Elsevier Editorial System(tm) for Neurocomputing

Manuscript Draft

Manuscript Number:

Title: Distributed and Asynchronous Stochastic Gradient Descent with Variance Reduction

Article Type: Full Length Article (LS)

Keywords: Stochastic gradient descent; Variance reduction; Asynchronous communication protocol; Distributed machine learning algorithms

Corresponding Author: Dr. Yuewei Ming,

Corresponding Author's Institution:

First Author: Yuewei Ming

Order of Authors: Yuewei Ming; Yawei Zhao; Chengkun Wu; Kuan Li; Jianping Yin

Abstract: Stochastic Gradient Descent (SGD) with variance reduction techniques has been proved powerful to train parameters of various machine learning models. However, it cannot support the distributed systems trivially due to the intrinsic design. Although conventional studies such as PetuumSGD perform well for distributed machine learning tasks, they mainly focus on the optimization of the communication protocol, which do not exploit the potential benefits of a specific machine learning algorithm. We analyze the asynchronous communication protocol in PetuumSGD, and propose a distributed version of variance reduced SGD named \emph{DisSVRG}. Specifically, DisSVRG adopts a variance reduction technique to update the parameters of a model, and then shares those newly learned parameters across nodes in a cluster by using the asynchronous communication protocol. Besides, we accelerate DisSVRG by using the learning rate with an acceleration factor. An adaptive sampling strategy is proposed in DisSVRG, which greatly reduces wait time during the iterations, and speeds up the convergence of DisSVRG. Extensive empirical studies verify that DisSVRG converges faster than the state-of-the-art variants of SGD, and gains almost linear speedup in a cluster.

Distributed and Asynchronous Stochastic Gradient Descent with Variance Reduction *

Yuewei Ming[1], Yawei Zhao[1], Chengkun Wu[1], Kuan Li[1], and Jianping Yin[2]

[1] College of Computer, National University of Defense Technology, Changsha 410073, China
[2] State Key Laboratory of High Performance Computing, National University of Defense Technology

Abstract. Stochastic Gradient Descent (SGD) with variance reduction techniques has been proved powerful to train parameters of various machine learning models. However, it cannot support the distributed systems trivially due to the intrinsic design. Although conventional studies such as PetuumSGD perform well for distributed machine learning tasks, they mainly focus on the optimization of the communication protocol, which do not exploit the potential benefits of a specific machine learning algorithm. We analyze the asynchronous communication protocol in PetuumSGD, and propose a distributed version of variance reduced SGD named *DisSVRG*. Specifically, DisSVRG adopts a variance reduction technique to update the parameters of a model, and then shares those newly learned parameters across nodes in a cluster by using the asynchronous communication protocol. Besides, we accelerate DisSVRG by using the learning rate with an acceleration factor. An adaptive sampling strategy is proposed in DisSVRG, which greatly reduces wait time during the iterations, and speeds up the convergence of DisSVRG. Extensive empirical studies verify that DisSVRG converges faster than the state-of-the-art variants of SGD, and gains almost linear speedup in a cluster.

Keywords: Stochastic gradient descent, Variance reduction, Asynchronous communication protocol, Distributed machine learning algorithms

* We thank the National Supercomputing Center in Changsha for providing *Tianhe-1* supercomputer as our experiment platform. This work was supported by the National Natural Science Foundation of China (Project NO. 61672528, 61170287, 61232016, 61303189 and 31501073).

Distributed and Asynchronous Stochastic Gradient Descent with Variance Reduction *

Yuewei Ming[1], Yawei Zhao[1], Chengkun Wu[1], Kuan Li[1], and Jianping Yin[2]

[1] College of Computer, National University of Defense Technology, Changsha 410073, China
[2] State Key Laboratory of High Performance Computing, National University of Defense Technology

Abstract. Stochastic Gradient Descent (SGD) with variance reduction techniques has been proved powerful to train parameters of various machine learning models. However, it cannot support the distributed systems trivially due to the intrinsic design. Although conventional studies such as PetuumSGD perform well for distributed machine learning tasks, they mainly focus on the optimization of the communication protocol, which do not exploit the potential benefits of a specific machine learning algorithm. We analyze the asynchronous communication protocol in PetuumSGD, and propose a distributed version of variance reduced SGD named *DisSVRG*. Specifically, DisSVRG adopts a variance reduction technique to update the parameters of a model, and then shares those newly learned parameters across nodes in a cluster by using the asynchronous communication protocol. Besides, we accelerate DisSVRG by using the learning rate with an acceleration factor. An adaptive sampling strategy is proposed in DisSVRG, which greatly reduces wait time during the iterations, and speeds up the convergence of DisSVRG. Extensive empirical studies verify that DisSVRG converges faster than the state-of-the-art variants of SGD, and gains almost linear speedup in a cluster.

Keywords: Stochastic gradient descent, Variance reduction, Asynchronous communication protocol, Distributed machine learning algorithms

1 Introduction

Machine learning based applications such as image recognition [2], speech recognition [4] and text processing [3] proliferate in the era of Big Data. Those underlying machine learning models are usually complex and big with a large number of parameters which can be trained or learned from a large amount of training data. For example, it is possible to train a large scale deep neural network which consists of millions or even billions of parameters by feeding it with terabytes of training data. Furthermore, it is worth noting that most of the machine learning algorithms are iterative convergent, which means those algorithms need many rounds of iterative calculations to update the parameters of their models. Considering the complexity of the underlying model, the huge size of

* We thank the National Supercomputing Center in Changsha for providing *Tianhe-1* supercomputer as our experiment platform. This work was supported by the National Natural Science Foundation of China (Project NO. 61672528, 61170287, 61232016, 61303189 and 31501073).

the training data and the massive amount of computation, an efficient training method is vitally important for performing a large scale learning task.

Many machine learning algorithms can be described the optimization problem like (1).

$$\min F(\omega), \quad F(\omega) = \frac{1}{n}\sum_{i=1}^{n} f_i(\omega) \tag{1}$$

, and the update rule is:

$$\omega_t = \begin{cases} \omega_{t-1} - \nabla F(\omega_{t-1}) \ \backslash\backslash \textit{the gradient descent} \\ \omega_{t-1} - \nabla f_i(\omega_{t-1}) \ \backslash\backslash \textit{the stochastic gradient descent} \end{cases} \tag{2}$$

. Here, ∇ stands for the derivation operation. $F(\omega)$ is generally called the loss function. ω represents the parameters of a model, that is, the parameters needed to be updated during the iterations. n means the size of training data and t represents the tth iteration. The loss function $F(\omega)$ can be minimized by updating the parameters iteratively, which is called the learning process. Conventionally, the gradient descent method is used to compute the global average gradient, i.e., $\nabla F(\omega_{t-1})$, and then uses it to update the parameters during an iteration. Since$\nabla F(\omega_{t-1})$ needs n derivations, which are time-consuming, the gradient descent method is not practical for a large scale learning task. An alternative approach is Stochastic Gradient Descent (SGD) and its variants. SGD randomly samples an instance from the training data, and then use it to compute the local gradient, i.e., $\nabla f_i(\omega_{t-1})$, instead of the global average gradient. The parameters are thus updated by using the local gradient. Since $\nabla f_i(\omega_{t-1})$ merely needs one derivation for the local gradient during an iteration, it is efficient for the large scale learning tasks. However, since f_i is randomly sampled, the variance exists between the local gradient $\nabla f_i(\omega_{t-1})$ and the global average gradient $\nabla F(\omega_{t-1})$, which slows the convergence of the loss function i.e., $F(\omega)$, of a machine learning algorithm. Specifically, when the parameters are close to the optimal state, it is difficult to decrease the loss function due to the variance. Conventionally, the variance is reduced by using a decaying learning rate to update the parameters. That is, the value of the learning rate is decreased when the iteration proceeds. Although the variance can be reduced by the decaying learning rate, the small learning rate unavoidably leads to the slow convergence.

Recently, the SGD and its variants have been widely used to train the parameters of a model in distributed memory systems [7,9,14]. Those versions of SGD generally train and update parameters in a parameter-server system by using a cluster. The nodes in the parameter-server system are categorized into servers and workers. The underlying calculations of the gradients are conducted by workers. Those updates will be then pushed to servers, and be aggregated on servers for updating the global parameters. Finally, those newly learned global parameters will be shared with workers. Since there exists much communication between workers and servers, all the distributed versions of SGD like PetuumSGD focus on the optimization of communication [14]. In specific, PetuumSGD has proposed the asynchronous communication protocol denoted by Staleness Synchronous Protocol (SSP) to conduct communication across nodes. SSP outperforms the Bulk Synchronous Protocol (BSP) employed on Hadoop or Spark significantly. However, since SSP is designed for the general iterative convergent machine learning algorithms, it does not exploit the potential benefits of SGD to accelerate the iterative

calculations. For example, PetuumSGD updates the parameters by using a decaying learning rate. The learning rate in PetuumSGD in the current iteration denoted by η will become 0.95η in the next iteration. When the parameters are close to the optimal state, the loss function is difficult to be decreased due to the extremely small learning rate. In a nutshell, even though PetuumSGD adopts SSP to optimize the communication between workers and servers, the decay learning rate slows the convergence.

Meanwhile, a new technique of the variance reduction is proposed to speed up the convergence of SGD [8,13,17]. Such variance reduction technique reduces the variance of SGD, and keeps SGD converging at a constant rate. However, those versions of SGD are designed to be used in one node instead of a cluster. When the amount of parameters or the size of training data is extremely huge so that they cannot be stored in a node, the underlying variance reduction technique will not work. For instance, a deep network may have billions or even trillions of parameters, which cannot be stored in a node. Therefore, such versions of SGD are incapable of performing the training for a large scale learning task, or handling a large amount of training data.

In this paper, We design a distributed and asynchronous version of variance reduced SGD denoted by DisSVRG for large-scale machine learning tasks. It is noting that DisSVRG adopts the asynchronous communication protocol, i.e., Staleness Synchronous Protocol (SSP). In order to obtain a fast convergence, DisSVRG is accelerated by using a learning rate with an acceleration factor. It is unavoidable that the fast workers will spend much time on waiting for the slow workers when performing iterative calculations in a cluster, which is also known as the "straggler problem". The straggler problem wastes much time for the fast workers, thus leads to the slow convergence of a machine learning algorithm. In order to reduce the wait time, we propose an adaptive sampling strategy to alleviate the straggler problem. Specifically, we dynamically adjust the random sampling strategy during the iterations. When the worker is faster than other workers, it will sample more instances for the next iteration, which will take the faster worker more time to compute the local gradient. Thus, the slow workers have chance to catch up with the faster works, and the wait time is reduced significantly. Finally, we conduct empirical studies on the HPC cluster of the *Tianhe-1* supercomputer which is located in the National Supercomputing Center in Changsha. The performance evaluation verifies that DisSVRG outperforms the state-of-the-art version of SGD, and obtains approximately linear speedup in the cluster.

The rest of this paper is organized as follows. Section 2 outlines the related work of our work. Section 3 presents the assumptions and the overview of DisSVRG. Section 4 illustrates the details of DisSVRG. Section 5 highlights the optimization of DisSVRG. Section 6 discusses the major difference between our work and the previous studies. Section 7 shows the performance evaluation. Section 8 concludes the paper.

2 Related work

With the proliferation of data, a complex and big model can be learned by feeding it with a huge size of training data. An approach for such a large scale learning is to use a cluster to train the parameters of the underlying model [7,9,14]. Dean et al. propose a version of asynchronous and distributed SGD denoted by *DownpourSGD* in a parameter-server

system. DownpourSGD uses fully asynchronous communication protocol to conduct communication across nodes, which cannot guarantee the convergence. To increase the robustness of DownpourSGD, the Adagrad adaptive learning rate procedure is adopted [1]. However, the adopted learning rate is decaying with the iterations, and thus leads to the slow convergence. Besides, Li et al. and Xing et al. propose an implementation of the parameter-server system respectively. The similar asynchronous communication protocol denoted by SSP is adopted in both of their systems to share the updates of the parameters across nodes. SSP has been proved powerful in both theory and practice. However, SGD in [9] uses a constant learning rate without variance reduction technique, leading to slow convergence caused by the variance. SGD in [14], i.e.,PetuumSGD, adopts a decaying learning rate to reduce the variance, giving rise to slow convergence when the learning rate becomes small. Our version of the asynchronous and distributed SGD, i.e., DisSVRG, adopts SSP to implement the communication across nodes, and uses the variance reduction technique to reduce variance as well.

