encoding Scheme. That is also the reason why we do not use ℓ_2 -norm as in [Alistarh et al., 2017].*

3 Convex Minimization

In this section, we analyze the convex minimization problem in both finite-sum and stochastic settings. Note that even in the setting without quantization, if we use stochastic gradients in the update of FW, it might diverge [Hazan and Luo, 2016, Mokhtari et al., 2018a]. So appropriate variance reduction techniques are needed for communicating quantized gradients. Nguyen et al. [2017a,b, 2019] developed the Stochastic Recursive Gradient algorithm (SARAH), a stochastic recursive gradient update framework. Fang et al. [2018] proposed Stochastic Path-Integrated Differential Estimator (SPIDER) technique, a variant of SARAH, for centralized unconstrained optimization. Recently, Hassani et al. [2019], Shen et al. [2019a], Yurtsever et al. [2019] proposed the SPIDER variants of FW method for both convex and non-convex optimization problems. Similar variance reduction idea was also combined with SGD to solve non-convex finite-sum problems in [Zhou et al., 2018]. In this paper, we generalize SPIDER to the constrained and distributed settings.

We first consider the case where no quantization is performed. Let $\{p_i\} \in \mathbb{N}^+$ be a sequence of period parameters. At the beginning of each period i , namely, $t = \sum_{j=1}^{i-1} p_j + 1$, each worker m samples $S_{i,1}$ component functions in finite-sum case, or stochastic functions in stochastic case, which are denoted as $\mathcal{S}_{i,1}^m$. We define the local average gradient on set $\mathcal{S}_{i,1}^m$ as $g_{i,1}^m \triangleq \nabla f_{\mathcal{S}_{i,1}^m}(x_t) = \frac{1}{S_{i,1}} \sum_{j \in \mathcal{S}_{i,1}^m} \nabla f_j(x_t)$. Then each worker m computes the average of all these local gradients $g_{i,1}^m$ and sends it to the master. Then, master node calculates the average of the M received signals and broadcasts it to all workers. Then, the workers update their gradient estimation $\bar{g}_t$ as the averaged signal $\bar{g}_t = \frac{1}{M} \sum_{m=1}^M g_{i,1}^m$.

Note $\bar{g}_t$ is identical for all the workers. In the rest of that period, i.e., $t = \sum_{j=1}^{i-1} p_j + k$, where $2 \leq k \leq p_i$, each worker m samples a set of local functions, denoted

Algorithm 1 Quantized Frank-Wolfe (QFW)

- 1: **Input:** constraint set $\mathcal{K}$, total iteration number T , No. of workers M , period parameters $\{p_i\}$, sample sizes $\{S_{i,k}\}$, learning rate η_t , initial point $x_1 \in \mathcal{K}$
 - 2: **Output:** x_{T+1} or x_o , where x_o is chosen from $\{x_1, x_2, \dots, x_T\}$ uniformly at random
 - 3: **for** $t = 1$ **to** T **do**
 - 4: Set $x_{i,k} \leftarrow x_t$, where $t = \sum_{j=1}^{i-1} p_j + k$, $1 \leq k \leq p_i$
 - 5: Each worker m computes local gradient $g_{i,k}^m$ by $g_{i,1}^m = \nabla f_{\mathcal{S}_{i,1}^m}(x_{i,k}) = \nabla f_{\mathcal{S}_{i,1}^m}(x_t)$ for $k = 1$, or $g_{i,k}^m \triangleq \nabla f_{\mathcal{S}_{i,k}^m}(x_{i,k}) - \nabla f_{\mathcal{S}_{i,k}^m}(x_{i,k-1}) = \nabla f_{\mathcal{S}_{i,k}^m}(x_t) - \nabla f_{\mathcal{S}_{i,k}^m}(x_{t-1})$ for $k \geq 2$
 - 6: Each worker m encodes $g_{i,k}^m$ as $\Phi_{1,i,k}(g_{i,k}^m)$ and pushes it to the master
 - 7: Master decodes $\Phi_{1,i,k}(g_{i,k}^m)$ as $\Phi'_{1,i,k}(g_{i,k}^m)$, and computes $\tilde{g}_{i,k} \leftarrow \frac{1}{M} \sum_{m=1}^M \Phi'_{1,i,k}(g_{i,k}^m)$
 - 8: Master encodes $\tilde{g}_{i,k}$ as $\Phi_{2,i,k}(\tilde{g}_{i,k})$, and broadcasts it to all workers
 - 9: Workers decode $\Phi_{2,i,k}(\tilde{g}_{i,k})$ as $\Phi'_{2,i,k}(\tilde{g}_{i,k})$
 - 10: **if** $k = 1$ **then**
 - 11: Workers update $\bar{g}_{i,k} \leftarrow \Phi'_{2,i,k}(\tilde{g}_{i,k})$
 - 12: **else**
 - 13: Workers update $\bar{g}_{i,k} \leftarrow \Phi'_{2,i,k}(\tilde{g}_{i,k}) + \bar{g}_{i,k-1}$
 - 14: **end if**
 - 15: Each worker updates $x_{t+1} \leftarrow x_t + \eta_t(v_t - x_t) = x_{i,k} + \eta_{i,k}(v_{i,k} - x_{i,k})$ where $v_{i,k} \leftarrow \arg\min_{v \in \mathcal{K}} \langle v, \bar{g}_{i,k} \rangle$
 - 16: **end for**
-

as $\mathcal{S}_{i,k}^m$, of size $S_{i,k}$ uniformly at random, and computes the difference of averages of these gradients

$$g_{i,k}^m \triangleq \nabla f_{\mathcal{S}_{i,k}^m}(x_t) - \nabla f_{\mathcal{S}_{i,k}^m}(x_{t-1}), \quad (6)$$

and sends it to master. Then master node calculates the average of the M signals and broadcasts it to all the workers. The workers update their gradient estimation $\bar{g}_t$ as

$$\bar{g}_t = \bar{g}_{t-1} + \frac{1}{M} \sum_{m=1}^M g_{i,k}^m. \quad (7)$$

So $\bar{g}_t$ is still identical for all the workers. In order to incorporate quantization, each worker simply pushes the quantized version of the average gradients. Then the master decodes the quantizations, encodes the average of decoded signals in a quantized fashion, and broadcasts the quantization. Finally, each worker decodes the quantized signal and updates x_t locally. To be more specific, in the quantized setting, in each iteration t such that $t = \sum_{j=1}^{i-1} p_j + k$ where $1 \leq k \leq p_i$, each worker m sends the quantized version of its local gradient information $\Phi_{1,t}(g_{i,k}^m)$ to the master. Once master collects all the quantized information, it decodes them, i.e., finds $\{\Phi'_{1,t}(g_{i,k}^m)\}_{m=1}^M$, computes their average $\tilde{g}_t$,

and sends its quantized version $\Phi_{2,t}(\tilde{g}_t)$ to all workers. Then, all the workers decode the received quantized signal and use it as their new gradient approximation $\bar{g}_t$ and update their variable according to the update of Frank-Wolfe, i.e.,

$$x_{t+1} = x_t + \eta_t(v_t - x_t), \quad (8)$$

where $v_t \leftarrow \operatorname{argmin}_{v \in \mathcal{K}} \langle v, \bar{g}_t \rangle$. The description of our proposed **Quantized Frank-Wolfe** (QFW) method is shown in Figure 1 and outlined in Algorithm 1.

Remark 2 *The model update (Line 15 in Algorithm 1) should be performed at each worker. Since all the linear programming problems (to obtain v_t) are solved simultaneously, the total running time is the same with that where the model updating is performed in the master node. However, additional variance would be introduced if the master node updates the model and broadcasts it in quantized manners. Thus the master-updating method lacks theoretical justification regarding convergence, and we adopt the worker-updating approach.*

3.1 Finite-Sum Setting

Now we proceed to establish the convergence properties of our proposed QFW in the finite-sum setting. Recall that we assume that there are N functions and M workers in total, and each worker m has access to $n = N/M$ functions $f_{m,j}$ for $j \in [n]$. We first make two assumptions on the constraint set and component functions. Let $\|\cdot\|$ denote the l_2 norm in the Euclidean space through out the paper.

Assumption 1 *The constraint set $\mathcal{K}$ is convex and compact, with diameter $D = \sup_{x,y \in \mathcal{K}} \|x - y\|$.*

Assumption 2 *The functions $f_{m,i}$ are convex, L -smooth on $\mathcal{K}$, and satisfy that $\|\nabla f_{m,i}(x)\|_\infty \leq G_\infty$, for all $m \in [M], i \in [n], x, y \in \mathcal{K}$.*

Theorem 1 (Finite-Sum Convex) *Consider QFW outlined in Algorithm 1. Under Assumptions 1 and 2, if we set $p_i = 2^{i-1}, S_{i,1}^m = \{f_{m,j} : j \in [n]\}$ (i.e., each worker m samples all its n component functions), $S_{i,k} = p_i/M = 2^{i-1}/M$, for all $i \geq 1, k \geq 2$, and $\eta_{i,k} = 2/(p_i + k) = 2/(2^{i-1} + k)$, and use the $s_{1,i,1} = (\sqrt{\frac{dp_i^2}{M}})$ -Partition Encoding Scheme, $s_{2,i,1} = (\sqrt{dp_i^2})$ -Partition Encoding Scheme for $k = 1$, and $s_{1,i,k} = (\sqrt{\frac{dp_i}{M}})$ -Partition Encoding Scheme, $s_{2,i,k} = (\sqrt{dp_i})$ -Partition Encoding Scheme for $k \geq 2$ as $\Phi_{1,i,k}$ and $\Phi_{2,i,k}$ in Algorithm 1, then the output $x_{T+1} \in \mathcal{K}$ satisfies*

$$\mathbb{E}[f(x_{T+1})] - f(x^*) \leq \frac{4D\sqrt{2(G_\infty^2 + 6L^2D^2)} + 2LD^2}{T},$$

where x^* is a minimizer of f on $\mathcal{K}$.

Corollary 1 *To obtain an ϵ -suboptimal solution, we need to run the QFW method for at most $\mathcal{O}(1/\epsilon)$ iterations. The IFO complexity per worker in this case is $\mathcal{O}(\frac{N \ln(1/\epsilon) + 1/\epsilon^2}{M})$.*

Corollary 1 shows IFO complexity per worker is linear in $1/M$, which implicates that increasing the number of workers M will decrease the IFO complexity per worker effectively, thus accelerate the optimization procedure. Also, our numerical experiments in Section 5 showed that our proposed method requires significantly fewer bits than the unquantized version to achieve a specific accuracy.

3.2 Stochastic Setting

QFW can also be applied to the stochastic case. Recall that in the stochastic setting we assume that the objective function is $f(x) = \mathbb{E}_{z \sim P}[\tilde{f}(x, z)]$ and each worker has access to independent samples $\tilde{f}(x, z)$. Before proving the convergence properties of QFW for the stochastic setting, we first make a standard assumption on $\tilde{f}(x, z)$.

