Asynchronous Stochastic Frank-Wolfe Algorithms for Non-Convex Optimization

Bin Gu¹, Wenhan Xian² and Heng Huang²,¹ *

¹JD Finance America Corporation
²Department 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 have 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 \( \ell_1 \) norm and pairwise \( \ell_\infty \) 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.
problem as follows.

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

where \( f_i : \mathbb{R}^d \to \mathbb{R} \) 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 (1). 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.

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.

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.

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

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Reference</th>
<th>Function ( f(x) )</th>
<th>Stochastic gradient</th>
<th>Variance reduced</th>
<th>Asynchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frank-Wolfe</td>
<td>[Lacoste-Julien, 2016]</td>
<td>Non-convex</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SFW</td>
<td>[Reddi et al., 2016]</td>
<td>Non-convex</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SVFW</td>
<td>[Reddi et al., 2016]</td>
<td>Non-convex</td>
<td>Yes</td>
<td>Yes (SVRG)</td>
<td>No</td>
</tr>
<tr>
<td>SAGAFW</td>
<td>[Reddi et al., 2016]</td>
<td>Non-convex</td>
<td>Yes</td>
<td>Yes (SAGA)</td>
<td>No</td>
</tr>
<tr>
<td>AsySFW</td>
<td>Our</td>
<td>Non-convex</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>AsySVFW</td>
<td>Our</td>
<td>Non-convex</td>
<td>Yes</td>
<td>Yes (SVRG)</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### 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 \( \tilde{x} \). After that, AsySFW randomly chooses a mini-batch \( B \) of the size of \( b \) from \( \{1, \ldots, n\} \) with equal probability.

1. For each thread, do:
2. for \( t = 0, 1, 2, \ldots, T - 1 \) do
3. Randomly sample a mini-batch \( B \) of the size of \( b \) from \( \{1, \ldots, n\} \) with equal probability.
4. Compute \( \tilde{v}_t \leftarrow \frac{1}{b} \sum_{i \in B} \nabla f_i(\tilde{x}_t) \), and \( \tilde{d}_t = \arg \min_{d \in \mathcal{M}} \langle d, \tilde{v}_t \rangle \).
5. Update \( x_{t+1} \leftarrow x_t + \gamma (\tilde{d}_t - \tilde{x}_t) \).

#### 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 \( \nabla f(x^s) = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(x^s) \), 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.
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 $\gamma$.

Initialize: Initialize $x^0 \in M$.
1. for $s = 0, 1, 2, \ldots, S - 1$ do
2. Set $x^s \leftarrow x^s$, and parallelly compute the full gradient $\nabla \ell(x^s) = \frac{1}{b} \sum_{i \in B} \nabla f_i(x^s)$.
3. For each thread, do:
4. for $t = 0, 1, 2, \ldots, m - 1$ do
5. Randomly sample a mini-batch $B$ of the size of $b$ from $\{1, \ldots, n\}$ with equal probability.
6. Compute $\hat{v}_t^{s+1} \leftarrow \frac{1}{b} \sum_{i \in B} \nabla f_i(x^s) - \frac{1}{b} \sum_{i \in B} \nabla f_i(x^{s+1}) - \frac{1}{b} \sum_{i \in B} \nabla f_i(x^{s+1}) + \nabla f(x^s)$.
7. Compute $d_t^{s+1} = \arg \min_{d \in M} \langle d, \hat{v}_t^{s+1} \rangle$.
8. Update $x_t^{s+1} \leftarrow x_t^s + \gamma (d_t^{s+1} - x_t^{s+1})$.
9. end for
10. $x^{s+1} \leftarrow x^m$
11. end for

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

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

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 $M$ (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 (I), $f_i(x)$ is called $L$-Lipschitz smoothness, if for all $x$ and $y$, we have that:

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

Assumption 2 (Bounded gradients) For smooth function $f_i(x)$ ($\forall i \in \{1, \ldots, l\}$) in (I), 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 $x \in M$.