The latest variance reduced SGD denoted by *SVRG* is adopted in [8], which is effective to reduce the variance of SGD. However, SVRG is a serial version, and designed to be run on a single node. Thus, it is not suitable for a large scale learning task. Asynchronous and parallel versions of SVRG partially solve this problem [11–13, 17]. Such SVRG versions use the lock-free method to update the parameters in parallel for multiple learning threads in a node. However, the design of such work targets at a multicore system on a single node. When the size of training data or the number of parameters is huge so that they can not be stored in one node, SVRG and those variants fail immediately. Our proposed variance reduced SGD is designed for the large scale learning tasks in a cluster.

3 Assumptions and Overview

3.1 Assumptions

Conventionally, each $f_i(\omega)$ in the optimization problem 1 is a L-smooth function. That is, $\exists$ a non-negative L, and $\forall$ a and b, the following inequality holds.

$$f_i(a) \leq f_i(b) + \nabla f_i(b)^{\mathrm{T}}(a-b) + \frac{L}{2} \parallel a-b \parallel^2 \tag{3}$$

Besides, the loss function $F(\omega)$ in the optimization problem 1 is γ-strongly convex, which means that $\exists$ a non-negative γ, and $\forall$ a and b, the following inequality holds.

$$F(a) \geq F(b) + \nabla F(b)^{\mathrm{T}}(a-b) + \frac{\gamma}{2} \parallel a-b \parallel^2 \tag{4}$$

It is worth noting that these assumptions are not tight, and many machine learning algorithms are under these assumptions including regression and classification. These assumptions will be exploited to optimize our version of SGD in Section 5.

Algorithm 1 DisSVRG

1: Initialize $\tilde{\omega}^0$. \\ Pull the global parameters by all trainers from the servers.
2: **for** $s = 1, 2, ...$ **do** \\ Asynchronously update the parameters by all trainers.
3: $\tilde{\omega} = \tilde{\omega}^{s-1}$.
4: $\mu = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\omega_i)$.
5: $\omega_0 = \tilde{\omega}$.
6: **for** $t = 1, 2, ...m$ **do**
7: randomly sample $i_t \in \{0, 1, 2, ..., m\}$.
8: $v_t = \nabla f_{i_t}(\omega_{t-1}) - \nabla f_{i_t}(\tilde{\omega}) + \mu$. \\Variance reduced gradient.
9: $\omega_t = \omega_{t-1} - \eta v_t$. \\Update the parameters with variance reduction gradient.
10: $\tilde{\omega}_s = \omega_m$. \\Push the newly learned parameters to the servers, and aggregate them with the global parameters.

3.2 Overview

Our distributed and asynchronous SGD denoted by DisSVRG is presented in Algorithm 1. DisSVRG consists of three ingredients: the epochs of iterations (the outer *for* loop at Line 2), the random sampling strategy of the training data (the inner *for* loop at Line 6) and the update rule of the parameters (Lines 8-9). DisSVRG is launched by the servers, and all the workers will be informed by message passing. Once a worker receives the message from a server, it pulls a copy of the global parameters from a server, and begins conducting the calculations during the epoch. The random sampling strategy is conducted by the workers. When a worker randomly samples an instance from the training data, it computes the variance reduced gradient (Line 8), and updates the local parameters with the local gradient (Line 9). When the local parameters have been updated, the newly learned parameters will be sent to a server. The inter-node communication is conducted by the asynchronous communication protocol, i.e., SSP. The server receives these learned parameters and aggregates them. The aggregated parameters are the latest global parameters which will be sent to workers for the next iteration. The details of communication across nodes and aggregation of parameters will be demonstrated in Section 4.

4 System implementation

DisSVRG is designed for the distributed memory systems where the nodes can be categorized into workers and servers. Every worker caches a copy of the parameters which is called the *local parameters*; while all the servers maintains one copy of the parameters which is called the *global parameters*. The global parameters will be pulled by a worker and be used as the initial parameters for an iteration. Meanwhile, such initial parameters will replace the stale local parameters on the worker and become its new local parameters.

Server: The servers control the iterations by using the asynchronous communication protocol, i.e., SSP. Since the runtime environment of nodes in a cluster varies a lot, the time overhead of an epoch for different workers varies. Therefore, there are

Algorithm 2 Server

Initialize $\tilde{\omega}^0$.
while true **do**
  if receive a pull request from the trainer p. **then**
    $e_p = p.epoch$.
    if all the trainers have finished $(e_p - \tau)$th epoch. **then**
      send a copy of the global parameters.
  if receive a push request from the trainer p. **then**
    receive the newly learned parameters from the trainer p.
    aggregate the newly learned parameters with the global parameters on the server.

Algorithm 3 Trainer

while true **do**
  send a pull request to a server.
  while true **do**
    if receive a copy of the global parameters from the server. **then**
      cache it as the new local parameters, i.e., $\tilde{\omega}$.
      $\mu = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\tilde{\omega})$.
      for $t = 0, 1, 2, ..., m$ **do**
        randomly sample a non-negative number i with $i \in \{0, 1, 2, ..., n\}$.
        $v_t = \nabla f_{i_t}(\omega_{t-1}) - \nabla f_{i_t}(\tilde{\omega}) + \mu$.
        $\omega_t = \omega_{t-1} - \eta v_t$.
      send ω_m to a server.

fast and slow workers which conduct an epoch fast and slow respectively. It is worth noting that DisSVRG may not converge if all the workers update parameters in a fully asynchronous way. Therefore, we set a delay bound, i.e., τ. The delay τ is used to synchronize all the workers. For instance, when the fastest worker finishes the tth iteration, and the slowest worker does not finish the $(t - \tau)$th iteration, the fastest worker will be forced to stop and wait for the slowest one. In specific, the fastest worker can not pull a copy of the global parameters from the servers, and thus has to wait for the slowest worker. Until the slowest worker finishes the $(t - \tau)$th iteration, the fastest worker will re-start to conduct the iterations. When a server receives the newly learned parameters from a worker, it will aggregate them with the global parameters on the server. After that, the latest parameters on the server will be pulled by the worker for the next iteration. The details are illustrated in Algorithm 2.

Worker: Workers in a cluster conduct machine learning tasks in asynchronous way. They pull the parameters from the servers by message passing. If a copy of the global parameters is pulled to the workers, those workers begin iterative calculations. During an epoch, workers first randomly pick an instance from the training data, then use the instance to compute the gradient. All the workers are independent with peers when conducting iterative calculations. When an epoch is finished, a worker has learned the new parameters, and will send those newly learned parameters to a server. It is the servers that control when to synchronize among the workers.

Aggregation: When the workers push their newly learned parameters to the servers, those parameters will be aggregated with the global parameters on the servers. The average between the newly learned parameters and the global parameters will be identified as the latest global parameters, and will wait to be pushed to all the workers for the next iteration.

Update rule: As illustrated in Algorithm 3, DisSVRG is significantly different from the standard SGD because of the variance reduction technique. In the standard SGD, the update rule is shown as follows.

$$v_t = \nabla f_{i_t}(\omega_{t-1}) \tag{5}$$

v_t in the standard SGD is not the global gradient, which leads to variance, and slows the convergence of the loss function unavoidably. Instead, the variance reduction technique is used in DisSVRG effectively reduce the variance [8]. Considering the ith round of the iterations, since i_t is randomly picked, it holds that

$$\mathbb{E}\nabla f_{i_t}(\omega_{t-1}) = \frac{1}{n}\nabla F(\omega_{t-1}) \tag{6}$$

, and

$$\mathbb{E}(-\nabla f_{i_t}(\tilde{\omega}) + \mu) = -\frac{1}{n}\nabla F(\tilde{\omega}) + \mathbb{E}(\mu) = 0 \tag{7}$$

. Therefore,

$$\mathbb{E}(\nabla f_{i_t}(\omega_{t-1}) - \nabla f_{i_t}(\tilde{\omega}) + \mu) = \mathbb{E}(\nabla f_{i_t}(\omega_{t-1})) = \frac{1}{n}\nabla F(\omega_{t-1}) \tag{8}$$

. It is obvious that the variance of DisSVRG is reduced as expected. Unlike PetuumSGD and DownpourSGD, DisSVRG can converge fast without decaying the learning rate during the iterations. Benefiting from the variance reduction technique, DisSVRG can converge at a constant rate.

5 Optimization of DisSVRG

5.1 Learning rate with an acceleration factor

Generally, variance reduction technique accelerates machine learning algorithms by using a constant learning rate. Even though the constant learning rate performs well for L-smooth and γ-strongly convex objection function, some intrinsic properties can be exploited to accelerate the convergence of the machine learning algorithms.

We introduce an acceleration factor σ with $\sigma = \| \nabla f_i(\omega) \|$ to the learning rate of DisSVRG. That is,

$$\eta = \eta_0 + \sigma, \qquad \sigma = \| \nabla f_i(\omega) \| \tag{9}$$

. The learning rate of DisSVRG contains two ingredients: the constant and the acceleration factor. The constant part of the learning rate get the parameters out of the local optimal state; while the acceleration factor speedup the convergence of DisSVRG. This

setting of the acceleration factor is reasonable for the following reasons. First, the acceleration factor will become large when the parameters are far from the optimal state. If so, DisSVRG will converge fast accordingly. Second, the acceleration factor will not become so large that DisSVRG do not converge. Considering that the convex function f_i in the machine learning model is L-smooth, ∇f_i will not be changed out of a range for an iteration. That is, $\| \nabla f_i \| < C$. Here, C is a non-negative constant. Third, when the parameters are close to the global optimal state, the acceleration factor σ will become close to zero. DisSVRG thus almost converges to the global optimal state at a constant just like the original design in [8].

As illustrated in Fig. 1, we use the acceleration factor to conduct the linear regression tasks. Here, the evaluation test is conducted on a node instead of a cluster. The dataset is *YearPredictMSD* which is the largest dataset we can find to conduct linear regression tasks. Other settings of the evaluation are presented in Section 7. We can get two important observations from Fig. 1. First, the learning rate with an acceleration factor makes the underlying machine learning algorithm converges faster than the constant learning rate without the acceleration factor. Second, the benefits become significant with a small basic learning rate. Shortly, the acceleration factor gives rise to obvious benefits to accelerate the convergence of a machine learning algorithm.

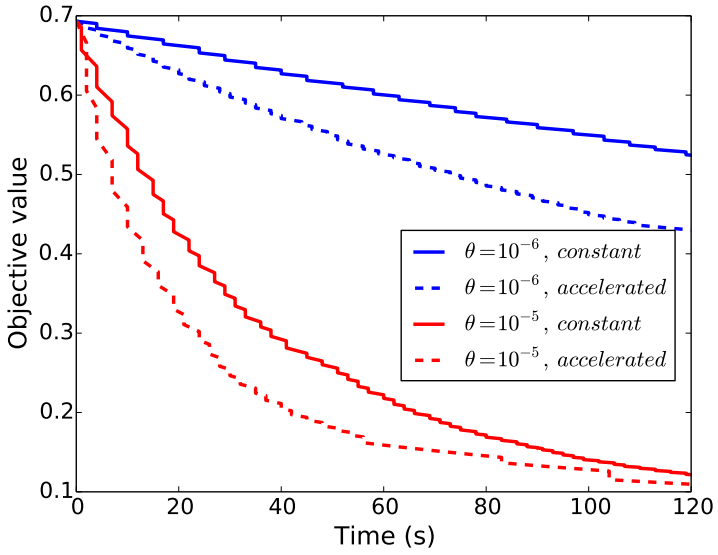


Fig. 1. The learning rate with the acceleration factor makes DisSVRG converge fast significantly.

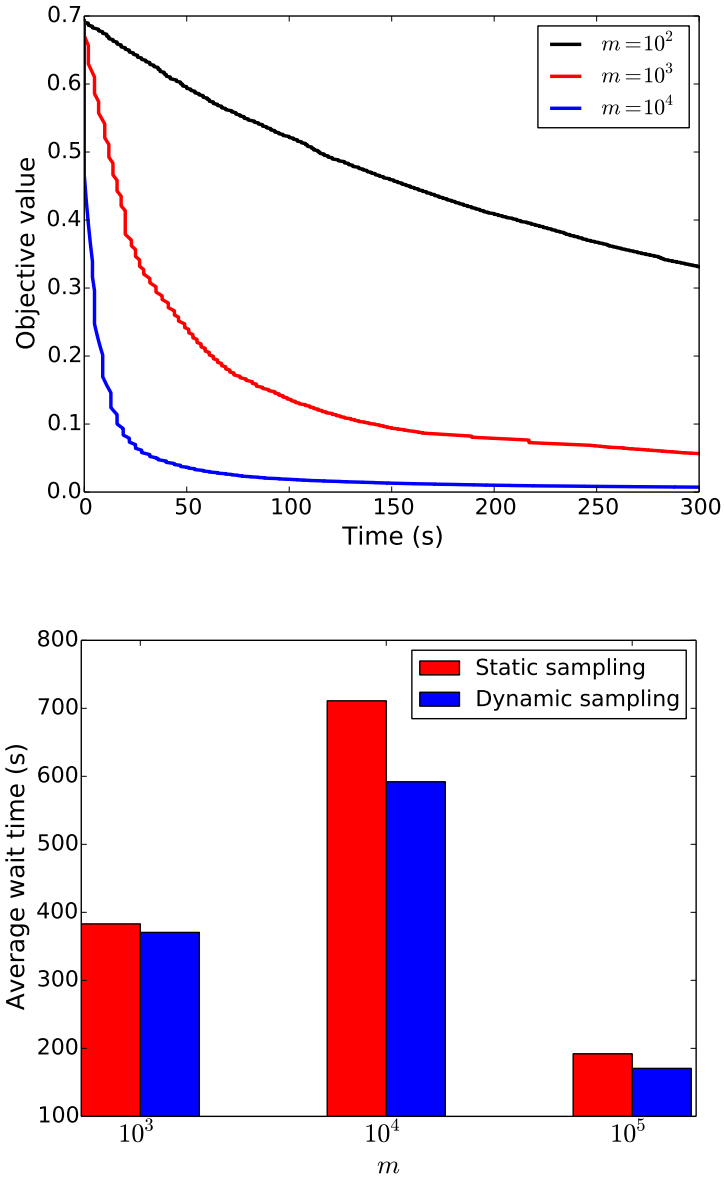


Fig. 2. A large m leads to a fast convergence for DisSVRG, and reduces much wait time during iterations.