Assumption 3 *The stochastic function $\tilde{f}(x, z)$ is convex, L -smooth on $\mathcal{K}$. The gradient $\nabla \tilde{f}(x, z)$ is an unbiased estimate of $\nabla f(x)$ with bounded variance σ^2 , and satisfies that $\|\nabla \tilde{f}(x, z)\|_\infty \leq G_\infty$, for all $x \in \mathcal{K}, z \in \mathbb{R}^q$.*

Theorem 2 (Stochastic Convex) *Consider QFW outlined in Algorithm 1. Under Assumptions 1 and 3, if we set $p_i = 2^{i-1}, S_{i,1} = \frac{\sigma^2 p_i^2}{ML^2 D^2}$, $S_{i,k} = p_i/M = 2^{i-1}/M$, for all $i \geq 1, k \geq 2$, and $\eta_{i,k} = 2/(p_i + k) = 2/(2^{i-1} + k)$, and use the $s_{1,i,1} = (\sqrt{\frac{dp_i^2}{M}})$ -Partition Encoding Scheme, $s_{2,i,1} = (\sqrt{dp_i^2})$ -Partition Encoding Scheme for $k = 1$, and $s_{1,i,k} = (\sqrt{\frac{dp_i}{M}})$ -Partition Encoding Scheme, $s_{2,i,k} = (\sqrt{dp_i})$ -Partition Encoding Scheme for $k \geq 2$ as $\Phi_{1,i,k}$ and $\Phi_{2,i,k}$ in Algorithm 1, then the output $x_{T+1} \in \mathcal{K}$ satisfies*

$$\mathbb{E}[f(x_{T+1})] - f(x^*) \leq \frac{4D\sqrt{13L^2D^2 + 2G_\infty^2} + 2LD^2}{T},$$

where x^* is a minimizer of f on $\mathcal{K}$.

Corollary 2 *To obtain an ϵ -suboptimal solution, we need to run the QFW method outlined in Algorithm 1 for at most $\mathcal{O}(1/\epsilon)$ iterations. The SFO complexity per worker in this case is $\mathcal{O}(1/(M\epsilon^2))$.*

Corollary 2 shows SFO complexity per worker is linear in $1/M$, which implies a speed-up for distributed

settings. It also shows that the dependency of QFW's complexity on ϵ for convex settings is optimal.

Remark 3 *In theory, the partitioning levels of quantization do depend on the number of iterations. Thus more transmission bits are required over the optimization procedure. But it will not render our QFW method communication-expensive. We set the partitioning levels conservatively to achieve the theoretical guarantees. However, as shown in the experiments (Section 5), much smaller quantization levels (which are actually constants) are usually preferred in practice.*

4 Non-Convex Optimization

With slightly different parameters, QFW can be applied to non-convex settings as well. In *unconstrained* non-convex optimization problems, the gradient norm $\|\nabla f\|$ is usually a good measure of convergence as $\|\nabla f\| \rightarrow 0$ implies convergence to a stationary point. However, in the constrained setting we study the Frank-Wolfe Gap [Jaggi, 2013, Lacoste-Julien, 2016] defined as

$$\mathcal{G}(x) = \max_{v \in \mathcal{K}} \langle v - x, -\nabla f(x) \rangle. \quad (9)$$

For constrained optimization problem, if a point x satisfies $\mathcal{G}(x) = 0$, then it is a first-order stationary point. Also, by definition, we have $\mathcal{G}(x) \geq 0$, for all $x \in \mathcal{K}$. We first analyze the finite-sum setting and then the more general stochastic setting.

4.1 Finite-Sum Setting

To extend our results to the nonconvex setting we first assume that the following condition is satisfied.

Assumption 4 *The component functions $f_{m,i}$ are L -smooth on $\mathcal{K}$ and uniformly bounded, i.e., $\sup_{x \in \mathcal{K}} |f_{m,i}(x)| \leq M_0$. Further, $\|\nabla f_{m,i}(x)\|_\infty \leq G_\infty$, for all $m \in [M], i \in [n], x, y \in \mathcal{K}$.*

Theorem 3 (Finite-Sum Non-Convex) *Under Assumptions 1 and 4, if we set $p_i = \sqrt{n}$, $\mathcal{S}_{i,1}^m = \{f_{m,j} : j \in [n]\}$ (i.e., each worker m samples all its n component functions), $S_{i,k} = \sqrt{n}/M$ for all $i \geq 1, k \geq 2, \eta_t = T^{-1/2}$ for all t , and use the $s_{1,i,1} = (\sqrt{\frac{Td}{M}})$ -Partition Encoding Scheme, $s_{2,i,1} = (\sqrt{Td})$ -Partition Encoding Scheme for $k = 1$, and $s_{1,i,k} = (\frac{d^{1/2}n^{1/4}}{\sqrt{M}})$ -Partition Encoding Scheme, $s_{2,i,k} = (d^{1/2}n^{1/4})$ -Partition Encoding Scheme for $k \geq 2$ as $\Phi_{1,i,k}$ and $\Phi_{2,i,k}$ in Algorithm 1, then the output $x_o \in \mathcal{K}$ satisfies*

$$\mathbb{E}[\mathcal{G}(x_o)] \leq \frac{2M_0 + D\sqrt{3L^2D^2 + 2G_\infty^2} + \frac{LD^2}{2}}{\sqrt{T}}.$$

Algorithm 2 Stochastic Non-Convex Quantized Frank-Wolfe (SNC-QFW)

- 1: **Input:** constraint set $\mathcal{K}$, iteration number T , No. of workers M , initial point $x_1 \in \mathcal{K}$
 - 2: Obtain T independent samples of z_i , and define finite-sum $\hat{f}(x) = \frac{1}{T} \sum_{i=1}^T \tilde{f}(x, z_i)$
 - 3: Apply Algorithm 1 on $\hat{f}$ with $N = T$ and all other parameters being the same as in Theorem 3
 - 4: **Output:** x_o , where x_o is chosen from $\{x_1, x_2, \dots, x_T\}$ uniformly at random
-

Corollary 3 *Algorithm 1 finds an ϵ -first order stationary point after at most $\mathcal{O}(1/\epsilon^2)$ iterations. The IFO complexity per worker is $\mathcal{O}(\sqrt{N}/(\epsilon^2\sqrt{M}))$.*

Corollary 3 shows IFO complexity per worker is linear in $1/\sqrt{M}$, implicating that increasing the number of workers M will decrease the IFO complexity per worker effectively, thus accelerate the optimization procedure.

4.2 Stochastic Setting

For non-convex objective function, the stochastic optimization problem in (1) can be solved approximately by the QFW method described in Algorithm 1. Specifically, the objective function $f(x) = \mathbb{E}_{z \sim P}[\tilde{f}(x, z)]$ can be approximated by a finite-sum problems with B samples where the samples $\{z_1, \dots, z_B\}$ are independently drawn according to the probability distribution P . Thus we define the surrogate function $\hat{f}$

$$\hat{f}(x) = \frac{1}{B} \sum_{i=1}^B \tilde{f}(x, z_i), \quad (10)$$

as the finite-sum approximation of $f(x)$. As a result, we can apply QFW on $\hat{f}$, thus optimize f approximately. The algorithm is outlined in Algorithm 2

In the non-convex setting, if we further assume that $\tilde{f}(x, z)$ is G -Lipschitz for all $z \in \mathbb{R}^q$, then we have the following lemma:

Lemma 3 (Theorem 5 of [Reddi et al., 2016])

If we define $\hat{\mathcal{G}}(x) = \max_{v \in \mathcal{K}} \langle v - x, -\nabla \hat{f}(x) \rangle$, then $\mathbb{E}[\mathcal{G}(x) - \hat{\mathcal{G}}(x)] \leq \frac{GD}{\sqrt{B}}$. Recall that D is the diameter of $\mathcal{K}$ as defined in Assumption 1, $\mathcal{G}(x) = \max_{v \in \mathcal{K}} \langle v - x, -\nabla f(x) \rangle$. Thus for the output x_o , we have

$$\mathbb{E}[\mathcal{G}(x_o)] \leq \mathbb{E}[\hat{\mathcal{G}}(x_o)] + \frac{GD}{\sqrt{B}}. \quad (11)$$

Baesd on Theorem 3 and Lemma 3, we have the following theoretical guarantee for stochastic non-convex minimization.

Theorem 4 (Stochastic Non-Convex) *Assuming that for all $z \in \mathbb{R}^q$, $\hat{f}(x, z)$ is G -Lipschitz, L -smooth, and satisfies $|\hat{f}(x, z)| \leq M_0$ for all $x \in \mathcal{K}$. If we obtain T independent samples of z_i , and apply Algorithm 1 on $\hat{f}(x) = \frac{1}{T} \sum_{i=1}^T \hat{f}(x, z_i)$ with $N = T, n = T/M$, and all the other parameters set the same as in Theorem 3, then after T iterations, the output $x_o \in \mathcal{K}$ satisfies*

$$\mathbb{E}[\mathcal{G}(x_o)] \leq \frac{2M_0 + D\sqrt{3L^2D^2 + 2G^2} + \frac{LD^2}{2}}{\sqrt{T}} + \frac{GD}{\sqrt{T}}.$$

We note that the algorithm finds an ϵ -first order stationary point with at most $\mathcal{O}(1/\epsilon^2)$ rounds. The SFO complexity per worker is $\mathcal{O}(\sqrt{N}/(\sqrt{M}\epsilon^2)) = \mathcal{O}(\frac{1}{\epsilon^3\sqrt{M}})$.

Thus the SFO complexity per worker is linear in $1/\sqrt{M}$, which implicates that increasing the number of workers M will decrease the SFO complexity per worker.

5 Experiments

We evaluate the performance of algorithms by visualizing their loss $f(x_i)$ vs. the number of transmitted bits. The experiments were performed on 20 Intel Xeon E5-2660 cores and thus the number of workers is 20. For each curve in the figures below, we ran at least 50 repeated experiments, and the height of shaded regions represents two standard deviations.

In our first setup, we consider a multinomial logistic regression problem. Consider the dataset $\{(x_i, y_i)\}_{i=1}^N \subseteq \mathbb{R}^d \times \{1, \dots, C\}$ with N samples that have C different labels. We aim to find a model w to classify these sample points under the condition that the solution has a small ℓ_1 -norm. Therefore, we aim to solve the following convex problem

$$\begin{aligned} \min_w f(w) &:= - \sum_{i=1}^N \sum_{c=1}^C 1\{y_i = c\} \log \frac{\exp(w_c^\top x_i)}{\sum_{j=1}^C \exp(w_j^\top x_i)}, \\ \text{subject to } &\|w\|_1 \leq 1. \end{aligned} \quad (12)$$

In our experiments, we use the MNIST dataset and assume that each worker stores 3000 images. Therefore, the overall number of samples in the training set is $N = 60000$.

