Asynchronous Stochastic Frank-Wolfe Algorithms for Non-Convex Optimization

Bin Gu[◇†], **Wenhan Xian**[†] and **Heng Huang**[†◇ *]

[◇]JD Finance America Corporation

[†]Department of Electrical & Computer Engineering, University of Pittsburgh, USA

jsgubin@gmail.com, WEX37@pitt.edu, heng.huang@pitt.edu

Abstract

Asynchronous parallel stochastic optimization for non-convex problems becomes more and more important in machine learning especially due to the popularity of deep learning. The Frank-Wolfe (a.k.a. conditional gradient) algorithms has regained much interest because of its projection-free property and the ability of handling structured constraints. However, our understanding of asynchronous stochastic Frank-Wolfe algorithms is extremely limited especially in the non-convex setting. To address this challenging problem, in this paper, we propose our asynchronous stochastic Frank-Wolfe algorithm (AsySFW) and its variance reduction version (AsySVFW) for solving the constrained non-convex optimization problems. More importantly, we prove the fast convergence rates of AsySFW and AsySVFW in the non-convex setting. To the best of our knowledge, AsySFW and AsySVFW are the first asynchronous parallel stochastic algorithms with convergence guarantees for solving the constrained non-convex optimization problems. The experimental results on real high-dimensional gray-scale images not only confirm the fast convergence of our algorithms, but also show a near-linear speedup on a parallel system with shared memory due to the lock-free implementation.

1 Introduction

Asynchronous parallel stochastic optimization algorithms are popular in the current big data era because of the outstanding scalability w.r.t. the sample size and the full exploitation of the computing resources of multi-core machines. For example, Hogwild! [Recht *et al.*, 2011] is a famous asynchronous parallel stochastic gradient descent algorithm for solving smooth finite-sum optimization problems. Non-convex optimization is now ubiquitous in machine learning especially due to the popularity of deep learning. Thus, the research of asynchronous parallel stochastic optimization for non-convex problems becomes more and more important in machine learning.

In machine learning, a lot of learning problems are formulated as a finite-sum of loss functions with one or more structured constraints. For example, multi-class classification [Dudik *et al.*, 2012] and matrix completion [Hsieh *et al.*, 2015] consider a finite-sum function with the trace-norm constraint. Octagonal shrinkage and clustering algorithm for regression (OSCAR) [Gu *et al.*, 2017] considers a finite-sum function with ℓ_1 norm and pairwise ℓ_∞ norm constraints. Normally, we use proximal gradient descent algorithms [Gu *et al.*, 2018b] to solve the above constrained optimization problems, in which each iteration solves a projection mapping. However, it is maybe time-consuming to solve the projection mapping which is a quadratic problem for a lot of norms (including the trace norm [Dudik *et al.*, 2012] and the OSCAR norm [Gu *et al.*, 2017]). Alternatively, [Frank and Wolfe, 1956] first proposed the Frank-Wolfe (a.k.a. conditional gradient) algorithm where only a less expensive linear subproblem need to be solved per iteration. Recently, the Frank-Wolfe algorithms have regained much interest due to its projection-free property and its ability to handle structured constraints.

Most of existing Frank-Wolfe algorithms were designed for the constrained smooth convex optimization problems. Specifically, several variants of deterministic Frank-Wolfe algorithms [Frank and Wolfe, 1956; Jaggi, 2013; Garber and Hazan, 2015; Lacoste-Julien *et al.*, 2013; Lacoste-Julien and Jaggi, 2015] were proposed, and their convergence rates in the convex setting were provided. To handle large-scale problems, corresponding stochastic Frank-Wolfe algorithms [Lan, 2013; Hazan and Luo, 2016; Goldfarb *et al.*, 2017] were proposed recently, and their convergence rates were also provided. In addition to the stochastic Frank-Wolfe algorithms, distributed parallel Frank-Wolfe algorithms [Wang *et al.*, 2016; Moharrer and Ioannidis, 2017; Zhang *et al.*, 2017] were also proposed recently and their convergence rates were provided. As mentioned above, the convergence analysis of most of existing Frank-Wolfe algorithms were provided only for the convex setting.

Because non-convex optimization is now ubiquitous in machine learning as mentioned previously, in this paper, we mainly consider the constrained non-convex optimization

*To whom all correspondence should be addressed.

Algorithm	Reference	Function $f(x)$	Stochastic gradient	Variance reduced	Asynchronous
Frank-Wolfe	[Lacoste-Julien, 2016]	Non-convex	No	No	No
SFW	[Reddi *et al.*, 2016]	Non-convex	Yes	No	No
SVFW	[Reddi *et al.*, 2016]	Non-convex	Yes	Yes (SVRG)	No
SAGAFW	[Reddi *et al.*, 2016]	Non-convex	Yes	Yes (SAGA)	No
AsySFW	Our	Non-convex	Yes	No	Yes
AsySVFW	Our	Non-convex	Yes	Yes (SVRG)	Yes

Table 1: Representative Frank-Wolfe algorithms to solve the constrained non-convex optimization problem (1).

problem as follows.