5.2 Adaptive sampling strategy

Since DisSVRG needs the sampling strategy to update the parameters during an epoch, it is important to identify how many random updates in an epoch is appropriate. As illustrated in Algorithm 1, m represents the number of updates for an epoch. First, m cannot be given an extremely large number, which takes much time to update the local parameters during an epoch on a worker. Second, m cannot be set a small value straightly. If so, DisSVRG has to conduct many epochs to reduce the value of the loss function. Considering that the average gradient of the loss function is needed to conduct an epoch, a small m means much computation of the average gradients, which is time-consuming.

Additionally, it is worth noting that the workers in a cluster perform asynchronously due to the diversity of the runtime environment or the hardware in the heterogenous cluster. Although the asynchronous communication protocol, i.e., SSP, allows a delay bound to relax the synchronization among the workers, those fast workers have to wait for the slow peers when the delay is met. Such wait time impairs the convergence of DisSVRG. An approach to reduce the underlying wait time is to adjust the sampling strategy dynamically. Intuitively, the fast workers in the cluster should sample more instances during an epoch than the slow workers. That is, m in the fast workers should be larger than that in the slow workers.

We adopt an adaptive dynamic sampling strategy which dynamically adjust m. When an epoch is completed, the newly learned local parameters will be pushed to a server. The server will check the epochs of the worker. If the worker is too fast, it needs to wait for other slow peers. The fast workers will adjust m to be a large value, i.e., $m+\delta$. Here, δ is a non-negative integer with $\delta = 0.05m$ in DisSVRG. By using this adaptive strategy, the fast workers will sample more instances during an epoch, thus spend more time on computing the updates of parameters than the slow workers. The slow workers have chance to catch up with the fast workers. Therefore, the wait time is reduced as a result. This adaptive sampling strategy has many benefits. The most important benefit is that the fast workers reduce the wait time, and use it to converge DisSVRG. We conduct an evaluation test on a node by varying the value of m. Here, the dataset is still *YearPredictMSD*, and the learning rate, i.e., η is set to be a constant with $\eta = 10^{-5}$. As illustrated in Fig. 2(a), it is obvious that a large m brings a fast convergence of the objective function. Therefore, a large m benefits to decrease the value of the loss function. Additionally, the wait time is compared in a cluster which consists of 5 nodes. As shown in Fig. 2(b), the average wait time has been evaluated by varying m. Here, the delay is set to be 0. It is obvious that the adaptive sampling strategy decreases the average wait time during iterations, and thus spend much time to accelerate the convergence of DisSVRG in reverse.

6 Discussion

The machine learning tasks such as deep learning are generally fed with an extremely large volume of training data. However, conventional serial versions of SGD cannot handle a huge training data on a single node within the available time. Although some

distributed machine learning systems such as Petuum [14] and DMTK [15] have designed to solve this problem. Those variants of SGD have their inner weakness, namely the variance. Such those distributed machine learning systems thus decrease the learning rate to reduce the variance, which leads to slow convergence of SGD. We do not aim to proposing another a general platform for distributed machine learning algorithms, but focus on the optimization of SGD in a distributed system. Our implementation of the distributed SGD, i.e., DisSVRG, adopts the variance reduction technique to reduce the variance. Such variance reduction technique is the most difference between DisSVRG and PetummSGD. Although PetuumSGD adopts a decaying learning rate, it slows to converge the loss function. DisSVRG has been accelerated with two ingredients: the constant and the acceleration factor. The constant factor is effective to get DisSVRG out of the local optimal state, and keeps converge at a constant rate. The acceleration factor will speedup the convergence of the machine learning algorithm. Even though DisSVRG adopts the same asynchronous communication protocol with PetuumSGD, it converges faster than PetuumSGD by adopting the powerful variance reduction technique.

DisSVRG improves SVRG with at least three aspects. First, we extend the serial SVRG to an asynchronous and distributed version by using the asynchronous consistency protocol, i.e., SSP. Second, comparing with the constant learning rate in SVRG, the learning rate in our SGD contains an acceleration factor which exploits the potential benefits of the loss function, and makes the machine learning algorithms converge fast. Third, SVRG needs to sample m instances randomly to update parameters. SVRG sets m multiple times of the size of training data, which is not practical for a large volume of training data. Instead, DisSVRG adopts an adaptive sampling strategy which adjusts the value of m by the runtime environment of the worker dynamically. Specifically, the fast workers will sample more instances during the next epoch than the slow peers. The slow workers thus have chance to catch up with the fast workers. Thus, wait time is reduced significantly, which makes DisSVRG converge fast in the end. In a nutshell, considering the diversity of the runtime environment and the hardware in a cluster, DisSVRG is suitable to the practical scenarios.

Additionally, comparing with the version of SGD in [16] denoted by *SSGD*, our SGD adopts a more natural way to implement the distributed SGD with the variance reduction. SSGD implements the τ-delay bound inconsistent protocol within an epoch, but keeps fully consistent protocol among different workers. Therefore, SGD in [16] has at least two weaknesses. First, the fully consistency protocol for the workers is not suitable to the iterative convergence machine learning tasks [5, 6, 10]. Since the straggler problem usually exists among workers due to the variety of the system runtime environment or the hardware in the heterogenous cluster, the fast workers in [16] have to wait for the slow workers, and start the next iteration in a synchronous way. Considering that machine learning algorithms are iterative convergent, the fully consistency protocol wastes too much time. Second, SSGD is coupled with specific hardware settings of clusters, which is less practical and natural. SSGD uses m learning threads to perform machine learning tasks where m is also the number of instances sampled in an epoch. That is, if the sampling strategy in SSGD adopts a large m, SSGD should be run in a cluster which can support m learning threads at a same time. Meanwhile, m in SSGD

is $O(n)$ where n represents the size of the training data. Considering the huge size of training data, m is really large in SSGD, which is not practical in a real cluster. In fact, a practical SGD should be designed to be flexible to work in different clusters by merely adjusting its settings. Instead, m in DisSVRG is identified by the adaptive sampling mechanism, which is flexible to be adjusted in the runtime environment. This design of the sampling strategy shields the difference of a specific cluster, which is more general and natural.

7 Performance evaluation

In this section, we evaluate the performance of DisSVRG by using a regression problem and a classification problem on two datasets.

First we consider a regression problem:

$$\min \frac{1}{2n}\left(\frac{1}{1+e^{-\omega^{\mathrm{T}}x_i}}-y_i\right)^2 \tag{10}$$

Here, n is the size of training data. The dataset named *YearPredictMSD*[3] is the biggest dataset on the LibSVM for the regression problem. It contains 463715 samples, and each sample has 90 dimensions.

Additionally, we consider a classification problem:

$$\min -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log\frac{1}{1+e^{-\omega x_i}}+(1-y_i)\log(1-\frac{1}{1+e^{-\omega x_i}})\right) \tag{11}$$

Similarly, n is the size of training data. The datasets named *dna*[4] is used for the evaluation tests. Every sample in the dataset has 200 dimensions. The number of samples in the *dna* is $50,000,000$.

We conduct the evaluation test on the HPC cluster of the *Tianhe-1* supercomputer which is located in the National Supercomputing Center in Changsha. We have a maximum of 128 computing nodes, and each such node is equipped with two Intel Xeon X5670 CPUs and one Nvidia M2050 GPU. Each a CPU has 6 cores; while GPUs are not used in the evaluation test. m is set to be 1000. The learning rate, i.e., η is set to be 10^{-6}. For the fairness of the evaluation test, we use the third-party open source distributed machine learning system denoted by DMTK [15] to conduct the performance evaluation. All the compared algorithms are implemented based on the DMTK.

The following algorithms will be used for comparison.

- **PetuumSGD:** The distributed version of SGD is implemented by using the asynchronous communication protocol, i.e., SSP [14]. The learning rate in PetuumSGD is decayed with a fixed factor 0.95 at the end of an epoch.
- **SSGD:** It is the state-of-the-art distributed version of SGD, which adopts the variance reduction technique [16]. The update rule in the SSGD has a variable θ which is used to update the parameters asynchronously. The details of SSGD can be referred in [16]. Here, we set $\theta = 0.5$.

[3] http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/

[4] ftp://largescale.ml.tu-berlin.de/largescale

- **DisSVRG-tricks:** DisSVRG is evaluated with all the optimization tricks.
- **DisSVRG-without-tricks:** DisSVRG is evaluated without any an optimization tricks.

7.1 Convergence

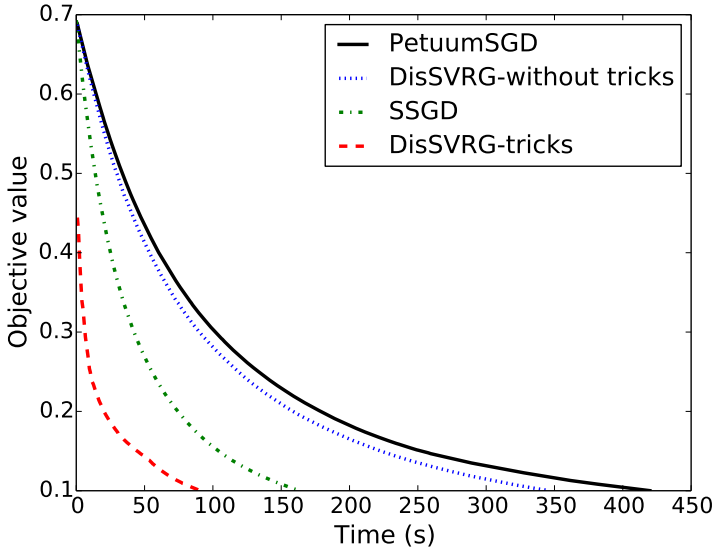
As illustrated in Fig. 3, the convergence performance of the algorithms has been evaluated. The delay is set to be 50 when those machine learning algorithms adopt the asynchronous communication protocol. All the datasets are handled on 32 workers. It is obvious that DisSVRG with the optimization tricks outperforms other algorithms for all the datasets. Even though DisSVRG does not use any an optimization trick, it always performs better than PetuumSGD, and gains a better performance than SSGD for the *dna* dataset. In specific, in order to decrease the loss function to 0.1, DisSVRG with all the optimization tricks spends one forth and one third time of the PetuumSGD for the datasets *YearPredictMSD* and *dna*, respectively. The main reason is that DisSVRG adopts asynchronous communication protocol as well as the variance reduction technique to update the parameters, thus better than any of the existing algorithms. Additionally, the learning rate with an acceleration factor speeds up the convergence. Meanwhile, the adaptive sampling strategy significantly reduces the underlying wait time which is used to accelerate the convergence of DisSVRG in reverse.