In our second setup, our goal is to minimize the loss of a three-layer neural network under some conditions on the norm of the solution. Before stating the problem precisely, let us define the log-loss function as $h(y, p) \triangleq -\sum_{c=1}^C 1\{y = c\} \log p_c$ for $y \in \{1, \dots, C\}$ and a C -dimensional probability vector $p := (p_1, \dots, p_C)$. We aim to solve the following non-convex problem

$$\begin{aligned} \min_{W_1, W_2} &\sum_{i=1}^N h(y_i, \phi(W_2 \text{relu}(W_1 x_i + b) + b_2)), \\ \text{subject to } &\|W_i\|_1 \leq a_1, \|b_i\|_1 \leq a_2, \end{aligned} \quad (13)$$

where $\text{relu}(x) \triangleq \max\{0, x\}$ is the ReLU function and ϕ is the softmax function. The imposed ℓ_1 constraint on the weights leads to a sparse network. We further remark that Frank-Wolfe methods are suitable for training a neural network subject to an ℓ_1 constraint as they are equivalent to a dropout regularization [Ravi et al., 2018]. In our setup, the size of matrices W_1 and W_2 are 784×50 and 50×10 , respectively, and the constraints parameters are $a_1 = a_2 = 5$.

For all of the considered settings, we vary the quantization level, use the s_1 -partition encoding scheme when workers send encoded tensors to the master and use the s_2 -partition encoding scheme when the master broadcasts encoded tensors to the workers ($s_i = uq$ indicates FW without quantization and $s_i = thm$ indicates QFW with the quantization level recommended by our theorems, where $i = 1, 2$). We also propose the federated learning approach FL, an effective heuristic based on QFW, where each worker performs its local Frank-Wolfe update autonomously without communicating with each other and synchronizes the model only at the end of each round. This method may not enjoy the strong theoretical guarantees of QFW and we observe in our experiments that it is even prone to divergence. In stochastic minimization, each worker samples 1000 images uniformly at random and without replacement.

In Figure 2, we observe the performance of FL, FW without quantization, and different variants of QFW for solving the multinomial logistic regression problem in (12). The stochastic minimization is presented in Figure 2a and the finite-sum minimization is shown in Figure 2b. We observe that QFW with **Partition Encoding Scheme** ($s_1 = 1, s_2 = 3$) has the best performance in terms of the amount of transmitted bits. Specifically, QFW with **Partition Encoding Scheme** ($s_1 = 1, s_2 = 3$) requires 3×10^8 bits to hit the lowest loss in Figures 2a and 2b, while FL with the same level of quantization only achieves a suboptimal loss (approximately 2.24) with the same amount of communication. Furthermore, FW without quantization requires more than 3.8×10^9 bits to reach the same error, *i.e.*, quantization reduces communication load by at least an order of magnitude.

Figure 3 demonstrates the performance of FL, FW without quantization, and different variants of QFW for solving the three-layer neural network in (13). Again we show the stochastic minimization on the left (Figure 3a) and the finite-sum minimization on the right (Figure 3b). We observe four divergent curves of the federated learning method FL ($s_1 = 3, s_2 = 1$; $s_1 = thm, s_2 = 1$; $s_1 = 1, s_2 = thm$; and $s_1 = 1, s_2 = uq$), while all QFW curves converge. This observation is in accordance with the fact that FL has no theoretical guarantee, in contrast to the proposed QFW method. FW

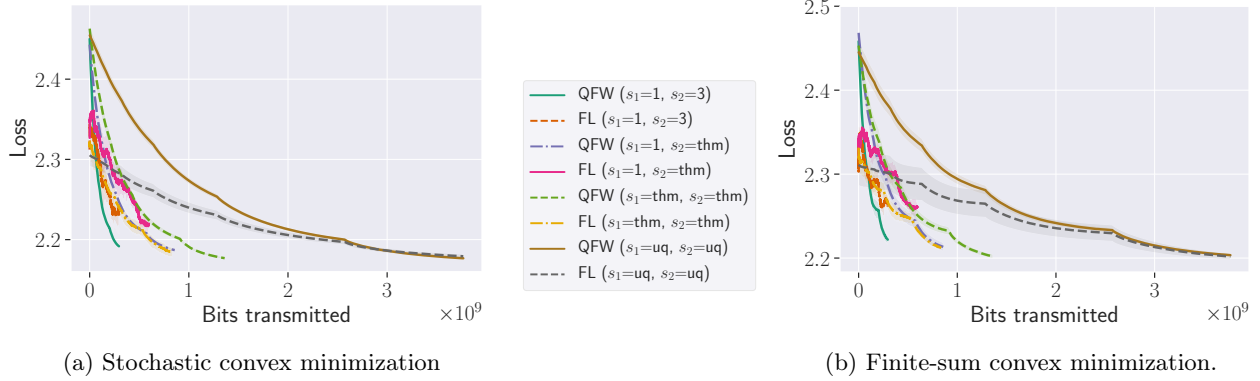


Figure 2: Comparison in terms of the loss versus the number of transmitted bits for a multinomial logistic regression problem. The best performance belongs to QFW with Partition Encoding Scheme ($s_1 = 1, s_2 = 3$), and FW without quantization has the worst performance.

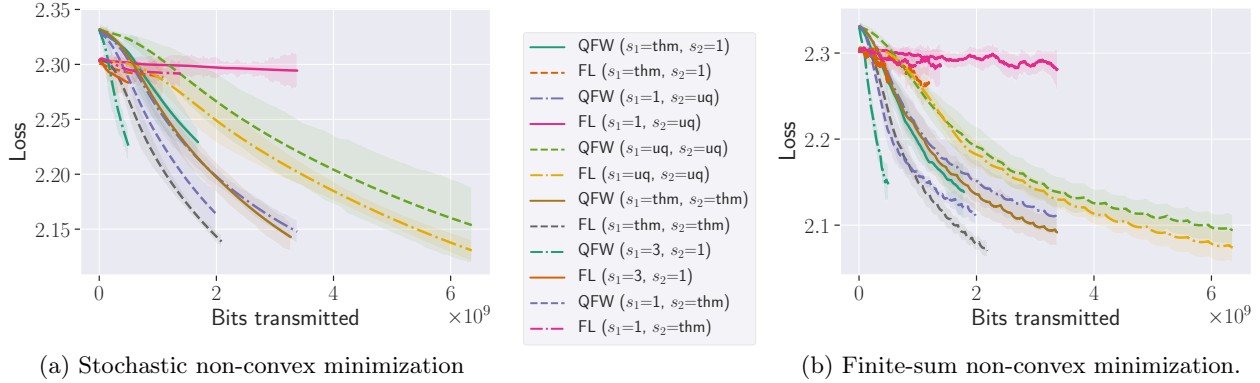


Figure 3: Comparison of algorithms in terms of the loss versus the number of transmitted bits for a three-layer neural network. FW without quantization ($s = uq$) significantly underperforms the quantized FW methods. We observe four divergent curves of the federated learning method FL ($s_1 = 3, s_2 = 1$; $s_1 = thm, s_2 = 1$; $s_1 = 1, s_2 = thm$; and $s_1 = 1, s_2 = uq$).

without quantization consumes approximately 6.5×10^9 bits to achieve the lowest loss. Its amount of communication is twice that of QFW with the quantization levels recommended by Theorems 3 and 4.

In both convex and non-convex setups, the theoretically guaranteed quantization levels recommended by our theorems may be conservative. In fact, a **Partition Encoding Scheme** with partitions much fewer than our theorems recommend achieves a similar loss level and saves even more communication bits. For example, QFW with $s_1 = 1, s_2 = 3$ and $s_1 = 1, s_2 = thm$ exhibits a higher communication efficiency than QFW with $s_1 = thm, s_2 = thm$ in Figures 2a and 2b.

We provide more experimental results in the longer version of this paper [Zhang et al., 2019a].

6 Conclusion

In this paper, we developed **Quantized Frank-Wolfe (QFW)**, the first general-purpose projection-free and communication-efficient framework for constrained optimization. Along with proposing various quantization schemes, QFW can address both convex and non-convex optimization settings in stochastic and finite-sum cases. We provided theoretical guarantees on the convergence rate of QFW and validated its efficiency empirically on training multinomial logistic regression and neural networks. Our theoretical results highlighted the importance of variance reduction techniques to stabilize FW and achieve a sweet trade-off between the communication complexity and convergence rate in distributed settings. We also note that it might be possible to design simpler Quantized FW methods based on the new developments [Zhang et al., 2019b].

Acknowledgements

Lin Chen is supported by Google PhD Fellowship. Hamed Hassani is supported by AFOSR Award 19RT0726, NSF HDR TRIPODS award 1934876, NSF award CPS-1837253, NSF award CIF-1910056, and NSF CAREER award CIF-1943064. Amin Karbasi is partially supported by NSF (IIS-1845032), ONR (N00014-19-1-2406), and AFOSR (FA9550-18-1-0160).

References

- Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.
- Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pages 1709–1720, 2017.
- Aurélien Bellet, Yingyu Liang, Alireza Bagheri Garakani, Maria-Florina Balcan, and Fei Sha. A distributed frank-wolfe algorithm for communication-efficient sparse learning. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 478–486. SIAM, 2015.
- Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Anima Anandkumar. signsgd: compressed optimisation for non-convex problems. *arXiv preprint arXiv:1802.04434*, 2018.
- Leonard Berrada, Andrew Zisserman, and M Pawan Kumar. Deep frank-wolfe for neural network optimization. *arXiv preprint arXiv:1811.07591*, 2018.
- Trishul M Chilimbi, Yutaka Suzue, Johnson Apacible, and Karthik Kalyanaraman. Project adam: Building an efficient and scalable deep learning training system. In *OSDI*, volume 14, pages 571–582, 2014.
- Christopher M De Sa, Ce Zhang, Kunle Olukotun, and Christopher Ré. Taming the wild: A unified analysis of hogwild-style algorithms. In *Advances in neural information processing systems*, pages 2674–2682, 2015.
- Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. In *Advances in Neural Information Processing Systems*, pages 687–697, 2018.
- Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Research Logistics (NRL)*, 3(1-2):95–110, 1956.
- Dan Garber and Elad Hazan. Faster rates for the frank-wolfe method over strongly-convex sets. *arXiv preprint arXiv:1406.1305*, 2014.
- Dan Garber and Elad Hazan. Faster rates for the frank-wolfe method over strongly-convex sets. In *ICML*, volume 15, pages 541–549, 2015.
- Hamed Hassani, Amin Karbasi, Aryan Mokhtari, and Zebang Shen. Stochastic continuous greedy++: When upper and lower bounds match. In *Advances in Neural Information Processing Systems*, pages 13066–13076, 2019.
- Elad Hazan and Haipeng Luo. Variance-reduced and projection-free stochastic optimization. In *ICML*, pages 1263–1271, 2016.
- Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *ICML*, pages 427–435, 2013.
- Simon Lacoste-Julien. Convergence rate of frank-wolfe for non-convex objectives. *arXiv preprint arXiv:1607.00345*, 2016.
- Simon Lacoste-Julien and Martin Jaggi. On the global linear convergence of frank-wolfe optimization variants. In *Advances in Neural Information Processing Systems*, pages 496–504, 2015.
- Jean Lafond, Hoi-To Wai, and Eric Moulines. D-fw: Communication efficient distributed algorithms for high-dimensional sparse optimization. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 4144–4148. IEEE, 2016.
- Aryan Mokhtari, Hamed Hassani, and Amin Karbasi. Stochastic conditional gradient methods: From convex minimization to submodular maximization. *arXiv preprint arXiv:1804.09554*, 2018a.
- Aryan Mokhtari, Asuman Ozdaglar, and Ali Jadbabaie. Escaping saddle points in constrained optimization. In *Advances in Neural Information Processing Systems*, pages 3629–3639, 2018b.
- Lam M Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. Sarah: A novel method for machine learning problems using stochastic recursive gradient. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2613–2621. JMLR.org, 2017a.
- Lam M Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. Stochastic recursive gradient algorithm for nonconvex optimization. *arXiv preprint arXiv:1705.07261*, 2017b.
- Lam M Nguyen, Marten van Dijk, Dzung T Phan, Phuong Ha Nguyen, Tsui-Wei Weng, and Jayant R Kalagnanam. Optimal finite-sum smooth non-convex optimization with sarah. *arXiv preprint arXiv:1901.07648*, 2019.