$$\min_{x\in\mathcal{M}} f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)\,, \tag{1}$$

where $f_i : \mathbb{R}^d \mapsto \mathbb{R}$ is a non-convex and smooth loss function, and $\mathcal{M}$ is a convex and compact set. To the best of our knowledge, there are only two works to solve the non-convex problem (1) by Frank-Wolfe algorithms. Specifically, [Lacoste-Julien, 2016] proposed the deterministic Frank-Wolfe algorithm to solve (1) and provide its convergence rate. [Reddi *et al.*, 2016] proposed vanilla stochastic Frank-Wolfe algorithm and its variants of SVRG and SAGA to solve (1), and provided their convergence rates. We also summarize these representative Frank-Wolfe algorithms to solve (1) in Table 1. As far as we know, the convergence guarantee of asynchronous stochastic Frank-Wolfe algorithms for solving the constrained non-convex optimization problem (1) is still an open question.

To address this challenging problem, in this paper, we propose our asynchronous stochastic Frank-Wolfe algorithm (AsySFW) and its variance reduction version (AsySVFW) for solving the constrained non-convex optimization problem. More importantly, we prove the fast convergence rates of AsySFW and AsySVFW in the non-convex setting. To the best of our knowledge, AsySFW and AsySVFW are the first asynchronous parallel stochastic Frank-Wolfe algorithms with convergence guarantees for solving the constrained non-convex optimization problems. The experimental results on real high-dimensional gray-scale images not only confirm the fast convergence of our algorithms, but also show a near-linear speedup on a parallel system with shared memory due to the lock-free implementation.

Contributions. The main contributions of this paper are summarized as follows.

1. We propose vanilla asynchronous stochastic Frank-Wolfe algorithm (*i.e.*, AsySFW) and prove the ergodic convergence rate of AsySFW in the non-convex setting.

2. We propose the SVRG version of AsySFW (*i.e.*, AsySVFW) and prove the ergodic convergence rate of AsySVFW in the non-convex setting.

Organization. We organize the rest of paper as follows. In Section 2, we propose our AsySFW and AsySVFW algorithms. In Section 3, we provide the convergence analysis to AsySFW and AsySVFW. In Section 4, we show the experimental results. Finally, in Section 5, we conclude the paper.

2 Asynchronous Stochastic Frank-Wolfe Algorithms

In this section, we first propose our asynchronous stochastic Frank-Wolfe algorithm (*i.e.*, AsySFW) to solve the constrained non-convex optimization problem (1), and then propose its SVRG version (*i.e.*, AsySVFW). Finally, we provide several discussions to AsySFW and AsySVFW.

2.1 AsySFW Algorithm

AsySFW repeats the following three steps concurrently for each thread without any lock. First, AsySFW reads the vector x from the shared memory to the local memory which is denoted as $\widehat{x}$. After that, AsySFW randomly chooses a mini-batch $\mathcal{B}$ of the size of b from $\{1, ..., n\}$ with equal probability, and locally computes $\widehat{v}_t \leftarrow \frac{1}{b}\sum_{i\in\mathcal{B}} \nabla f_i(\widehat{x}_t)$, and $\widehat{d}_t = \arg\min_{d\in\mathcal{M}} \langle d, \widehat{v}_t\rangle$. Finally, AsySFW updates the vector x in the shared memory by the following formulation.

$$x_{t+1} \leftarrow x_t + \gamma\left(\widehat{d}_t - \widehat{x}_t\right) \tag{2}$$

where γ is the learning rate. We summarize our AsySFW in Algorithm 1.

Algorithm 1 Asynchronous Stochastic Frank-Wolfe Algorithm (AsySFW)

Parameter: The number of iterations T, the mini-batch size b and the learning rate γ.
Initialize: Initialize $x^0 \in \mathcal{M}$.
1: *For each thread*, do:
2: **for** $t = 0, 1, 2, \cdots, T-1$ **do**
3: Randomly sample a mini-batch $\mathcal{B}$ of the size of b from $\{1, ..., n\}$ with equal probability.
4: Compute $\widehat{v}_t \leftarrow \frac{1}{b}\sum_{i\in\mathcal{B}} \nabla f_i(\widehat{x}_t)$, and $\widehat{d}_t = \arg\min_{d\in\mathcal{M}} \langle d, \widehat{v}_t\rangle$.
5: Update $x_{t+1} \leftarrow x_t + \gamma\left(\widehat{d}_t - \widehat{x}_t\right)$.
6: **end for**

2.2 AsySVFW Algorithm

We use the variance reduction technique specifically SVRG [Gu *et al.*, 2018a; Huo *et al.*, 2018] to accelerate AsySFW algorithm, which is called as AsySVFW. AsySVFW has two-layer loops. The outer layer is to parallelly compute the full gradient $\nabla f(x^s) = \frac{1}{n}\sum_{i=1}^{l} \nabla f_i(x^s)$, where the superscript s denotes the s-th outer loop. The inner layer is to parallelly and repeatedly update the vector x in the shared memory.

Specifically, all cores repeat the following steps independently and concurrently without any lock.