7.2 Speedup

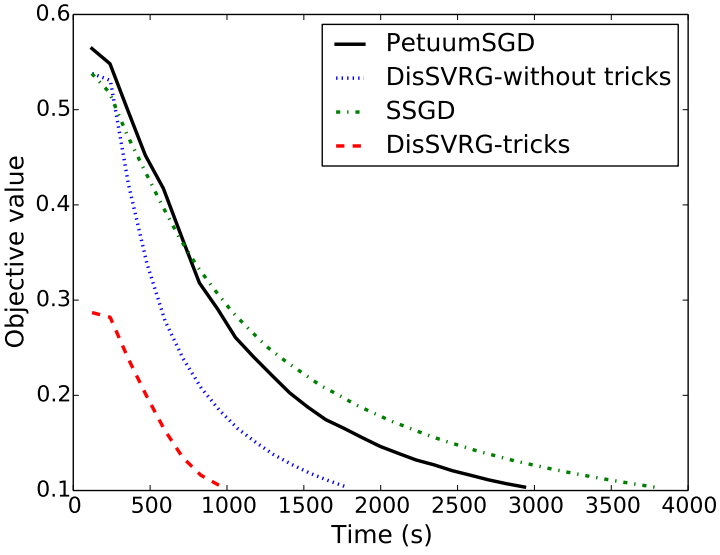
As illustrated in Fig. 4, we compare the convergence performance of DisSVRG by varying the number of workers in a cluster. It is obvious that DisSVRG converges fast with a large number of workers. During the iterations, the more workers are used to update the local parameters, the more newly learned parameters will be aggregated with the global parameters. After that, the global parameters which contains the newly learned updates of parameters will be shared with other workers for the next iteration, thus accelerating the convergence of other workers. In specific, DisSVRG obtains apprximately linear speedup when varying the number of workers. For instance, DisSVRG gains 19$\times$ speedup on the dataset *YearPredictMSD* when using 32 workers. DisSVRG even keeps the linear speedup when using 128 workers for the dataset *dna*. The approximately linear speedup mainly benefits from the asynchronous communication protocol and the variance reduction technique. The asynchronous communication protocol relaxes the bound of the synchronization among workers, which alleviates the straggler problem, and thus decreases the wait time. The variance reduction technique reduces the variance of SGD, and keeps DisSVRG converging at a constant rate.

7.3 Wait time

As shown in Fig. 5, we evaluate the average time consumption of DisSVRG by varying the value of the delay τ. It is obvious when the delay τ is small, the average wait time is large. A small delay means a tight bound among workers, which usually leads to the wait time for the fast workers due to the asynchronous communication protocol. For

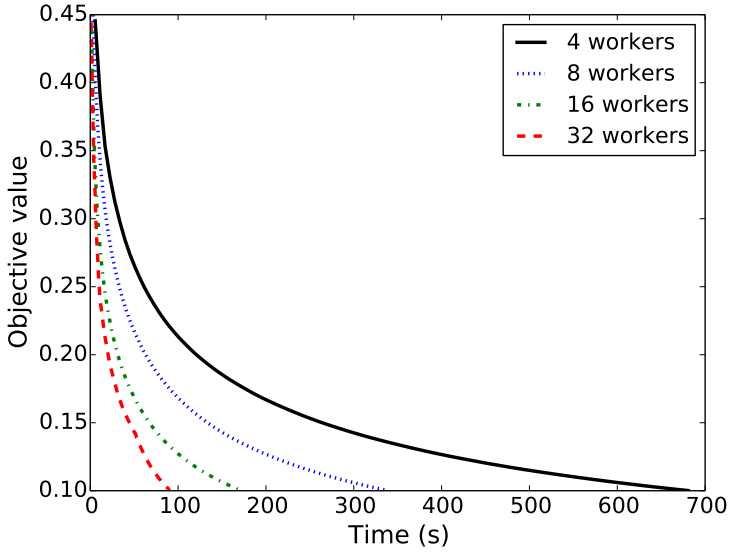


(a) *YearPredictMSD*

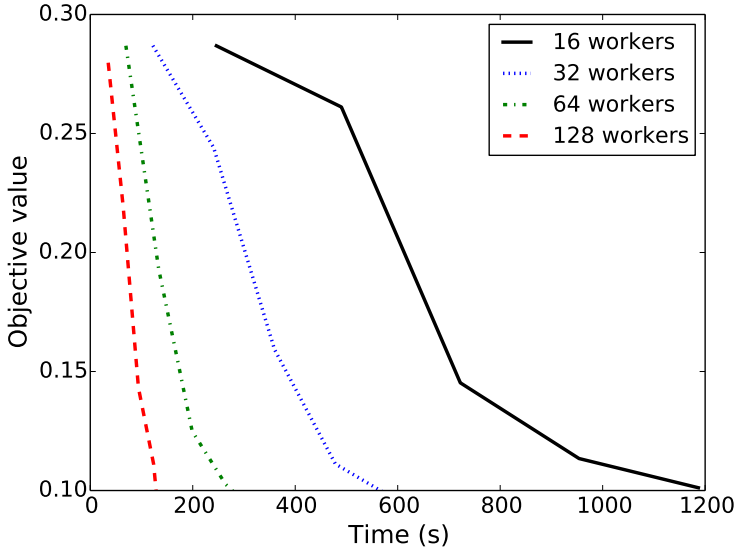


(b) *dna*

Fig. 3. The performance of the convergence is compared by using 32 computing nodes.

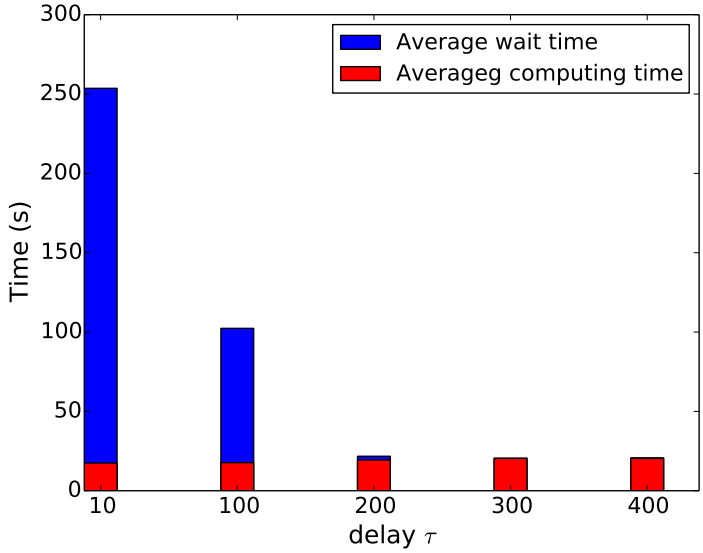


(a) *YearPredictMSD*

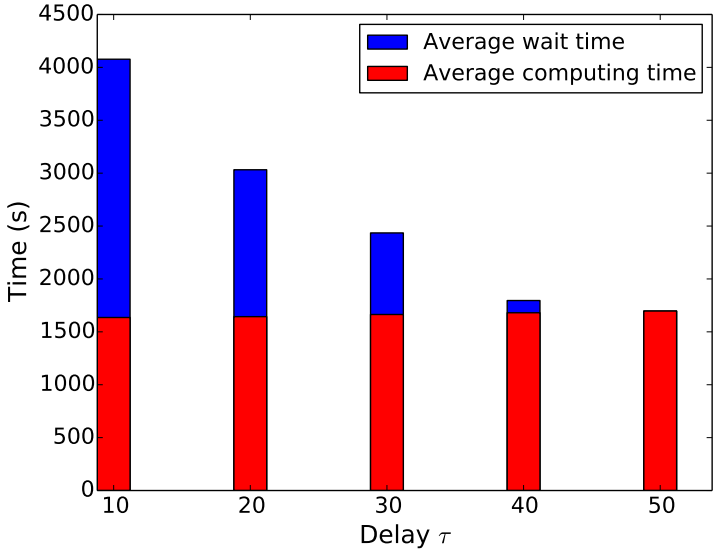


(b) *dna*

Fig. 4. DisSVRG obtains almost linear speedup when varying the number of workers.



(a) *YearPredictMSD*



(b) *dna*

Fig. 5. The time consumption is compared by varying the delay τ for 128 workers.

example, when the delay is set to be 0, all the workers should be synchronized for each iteration, thus waiting the longest time. It is worth noting that the wait time decreases sharply when the delay becomes large. We concludes that a relax bound is effective to reduce the average wait time. Meanwhile, the average computing time will increase slightly with a relax bound of the delay. Although the fast workers have more freedom to conduct the iterations with the relax bound of the delay, the slow workers cannot obtain the newly learned parameters from the fast workers within the large delay. The slow workers thus cannot benefit from the fast peers. In a nutshell, the delay should not be identified either too small or too large. It is a tradeoff between the wait time and the computing time for the workers. Generally, the delay should be set to minimize the total time consumption of DisSVRG. In our evaluation tests, the delay τ should be set to be 200 for the dataset *YearPredictMSD*, and 50 for the dataset *dna*.

8 Conclusion

Distributed SGD is an effective way to solve the large scale learning problems. In this paper, we propose a version of distributed SGD named DisSVRG with combing the asynchronous communication protocol and the variance reduction technique. Exploiting the properties of the loss function, DisSVRG is optimized by using a learning rate with the acceleration factor. Additionally, in order to reduce the wait time caused by the straggler problem, we propose an adaptive sampling strategy during the iterations. The adaptive sampling strategy gives the slow workers a chance to catch up with the fast peers during iterations, thus alleviating the straggler problem. Extensive empirical studies show that DisSVRG converges faster than the state-of-the-art version of SGD, and can gain approximately linear speedup in a cluster.

Acknowledgment

We thank the National Supercomputing Center in Changsha for providing *Tianhe-1* supercomputer as our experiment platform. This work was supported by the National Natural Science Foundation of China (Project NO. 61672528, 61170287, 61232016, 61303189 and 31501073).

References

1. Cavalcante, R.L., Yamada, I., Mulgrew, B.: An adaptive projected subgradient approach to learning in diffusion networks. Transactions on Signal Processing 57(7), 2762–2774 (2009)
2. Coates, A., Ng, A.Y., Lee, H.: An analysis of single-layer networks in unsupervised feature learning. Journal of Machine Learning Research 15, 215–223 (2011)
3. Collobert, R., Weston, J.: A unified architecture for natural language processing: Deep neural networks with multitask learning. In: Proc. ACM ICML. pp. 160–167 (2008)
4. Dahl, G.E., Yu, D., Deng, L., Acero, A.: Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. Transactions on Audio, Speech, and Language Processing 20(1), 30–42 (2012)

5. Dai, W., Kumar, A., Wei, J., Ho, Q., Gibson, G.A., Xing, E.P.: High-performance distributed ml at scale through parameter server consistency models. In: Proc. AAAI. pp. 79–87 (2015)
6. Dai, W., Wei, J., Zheng, X., Kim, J.K., Lee, S., Yin, J., Ho, Q., Xing, E.P.: Petuum: A framework for iterative-convergent distributed ml. arXiv:1312.7651 (2013)
7. Dean, J., Corrado, G., Monga, R., 0010, K.C., Devin, M., Le, Q.V., Mao, M.Z., Ranzato, M., Senior, A.W., Tucker, P.A., Yang, K., Ng, A.Y.: Large scale distributed deep networks. In: Proc. NIPS. pp. 1232–1240 (2012)
8. Johnson, R., Zhang, T.: Accelerating stochastic gradient descent using predictive variance reduction. Advances in Neural Information Processing Systems pp. 315–323 (2013)
9. Li, M., Andersen, D.G., Park, J.W., Smola, A.J., Ahmed, A., Josifovski, V., Long, J., Shekita, E.J., Su, B.Y.: Scaling distributed machine learning with the parameter server. In: Proc. UENSIX OSDI. pp. 583–598 (2014)
10. Li, M., Andersen, D.G., Smola, A.J., Yu, K.: Communication efficient distributed machine learning with the parameter server. Proc. NIPS pp. 19–27 (2014)
11. Lian, X., Huang, Y., Li, Y., Liu, J.: Asynchronous parallel stochastic gradient for nonconvex optimization. In: Advances in Neural Information Processing Systems. pp. 2719–2727 (2015)
12. Mania, H., Pan, X., Papailiopoulos, D., Recht, B., Ramchandran, K., Jordan, M.I.: Perturbed iterate analysis for asynchronous stochastic optimization. arXiv:1507.06970 (2015)
13. Reddi, S.J., Hefny, A., Sra, S., Póczos, B., Smola, A.J.: On variance reduction in stochastic gradient descent and its asynchronous variants pp. 2629–2637 (2015)
14. Xing, E.P., Ho, Q., Dai, W., Kim, J.K., Wei, J., Lee, S., Zheng, X., Xie, P., Kumar, A., Yu, Y.: Petuum: a new platform for distributed machine learning on big data. Transactions on Big Data 1(2), 49–67 (2015)
15. Yuan, J., Gao, F., Ho, Q., Dai, W., Wei, J., Zheng, X., Xing, E.P., Liu, T.Y., Ma, W.Y.: Lightlda: Big topic models on modest computer clusters. In: Proc. WWW. pp. 1351–1361 (2015)
16. Zhang, Ruiliang, S.Z., Kwok, J.T.: Fast distributed asynchronous sgd with variance reduction. arXiv:1508.01633 (2015)
17. Zhao, S.Y., Li, W.J.: Fast asynchronous parallel stochastic gradient decent. arXiv:1508.05711 (2015)

```
This is pdfTeX, Version 3.14159265-2.6-1.40.15 (TeX Live 2014/W32TeX)
(preloaded format=latex 2015.9.2)  27 SEP 2016 21:36
entering extended mode
 restricted \write18 enabled.
 %&-line parsing enabled.
**"asynchronsou and distributed stochastic gradient descent with variance
reduc
tion.tex"

(./asynchronsou and distributed stochastic gradient descent with variance
reduc
tion.tex
LaTeX2e <2014/05/01>
Babel <3.9k> and hyphenation patterns for 78 languages loaded.
(./llncs.cls
Document Class: llncs 2010/07/12 v2.17
 LaTeX document class for Lecture Notes in Computer Science
(c:/texlive/2014/texmf-dist/tex/latex/base/article.cls
Document Class: article 2007/10/19 v1.4h Standard LaTeX document class
(c:/texlive/2014/texmf-dist/tex/latex/base/size10.clo
File: size10.clo 2007/10/19 v1.4h Standard LaTeX file (size option)
)
\c@part=\count79
\c@section=\count80
\c@subsection=\count81
\c@subsubsection=\count82
\c@paragraph=\count83
\c@subparagraph=\count84
\c@figure=\count85
\c@table=\count86
\abovecaptionskip=\skip41
\belowcaptionskip=\skip42
\bibindent=\dimen102
) (c:/texlive/2014/texmf-dist/tex/latex/tools/multicol.sty
Package: multicol 2014/04/23 v1.8e multicolumn formatting (FMi)
\c@tracingmulticols=\count87
\mult@box=\box26
\multicol@leftmargin=\dimen103
\c@unbalance=\count88
\c@collectmore=\count89
\doublecol@number=\count90
\multicoltolerance=\count91
\multicolpretolerance=\count92
\full@width=\dimen104
\page@free=\dimen105
\premulticols=\dimen106
\postmulticols=\dimen107
\multicolsep=\skip43
\multicolbaselineskip=\skip44
\partial@page=\box27
\last@line=\box28
\maxbalancingoverflow=\dimen108
\mult@rightbox=\box29
\mult@grightbox=\box30
\mult@gfirstbox=\box31
\mult@firstbox=\box32
\@tempa=\box33
\@tempa=\box34
```