- Wei Ping, Qiang Liu, and Alexander T Ihler. Learning infinite rbms with frank-wolfe. In *Advances in Neural Information Processing Systems*, pages 3063–3071, 2016.
- Sathya N Ravi, Tuan Dinh, Vishnu Sai Rao Lokhande, and Vikas Singh. Constrained deep learning using conditional gradient and applications in computer vision. *arXiv preprint arXiv:1803.06453*, 2018.
- Sashank J Reddi, Suvrit Sra, Barnabás Póczos, and Alex Smola. Stochastic frank-wolfe methods for nonconvex optimization. *arXiv preprint arXiv:1607.08254*, 2016.
- Patrick Schramowski, Christian Bauckhage, and Kristian Kersting. Neural conditional gradients. *arXiv preprint arXiv:1803.04300*, 2018.
- Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.
- Zebang Shen, Cong Fang, Peilin Zhao, Junzhou Huang, and Hui Qian. Complexities in projection-free stochastic non-convex minimization. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2868–2876, 2019a.
- Zebang Shen, Cong Fang, Peilin Zhao, Junzhou Huang, and Hui Qian. Complexities in projection-free stochastic non-convex minimization. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2868–2876, 2019b.
- Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified sgd with memory. In *Advances in Neural Information Processing Systems*, pages 4452–4463, 2018.
- Nikko Strom. Scalable distributed dnn training using commodity gpu cloud computing. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.
- Yu-Xiang Wang, Veeranjanyulu Sadhanala, Wei Dai, Willie Neiswanger, Suvrit Sra, and Eric Xing. Parallel and distributed block-coordinate frank-wolfe algorithms. In *International Conference on Machine Learning*, pages 1548–1557, 2016.
- Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In *Advances in neural information processing systems*, pages 1509–1519, 2017.
- Alp Yurtsever, Suvrit Sra, and Volkan Cevher. Conditional gradient methods via stochastic path-integrated differential estimator. In *International Conference on Machine Learning*, pages 7282–7291, 2019.
- Mingrui Zhang, Lin Chen, Aryan Mokhtari, Hamed Hassani, and Amin Karbasi. Quantized frank-wolfe: Faster optimization, lower communication, and projection free. *arXiv preprint arXiv:1902.06332*, 2019a.
- Mingrui Zhang, Zebang Shen, Aryan Mokhtari, Hamed Hassani, and Amin Karbasi. One sample stochastic frank-wolfe. *arXiv preprint arXiv:1910.04322*, 2019b.
- Wenpeng Zhang, Peilin Zhao, Wenwu Zhu, Steven CH Hoi, and Tong Zhang. Projection-free distributed online learning in networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 4054–4062. JMLR. org, 2017.
- Wenjie Zheng, Aurélien Bellet, and Patrick Gallinari. A distributed frank-wolfe framework for learning low-rank matrices with the trace norm. *Machine Learning*, 107(8-10):1457–1475, 2018.
- Dongruo Zhou, Pan Xu, and Quanquan Gu. Stochastic nested variance reduction for nonconvex optimization. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 3925–3936. Curran Associates Inc., 2018.

Submodular Streaming in All Its Glory: Tight Approximation, Minimum Memory and Low Adaptive Complexity

Ehsan Kazemi¹ Marko Mitrovic¹ Morteza Zadimoghaddam² Silvio Lattanzi² Amin Karbasi¹

Abstract

Streaming algorithms are generally judged by the quality of their solution, memory footprint, and computational complexity. In this paper, we study the problem of maximizing a monotone submodular function in the streaming setting with a cardinality constraint k . We first propose SIEVE-STREAMING++, which requires just one pass over the data, keeps only $O(k)$ elements and achieves the tight $1/2$ -approximation guarantee. The best previously known streaming algorithms either achieve a suboptimal $1/4$ -approximation with $\Theta(k)$ memory or the optimal $1/2$ -approximation with $O(k \log k)$ memory. Next, we show that by buffering a small fraction of the stream and applying a careful filtering procedure, one can heavily reduce the number of adaptive computational rounds, thus substantially lowering the computational complexity of SIEVE-STREAMING++. We then generalize our results to the more challenging multi-source streaming setting. We show how one can achieve the tight $1/2$ -approximation guarantee with $O(k)$ shared memory while minimizing not only the required rounds of computations but also the total number of communicated bits. Finally, we demonstrate the efficiency of our algorithms on real-world data summarization tasks for multi-source streams of tweets and of YouTube videos.

1. Introduction

Many important problems in machine learning, including data summarization, network inference, active set selection, facility location, and sparse regression can be cast as instances of constrained submodular maximization (Krause & Golovin, 2012). Submodularity captures an intuitive diminishing returns property where the gain of adding an element to a set decreases as the set gets larger. More formally, a

non-negative set function $f : 2^V \rightarrow \mathbb{R}_{\geq 0}$ is **submodular** if for all sets $A \subseteq B \subset V$ and every element $e \in V \setminus B$, we have $f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B)$. The submodular function f is **monotone** if for all $A \subseteq B$ we have $f(A) \leq f(B)$.

In this paper, we consider the following canonical optimization problem: given a non-negative monotone submodular function f , find the set S^* of size at most k that maximizes the function f :

$$S^* = \arg \max_{S \subseteq V, |S| \leq k} f(S). \quad (1)$$

We define $\text{OPT} = f(S^*)$. When the data is relatively small and it does not change over time, the greedy algorithm and other fast centralized algorithms provide near-optimal solutions. Indeed, it is well known that for problem (1) the greedy algorithm (which iteratively adds elements with the largest marginal gain) achieves a $1 - 1/e$ approximation guarantee (Nemhauser et al., 1978).

In many real-world applications, we are dealing with massive streams of images, videos, texts, sensor logs, tweets, and high-dimensional genomics data which are produced from different data sources. These data streams have an unprecedented volume and are produced so rapidly that they cannot be stored in memory, which means we cannot apply classical submodular maximization algorithms. In this paper, our goal is to design efficient algorithms for streaming submodular maximization in order to simultaneously provide the best approximation factor, memory complexity, running time, and communication cost.

For problem (1), Norouzi-Fard et al. (2018) proved that any streaming algorithm¹ with a memory $o(n/k)$ cannot provide an approximation guarantee better than $1/2$. SIEVE-STREAMING is the first streaming algorithm with a constant approximation factor (Badanidiyuru et al., 2014). This algorithm guarantees an approximation factor of $1/2 - \varepsilon$ and memory complexity of $O(k \log(k)/\varepsilon)$. While the approximation guarantee of their SIEVE-STREAMING is optimal, the memory complexity is a factor of $\log(k)$ away from the desired lower bound $\Theta(k)$. In contrast, Buchbinder et al. (2015)

¹Yale University ²Google Research. Correspondence to: Ehsan Kazemi <ehsan.kazemi@yale.edu>.

¹They assume the submodular function is evaluated only on feasible sets of cardinality at most k . In this paper, we make the same natural assumption regarding the feasible queries.

designed a streaming algorithm with a $1/4$ -approximation factor and optimal memory $\Theta(k)$. The first contribution of this paper is to answer the following question: Is there a streaming algorithm with an approximation factor arbitrarily close to $1/2$ whose memory complexity is $O(k)$?

Our new algorithm, SIEVE-STREAMING++, closes the gap between the optimal approximation factor and memory complexity, but it still has some drawbacks. In fact, in many applications of submodular maximization, the function evaluations (or equivalently Oracle queries)² are computationally expensive and can take a long time to process.

In this context, the fundamental concept of adaptivity quantifies the number of sequential rounds required to maximize a submodular function, where in each round, we can make polynomially many independent Oracle queries in parallel. More formally, given an Oracle f , an algorithm is ℓ -adaptive if every query Q to the Oracle f at a round $1 \leq i \leq \ell$ is independent of the answers $f(Q')$ to all other queries Q' at rounds j , $i \leq j \leq \ell$ (Balkanski & Singer, 2018). The adaptivity of an algorithm has important practical consequences as low adaptive complexity results in substantial speedups in parallel computing time.

All the existing streaming algorithms require at least one Oracle query for each incoming element. This results in an adaptive complexity of $\Omega(n)$ where n is the total number of elements in the stream. Furthermore, in many real-world applications, data streams arrive at such a fast pace that it is not possible to perform multiple Oracle queries in real time. This could result in missing potentially important elements or causing a huge delay.

Our idea to tackle the problem of adaptivity is to introduce a hybrid model where we allow a machine to buffer a certain amount of data, which allows us to perform many queries in parallel. We design a sampling algorithm that, in only a few adaptive rounds, picks items with good marginal gain and discards the rest. The main benefit of this method is that we can quickly empty the buffer and continue the optimization process. In this way, we obtain an algorithm with optimal approximation, query footprint, and near-optimal adaptivity.

Next, we focus on an additional challenge posed by real-world data where often multiple streams co-exist at the same time. In fact, while submodular maximization over only one stream of data is challenging, in practice there are many massive data streams generated simultaneously from a variety of sources. For example, these multi-source streams are generated by tweets from news agencies, videos and images from sporting events, or automated security systems and sensor logs. These data streams have an enormous volume and are produced so rapidly that they cannot be even transferred

²The Oracle for a submodular function f receives a set S and returns its value $f(S)$.

to a central machine. Therefore, in the multi-source streaming setting, other than approximation factor, memory and adaptivity, it is essential to keep communication cost low. To solve this problem, we show that a carefully-designed generalization of our proposed algorithm for single-source streams also has an optimal communication cost.

2. Related Work

Badanidiyuru et al. (2014) were the first to consider a one-pass streaming algorithm for maximizing a monotone submodular function under a cardinality constraint. Buchbinder et al. (2015) improved the memory complexity of (Badanidiyuru et al., 2014) to $\Theta(k)$ by designing a $1/4$ approximation algorithm. Norouzi-Fard et al. (2018) introduced an algorithm for random order streams that beats the $1/2$ bound. Recently, Agrawal et al. (2019) substantially improved the result for random order streams to an almost tight $1 - 1/e - \varepsilon - O(k^{-1})$ approximation factor. Norouzi-Fard et al. (2018) also studied the multi-pass streaming submodular maximization problem.