1. **Read:** Read the vector x from the shared memory to the local memory without reading lock. We use $\widehat{x}_t^{s+1}$ to denote its value, where the subscript t denotes the t-th inner loop.
2. **Compute:** Randomly choose a mini-batch $\mathcal{B}$ of the size of b from $\{1, ..., n\}$ with equal probability, and locally compute $\widehat{v}_t^{s+1} \leftarrow \frac{1}{b}\sum_{i\in\mathcal{B}} \nabla f_i(\widehat{x}_t^{s+1}) - \frac{1}{b}\sum_{i\in\mathcal{B}} \nabla f_i(\widetilde{x}^s) + \nabla f(\widetilde{x}^s)$, and $\widehat{d}_t^{s+1} = \arg\min_{d\in\mathcal{M}} \langle d, \widehat{v}_t^{s+1}\rangle$.
3. **Update:** Update the vector x in the shared memory as $x_{t+1}^{s+1} \leftarrow x_t^{s+1} + \gamma\left(\widehat{d}_t^{s+1} - \widehat{x}_t^{s+1}\right)$ without writing lock.

We summarize our AsySVFW in Algorithm 2. Note that the expectation of $\widehat{v}_t^{s+1}$ on $\mathcal{B}$ is equal to $\nabla f(\widehat{x}_t^{s+1})$, *i.e.*, $\mathbb{E}\widehat{v}_t^{s+1} = \nabla f(\widehat{x}_t^{s+1}) - \nabla f(\widetilde{x}^s) + \nabla f(\widetilde{x}^s) = \nabla f(\widehat{x}_t^{s+1})$. Thus, $\widehat{v}_t^{s+1}$ is called an unbiased stochastic gradient of $\nabla f(\widehat{x}_t^{s+1})$.

Algorithm 2 AsySVFW Algorithm

Parameter: The number of outer loop iterations S, the number of inner loop iterations m where $T = S \times m$, the mini-batch size b and the learning rate γ.
Initialize: Initialize $x^0 \in \mathcal{M}$.
1: **for** $s = 0, 1, 2, \cdots, S-1$ **do**
2: Set $\widetilde{x}^s \leftarrow x^s$, and parallelly compute the full gradient $\nabla f(\widetilde{x}^s) = \frac{1}{n}\sum_i^l \nabla f_i(\widetilde{x}^s)$.
3: *For each thread*, do:
4: **for** $t = 0, 1, 2, \cdots, m-1$ **do**
5: Randomly sample a mini-batch $\mathcal{B}$ of the size of b from $\{1, ..., n\}$ with equal probability.
6: Compute $\widehat{v}_t^{s+1} \leftarrow \frac{1}{b}\sum_{i\in\mathcal{B}} \nabla f_i(\widehat{x}_t^{s+1}) - \frac{1}{b}\sum_{i\in\mathcal{B}} \nabla f_i(\widetilde{x}^s) + \nabla f(\widetilde{x}^s)$.
7: Compute $\widehat{d}_t^{s+1} = \arg\min_{d\in\mathcal{M}} \langle d, \widehat{v}_t^{s+1}\rangle$.
8: Update $x_{t+1}^{s+1} \leftarrow x_t^{s+1} + \gamma\left(\widehat{d}_t^{s+1} - \widehat{x}_t^{s+1}\right)$.
9: **end for**
10: $x^{s+1} \leftarrow x_m^{s+1}$
11: **end for**

2.3 Discussions

We provide several discussions of AsySFW and AsySVFW in terms of solving the subproblem and parallel computing environment respectively as follows.

1. **Solving the Subproblem:** Different to the proximal gradient descent algorithms [Gu *et al.*, 2018b] which need to solve a quadratic subproblem in each iteration, AsySFW and AsySVFW solve a linear subproblem (lines 4 and 7 in Algorithms 1 and 2 respectively) which can be solved easier compared to the quadratic subproblem. [Jaggi, 2013] provides a lot of discussions and examples to solve the linear subproblems associated to the convex and compact sets $\mathcal{M}$ (including the trace-norm constraint [Dudik *et al.*, 2012] considered in this paper).
2. **Parallel Computing Environment:** AsySFW and AsySVFW are designed for the parallel environment with shared memory, such as multi-core processors and GPU-accelerators which can allow for the situation of concurrent write-read to happen in the process. Besides, AsySFW and AsySVFW can also work in the parallel environment with distributed memory.

3 Convergence Analysis

In this section, we first give several basic assumptions and notations, and then discuss three major difficulties for proving the convergence rates of AsySFW and AsySVFW for constrained non-convex problems. Finally, we give the convergence rates of our AsySFW and AsySVFW.

3.1 Basic Assumptions and Notations

We introduce the assumptions of Lipschitz smoothness (*i.e.*, Assumption 1) and bounded gradients (*i.e.*, Assumption 2) for the function $f_i(x)$, which are standard for non-convex optimization [Lacoste-Julien, 2016; Gu *et al.*, 2018b; Lei *et al.*, 2017; Allen-Zhu and Li, 2018].