\@tempa=\box35
\@tempa=\box36
\@tempa=\box37
\@tempa=\box38
\@tempa=\box39
\@tempa=\box40
\@tempa=\box41
\@tempa=\box42
\@tempa=\box43
\@tempa=\box44
\@tempa=\box45
\@tempa=\box46
\@tempa=\box47
\@tempa=\box48
\@tempa=\box49
\c@columnbadness=\count93
\c@finalcolumnbadness=\count94
\last@try=\dimen109
\multicolovershoot=\dimen110
\multicolundershoot=\dimen111
\mult@nat@firstbox=\box50
\colbreak@box=\box51
\mc@col@check@num=\count95
) (c:/texlive/2014/texmf-dist/tex/latex/oberdiek/aliascnt.sty
Package: aliascnt 2009/09/08 v1.3 Alias counters (HO)
(c:/texlive/2014/texmf-dist/tex/latex/carlisle/remreset.sty))
\c@chapter=\count96
LaTeX Font Info: Redeclaring math symbol \Gamma on input line 362.
LaTeX Font Info: Redeclaring math symbol \Delta on input line 363.
LaTeX Font Info: Redeclaring math symbol \Theta on input line 364.
LaTeX Font Info: Redeclaring math symbol \Lambda on input line 365.
LaTeX Font Info: Redeclaring math symbol \Xi on input line 366.
LaTeX Font Info: Redeclaring math symbol \Pi on input line 367.
LaTeX Font Info: Redeclaring math symbol \Sigma on input line 368.
LaTeX Font Info: Redeclaring math symbol \Upsilon on input line 369.
LaTeX Font Info: Redeclaring math symbol \Phi on input line 370.
LaTeX Font Info: Redeclaring math symbol \Psi on input line 371.
LaTeX Font Info: Redeclaring math symbol \Omega on input line 372.
\tocchpnum=\dimen112
\tocsecnum=\dimen113
\tocsectotal=\dimen114
\tocsubsecnum=\dimen115
\tocsubsectotal=\dimen116
\tocsubsubsecnum=\dimen117
\tocsubsubsectotal=\dimen118
\tocparanum=\dimen119
\tocparatotal=\dimen120
\tocsubparanum=\dimen121
\@tempcntc=\count97
\fnindent=\dimen122
\c@@inst=\count98
\c@@auth=\count99
\c@auco=\count100
\instindent=\dimen123
\authrun=\box52
\authorrunning=\toks14
\tocauthor=\toks15
\titrun=\box53

```
\titlerunning=\toks16
\toctitle=\toks17
\c@theorem=\count101
\c@case=\count102
\c@conjecture=\count103
\c@corollary=\count104
\c@definition=\count105
\c@example=\count106
\c@exercise=\count107
\c@lemma=\count108
\c@note=\count109
\c@problem=\count110
\c@property=\count111
\c@proposition=\count112
\c@question=\count113
\c@solution=\count114
\c@remark=\count115
\headlineindent=\dimen124
) (c:/texlive/2014/texmf-dist/tex/latex/cite/cite.sty
LaTeX Info: Redefining \cite on input line 302.
LaTeX Info: Redefining \nocite on input line 373.
Package: cite 2010/09/10  v 5.3
) (c:/texlive/2014/texmf-dist/tex/latex/hyperref/hyperref.sty
Package: hyperref 2012/11/06 v6.83m Hypertext links for LaTeX
(c:/texlive/2014/texmf-dist/tex/generic/oberdiek/hobsub-hyperref.sty
Package: hobsub-hyperref 2012/05/28 v1.13 Bundle oberdiek, subset
hyperref (HO)

(c:/texlive/2014/texmf-dist/tex/generic/oberdiek/hobsub-generic.sty
Package: hobsub-generic 2012/05/28 v1.13 Bundle oberdiek, subset generic
(HO)
Package: hobsub 2012/05/28 v1.13 Construct package bundles (HO)
Package: infwarerr 2010/04/08 v1.3 Providing info/warning/error messages
(HO)
Package: ltxcmds 2011/11/09 v1.22 LaTeX kernel commands for general use
(HO)
Package: ifluatex 2010/03/01 v1.3 Provides the ifluatex switch (HO)
Package ifluatex Info: LuaTeX not detected.
Package: ifvtex 2010/03/01 v1.5 Detect VTeX and its facilities (HO)
Package ifvtex Info: VTeX not detected.
Package: intcalc 2007/09/27 v1.1 Expandable calculations with integers
(HO)
Package: ifpdf 2011/01/30 v2.3 Provides the ifpdf switch (HO)
Package ifpdf Info: pdfTeX in PDF mode is not detected.
Package: etexcmds 2011/02/16 v1.5 Avoid name clashes with e-TeX commands
(HO)
Package etexcmds Info: Could not find \expanded.
(etexcmds)             That can mean that you are not using pdfTeX 1.50
or
(etexcmds)             that some package has redefined \expanded.
(etexcmds)             In the latter case, load this package earlier.
Package: kvsetkeys 2012/04/25 v1.16 Key value parser (HO)
Package: kvdefinekeys 2011/04/07 v1.3 Define keys (HO)
Package: pdftexcmds 2011/11/29 v0.20 Utility functions of pdfTeX for
LuaTeX (HO
)
Package pdftexcmds Info: LuaTeX not detected.
Package pdftexcmds Info: \pdf@primitive is available.
```

```
Package pdftexcmds Info: \pdf@ifprimitive is available.
Package pdftexcmds Info: \pdfdraftmode is ignored in DVI mode.
Package: pdfescape 2011/11/25 v1.13 Implements pdfTeX's escape features
(HO)
Package: bigintcalc 2012/04/08 v1.3 Expandable calculations on big
integers (HO
)
Package: bitset 2011/01/30 v1.1 Handle bit-vector datatype (HO)
Package: uniquecounter 2011/01/30 v1.2 Provide unlimited unique counter
(HO)
)
Package hobsub Info: Skipping package `hobsub' (already loaded).
Package: letltxmacro 2010/09/02 v1.4 Let assignment for LaTeX macros (HO)
Package: hopatch 2012/05/28 v1.2 Wrapper for package hooks (HO)
Package: xcolor-patch 2011/01/30 xcolor patch
Package: atveryend 2011/06/30 v1.8 Hooks at the very end of document (HO)
Package atveryend Info: \enddocument detected (standard20110627).
Package: atbegshi 2011/10/05 v1.16 At begin shipout hook (HO)
Package: refcount 2011/10/16 v3.4 Data extraction from label references
(HO)
Package: hycolor 2011/01/30 v1.7 Color options for hyperref/bookmark (HO)
) (c:/texlive/2014/texmf-dist/tex/latex/graphics/keyval.sty
Package: keyval 2014/05/08 v1.15 key=value parser (DPC)
\KV@toks@=\toks18
) (c:/texlive/2014/texmf-dist/tex/generic/ifxetex/ifxetex.sty
Package: ifxetex 2010/09/12 v0.6 Provides ifxetex conditional
) (c:/texlive/2014/texmf-dist/tex/latex/oberdiek/auxhook.sty
Package: auxhook 2011/03/04 v1.3 Hooks for auxiliary files (HO)
) (c:/texlive/2014/texmf-dist/tex/latex/oberdiek/kvoptions.sty
Package: kvoptions 2011/06/30 v3.11 Key value format for package options
(HO)
)
\@linkdim=\dimen125
\Hy@linkcounter=\count116
\Hy@pagecounter=\count117
(c:/texlive/2014/texmf-dist/tex/latex/hyperref/pd1enc.def
File: pd1enc.def 2012/11/06 v6.83m Hyperref: PDFDocEncoding definition
(HO)
)
\Hy@SavedSpaceFactor=\count118
(c:/texlive/2014/texmf-dist/tex/latex/latexconfig/hyperref.cfg
File: hyperref.cfg 2002/06/06 v1.2 hyperref configuration of TeXLive
)
Package hyperref Info: Option `hyperfootnotes' set `true' on input line
4319.
Package hyperref Info: Hyper figures OFF on input line 4443.
Package hyperref Info: Link nesting OFF on input line 4448.
Package hyperref Info: Hyper index ON on input line 4451.
Package hyperref Info: Plain pages OFF on input line 4458.
Package hyperref Info: Backreferencing OFF on input line 4463.
Package hyperref Info: Implicit mode ON; LaTeX internals redefined.
Package hyperref Info: Bookmarks ON on input line 4688.
\c@Hy@tempcnt=\count119
(c:/texlive/2014/texmf-dist/tex/latex/url/url.sty
\Urlmuskip=\muskip10
Package: url 2013/09/16  ver 3.4  Verb mode for urls, etc.
)
LaTeX Info: Redefining \url on input line 5041.
```

```
\XeTeXLinkMargin=\dimen126
\Fld@menulength=\count120
\Field@Width=\dimen127
\Fld@charsize=\dimen128
Package hyperref Info: Hyper figures OFF on input line 6295.
Package hyperref Info: Link nesting OFF on input line 6300.
Package hyperref Info: Hyper index ON on input line 6303.
Package hyperref Info: backreferencing OFF on input line 6310.
Package hyperref Info: Link coloring OFF on input line 6315.
Package hyperref Info: Link coloring with OCG OFF on input line 6320.
Package hyperref Info: PDF/A mode OFF on input line 6325.
LaTeX Info: Redefining \ref on input line 6365.
LaTeX Info: Redefining \pageref on input line 6369.
\Hy@abspage=\count121
\c@Item=\count122
\c@Hfootnote=\count123
)

Package hyperref Message: Driver (default): hdvips.

(c:/texlive/2014/texmf-dist/tex/latex/hyperref/hdvips.def
File: hdvips.def 2012/11/06 v6.83m Hyperref driver for dvips
(c:/texlive/2014/texmf-dist/tex/latex/hyperref/pdfmark.def
File: pdfmark.def 2012/11/06 v6.83m Hyperref definitions for pdfmark
specials
\pdf@docset=\toks19
\pdf@box=\box54
\pdf@toks=\toks20
\pdf@defaulttoks=\toks21
\HyField@AnnotCount=\count124
\Fld@listcount=\count125
\c@bookmark@seq@number=\count126
(c:/texlive/2014/texmf-dist/tex/latex/oberdiek/rerunfilecheck.sty
Package: rerunfilecheck 2011/04/15 v1.7 Rerun checks for auxiliary files
(HO)
Package uniquecounter Info: New unique counter `rerunfilecheck' on input
line 2
82.
)
\Hy@SectionHShift=\skip45
)) (c:/texlive/2014/texmf-dist/tex/latex/amsfonts/amssymb.sty
Package: amssymb 2013/01/14 v3.01 AMS font symbols
(c:/texlive/2014/texmf-dist/tex/latex/amsfonts/amsfonts.sty
Package: amsfonts 2013/01/14 v3.01 Basic AMSFonts support
\@emptytoks=\toks22
\symAMSa=\mathgroup4
\symAMSb=\mathgroup5
LaTeX Font Info:    Overwriting math alphabet `\mathfrak' in version
`bold'
(Font)                  U/euf/m/n --> U/euf/b/n on input line 106.
)) (c:/texlive/2014/texmf-dist/tex/latex/algorithms/algorithm.sty
Package: algorithm 2009/08/24 v0.1 Document Style `algorithm' - floating
enviro
nment
(c:/texlive/2014/texmf-dist/tex/latex/float/float.sty
Package: float 2001/11/08 v1.3d Float enhancements (AL)
\c@float@type=\count127
\float@exts=\toks23
```

```
\float@box=\box55
\@float@everytoks=\toks24
\@floatcapt=\box56
) (c:/texlive/2014/texmf-dist/tex/latex/base/ifthen.sty
Package: ifthen 2001/05/26 v1.1c Standard LaTeX ifthen package (DPC)
)
\@float@every@algorithm=\toks25
\c@algorithm=\count128
) (c:/texlive/2014/texmf-dist/tex/latex/amsmath/amsmath.sty
Package: amsmath 2013/01/14 v2.14 AMS math features
\@mathmargin=\skip46
For additional information on amsmath, use the `?' option.
(c:/texlive/2014/texmf-dist/tex/latex/amsmath/amstext.sty
Package: amstext 2000/06/29 v2.01
(c:/texlive/2014/texmf-dist/tex/latex/amsmath/amsgen.sty
File: amsgen.sty 1999/11/30 v2.0
\@emptytoks=\toks26
\ex@=\dimen129
)) (c:/texlive/2014/texmf-dist/tex/latex/amsmath/amsbsy.sty
Package: amsbsy 1999/11/29 v1.2d
\pmbraise@=\dimen130
) (c:/texlive/2014/texmf-dist/tex/latex/amsmath/amsopn.sty
Package: amsopn 1999/12/14 v2.01 operator names
)
\inf@bad=\count129
LaTeX Info: Redefining \frac on input line 210.
\uproot@=\count130
\leftroot@=\count131
LaTeX Info: Redefining \overline on input line 306.
\classnum@=\count132
\DOTSCASE@=\count133
LaTeX Info: Redefining \ldots on input line 378.
LaTeX Info: Redefining \dots on input line 381.
LaTeX Info: Redefining \cdots on input line 466.
\Mathstrutbox@=\box57
\strutbox@=\box58
\big@size=\dimen131
LaTeX Font Info:    Redeclaring font encoding OML on input line 566.
LaTeX Font Info:    Redeclaring font encoding OMS on input line 567.

Package amsmath Warning: Unable to redefine math accent \vec.

\macc@depth=\count134
\c@MaxMatrixCols=\count135
\dotsspace@=\muskip11
\c@parentequation=\count136
\dspbrk@lvl=\count137
\tag@help=\toks27
\row@=\count138
\column@=\count139
\maxfields@=\count140
\andhelp@=\toks28
\eqnshift@=\dimen132
\alignsep@=\dimen133
\tagshift@=\dimen134
\tagwidth@=\dimen135
\totwidth@=\dimen136
\lineht@=\dimen137
```

```
\@envbody=\toks29
\multlinegap=\skip47
\multlinetaggap=\skip48
\mathdisplay@stack=\toks30
LaTeX Info: Redefining \[ on input line 2665.
LaTeX Info: Redefining \] on input line 2666.
) (c:/texlive/2014/texmf-dist/tex/latex/tools/array.sty
Package: array 2008/09/09 v2.4c Tabular extension package (FMi)
\col@sep=\dimen138
\extrarowheight=\dimen139
\NC@list=\toks31
\extratabsurround=\skip49
\backup@length=\skip50
) (c:/texlive/2014/texmf-dist/tex/latex/graphics/graphicx.sty
Package: graphicx 2014/04/25 v1.0g Enhanced LaTeX Graphics (DPC,SPQR)
(c:/texlive/2014/texmf-dist/tex/latex/graphics/graphics.sty
Package: graphics 2009/02/05 v1.0o Standard LaTeX Graphics (DPC,SPQR)
(c:/texlive/2014/texmf-dist/tex/latex/graphics/trig.sty
Package: trig 1999/03/16 v1.09 sin cos tan (DPC)
) (c:/texlive/2014/texmf-dist/tex/latex/latexconfig/graphics.cfg
File: graphics.cfg 2010/04/23 v1.9 graphics configuration of TeX Live
)
Package graphics Info: Driver file: dvips.def on input line 91.
(c:/texlive/2014/texmf-dist/tex/latex/graphics/dvips.def
File: dvips.def 2014/04/23 v3.0j Driver-dependant file (DPC,SPQR)
))
\Gin@req@height=\dimen140
\Gin@req@width=\dimen141
) (c:/texlive/2014/texmf-dist/tex/latex/psnfss/pifont.sty
Package: pifont 2005/04/12 PSNFSS-v9.2a Pi font support (SPQR)
LaTeX Font Info:    Try loading font information for U+pzd on input line
63.
(c:/texlive/2014/texmf-dist/tex/latex/psnfss/upzd.fd
File: upzd.fd 2001/06/04 font definitions for U/pzd.
)
LaTeX Font Info:    Try loading font information for U+psy on input line
64.
(c:/texlive/2014/texmf-dist/tex/latex/psnfss/upsy.fd
File: upsy.fd 2001/06/04 font definitions for U/psy.
)) (c:/texlive/2014/texmf-dist/tex/latex/graphics/epsfig.sty
Package: epsfig 1999/02/16 v1.7a (e)psfig emulation (SPQR)
\epsfxsize=\dimen142
\epsfysize=\dimen143
) (c:/texlive/2014/texmf-dist/tex/latex/tools/tabularx.sty
Package: tabularx 2014/05/13 v2.10 `tabularx' package (DPC)
\TX@col@width=\dimen144
\TX@old@table=\dimen145
\TX@old@col=\dimen146
\TX@target=\dimen147
\TX@delta=\dimen148
\TX@cols=\count141
\TX@ftn=\toks32
) (c:/texlive/2014/texmf-dist/tex/latex/aries/subfigure.sty
Package: subfigure 2002/03/15 v2.1.5 subfigure package
\subfigtopskip=\skip51
\subfigcapskip=\skip52
\subfigcaptopadj=\dimen149
\subfigbottomskip=\skip53
```

```
\subfigcapmargin=\dimen150
\subfiglabelskip=\skip54
\c@subfigure=\count142
\c@lofdepth=\count143
\c@subtable=\count144
\c@lotdepth=\count145
****************************************
* Local config file subfigure.cfg used *
****************************************
(c:/texlive/2014/texmf-dist/tex/latex/aries/subfigure.cfg)
\subfig@top=\skip55
\subfig@bottom=\skip56
) (c:/texlive/2014/texmf-dist/tex/latex/multirow/multirow.sty
\bigstrutjot=\dimen151
) (c:/texlive/2014/texmf-dist/tex/latex/booktabs/booktabs.sty
Package: booktabs 2005/04/14 v1.61803 publication quality tables
\heavyrulewidth=\dimen152
\lightrulewidth=\dimen153
\cmidrulewidth=\dimen154
\belowrulesep=\dimen155
\belowbottomsep=\dimen156
\aboverulesep=\dimen157
\abovetopsep=\dimen158
\cmidrulesep=\dimen159
\cmidrulekern=\dimen160
\defaultaddspace=\dimen161
\@cmidla=\count146
\@cmidlb=\count147
\@aboverulesep=\dimen162
\@belowrulesep=\dimen163
\@thisruleclass=\count148
\@lastruleclass=\count149
\@thisrulewidth=\dimen164
) (c:/texlive/2014/texmf-dist/tex/latex/preprint/balance.sty
Package: balance 1999/02/23 4.3 (PWD)
\oldvsize=\dimen165
) (c:/texlive/2014/texmf-dist/tex/latex/tools/bm.sty
Package: bm 2004/02/26 v1.1c Bold Symbol Support (DPC/FMi)
\symboldoperators=\mathgroup6
\symboldletters=\mathgroup7
\symboldsymbols=\mathgroup8
LaTeX Font Info:    Redeclaring math alphabet \mathbf on input line 137.
LaTeX Info: Redefining \bm on input line 203.
) (c:/texlive/2014/texmf-dist/tex/latex/algorithmicx/algpseudocode.sty
Package: algpseudocode
(c:/texlive/2014/texmf-dist/tex/latex/algorithmicx/algorithmicx.sty
Package: algorithmicx 2005/04/27 v1.2 Algorithmicx
Document Style algorithmicx 1.2 - a greatly improved `algorithmic' style
\c@ALG@line=\count150
\c@ALG@rem=\count151
\c@ALG@nested=\count152
\ALG@tlm=\skip57
\ALG@thistlm=\skip58
\c@ALG@Lnr=\count153
\c@ALG@blocknr=\count154
\c@ALG@storecount=\count155
\c@ALG@tmpcounter=\count156
\ALG@tmplength=\skip59
```

```
)
Document Style - pseudocode environments for use with the `algorithmicx'
style
) (c:/texlive/2014/texmf-dist/tex/latex/psnfss/mathptmx.sty
Package: mathptmx 2005/04/12 PSNFSS-v9.2a Times w/ Math, improved (SPQR,
WaS)
LaTeX Font Info:    Redeclaring symbol font `operators' on input line 28.
LaTeX Font Info:    Overwriting symbol font `operators' in version
`normal'
(Font)                  OT1/cmr/m/n --> OT1/ztmcm/m/n on input line 28.
LaTeX Font Info:    Overwriting symbol font `operators' in version `bold'
(Font)                  OT1/cmr/bx/n --> OT1/ztmcm/m/n on input line 28.
LaTeX Font Info:    Redeclaring symbol font `letters' on input line 29.
LaTeX Font Info:    Overwriting symbol font `letters' in version `normal'
(Font)                  OML/cmm/m/it --> OML/ztmcm/m/it on input line 29.
LaTeX Font Info:    Overwriting symbol font `letters' in version `bold'
(Font)                  OML/cmm/b/it --> OML/ztmcm/m/it on input line 29.
LaTeX Font Info:    Redeclaring symbol font `symbols' on input line 30.
LaTeX Font Info:    Overwriting symbol font `symbols' in version `normal'
(Font)                  OMS/cmsy/m/n --> OMS/ztmcm/m/n on input line 30.
LaTeX Font Info:    Overwriting symbol font `symbols' in version `bold'
(Font)                  OMS/cmsy/b/n --> OMS/ztmcm/m/n on input line 30.
LaTeX Font Info:    Redeclaring symbol font `largesymbols' on input line
31.
LaTeX Font Info:    Overwriting symbol font `largesymbols' in version
`normal'
(Font)                  OMX/cmex/m/n --> OMX/ztmcm/m/n on input line 31.
LaTeX Font Info:    Overwriting symbol font `largesymbols' in version
`bold'
(Font)                  OMX/cmex/m/n --> OMX/ztmcm/m/n on input line 31.
\symbold=\mathgroup9
\symitalic=\mathgroup10
LaTeX Font Info:    Redeclaring math alphabet \mathbf on input line 34.
LaTeX Font Info:    Redeclaring math alphabet \mathit on input line 35.
LaTeX Font Info:    Overwriting math alphabet `\mathit' in version
`normal'
(Font)                  OT1/cmr/m/it --> OT1/ptm/m/it on input line 35.
LaTeX Font Info:    Overwriting math alphabet `\mathit' in version `bold'
(Font)                  OT1/cmr/bx/it --> OT1/ptm/m/it on input line 35.
LaTeX Info: Redefining \hbar on input line 50.
) (c:/texlive/2014/texmf-dist/tex/latex/footmisc/footmisc.sty
Package: footmisc 2011/06/06 v5.5b a miscellany of footnote facilities
\FN@temptoken=\toks33
\footnotemargin=\dimen166
\c@pp@next@reset=\count157
Package footmisc Info: Declaring symbol style bringhurst on input line
855.
Package footmisc Info: Declaring symbol style chicago on input line 863.
Package footmisc Info: Declaring symbol style wiley on input line 872.
Package footmisc Info: Declaring symbol style lamport-robust on input
line 883.

Package footmisc Info: Declaring symbol style lamport* on input line 903.
Package footmisc Info: Declaring symbol style lamport*-robust on input
line 924
.
)
```

```
(./asynchronsou and distributed stochastic gradient descent with variance
reduc
tion.aux

LaTeX Warning: Label `standard_sgd' multiply defined.


LaTeX Warning: Label `standard_sgd' multiply defined.

)
\openout1 = `"asynchronsou and distributed stochastic gradient descent
with var
iance reduction.aux"'.

LaTeX Font Info:    Checking defaults for OML/cmm/m/it on input line 37.
LaTeX Font Info:    ... okay on input line 37.
LaTeX Font Info:    Checking defaults for T1/cmr/m/n on input line 37.
LaTeX Font Info:    ... okay on input line 37.
LaTeX Font Info:    Checking defaults for OT1/cmr/m/n on input line 37.
LaTeX Font Info:    ... okay on input line 37.
LaTeX Font Info:    Checking defaults for OMS/cmsy/m/n on input line 37.
LaTeX Font Info:    ... okay on input line 37.
LaTeX Font Info:    Checking defaults for OMX/cmex/m/n on input line 37.
LaTeX Font Info:    ... okay on input line 37.
LaTeX Font Info:    Checking defaults for U/cmr/m/n on input line 37.
LaTeX Font Info:    ... okay on input line 37.
LaTeX Font Info:    Checking defaults for PD1/pdf/m/n on input line 37.
LaTeX Font Info:    ... okay on input line 37.
LaTeX Font Info:    Try loading font information for OT1+ptm on input
line 37.
(c:/texlive/2014/texmf-dist/tex/latex/psnfss/ot1ptm.fd
File: ot1ptm.fd 2001/06/04 font definitions for OT1/ptm.
)
\AtBeginShipoutBox=\box59
Package hyperref Info: Link coloring OFF on input line 37.
(c:/texlive/2014/texmf-dist/tex/latex/hyperref/nameref.sty
Package: nameref 2012/10/27 v2.43 Cross-referencing by name of section
(c:/texlive/2014/texmf-dist/tex/generic/oberdiek/gettitlestring.sty
Package: gettitlestring 2010/12/03 v1.4 Cleanup title references (HO)
)
\c@section@level=\count158
)
LaTeX Info: Redefining \ref on input line 37.
LaTeX Info: Redefining \pageref on input line 37.
LaTeX Info: Redefining \nameref on input line 37.

(./asynchronsou and distributed stochastic gradient descent with variance
reduc
tion.out)
(./asynchronsou and distributed stochastic gradient descent with variance
reduc
tion.out)
\@outlinefile=\write3
\openout3 = `"asynchronsou and distributed stochastic gradient descent
with var
iance reduction.out"'.
```

```
LaTeX Font Info:    Font shape `OT1/ptm/bx/n' in size <14.4> not
available
(Font)              Font shape `OT1/ptm/b/n' tried instead on input line
71.

Package mathptmx Warning: There are no bold math fonts on input line 71.

LaTeX Font Info:    Try loading font information for OT1+ztmcm on input
line 71
.
(c:/texlive/2014/texmf-dist/tex/latex/psnfss/ot1ztmcm.fd
File: ot1ztmcm.fd 2000/01/03 Fontinst v1.801 font definitions for
OT1/ztmcm.
)
LaTeX Font Info:    Try loading font information for OML+ztmcm on input
line 71
.
(c:/texlive/2014/texmf-dist/tex/latex/psnfss/omlztmcm.fd
File: omlztmcm.fd 2000/01/03 Fontinst v1.801 font definitions for
OML/ztmcm.
)
LaTeX Font Info:    Try loading font information for OMS+ztmcm on input
line 71
.
(c:/texlive/2014/texmf-dist/tex/latex/psnfss/omsztmcm.fd
File: omsztmcm.fd 2000/01/03 Fontinst v1.801 font definitions for
OMS/ztmcm.
)
LaTeX Font Info:    Try loading font information for OMX+ztmcm on input
line 71
.
(c:/texlive/2014/texmf-dist/tex/latex/psnfss/omxztmcm.fd
File: omxztmcm.fd 2000/01/03 Fontinst v1.801 font definitions for
OMX/ztmcm.
)
LaTeX Font Info:    Font shape `OT1/ptm/bx/n' in size <10.95> not
available
(Font)              Font shape `OT1/ptm/b/n' tried instead on input line
71.
LaTeX Font Info:    Font shape `OT1/ptm/bx/n' in size <8> not available
(Font)              Font shape `OT1/ptm/b/n' tried instead on input line
71.
LaTeX Font Info:    Try loading font information for OMS+ptm on input
line 71.
(c:/texlive/2014/texmf-dist/tex/latex/psnfss/omsptm.fd
File: omsptm.fd
)
LaTeX Font Info:    Font shape `OMS/ptm/m/n' in size <10.95> not
available
(Font)              Font shape `OMS/cmsy/m/n' tried instead on input line
71.
LaTeX Font Info:    Font shape `OT1/ptm/bx/n' in size <10> not available
(Font)              Font shape `OT1/ptm/b/n' tried instead on input line
71.
LaTeX Font Info:    Font shape `OT1/ptm/bx/n' in size <7.4> not available
(Font)              Font shape `OT1/ptm/b/n' tried instead on input line
71.
LaTeX Font Info:    Font shape `OT1/ptm/bx/n' in size <6> not available
```

```
(Font)              Font shape `OT1/ptm/b/n' tried instead on input line
71.
LaTeX Font Info:    Font shape `OT1/ptm/bx/n' in size <9> not available
(Font)              Font shape `OT1/ptm/b/n' tried instead on input line
71.
LaTeX Font Info:    Font shape `OT1/ptm/bx/n' in size <7> not available
(Font)              Font shape `OT1/ptm/b/n' tried instead on input line
71.
LaTeX Font Info:    Font shape `OT1/ptm/bx/n' in size <5> not available
(Font)              Font shape `OT1/ptm/b/n' tried instead on input line
71.
LaTeX Font Info:    Font shape `OMS/ptm/m/n' in size <7> not available
(Font)              Font shape `OMS/cmsy/m/n' tried instead on input line
71.

Overfull \hbox (2.70622pt too wide) in paragraph at lines 74--75
[]\OT1/ptm/m/n/9 Stochastic Gra-di-ent De-scent (SGD) with vari-ance re-
duc-tio
n tech-niques
 []


Overfull \hbox (0.16408pt too wide) in paragraph at lines 76--77
\OT1/ptm/b/n/9 Keywords:\OT1/ptm/m/n/9 Stochastic gra-di-ent de-scent,
Vari-anc
e re-duc-tion, Asyn-chronous com-
 []

LaTeX Font Info:    Font shape `OT1/ptm/bx/n' in size <12> not available
(Font)              Font shape `OT1/ptm/b/n' tried instead on input line
80.
[1


] [2] [3] [4]
Package hyperref Info: bookmark level for unknown algorithm defaults to 0
on in
put line 137.
[5] [6] [7]

! LaTeX Error: File `figures/evaluation_accelerated_factor' not found.

See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
 ...

l.242 ...]{figures/evaluation_accelerated_factor}}

I could not locate the file with any of these extensions:
.eps,.ps,.eps.gz,.ps.gz,.eps.Z,.mps
Try typing  <return>  to proceed.
If that doesn't work, type  X <return>  to quit.


! LaTeX Error: File `figures/evaluation_random_strategy' not found.

See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
```

```
 ...

l.248 ...tion_random_strategy_accelerated_factor}}

I could not locate the file with any of these extensions:
.eps,.ps,.eps.gz,.ps.gz,.eps.Z,.mps
Try typing  <return>  to proceed.
If that doesn't work, type  X <return>  to quit.


! LaTeX Error: File `figures/evaluation_random_strategy_wait_time' not
found.

See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
 ...

l.249 ...re_evaluation_random_strategy_wait_time}}

I could not locate the file with any of these extensions:
.eps,.ps,.eps.gz,.ps.gz,.eps.Z,.mps
Try typing  <return>  to proceed.
If that doesn't work, type  X <return>  to quit.

[8] [9]
Underfull \vbox (badness 1152) has occurred while \output is active []

[10]
Overfull \hbox (8.57199pt too wide) in paragraph at lines 300--301
[]\OT1/ptm/b/n/10 DisSVRG-without-tricks: \OT1/ptm/m/n/10 DisSVRG is
eval-u-ate
d with-out any an op-ti-miza-tion tricks.
 []

[11]

! LaTeX Error: File `figures/evaluation1_converage' not found.

See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
 ...

l.308 ...umnwidth]{figures/evaluation1_converage}}

I could not locate the file with any of these extensions:
.eps,.ps,.eps.gz,.ps.gz,.eps.Z,.mps
Try typing  <return>  to proceed.
If that doesn't work, type  X <return>  to quit.


Overfull \hbox (9.98987pt too wide) in paragraph at lines 308--308
[]\OT1/ptm/m/n/9 (a)
 []


Overfull \hbox (31.83261pt too wide) in paragraph at lines 308--308
\OT1/ptm/m/it/9 YearPre-
 []
```