Chakrabarti & Kale (2015) studied streaming submodular maximization problem subject to the intersection of p matroid constraints. These results were further extended to more general constraints such as p -matchoids (Chekuri et al., 2015; Feldman et al., 2018). Also, there have been some very recent works to generalize these results to non-monotone submodular functions (Chakrabarti & Kale, 2015; Chekuri et al., 2015; Chan et al., 2017; Mirzasoleiman et al., 2018; Feldman et al., 2018). Elenberg et al. (2017) provide a streaming algorithm with a constant factor approximation for a generalized notion of submodular objective functions, called weak submodularity. In addition, a few other works study the streaming submodular maximization over sliding windows (Chen et al., 2016; Epasto et al., 2017).

To scale to very large datasets, several solutions to the problem of submodular maximization have been proposed in recent years (Mirzasoleiman et al., 2015; 2016a; Feldman et al., 2017; Badanidiyuru & Vondrák, 2014; Mitrovic et al., 2017a). Mirzasoleiman et al. (2015) proposed the first linear-time algorithm for maximizing a monotone submodular function subject to a cardinality constraint that achieves a $(1 - 1/e - \varepsilon)$ -approximation. Buchbinder et al. (2017) extended these results to non-monotone submodular functions.

Another line of work investigates the problem of scalable submodular maximization in the MapReduce setting where the data is split amongst several machines (Kumar et al., 2015; Mirzasoleiman et al., 2016b; Barbosa et al., 2015; Mirrokni & Zadimoghaddam, 2015; Mirzasoleiman et al., 2016c; Barbosa et al., 2016; Liu & Vondrák, 2018). Each machine runs a centralized algorithm on its data and passes the result to a central machine. Then, the central machine outputs the final answer. Since each machine runs a variant of the greedy algorithm, the adaptivity of all these ap-

proaches is linear in k , i.e., it is $\Omega(n)$ in the worst-case.

Practical concerns of scalability have motivated studying the adaptivity of submodular maximization algorithms. Balkanski & Singer (2018) showed that no algorithm can obtain a constant factor approximation in $o(\log n)$ adaptive rounds for monotone submodular maximization subject to a cardinality constraint. They introduced the first constant factor approximation algorithm for submodular maximization with logarithmic adaptive rounds. Their algorithm, in $O(\log n)$ adaptive rounds, finds a solution with an approximation arbitrarily close to $1/3$. These bounds were recently improved by $(1 - 1/e - \varepsilon)$ -approximation algorithm with $O(\log(n)/\text{poly}(\varepsilon))$ adaptivity (Fahrback et al., 2019; Balkanski et al., 2019a; Ene & Nguyen, 2019). More recently and independently, Balkanski et al. (2019b) and Chekuri & Quanrud (2019) studied the additivity of submodular maximization under a matroid constraint. In addition, Balkanski et al. (2018) proposed an algorithm for maximizing a non-monotone submodular function with cardinality constraint k whose approximation factor is arbitrarily close to $1/(2e)$ in $O(\log^2 n)$ adaptive rounds. Fahrback et al. (2018) improved the adaptive complexity of this problem to $O(\log(n))$. Chen et al. (2018) considered the unconstrained submodular maximization problem and proposed the first algorithm that achieves the optimal approximation guarantee in a constant number of adaptive rounds.

Contributions The main contributions of our paper are:

- We introduce SIEVE-STREAMING++ which is the first streaming algorithm with optimal approximation factor and memory complexity. Note that our optimality result for the approximation factor is under the natural assumption that the Oracle is allowed to make queries only over the feasible sets of cardinality at most k .
- We design an algorithm for a hybrid model of submodular maximization, where it enjoys a near-optimal adaptive complexity and it still guarantees both optimal approximation factor and memory complexity. We also prove that our algorithm has a very low communication cost in a multi-source streaming setting.
- We use multi-source streams of data from Twitter and YouTube to compare our algorithms against state-of-the-art streaming approaches.
- We significantly improve the memory complexity for several important problems in the submodular maximization literature by applying the main idea of SIEVE-STREAMING++ (see the Supplementary material).

3. Streaming Submodular Maximization

In this section, we propose an algorithm called SIEVE-STREAMING++ that has the optimal $1/2$ -approximation factor and memory complexity $O(k)$. Our algorithm is designed based on the SIEVE-STREAMING algorithm (Badanidiyuru et al., 2014).

The general idea behind SIEVE-STREAMING is that choosing elements with marginal gain at least $\tau^* = \frac{\text{OPT}}{2k}$ from a data stream returns a set S with an objective value of at least $f(S) \geq \frac{\text{OPT}}{2}$. The main problem with this primary idea is that the value of OPT is not known. Badanidiyuru et al. (2014) pointed out that, from the submodularity of f , we can trivially deduce $\Delta_0 \leq \text{OPT} \leq k\Delta_0$ where Δ_0 is the largest value in the set $\{f(\{e\}) \mid e \in V\}$. It is also possible to find an accurate guess for OPT by dividing the range $[\Delta_0, k\Delta_0]$ into small intervals of $[\tau_i, \tau_{i+1})$. For this reason, it suffices to try $\log k$ different thresholds τ to obtain a close enough estimate of OPT . Furthermore, in a streaming setting, where we do not know the maximum value of singletons a priori, Badanidiyuru et al. (2014) showed it suffices to only consider the range $\Delta \leq \text{OPT} \leq 2k\Delta$, where Δ is the maximum value of singleton elements observed so far. The memory complexity of SIEVE-STREAMING is $O(k \log k/\varepsilon)$ because there are $O(\log k/\varepsilon)$ different thresholds and, for each one, we could keep at most k elements.

3.1. The SIEVE-STREAMING++ Algorithm

In the rest of this section, we show that with a novel modification to SIEVE-STREAMING it is possible to significantly reduce the memory complexity of the streaming algorithm.

Our main observation is that in the process of guessing OPT , the previous algorithm uses Δ as a lower bound for OPT ; but as new elements are added to sets S_τ , it is possible to get better and better estimates of a lower bound on OPT . More specifically, we have $\text{OPT} \geq \text{LB} \triangleq \max_\tau f(S_\tau)$ and as a result, there is no need to keep thresholds smaller than $\frac{\text{LB}}{2k}$. Also, for a threshold τ we can conclude that there is at most $\frac{\text{LB}}{\tau}$ elements in set S_τ . These two important observations allow us to get a geometrically decreasing upper bound on the number of items stored for each guess τ , which gives the asymptotically optimal memory complexity of $O(k)$. The details of our algorithm (SIEVE-STREAMING++) are described in Algorithm 1. Note that we represent the marginal gain of a set A to the set B with $f(A \mid B) = f(A \cup B) - f(B)$. Theorem 1 guarantees the performance of SIEVE-STREAMING++. Table 1 compares the state-of-the-art streaming algorithms based on approximation ratio, memory complexity and queries per element.

Theorem 1. *For a non-negative monotone submodular function f subject to a cardinality constraint k , SIEVE-STREAMING++ returns a solution S such that (i) $f(S) \geq (1/2 - \varepsilon) \cdot \max_{A \subseteq V, |A| \leq k} f(A)$, (ii) memory complexity is $O(k/\varepsilon)$, and (iii) number of queries is $O(\log(k)/\varepsilon)$ per each element.*

3.2. The BATCH-SIEVE-STREAMING++ Algorithm

The SIEVE-STREAMING++ algorithm, for each incoming element of the stream, requires at least one query to the Oracle which increases its adaptive complexity to $\Omega(n)$. Since the adaptivity of an algorithm has a significant impact

Table 1. Streaming algorithms for non-negative and monotone submodular maximization subject to a cardinality constraint k . The SIEVE-STREAMING++ provides the best approximation ratio, memory complexity (up to a constant factor), and query complexity.

Algorithm	Approx. Ratio	Memory	Queries per Element	Reference
PREEMPTION-STREAMING	$1/4$	$O(k)$	$O(k)$	Buchbinder et al. (2015)
SIEVE-STREAMING	$1/2 - \varepsilon$	$O(k \log(k)/\varepsilon)$	$O(\log(k)/\varepsilon)$	Badanidiyuru et al. (2014)
SIEVE-STREAMING++	$1/2 - \varepsilon$	$O(k/\varepsilon)$	$O(\log(k)/\varepsilon)$	Ours

Algorithm 1 SIEVE-STREAMING++

Input: Submodular function f , data stream V , cardinality constraint k and error term ε

```

1:  $\tau_{\min} \leftarrow 0, \Delta \leftarrow 0$  and  $\text{LB} \leftarrow 0$ 
2: while there is an incoming item  $e$  from  $V$  do
3:    $\Delta \leftarrow \max\{\Delta, f(\{e\})\}$ 
4:    $\tau_{\min} = \frac{\max(\text{LB}, \Delta)}{2k}$ 
5:   Discard all sets  $S_\tau$  with  $\tau < \tau_{\min}$ 
6:   for  $\tau \in \{(1 + \varepsilon)^i | \tau_{\min}/(1 + \varepsilon) \leq (1 + \varepsilon)^i \leq \Delta\}$  do
7:     if  $\tau$  is a new threshold then  $S_\tau \leftarrow \emptyset$ 
8:     if  $|S_\tau| < k$  and  $f(\{e\} | S_\tau) \geq \tau$  then
9:        $S_\tau \leftarrow S_\tau \cup \{e\}$  and  $\text{LB} \leftarrow \max\{\text{LB}, f(S_\tau)\}$ 
10: return  $\arg \max_{S_\tau} f(S_\tau)$ 
    
```

on its ability to be executed in parallel, there is a dire need to implement streaming algorithms with low adaptivity. To address this concern, our proposal is to first buffer a fraction of the data stream and then, through a parallel threshold filtering procedure, reduce the adaptive complexity, thus substantially lower the running time. Our results show that a small buffer memory can significantly parallelize streaming submodular maximization.

One natural idea for parallelization is to iteratively perform the following two steps: (i) for a threshold τ , in one adaptive round, compute the marginal gain of elements to set S_τ and discard those with a gain less than τ , and (ii) pick one of the remaining items with a good marginal gain and add it to S_τ . This process is repeated at most k times. We refer to this algorithm as SAMPLE-ONE-STREAMING and we will use it as a baseline in Section 5.3.

Although this method gives a $1/2 - \varepsilon$ approximation factor, the adaptive complexity of this algorithm is $\Omega(k)$ which is still prohibitive in the worst case. For this reason, we introduce a hybrid algorithm called BATCH-SIEVE-STREAMING++. This algorithm enjoys two important properties: (i) the number of adaptive rounds is near-optimal, and (ii) it has an optimal memory complexity (by adopting an idea similar to SIEVE-STREAMING++). Next, we explain BATCH-SIEVE-STREAMING++ (Algorithm 2) in detail.