Assumption 1 (Lipschitz smoothness) *For smooth function $f_i(x)$ ($\forall i \in \{1, \ldots, l\}$) in (1), $f_i(x)$ is called L-Lipschitz smoothness, if for all x and y, we have that:*

$$f_i(x) \leq f_i(y) + \langle \nabla f_i(y), x - y\rangle + \frac{L}{2}\|x - y\|^2 \qquad (3)$$

Assumption 2 (Bounded gradients) *For smooth function $f_i(x)$ ($\forall i \in \{1, \ldots, l\}$) in (1), the gradient $\nabla f_i(x)$ is called bounded if there exists a parameter G such that $\|\nabla f_i(x)\| \leq G$ for all $x \in \mathcal{M}$.*

Notations. To make this section easier to follow, we give a summary of the notations in the following list.

- $\Lambda = \{x : g(x) \leq c\}$ is the original constrained set.
- Ω is the expanded constrained set.
- D is the diameter of the compact convex set Ω.
- $(x)_i$ is the i-th coordinate of the vector x.
- $B_{t'}$ is a diagonal matrix with diagonal entries either 1 or 0 (if the corresponding updating has been received, the entry is 0, otherwise 1).

3.2 Difficulties of the Convergence Rate Analysis

In this subsection, we discuss the difficulties of convergence criteria, write-write conflict, inconsistent reading and beyonding the feasible set respectively when analyzing the convergence rates of AsySFW and AsySVFW for constrained non-convex problems.

Convergence Criteria: Because the objective function $f(x)$ is non-convex, our AsySFW and AsySVFW algoritms cannot guarantee to produce the global optimum solution. Thus, the closeness to the optimal solution (*i.e.*, $f(x) - f(x^*)$ and $\|x - x^*\|$) cannot be used for the convergence analysis, where x^* is the global optimal solution. For unconstrained problems, the gradient norm $\|\nabla f(x)\|$ is normally used to measure convergence, because $\|\nabla f(x)\|$ translates into convergence to a stationary point. However, this criterion cannot be used

(a) Original constrained set $\mathcal{M} = \{x : \|x\|_1 \leq 1\}$

(b) Expanded constrained set

Figure 1: An illustration of expanded constrained set associated for the set $\mathcal{M} = \{x : \|x\|_1 \leq 1\}$.

for constrained problems of the form (1). Similar to [Lacoste-Julien, 2016; Reddi *et al.*, 2016], we use the Frank-Wolfe gap (4) as the convergence criteria.

$$g(x) = \max_{d \in \mathcal{M}} \langle d - x, -\nabla f(x) \rangle \tag{4}$$

Specifically, for the constrained non-convex optimization problem (1), the gap $g(x) = 0$ if and only if x is a stationary point.

Write-Write Conflict: Because we do not use any writing lock in AsySFW and AsySVFW, all cores can update the vector x in the shared memory, which could lead to write-write conflict and make the convergence analysis more complicated.

To address this challenge, we first assume that all writes are atomic, in the sense that they will be successfully recorded in the shared memory at some point (Assumption 3).

Assumption 3 *Each coordinate of vector x in Lines 5 and 8 in Algorithms 1 and 2 respectively will be updated successfully.*

This assumption implies that the order in which these updates are recorded in the shared memory is irrelevant which is also called as the *commutativity*. Because the property of commutativity holds for our updating rules (*i.e.*, Lines 5 and 8 in Algorithms 1 and 2 respectively), the updating rule with the write-write conflict can be formalized as follows:

$$x_{t+1} \leftarrow x_t + \Delta_t, \qquad \text{where } \Delta_t = \gamma \left(\widehat{d}_t - \widehat{x}_t \right), \tag{5}$$

where x_t denotes the ideal one in the shared memory. Thus, we can analyze the convergence rates through the ideal x like [Recht *et al.*, 2011; Mania *et al.*, 2017; Gu and Huo, 2018]. Note that we use $\widehat{x}_t$ for computing Δ_t instead of x_t which reduces the frequency of reading x from the shared memory, but makes our analysis more difficult.

Inconsistent Read: Because AsySFW and AsySVFW do not use the reading and writing locks, the vector $\widehat{x}_t$ read into the local memory may be inconsistent to the vector x_t in the shared memory. We call this as *inconsistent read*. To address the challenge, we denote the delay updating iterations for x_t by a set $K(t)$ such that the relationship between x_t and $\widehat{x}_t$ can be builded as follows:

$$x_t = \widehat{x}_t + \sum_{t' \in K(t)} B_{t'} \Delta_{t'}, \tag{6}$$

where $t' \leq t - 1$. It is reasonable to assume there exists an upper bound τ to the delay of updating such that $\tau \geq t - \min\{t' | t' \in K(t)\}$ (*i.e.*, Assumption 4).

Assumption 4 (Bound of delay) *There exists an upper bound τ such that $\tau \geq t - \min\{t' | t' \in K(t)\}$ for all iterations t in AsySFW or AsySVFW.*

Beyond the Feasible Set: Due to the inconsistent writing and write-write conflict, $\widehat{x}_t$ read into the local memory could be beyond the original constrained set $\mathcal{M}$. To handle this challenge, we define an expanded constrained set Ω (see Definition 1 and Figure 1) based on $\mathcal{M}$ which is used only for our analysis. It is easy to verify that Ω is a compact convex set with $\mathcal{M} \subseteq \Omega$ and $\widehat{x}_t$ belongs to the expanded constrained set Ω.