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.
- $\Omega$ is the expanded constrained set.
- $D$ is the diameter of the compact convex set $\Omega$.
- $(x)_i$ is the $i$-th coordinate of the vector $x$.

$B_{\Omega}$ 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 beyond 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 algorithms cannot guarantee to produce the optimal solution. Thus, the closeness to the optimal solution (i.e., $f(x) - f(x^*)$) 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 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.

\[ g(x) = \max_{d \in M} \langle d - x, -\nabla f(x) \rangle \]

(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:

\[ x_{t+1} \leftarrow x_t + \Delta_t, \quad \text{where} \quad \Delta_t = \gamma \left( d_t - \hat{x}_t \right), \]

(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 \( \hat{x}_t \) for computing \( \Delta_t \) 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 lock, the vector \( \hat{x}_t \) 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 \( \hat{x}_t \) can be builded as follows:

\[ x_t = \hat{x}_t + \sum_{t' \in K(t)} B_{t'} \Delta_{t'}, \]

(6)

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

**Assumption 4** There exists an upper bound \( \tau \) 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, \( \hat{x}_t \) read into the local memory could be beyond the original constrained set \( M \). To handle this challenge, we define an expanded constrained set \( \Omega \) (see Definition 1 and Figure 1) based on \( M \) which is used only for our analysis. It is easy to verify that \( \Omega \) is a compact convex set with \( M \subseteq \Omega \) and \( \hat{x}_t \) belongs to the expanded constrained set \( \Omega \).

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

### 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.

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

**Theorem 1** Let \( \gamma = \frac{\sqrt{f(x_0) - f(x^*')}}{T((2\tau + \frac{1}{2})LD^2 + \tau GD)} \) and \( b = T \).

Under Assumptions 1, 2, 3 and 4, for AsySFW algorithm, we have that

\[ \frac{1}{T} \sum_{i=0}^{T-1} E g(x_i) \leq \frac{1}{\sqrt{T}} \left( 2GD + \sqrt{f(x_0) - f(x^*)} \right) \left( 2\tau + \frac{1}{2} \right) LD^2 + \tau GD \left( \sqrt{\beta} + \frac{1}{\sqrt{\beta}} \right) \]  

(7)

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

\[ \]
\( \tau \) of delay is equal to 0, the convergence rate is
\[
\frac{D}{\sqrt{T}} \left( 2G + \sqrt{\frac{L(f(x_0) - f(x^*))}{\sigma^2}} \right) \sqrt{\beta + \frac{1}{\sqrt{\beta}}} \] 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 \( \gamma = \sqrt{\frac{f(x_0) - f(x^*)}{T (2\tau LD^2 + \tau GD + \frac{3LD^2}{2}) \beta}} \) and \( b = m^2 \), where \( T = m \times S \). Under Assumptions 1, 2, 3 and 4, for AsySVFW algorithm, we have that
\[
\frac{1}{T} \sum_{s=0}^{S-1} \sum_{i=0}^{m-1} \mathbb{E}g(x_i^{s+1}) \leq \frac{1}{\sqrt{T}} \left( \sqrt{f(x_0) - f(x^*)} \right) a \left( \sqrt{\beta + \frac{1}{\sqrt{\beta}}} \right)
\]
where \( a = 2\tau LD^2 + \tau GD + \frac{3LD^2}{2} \).

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

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

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

where \( d_1 \times d_2 \) is the size of the matrix, \( \sigma > 0 \) is a scale parameter and \( \Lambda \) 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.

**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 \( \sigma \) and \( c \) is less important. Thus, the parameters \( \sigma \) 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 \( X = \frac{1}{\alpha} Y \) as the initial solution for AsySFW and AsySVFW, where \( \alpha \) 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, Lena and Pepper from the same image source. These images are originally with pixels of the size of 728 \( \times \) 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/
\[ \sum_{i=1}^{n} x_{ij} = 327, 163. \] 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