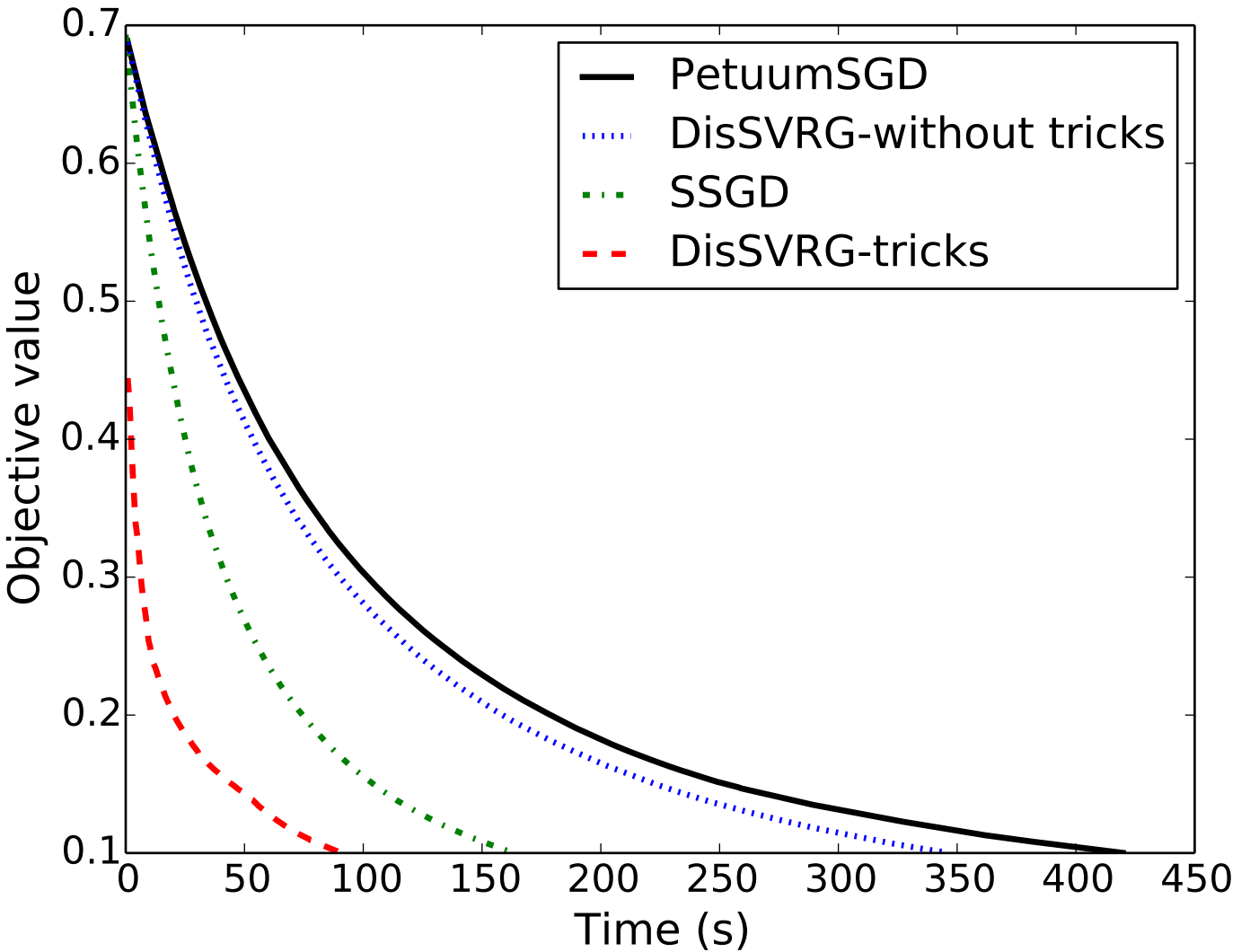
```
Overfull \hbox (31.99469pt too wide) in paragraph at lines 308--308
\OT1/ptm/m/it/9 dictMSD
 []


! LaTeX Error: File `figures/evaluation1_dna_converage' not found.

See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
 ...

l.309 ...idth]{figures/evaluation1_dna_converage}}

I could not locate the file with any of these extensions:
.eps,.ps,.eps.gz,.ps.gz,.eps.Z,.mps
Try typing  <return>  to proceed.
If that doesn't work, type  X <return>  to quit.


Overfull \hbox (10.49393pt too wide) in paragraph at lines 309--309
[]\OT1/ptm/m/n/9 (b)
 []


Overfull \hbox (13.5pt too wide) in paragraph at lines 309--309
\OT1/ptm/m/it/9 dna
 []


! LaTeX Error: File `figures/evaluation2_speedup' not found.

See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
 ...

l.316 ...olumnwidth]{figures/evaluation2_speedup}}

I could not locate the file with any of these extensions:
.eps,.ps,.eps.gz,.ps.gz,.eps.Z,.mps
Try typing  <return>  to proceed.
If that doesn't work, type  X <return>  to quit.


Overfull \hbox (9.98987pt too wide) in paragraph at lines 316--316
[]\OT1/ptm/m/n/9 (a)
 []


Overfull \hbox (31.83261pt too wide) in paragraph at lines 316--316
\OT1/ptm/m/it/9 YearPre-
 []


Overfull \hbox (31.99469pt too wide) in paragraph at lines 316--316
\OT1/ptm/m/it/9 dictMSD
 []
```

```
! LaTeX Error: File `figures/evaluation2_speedup_dna' not found.

See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
 ...

l.317 ...nwidth]{figures/evaluation2_speedup_dna}}

I could not locate the file with any of these extensions:
.eps,.ps,.eps.gz,.ps.gz,.eps.Z,.mps
Try typing  <return>  to proceed.
If that doesn't work, type  X <return>  to quit.


Overfull \hbox (10.49393pt too wide) in paragraph at lines 317--317
[]\OT1/ptm/m/n/9 (b)
 []


Overfull \hbox (13.5pt too wide) in paragraph at lines 317--317
\OT1/ptm/m/it/9 dna
 []


! LaTeX Error: File `figures/evaluation3_delay' not found.

See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
 ...

l.324 ...\columnwidth]{figures/evaluation3_delay}}

I could not locate the file with any of these extensions:
.eps,.ps,.eps.gz,.ps.gz,.eps.Z,.mps
Try typing  <return>  to proceed.
If that doesn't work, type  X <return>  to quit.


Overfull \hbox (9.98987pt too wide) in paragraph at lines 324--324
[]\OT1/ptm/m/n/9 (a)
 []


Overfull \hbox (31.83261pt too wide) in paragraph at lines 324--324
\OT1/ptm/m/it/9 YearPre-
 []


Overfull \hbox (31.99469pt too wide) in paragraph at lines 324--324
\OT1/ptm/m/it/9 dictMSD
 []


! LaTeX Error: File `figures/evaluation3_delay_dna' not found.

See the LaTeX manual or LaTeX Companion for explanation.
```