Definition 1 (Extended constrained set) *Given a constrained set $\mathcal{M}$. Let $\mathcal{M}_i^{\min} = \min_{x \in \mathcal{M}}(x)_i$ and $\mathcal{M}_i^{\max} = \max_{x \in \mathcal{M}}(x)_i$. We define the expanded constrained set of $\mathcal{M}$ as $\Omega = \{x : (x)_i \in [\mathcal{M}_i^{\min}, \mathcal{M}_i^{\max}], \forall i = 1, \ldots, n\}$.*

3.3 Convergence Rate Analysis

After addressing the above challenges, we provide the convergence rates to our AsySFW (Theorem 1) and AsySVFW (Theorem 2). All the detailed proofs are provided in our Appendix[1].

AsySFW: We first give the convergence rate of AsySFW in Theorem 1.

Theorem 1 *Let $\gamma = \sqrt{\frac{f(x_0) - f(x^*)}{T\left(\left(2\tau + \frac{1}{2}\right) LD^2 + \tau GD\right)\beta}}$ and $b = T$. Under Assumptions 1, 2, 3 and 4, for AsySFW algorithm, we have that*

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E} g(x_t) \leq \frac{1}{\sqrt{T}} \left(2GD + \sqrt{f(x_0) - f(x^*)} \cdot \sqrt{\left(2\tau + \frac{1}{2}\right) LD^2 + \tau GD} \left(\sqrt{\beta} + \frac{1}{\sqrt{\beta}} \right) \right) \tag{7}$$

Remark 1 *To achieve $1 - \epsilon$ accuracy, AsySFW algorithm requires $O(\frac{1}{\epsilon^4})$ stochastic gradient evaluations and $\mathcal{O}(\frac{1}{\epsilon^2})$ linear optimizations. If the bound*

[1]The Appendix is available at https://drive.google.com/open?id=1FeXGqQBr6sqmO1zmeHaPx80imvOrYfXH.

τ of delay is equal to 0, the convergence rate is $\frac{D}{\sqrt{T}}\left(2G+\sqrt{\frac{L(f(x_0)-f(x^*))}{2}}\left(\sqrt{\beta}+\frac{1}{\sqrt{\beta}}\right)\right)$ *which matches the convergence rate of sequential SFW algorithm [Reddi* et al.*, 2016].*

AsySVFW: We also give the convergence rate of AsySVFW in Theorem 2.

Theorem 2 *Let* $\gamma = \sqrt{\frac{f(x_0)-f(x^*)}{T\left(2\tau LD^2+\tau GD+\frac{3LD^2}{2}\right)\beta}}$ *and* $b = m^2$*, where* $T = m \times S$*. Under Assumptions 1, 2, 3 and 4, for AsySVFW algorithm, we have that*

$$\frac{1}{T}\sum_{s=0}^{S-1}\sum_{t=0}^{m-1}\mathbb{E}g(x_t^{s+1}) \tag{8}$$
$$\leq \frac{1}{\sqrt{T}}\left(\sqrt{(f(x_0)-f(x^*))\,a}\left(\sqrt{\beta}+\frac{1}{\sqrt{\beta}}\right)\right)$$

where $a = 2\tau LD^2 + \tau GD + \frac{3LD^2}{2}$.

Remark 2 *To achieve* $1-\epsilon$ *accuracy, AsySVFW algorithm requires* $O\left(\frac{\frac{1}{\epsilon^2}}{m^2}\right)$ *exact gradient evaluations,* $O(\frac{m^2}{\epsilon^2})$ *stochastic gradient evaluations and* $O(\frac{1}{\epsilon^2})$ *linear optimizations. If the bound* τ *of delay is equal to* 0*, the convergence rate is* $\frac{D}{\sqrt{T}}\left(\sqrt{\frac{3L(f(x_0)-f(x^*))}{2}}\left(\sqrt{\beta}+\frac{1}{\sqrt{\beta}}\right)\right)$ *which matches the convergence rate of sequential SVFW algorithm [Reddi* et al.*, 2016].*

4 Experiments

In this section, we first give the experimental setup, and then present our experimental results with discussions.