First, we assume that the machine has a buffer $\mathcal{B}$ that can store at most B elements. For a data stream V , whenever Threshold fraction of the buffer is full, the optimization

process begins. The purpose of Threshold is to empty the buffer before it gets completely full and to avoid losing arriving elements. Similar to the other sieve streaming methods, BATCH-SIEVE-STREAMING++ requires us to guess the value of $\tau^* = \frac{\text{OPT}}{2k}$. For each guess τ , BATCH-SIEVE-STREAMING++ uses THRESHOLD-SAMPLING (Algorithm 3) as a subroutine. THRESHOLD-SAMPLING iteratively picks random batches of elements T . If their average marginal gain to the set of picked elements S_τ is at least $(1 - \varepsilon)\tau$ it adds that batch to S_τ . Otherwise, all elements with marginal gain less than τ to the set S_τ are filtered out. THRESHOLD-SAMPLING repeats this process until the buffer is empty or $|S_\tau| = k$.

Note that in Algorithm 2, we define the function f_S as $f_S(A) = f(A | S)$, which calculates the marginal gain of adding a set A to S . It is straightforward to show that if f is a non-negative and monotone submodular function, then f_S is also non-negative and monotone submodular.

The adaptive complexity of BATCH-SIEVE-STREAMING++ is the number of times its buffer gets full (which is at most N/B) multiplied by the adaptive complexity of THRESHOLD-SAMPLING. The reason for the low adaptive complexity of THRESHOLD-SAMPLING is quite subtle. In Line 3 of Algorithm 3, with a non-negligible probability, a constant fraction of items is discarded from the buffer. Thus, the while loop continues for at most $O(\log B)$ steps. Since we increase the batch size by a constant factor of $(1 + \varepsilon)$ each time, the for loops within each while loop will run at most $O(\log(k)/\varepsilon)$ times. Therefore, the total adaptive complexity of BATCH-SIEVE-STREAMING++ is $O(\frac{N \log(B) \log(k)}{B\varepsilon})$. Note that when $|S| < 1/\varepsilon$, multiplying the size by $(1 + \varepsilon)$ would increase it less than one, so we increase the batch size one by one for the first loop in Lines 4–10 of Algorithm 3. Theorem 2 guarantees the performance of BATCH-SIEVE-STREAMING++.

Theorem 2. *For a non-negative monotone submodular function f subject to a cardinality constraint k , define N to be the total number of elements in the stream, B to be the buffer size and $\varepsilon < 1/3$ to be a constant. For BATCH-SIEVE-STREAMING++ we have: (i) the approximation factor is $1/2 - 3\varepsilon/2$, (ii) the memory complexity is $O(B + k/\varepsilon)$, and (iii) the expected adaptive complexity is $O(\frac{N \log(B) \log(k)}{B\varepsilon})$.*

Algorithm 2 BATCH-SIEVE-STREAMING++

Input: Stream of data V , submodular set function f , cardinality constraint k , buffer $\mathcal{B}$ with a memory B , Threshold, and error term ε .

```

1:  $\Delta \leftarrow 0, \tau_{\min} \leftarrow 0, LB \leftarrow 0$  and  $\mathcal{B} \leftarrow \emptyset$ 
2: while there is an incoming element  $e$  from  $V$  do
3:   Add  $e$  to  $\mathcal{B}$ 
4:   if the buffer  $\mathcal{B}$  is Threshold percent full then
5:      $\Delta \leftarrow \max\{\Delta, \max_{e \in \mathcal{B}} f(e)\}, \tau_{\min} = \frac{\max(LB, \Delta)}{2k(1+\varepsilon)}$ 
     and discard all sets  $S_\tau$  with  $\tau < \tau_{\min}$ 
6:     for  $\tau \in \{(1+\varepsilon)^i | \tau_{\min} \leq (1+\varepsilon)^i \leq \Delta\}$  do
7:       If  $\tau$  is a new threshold then assign a new set  $S_\tau$ 
       to it, i.e.,  $S_\tau \leftarrow \emptyset$ 
8:       if  $|S_\tau| < k$  then
9:          $T \leftarrow \text{THRESHOLD-SAMPLING}(f_{S_\tau}, \mathcal{B}, k - |S_\tau|, \tau, \varepsilon)$ 
10:         $S_\tau \leftarrow S_\tau \cup T$ 
11:       $LB = \max_{S_\tau} f(S_\tau)$  and  $\mathcal{B} \leftarrow \emptyset$ 
12: return  $\arg \max_{S_\tau} f(S_\tau)$ 
    
```

Algorithm 3 THRESHOLD-SAMPLING

Input: Submodular set function f , set of buffered items $\mathcal{B}$, cardinality constraint k , threshold τ , and error term ε

```

1:  $S \leftarrow \emptyset$ 
2: while  $|\mathcal{B}| > 0$  and  $|S| < k$  do
3:   update  $\mathcal{B} \leftarrow \{x \in \mathcal{B} : f(\{x\} \mid S) \geq \tau\}$  and filter
   out the rest
4:   for  $i = 1$  to  $\lceil \frac{1}{\varepsilon} \rceil$  do
5:     Sample  $x$  uniformly at random from  $\mathcal{B} \setminus S$ 
6:     if  $f(\{x\} \mid S) \leq (1-\varepsilon)\tau$  then
7:       break and go to Line 2
8:     else
9:        $S \leftarrow S \cup \{x\}$ 
10:    if  $|S| = k$  then return  $S$ 
11:  for  $i = \lfloor \log_{1+\varepsilon}(1/\varepsilon) \rfloor$  to  $\lceil \log_{1+\varepsilon} k \rceil - 1$  do
12:     $t \leftarrow \min\{\lfloor (1+\varepsilon)^{i+1} - (1+\varepsilon)^i \rfloor, |\mathcal{B} \setminus S|, k - |S|\}$ 
13:    Sample a random set  $T$  of size  $t$  from  $\mathcal{B} \setminus S$ 
14:    if  $|S \cup T| = k$  then return  $S \cup T$ 
15:    if  $\frac{f(T \mid S)}{|T|} \leq (1-\varepsilon)\tau$  then
16:       $S \leftarrow S \cup T$  and break
17:    else
18:       $S \leftarrow S \cup T$ 
19: return  $S$ 
    
```

Remark It is important to note that THRESHOLD-SAMPLING is inspired by recent progress for maximizing submodular functions with low adaptivity (Fahrback et al., 2019; Balkanski et al., 2019a; Ene & Nguyen, 2019) but it uses a few new ideas to adapt the result to our setting. Indeed, if we had used the sampling routines from these previous works, it was even possible to slightly improve the

adaptivity of the hybrid model. The main issue with these methods is that their adaptivity heavily depends on evaluating many random subsets of the ground set in each round. As it is discussed in the next section, we are interested in algorithms that are efficient in the multi-source setting. In that scenario, the data is distributed among several machines, so existing sampling methods dramatically increases the communication cost of our hybrid algorithm.

4. Multi-Source Data Streams

In general, the important aspects to consider for a single source streaming algorithm are approximation factor, memory complexity, and adaptivity. In the multi-source setting, the communication cost of an algorithm also plays an important role. While the main ideas of SIEVE-STREAMING++ give us an optimal approximation factor and memory complexity, there is always a trade-off between adaptive complexity and communication cost in any threshold sampling procedure.

As we discussed before, existing submodular maximization algorithms with low adaptivity need to evaluate the utility of random subsets several times to guarantee the marginal gain of sampled items. Consequently, this incurs high communication cost. In this section, we explain how BATCH-SIEVE-STREAMING++ can be generalized to the multi-source scenario with both low adaptivity and low communication cost.

We assume elements arrive from m different data streams and for each stream the elements are placed in a separate machine with a buffer $\mathcal{B}_i$. When the buffer memory of at least one of these m machines is Threshold% full, the process of batch insertion and filtering begins. The only necessary change to BATCH-SIEVE-STREAMING++ is to use a parallelized version of THRESHOLD-SAMPLING with inputs from $\{\mathcal{B}_i\}$. In this generalization, Lines 5 and 13 of Algorithm 3 are executed in a distributed way where the goal is to perform the random sampling procedure from the buffer memory of all machines. Indeed, in order to pick a batch of t random items, the central coordinator asks each machine to send a pre-decided number of items. Note that the set of picked elements S_τ for each threshold τ is shared among all machines. And therefore the filtering step at Line 3 of Algorithm 3 can be done independently for each stream in only one adaptive round. Our algorithm is shown pictorially in Figure 1.

Theorem 3 guarantees the communication cost of BATCH-SIEVE-STREAMING++ in the multi-source setting. Notably, the communication cost of our algorithm is independent of the buffer size B and the total number of elements N .

Theorem 3. For a non-negative and monotone submodular function f in a multi-source streaming setting subject to a cardinality constraint k , define Δ_0 as the largest singleton value when for the first time a buffer gets full, and $C = \frac{\text{OPT}}{\Delta_0}$. The total communication cost of BATCH-SIEVE-STREAMING++ is $O(\frac{k \log C}{\epsilon^2})$.

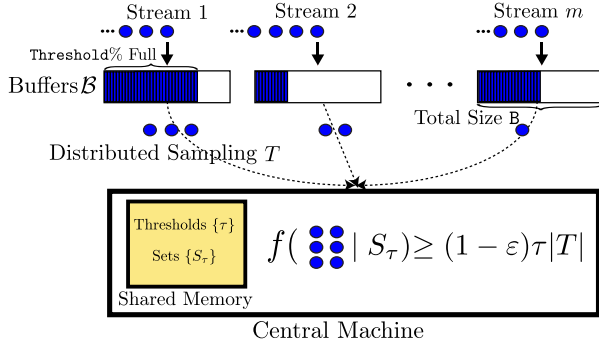


Figure 1. The schematic representation of our proposed hybrid algorithm: there are m simultaneous streams where data from each stream is buffered separately. When a buffer is `Threshold%` full, a central machine starts the sampling process. The thresholds $\{\tau\}$ and sets $\{S_\tau\}$ are stored in a shared memory. First, for each threshold τ , all elements with marginal gain less than τ are discarded from the buffers. Then the central machine randomly samples t items T (with geometrically increasing values of t) from the buffers of all streams and adds them to set S_τ if their average marginal gain is at least $(1 - \epsilon)\tau$. The sampling procedure stops when the average value of randomly picked items is not good enough. These iterative steps are performed until k items are picked or the buffer memories of all machines are emptied.

5. Experiments

In these experiments, we have three main goals:

1. For the single-source streaming scenario, we want to demonstrate the memory efficiency of SIEVE-STREAMING++ relative to SIEVE-STREAMING.
2. For the multi-source setting, we want to showcase how BATCH-SIEVE-STREAMING++ requires the fewest adaptive rounds amongst algorithms with optimal communication costs.
3. Lastly, we want to illustrate how a simple variation of BATCH-SIEVE-STREAMING++ can trade off communication cost for adaptivity, thus allowing the user to find the best balance for their particular problem.

5.1. Datasets

These experiments will be run on a Twitter stream summarization task and a YouTube Video summarization task, as described next.