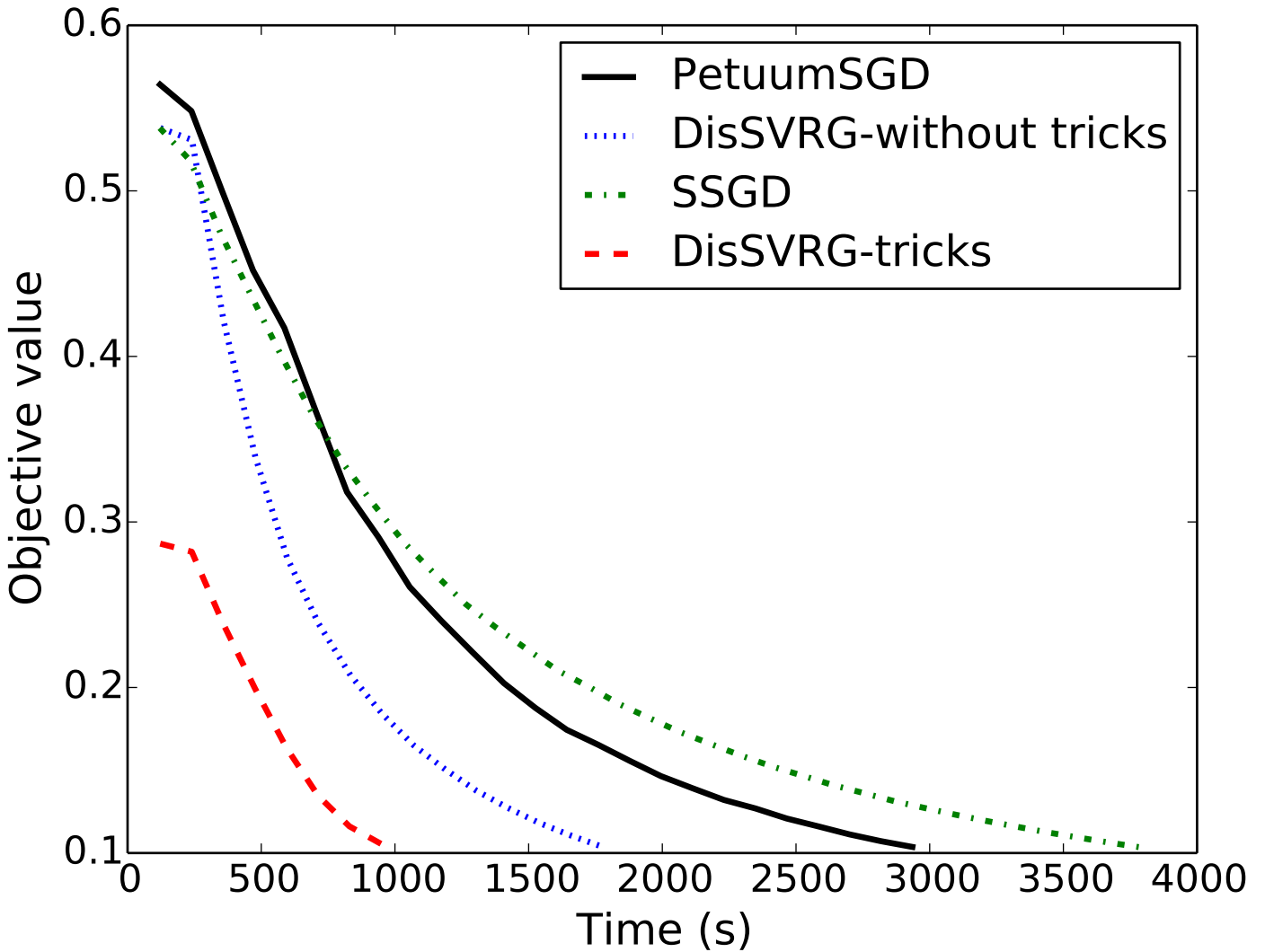
```
Type  H <return>  for immediate help.
 ...

l.325 ...umnwidth]{figures/evaluation3_delay_dna}}

I could not locate the file with any of these extensions:
.eps,.ps,.eps.gz,.ps.gz,.eps.Z,.mps
Try typing  <return>  to proceed.
If that doesn't work, type  X <return>  to quit.


Overfull \hbox (10.49393pt too wide) in paragraph at lines 325--325
[]\OT1/ptm/m/n/9 (b)
 []


Overfull \hbox (13.5pt too wide) in paragraph at lines 325--325
\OT1/ptm/m/it/9 dna
 []

[12]
(./asynchronsou and distributed stochastic gradient descent with variance
reduc
tion.bbl [13]
Underfull \hbox (badness 10000) in paragraph at lines 85--87
[]\OT1/ptm/m/n/9 Zhao, S.Y., Li, W.J.: Fast asyn-chronous par-al-lel
stochas-ti
c gra-di-ent de-cent.
 []

)
Package atveryend Info: Empty hook `BeforeClearDocument' on input line
347.
[14]
Package atveryend Info: Empty hook `AfterLastShipout' on input line 347.

(./asynchronsou and distributed stochastic gradient descent with variance
reduc
tion.aux)
Package atveryend Info: Empty hook `AtVeryEndDocument' on input line 347.
Package atveryend Info: Executing hook `AtEndAfterFileList' on input line
347.
Package rerunfilecheck Info: File `"asynchronsou and distributed
stochastic gra
dient descent with variance reduction".out' has not changed.
(rerunfilecheck)             Checksum:
96EB45D6009875143487DA4562EE609F;42.

LaTeX Warning: There were multiply-defined labels.

Package atveryend Info: Empty hook `AtVeryVeryEnd' on input line 347.
 )
Here is how much of TeX's memory you used:
 7408 strings out of 493118
 106015 string characters out of 6139011
 203820 words of memory out of 5000000
 10700 multiletter control sequences out of 15000+600000
 44420 words of font info for 123 fonts, out of 8000000 for 9000
```

```
 1141 hyphenation exceptions out of 8191
 31i,9n,26p,2092b,460s stack positions out of
5000i,500n,10000p,200000b,80000s

Output written on "asynchronsou and distributed stochastic gradient
descent wit
h variance reduction.dvi" (14 pages, 86344 bytes).
```