4.1 Experimental Setup

Design of Experiments

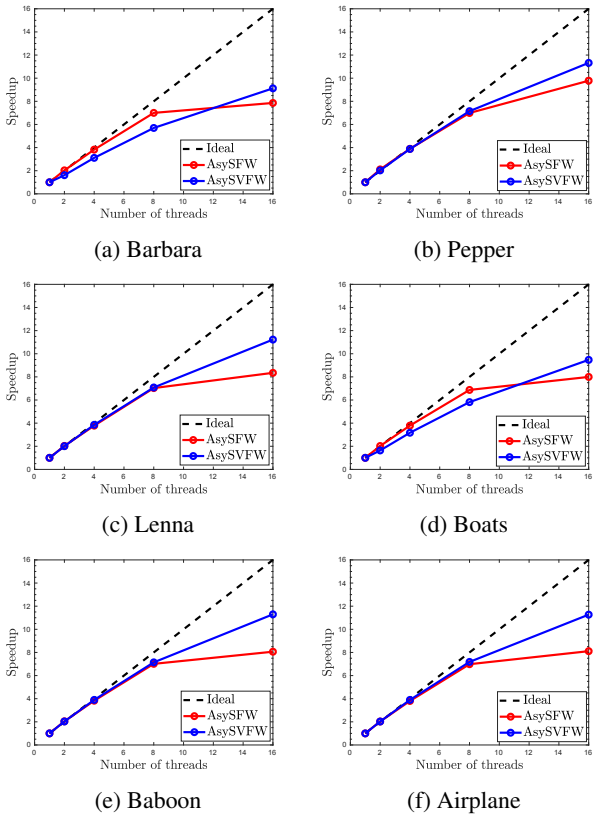
We consider the robust matrix completion problem [Yang *et al.*, 2018] in the experiments. Given an incomplete matrix Y, the robust matrix completion is to recover the incomplete matrix Y based on the correntropy-induced loss [Feng *et al.*, 2015; Chen and Wang, 2018] and trace-norm constraint [Dudik *et al.*, 2012]. Specifically, the robust matrix completion problem considers the objective function and the trace-norm constraint as follows.

$$f(X) = \frac{\sigma^2}{2}\sum_{(i,j)\in\Lambda}\left(1-\exp\left(-\frac{(X_{ij}-Y_{ij})^2}{\sigma^2}\right)\right) \tag{9}$$

$$\mathcal{M} = \{X\in\mathbb{R}^{d_1\times d_2} : \|X\|_* \leq c\} \tag{10}$$

where $d_1 \times d_2$ is the size of the matrix, $\sigma > 0$ is a scale parameter and Λ is the set of all observed entries.

In the experiments, we test the suboptimality (difference between the objective function and its closest minimum value) of AsySFW and AsySVFW with different numbers of threads to verify the fast convergence of AsySFW and AsySVFW. We also observe the speedup of AsySFW and AsySVFW on a parallel system with shared memory.



(a) Barbara (b) Pepper (c) Lenna (d) Boats (e) Baboon (f) Airplane

Figure 2: The speedup of our AsyFW and AsySVFW algorithms on the six real gray-scale images.