Twitter Stream Summarization In this application, we want to produce real-time summaries for Twitter feeds. As of January 2019, six of the top fifty Twitter accounts (also known as “handles”) are dedicated primarily to news reporting. Each of these handles has over thirty million followers, and there are many other news handles with tens of millions of followers as well. Naturally, such accounts commonly share the same stories. Whether we want to provide a periodic synopsis of major events or simply to reduce the clutter in a user’s feed, it would be very valuable if we could produce a succinct summary that still relays all the important information.

To collect the data, we scraped recent tweets from 30 different popular news accounts, giving us a total of 42,104 unique tweets. In the multi-source experiments, we assume that each machine is scraping one page of tweets, so we have 30 different streams to consider.

We want to define a submodular function that covers the important stories of the day without redundancy. To this end, we extract the keywords from each tweet and weight them proportionally to the number of retweets the post received. In order to encourage diversity in a selected set of tweets, we take the square root of the value assigned to each keyword. More formally, consider a function f defined over a ground set V of tweets. Each tweet $e \in V$ consists of a positive value val_e denoting its number of retweets and a set of ℓ_e keywords $W_e = \{w_{e,1}, \dots, w_{e,\ell_e}\}$ from a general set of keywords $\mathcal{W}$. The score of a word $w \in W_e$ for a tweet e is defined by $\text{score}(w, e) = \text{val}_e$. If $w \notin W_e$, we define $\text{score}(w, e) = 0$. For a set $S \subseteq V$ of tweets, the function f is defined as follows:

$$f(S) = \sum_{w \in \mathcal{W}} \sqrt{\sum_{e \in S} \text{score}(w, e)}.$$

Figure 2 gives an example and Appendix E gives a proof of submodularity for this function.

YouTube Video Summarization In this second task, we want to select representative frames from multiple simultaneous and related video feeds. In particular, we consider YouTube videos of New Year’s Eve celebrations from ten different cities around the world. Although the cities are not all in the same time zone, in our multi-source experiments we assume that we have one machine processing each video simultaneously.

Using the first 30 seconds of each video, we train an autoencoder that compresses each frame into a 4-dimensional representative vector. Given a ground set V of such vectors, we define a matrix M such that $M_{ij} = e^{-\text{dist}(v_i, v_j)}$, where $\text{dist}(v_i, v_j)$ is the euclidean distance between vectors $v_i, v_j \in V$. Intuitively, M_{ij} encodes the similarity between the frames represented by v_i and v_j .

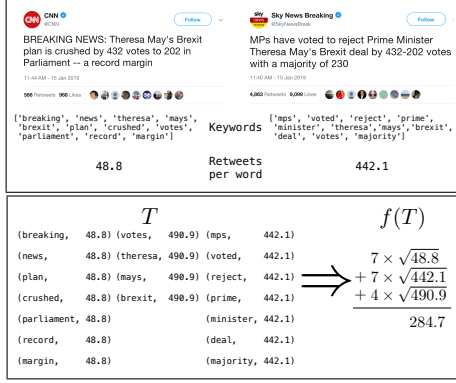


Figure 2. At the top, we show two tweets on the same subject from different accounts. We first extract the list of keywords, as well as the number of retweets per word. We combine these into a single list T of (keyword, score) pairs and then pass this list through our submodular function f .

The utility of a set $S \subseteq V$ is defined as a non-negative monotone submodular objective $f(S) = \log \det(I + \alpha M_S)$, where I is the identity matrix, $\alpha > 0$ and M_S is the principal sub-matrix of M indexed by S (Herbrich et al., 2003). Informally, this function is meant to measure the diversity of the vectors in S . Figure 3 shows the representative images selected by our BATCH-SIEVE-STREAMING++ algorithm for $k = 8$.



Figure 3. Eight representative frames chosen by BATCH-SIEVE-STREAMING++ from ten different simultaneous feeds of New Year's Eve fireworks from around the world.

5.2. Single-Source Experiments

In this section, we want to emphasize the power of SIEVE-STREAMING++ in the single-source streaming scenario. As discussed earlier, the two existing standard approaches for monotone k -cardinality submodular streaming are SIEVE-STREAMING and PREEMPTION-STREAMING.

As mentioned in Section 3, SIEVE-STREAMING++ theoretically has the best properties of both of these existing baselines, with optimal memory complexity and the optimal approximation guarantee. Figures 4a through 4d show the performance of these three algorithms on the YouTube task and confirm that this holds in practice as well.

For the purposes of this test, we simply combined the different video feeds into one single stream. We see that the mem-

ory required by SIEVE-STREAMING++ is much smaller than the memory required by SIEVE-STREAMING, but it still achieves the exact same utility. Furthermore, the memory requirement of SIEVE-STREAMING++ is within a constant factor of PREEMPTION-STREAMING, while its utility is much better. The Twitter experiment gives similar results so those graphs are deferred to Appendix F.

5.3. Multi-Source Experiments

Once we move into the multi-source setting, the communication cost of algorithms becomes a key concern also. In this section, we compare the performance of algorithms in terms of utility and adaptivity where their communication cost is optimal.

Our first baseline is a trivial extension of SIEVE-STREAMING++. The multi-source extension for this algorithm essentially functions by locally computing the marginal gain of each incoming element, and only communicating it to the central machine if the marginal gain is above the desired threshold. However, as mentioned at the beginning Section 3.1, this algorithm requires $\Omega(n)$ adaptive rounds. Our second baseline is SAMPLE-ONE-STREAMING, which was described in Section 3.2.

Figures 4e and 4f show the effect of the buffer size B on the performance of these algorithms for the Twitter task. The main observation is that BATCH-SIEVE-STREAMING++ can achieve roughly the same utility as the two baselines with many fewer adaptive rounds. Note that the number of adaptive rounds is shown in log scale.

Figures 4g and 4h show how these numbers vary with ε . Again, the utilities of the three baselines are similar. We also see that increasing ε results in a large drop in the number of adaptive rounds for BATCH-SIEVE-STREAMING++, but not for SAMPLE-ONE-STREAMING. Appendix F gives some additional graphs, as well as the results for the YouTube dataset.

5.4. Trade-off Between Communication and Adaptivity

In the multi-source setting, there is a natural exchange between communication cost and adaptivity. Intuitively, the idea is that if we sample items more aggressively (which translates into higher communication cost), a set S of k items is generally picked faster, thus it reduces the adaptivity. In the real world, the preference for one or the other can depend on a wide variety of factors ranging from resource constraints to the requirements of the particular problem.

In THRESHOLD-SAMPLING, we ensure the optimal communication performance by sampling $t_i = \lceil (1 + \varepsilon)^{i+1} - (1 + \varepsilon)^i \rceil$ items in each step of the for loop. Instead, to reduce the adaptivity by a factor of $\log(k)$, we could sample all the required k items in a single step. Thus, in one

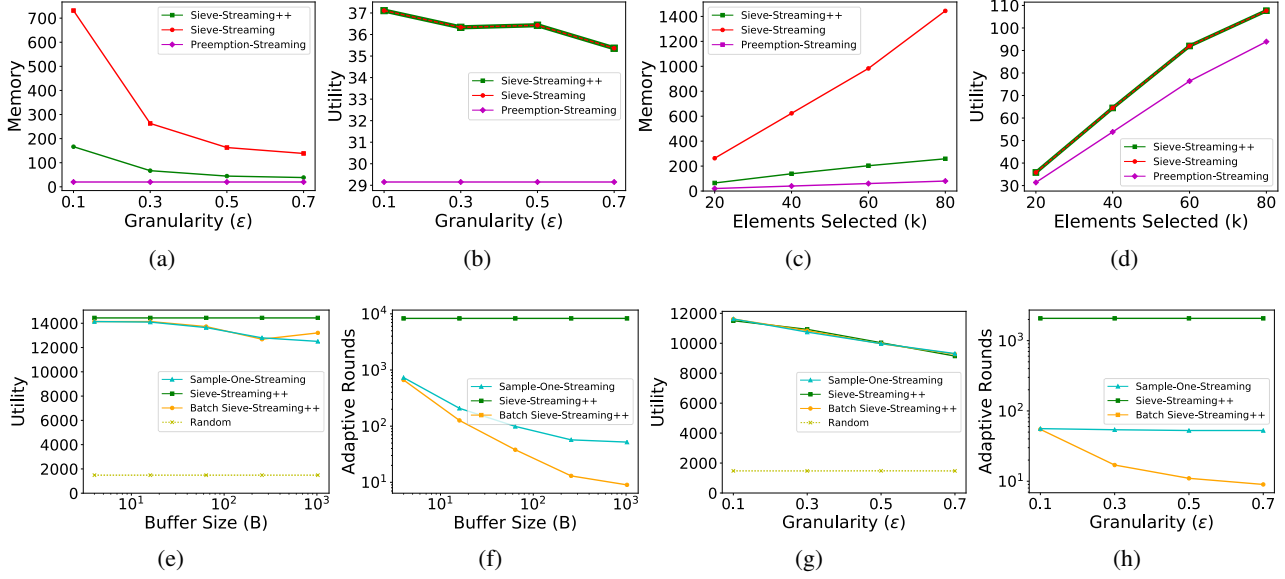


Figure 4. Graphs (a) to (d) show how the memory and utility of various single-source streaming algorithms vary with the cardinality k and granularity ϵ . Note that the utility of SIEVE-STREAMING++ and SIEVE-STREAMING exactly overlap. In (a) and (b) we use $k = 20$, while in (c) and (d) we use $\epsilon = 0.3$. Graphs (e) to (h) show how the utility and adaptivity of various multi-source streaming algorithms vary with the buffer size B and the granularity ϵ . Unless they are being varied on the x-axis, we set $\epsilon = 0.7$, $B = 100$, and $k = 50$.

adaptive round we mimic the two for loops of THRESHOLD-SAMPLING. Doing this in each call to Algorithm 3 would reduce the expected adaptive complexity of THRESHOLD-SAMPLING to the optimal $\log(B)$, but dramatically increase the communication cost to $O(k \log B)$.

In order to trade off between communication and adaptivity, we can instead sample $t_i^R = \lceil (1 + \epsilon)^{i+R} - (1 + \epsilon)^i \rceil$ elements to perform R consecutive adaptive rounds in only one round. However, to maintain the same chance of a successful sampling, we still need to check the marginal gain. Finally, we pick a batch of the largest size t_i^j such that the average marginal gain of the first t_i^{j-1} items is above the desired threshold. Then we just add just this subset to S_τ , meaning we have wasted $\lceil (1 + \epsilon)^{i+R} - (1 + \epsilon)^{i+j} \rceil$ communication.

Scatter plots of Figure 5 shows how the number of adaptive rounds varies with the communication cost. Each individual dot represents a single run of the algorithm on a different subset of the data. The different colors cluster the dots into groups based on the value of R that we used in that run. Note that the parameter R controls the communication cost.

The plot on the left comes from the Twitter experiment, while the plot on the right comes from the YouTube experiment. Although the shapes of the clusters are different in the two experiments, we see that increasing R increases the communication cost, but also decreases the number of adaptive rounds, as expected.