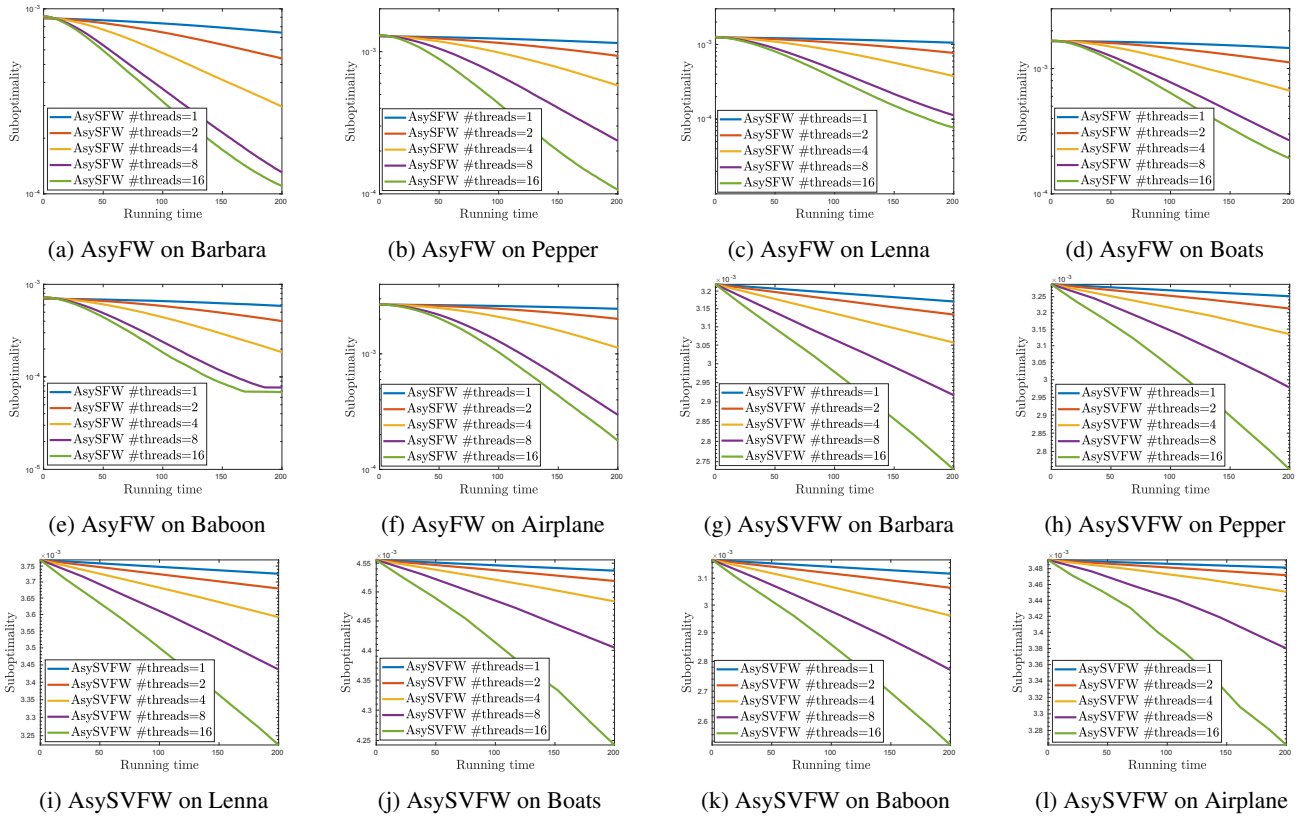
Implementation Details

Our experiments are performed on a 32-core two-socket Intel Xeon E5-2699 machine where each socket has 16 cores. We implement our AsySFW and AsySVFW using C++, where the shared memory parallel computation is handled via OpenMP. We focus on the convergence of AsySFW and AsySVFW so that the tuning of the parameters σ and c is less important. Thus, the parameters σ and c are fixed at 0.15 and 500 respectively. In addition, we set the learning rate $\gamma = 0.0001$, the mini-batch size $b = 500$, the inner loop size of AsySVFW $m = 50$. We choose $X = \frac{1}{2}^{\alpha}Y$ as the initial solution for AsySFW and AsySVFW, where α is the smallest value in $\{1, 2, \ldots, 10\}$ such that $\|X\|_* \leq c$.

Datasets

[Yang *et al.*, 2018] conducted an image recovery experiment based on real gray-scale images[2]. In our experiments, we used the robust matrix completion problem defined in (9)-(10) to achieve image recovery. Specifically, we select six images, *i.e.*, Airplane, Baboon, Barbara, Boats, Lenna and Pepper from the same image source. These images are originally with pixels of the size of 728×642, which means that $d_1 = 728$ and $d_2 = 642$. We miss 30% pixels randomly and let it be the incomplete matrix Y. Thus, the size of the set

[2]The real gray-scale images are available at https://homepages.cae.wisc.edu/~ece533/images/



(a) AsyFW on Barbara (b) AsyFW on Pepper (c) AsyFW on Lenna (d) AsyFW on Boats

(e) AsyFW on Baboon (f) AsyFW on Airplane (g) AsySVFW on Barbara (h) AsySVFW on Pepper

(i) AsySVFW on Lenna (j) AsySVFW on Boats (k) AsySVFW on Baboon (l) AsySVFW on Airplane

Figure 3: The convergence of the suboptimality *vs.* running time of our AsyFW and AsySVFW algorithms on the six real gray-scale images. (a)-(f) The results of AsyFW. (g)-(l) The results of AsySVFW.

Λ of all observed entries is $327,163$. In the experiments, all results are the average of 10 trials.

4.2 Results and Discussions

Figure 3 illustrates the lines of the suboptimality *vs.* running time of our AsyFW and AsySVFW algorithms with 1, 2, 4, 8 and 16 cores on the six real gray-scale images. The results show that our AsyFW and AsySVFW algorithms with different numbers of threads can always converge to a stationary (empirically local) solution. More importantly, the results verify that AsyFW and AsySVFW with more threads converge faster than the ones with less threads. To sum up, the results in Figure 3 confirm that our AsyFW and AsySVFW algorithms converge to a stationary solution with fast convergence.

Figure 2 presents the speedup results of AsyFW and AsySVFW with 1, 2, 4, 8 and 16 cores on the six real gray-scale images. The results show that AsyFW and AsySVFW can have a near-linear speedup on a parallel system with shared memory. This is because we do not use any lock in the implementations of AsyFW and AsySVFW which keeps all computational resources busy all the time.

5 Conclusion

The convergence guarantee of asynchronous stochastic Frank-Wolfe algorithms for solving the constrained non-convex optimization problem (1) is still an open question. To address this challenging problem, in this paper, we propose an asynchronous stochastic Frank-Wolfe algorithm (*i.e.*, AsySFW) and its variance reduction version (*i.e.*, AsySVFW) for solving the constrained non-convex optimization problems. We prove the fast convergence rates of AsySFW and AsySVFW in the non-convex setting. To the best of our knowledge, AsySFW and AsySVFW are the first asynchronous parallel stochastic algorithms with convergence guarantees for solving the constrained non-convex optimization problems. The experimental results on real high-dimensional gray-scale images not only confirm the fast convergence of our algorithms, but also show a near-linear speedup on a parallel system with shared memory due to the lock-free implementation.

Acknowledgments

H.H. was partially supported by U.S. NSF IIS 1836945, IIS 1836938, DBI 1836866, IIS 1845666, IIS 1852606, IIS 1838627, IIS 1837956. B.G. was partially supported by the National Natural Science Foundation of China (No: 61573191), and the Natural Science Foundation (No. BK20161534), Six talent peaks project (No. XYDXX-042) in Jiangsu Province.

References

[Allen-Zhu and Li, 2018] Zeyuan Allen-Zhu and Yuanzhi Li. Neon2: Finding local minima via first-order oracles. In *Advances in Neural Information Processing Systems*, pages 3720–3730, 2018.

[Chen and Wang, 2018] Hong Chen and Yulong Wang. Kernel-based sparse regression with the correntropy-induced loss. *Applied and Computational Harmonic Analysis*, 44(1):144–164, 2018.