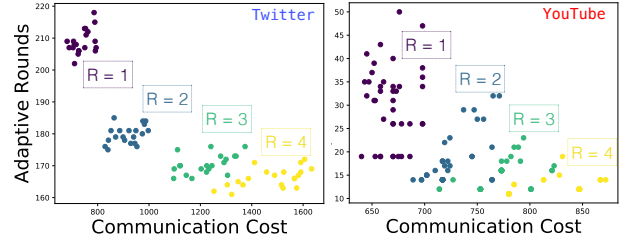


Figure 5. Scatter plots showing how we can lower the number of adaptive rounds by increasing communication. Each dot is the result of a single run of the algorithm and the colored clusters represent a particular setting for R .

6. Conclusion

In this paper, we studied the problem of maximizing a non-negative submodular function over a multi-source stream of data subject to a cardinality constraint k . We first proposed SIEVE-STREAMING++ with the optimum approximation factor and memory complexity for a single stream of data. Build upon this idea, we designed an algorithm for multi-source streaming setting with a $1/2$ approximation factor, $O(k)$ memory complexity, a very low communication cost, and near-optimal adaptivity. We evaluated the performance of our algorithms on two real-world data sets of multi-source tweet streams and video streams. Furthermore, by using the main idea of SIEVE-STREAMING++, we significantly improved the memory complexity of several important submodular maximization problems.

Acknowledgements

The work of Amin Karbasi is supported by AFOSR Young Investigator Award (FA9550-18-1-0160) and Grainger Award (PO 2000008083 2016012). We would like to thank Ola Svensson for his comment on the first version of this manuscript.

References

- Agrawal, S., Shadravan, M., and Stein, C. Submodular Secretary Problem with Shortlists. In *Innovations in Theoretical Computer Science Conference, ITCS*, pp. 1:1–1:19, 2019.
- Badanidiyuru, A. and Vondrák, J. Fast algorithms for maximizing submodular functions. In *Symposium on Discrete Algorithms, SODA*, pp. 1497–1514, 2014.
- Badanidiyuru, A., Mirzasoleiman, B., Karbasi, A., and Krause, A. Streaming Submodular Maximization: Massive Data Summarization on the Fly. In *International Conference on Knowledge Discovery and Data Mining, KDD*, pp. 671–680, 2014.
- Balkanski, E. and Singer, Y. The adaptive complexity of maximizing a submodular function. In *Symposium on Theory of Computing, STOC*, pp. 1138–1151, 2018.
- Balkanski, E., Mirzasoleiman, B., Krause, A., and Singer, Y. Learning sparse combinatorial representations via two-stage submodular maximization. In *International Conference on Machine Learning (ICML)*, 2016.
- Balkanski, E., Breuer, A., and Singer, Y. Non-monotone Submodular Maximization in Exponentially Fewer Iterations. In *Advances in Neural Information Processing Systems*, pp. 2359–2370, 2018.
- Balkanski, E., Rubinstein, A., and Singer, Y. An Exponential Speedup in Parallel Running Time for Submodular Maximization without Loss in Approximation. In *Symposium on Discrete Algorithms (SODA)*, pp. 283–302, 2019a.
- Balkanski, E., Rubinstein, A., and Singer, Y. An optimal approximation for submodular maximization under a matroid constraint in the adaptive complexity model. In *Symposium on Theory of Computing, STOC*, 2019b.
- Bansal, N. and Sviridenko, M. The santa claus problem. In *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing*, pp. 31–40. ACM, 2006.
- Barbosa, R., Ene, A., Nguyen, H., and Ward, J. The power of randomization: Distributed submodular maximization on massive datasets. In *International Conference on Machine Learning (ICML)*, pp. 1236–1244, 2015.
- Barbosa, R., Ene, A., Nguyen, H. L., and Ward, J. A New Framework for Distributed Submodular Maximization. In *Annual Symposium on Foundations of Computer Science, FOCS*, pp. 645–654, 2016.
- Buchbinder, N., Feldman, M., and Schwartz, R. Online submodular maximization with preemption. In *ACM-SIAM Symposium on Discrete Algorithms, SODA*, pp. 1202–1216, 2015.
- Buchbinder, N., Feldman, M., and Schwartz, R. Comparing Apples and Oranges: Query Trade-off in Submodular Maximization. *Math. Oper. Res.*, 42(2):308–329, 2017.
- Chakrabarti, A. and Kale, S. Submodular maximization meets streaming: matchings, matroids, and more. *Math. Program.*, 154(1-2):225–247, 2015.
- Chan, T. H., Huang, Z., Jiang, S. H., Kang, N., and Tang, Z. G. Online Submodular Maximization with Free Disposal: Randomization Beats $\frac{1}{4}$ for Partition Matroids. In *Symposium on Discrete Algorithms, SODA*, pp. 1204–1223, 2017.
- Chekuri, C. and Quanrud, K. Parallelizing greedy for submodular set function maximization in matroids and beyond. In *Symposium on Theory of Computing, STOC*, 2019.
- Chekuri, C., Gupta, S., and Quanrud, K. Streaming algorithms for submodular function maximization. In *International Colloquium on Automata, Languages, and Programming*, pp. 318–330. Springer, 2015.
- Chen, J., Nguyen, H. L., and Zhang, Q. Submodular Maximization over Sliding Windows. *CoRR*, abs/1611.00129, 2016.
- Chen, L., Feldman, M., and Karbasi, A. Unconstrained submodular maximization with constant adaptive complexity. *CoRR*, abs/1811.06603, 2018.
- Das, A. and Kempe, D. Submodular meets Spectral: Greedy Algorithms for Subset Selection, Sparse Approximation and Dictionary Selection. In *International Conference on Machine Learning (ICML)*, pp. 1057–1064, 2011.
- Elenberg, E. R., Dimakis, A. G., Feldman, M., and Karbasi, A. Streaming Weak Submodularity: Interpreting Neural Networks on the Fly. In *Advances in Neural Information Processing Systems*, pp. 4047–4057, 2017.
- Ene, A. and Nguyen, H. L. Submodular Maximization with Nearly-optimal Approximation and Adaptivity in Nearly-linear Time. In *Symposium on Discrete Algorithms (SODA)*, pp. 274–282, 2019.

- Epasto, A., Lattanzi, S., Vassilvitskii, S., and Zadimoghaddam, M. Submodular Optimization Over Sliding Windows. In *WWW*, pp. 421–430, 2017.
- Fahrbach, M., Mirrokni, V. S., and Zadimoghaddam, M. Non-monotone Submodular Maximization with Nearly Optimal Adaptivity Complexity. *CoRR*, abs/1808.06932, 2018.
- Fahrbach, M., Mirrokni, V. S., and Zadimoghaddam, M. Submodular Maximization with Nearly Optimal Approximation, Adaptivity and Query Complexity. In *Symposium on Discrete Algorithms (SODA)*, pp. 255–273, 2019.
- Feldman, M., Harshaw, C., and Karbasi, A. Greed Is Good: Near-Optimal Submodular Maximization via Greedy Optimization. In *Conference on Learning Theory*, 2017.
- Feldman, M., Karbasi, A., and Kazemi, E. Do Less, Get More: Streaming Submodular Maximization with Sub-sampling. In *Advances in Neural Information Processing Systems*, pp. 730–740, 2018.
- Herbrich, R., Lawrence, N. D., and Seeger, M. Fast sparse gaussian process methods: The informative vector machine. In *Advances in Neural Information Processing Systems*, pp. 625–632, 2003.
- Kazemi, E., Zadimoghaddam, M., and Karbasi, A. Scalable Deletion-Robust Submodular Maximization: Data Summarization with Privacy and Fairness Constraints. In *International Conference on Machine Learning (ICML)*, pp. 2549–2558, 2018.
- Krause, A. and Golovin, D. Submodular Function Maximization. In *Tractability: Practical Approaches to Hard Problems*. Cambridge University Press, 2012.
- Kumar, R., Moseley, B., Vassilvitskii, S., and Vattani, A. Fast Greedy Algorithms in MapReduce and Streaming. *TOPC*, 2(3):14:1–14:22, 2015.
- Liu, P. and Vondrák, J. Submodular Optimization in the MapReduce Model. *CoRR*, abs/1810.01489, 2018.
- Mirrokn, V. and Zadimoghaddam, M. Randomized composable core-sets for distributed submodular maximization. In *Symposium on Theory of Computing, , STOC*, pp. 153–162. ACM, 2015.
- Mirzasoleiman, B., Badanidiyuru, A., Karbasi, A., Vondrak, J., and Krause, A. Lazier than Lazy Greedy. In *AAAI Conference on Artificial Intelligence*, pp. 1812–1818, 2015.
- Mirzasoleiman, B., Badanidiyuru, A., and Karbasi, A. Fast constrained submodular maximization: Personalized data summarization. In *International Conference on Machine Learning (ICML)*, pp. 1358–1367, 2016a.
- Mirzasoleiman, B., Karbasi, A., Sarkar, R., and Krause, A. Distributed Submodular Maximization. *Journal of Machine Learning Research (JMLR)*, 17:1–44, 2016b.
- Mirzasoleiman, B., Zadimoghaddam, M., and Karbasi, A. Fast Distributed Submodular Cover: Public-Private Data Summarization. In *Advances in Neural Information Processing Systems*, 2016c.
- Mirzasoleiman, B., Karbasi, A., and Krause, A. Deletion-Robust Submodular Maximization: Data Summarization with “the Right to be Forgotten”. In *International Conference on Machine Learning (ICML)*, pp. 2449–2458, 2017.
- Mirzasoleiman, B., Jegelka, S., and Krause, A. Streaming Non-Monotone Submodular Maximization: Personalized Video Summarization on the Fly. In *AAAI Conference on Artificial Intelligence*, 2018.
- Mitrovic, M., Bun, M., Krause, A., and Karbasi, A. Differentially Private Submodular Maximization: Data Summarization in Disguise. In *International Conference on Machine Learning (ICML)*, pp. 2478–2487, 2017a.
- Mitrovic, M., Kazemi, E., Zadimoghaddam, M., and Karbasi, A. Data Summarization at Scale: A Two-Stage Submodular Approach. In *International Conference on Machine Learning (ICML)*, pp. 3593–3602, 2018.
- Mitrovic, S., Bogunovic, I., Norouzi-Fard, A., Tarnawski, J. M., and Cevher, V. Streaming Robust Submodular Maximization: A Partitioned Thresholding Approach. In *Advances in Neural Information Processing Systems*, pp. 4560–4569, 2017b.
- Nemhauser, G. L., Wolsey, L. A., and Fisher, M. L. An analysis of approximations for maximizing submodular set functions-I. *Mathematical programming*, 14(1):265–294, 1978.
- Norouzi-Fard, A., Tarnawski, J., Mitrovic, S., Zandieh, A., Mousavifar, A., and Svensson, O. Beyond 1/2-Approximation for Submodular Maximization on Massive Data Streams. In *International Conference on Machine Learning (ICML)*, pp. 3826–3835, 2018.
- Stan, S., Zadimoghaddam, M., Krause, A., and Karbasi, A. Probabilistic Submodular Maximization in Sub-Linear Time. In *International Conference on Machine Learning (ICML)*, 2017.