[Dudik *et al.*, 2012] Miroslav Dudik, Zaid Harchaoui, and Jérôme Malick. Lifted coordinate descent for learning with trace-norm regularization. In *Artificial Intelligence and Statistics*, pages 327–336, 2012.

[Feng *et al.*, 2015] Yunlong Feng, Xiaolin Huang, Lei Shi, Yuning Yang, and Johan AK Suykens. Learning with the maximum correntropy criterion induced losses for regression. *Journal of Machine Learning Research*, 16:993–1034, 2015.

[Frank and Wolfe, 1956] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Research Logistics (NRL)*, 3(1-2):95–110, 1956.

[Garber and Hazan, 2015] Dan Garber and Elad Hazan. Faster rates for the frank-wolfe method over strongly-convex sets. In *ICML*, volume 15, pages 541–549, 2015.

[Goldfarb *et al.*, 2017] Donald Goldfarb, Garud Iyengar, and Chaoxu Zhou. Linear convergence of stochastic frank wolfe variants. *arXiv preprint arXiv:1703.07269*, 2017.

[Gu and Huo, 2018] Bin Gu and Zhouyuan Huo. Asynchronous doubly stochastic group regularized learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS 2018)*, 2018.

[Gu *et al.*, 2017] Bin Gu, Guodong Liu, and Heng Huang. Groups-keeping solution path algorithm for sparse regression with automatic feature grouping. In *KDD*, pages 185–193. ACM, 2017.

[Gu *et al.*, 2018a] Bin Gu, Zhouyuan Huo, Cheng Deng, and Heng Huang. Faster derivative-free stochastic algorithm for shared memory machines. In *International Conference on Machine Learning*, pages 1807–1816, 2018.

[Gu *et al.*, 2018b] Bin Gu, De Wang, Zhouyuan Huo, and Heng Huang. Inexact proximal gradient methods for non-convex and non-smooth optimization. In *AAAI*, pages 3093–3100, 2018.

[Hazan and Luo, 2016] Elad Hazan and Haipeng Luo. Variance-reduced and projection-free stochastic optimization. In *International Conference on Machine Learning*, pages 1263–1271, 2016.

[Hsieh *et al.*, 2015] Cho-Jui Hsieh, Nagarajan Natarajan, and Inderjit S Dhillon. Pu learning for matrix completion. In *ICML*, pages 2445–2453, 2015.

[Huo *et al.*, 2018] Zhouyuan Huo, Bin Gu, Ji Liu, and Heng Huang. Accelerated method for stochastic composition optimization with nonsmooth regularization. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[Jaggi, 2013] Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *ICML (1)*, pages 427–435, 2013.

[Lacoste-Julien and Jaggi, 2015] 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.

[Lacoste-Julien *et al.*, 2013] Simon Lacoste-Julien, Martin Jaggi, Mark Schmidt, and Patrick Pletscher. Block-coordinate frank-wolfe optimization for structural svms. In *ICML 2013 International Conference on Machine Learning*, pages 53–61, 2013.

[Lacoste-Julien, 2016] Simon Lacoste-Julien. Convergence rate of frank-wolfe for non-convex objectives. *arXiv preprint arXiv:1607.00345*, 2016.

[Lan, 2013] Guanghui Lan. The complexity of large-scale convex programming under a linear optimization oracle. *arXiv preprint arXiv:1309.5550*, 2013.

[Lei *et al.*, 2017] Lihua Lei, Cheng Ju, Jianbo Chen, and Michael I Jordan. Non-convex finite-sum optimization via scsg methods. In *Advances in Neural Information Processing Systems*, pages 2348–2358, 2017.

[Mania *et al.*, 2017] Horia Mania, Xinghao Pan, Dimitris Papailiopoulos, Benjamin Recht, Kannan Ramchandran, and Michael I Jordan. Perturbed iterate analysis for asynchronous stochastic optimization. *SIAM Journal on Optimization*, 27(4):2202–2229, 2017.

[Moharrer and Ioannidis, 2017] Armin Moharrer and Stratis Ioannidis. Distributing frank-wolfe via map-reduce. In *2017 IEEE International Conference on Data Mining, ICDM 2017, New Orleans, LA, USA, November 18-21, 2017*, pages 317–326, 2017.

[Recht *et al.*, 2011] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in neural information processing systems*, pages 693–701, 2011.

[Reddi *et al.*, 2016] Sashank J Reddi, Suvrit Sra, Barnabás Póczos, and Alex Smola. Stochastic frank-wolfe methods for nonconvex optimization. In *Communication, Control, and Computing (Allerton), 2016 54th Annual Allerton Conference on*, pages 1244–1251. IEEE, 2016.

[Wang *et al.*, 2016] Yu-Xiang Wang, Veeranjaneyulu 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.

[Yang *et al.*, 2018] Yuning Yang, Yunlong Feng, and Johan AK Suykens. Correntropy based matrix completion. *Entropy*, 20(3):171, 2018.

[Zhang *et al.*, 2017] Wenpeng Zhang, Peilin Zhao, Wenwu Zhu, Steven CH Hoi, and Tong Zhang. Projection-free distributed online learning in networks. In *International Conference on Machine Learning*, pages 4054–4062, 2017.