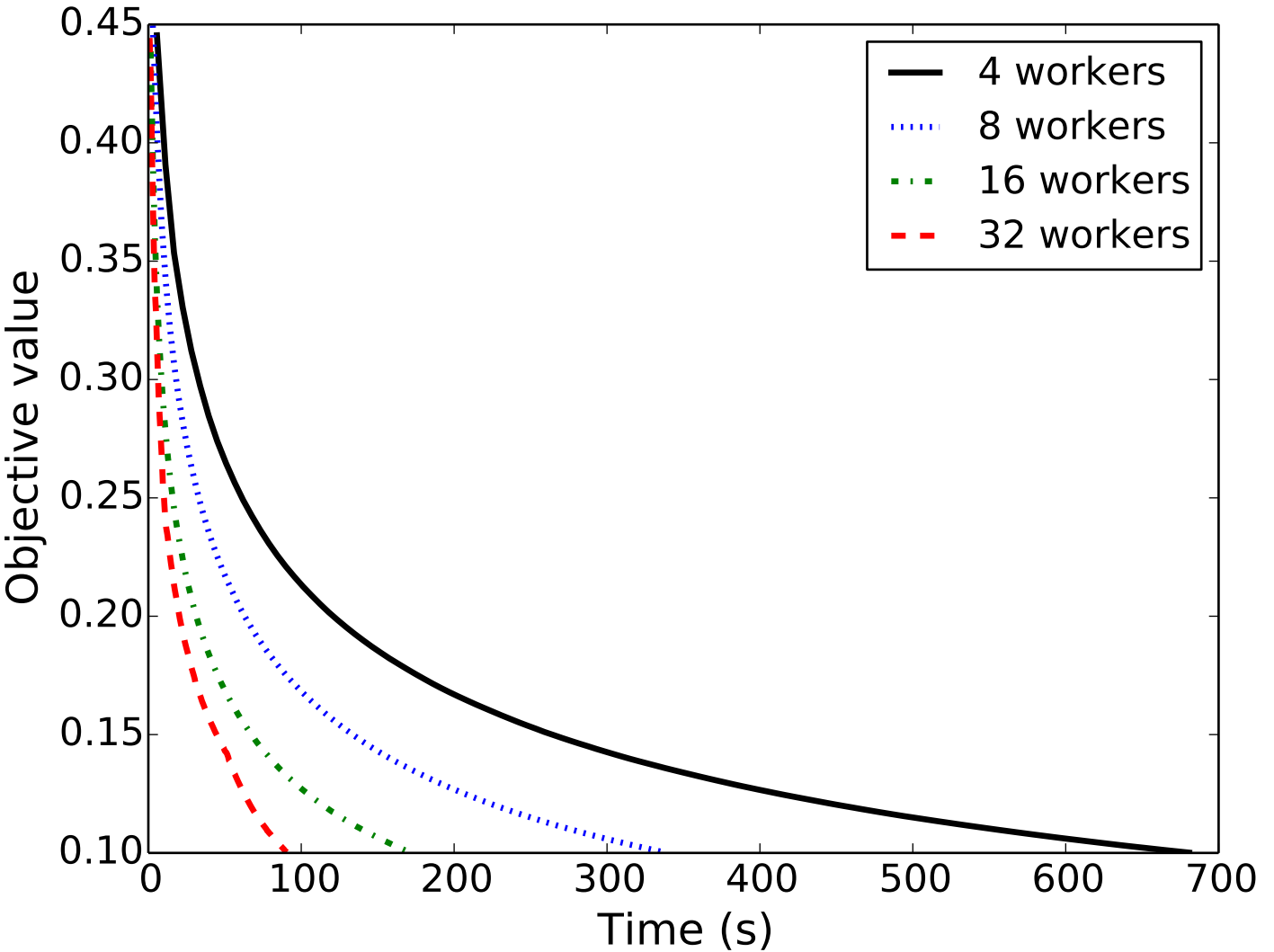
evaluation1_converage.pdf



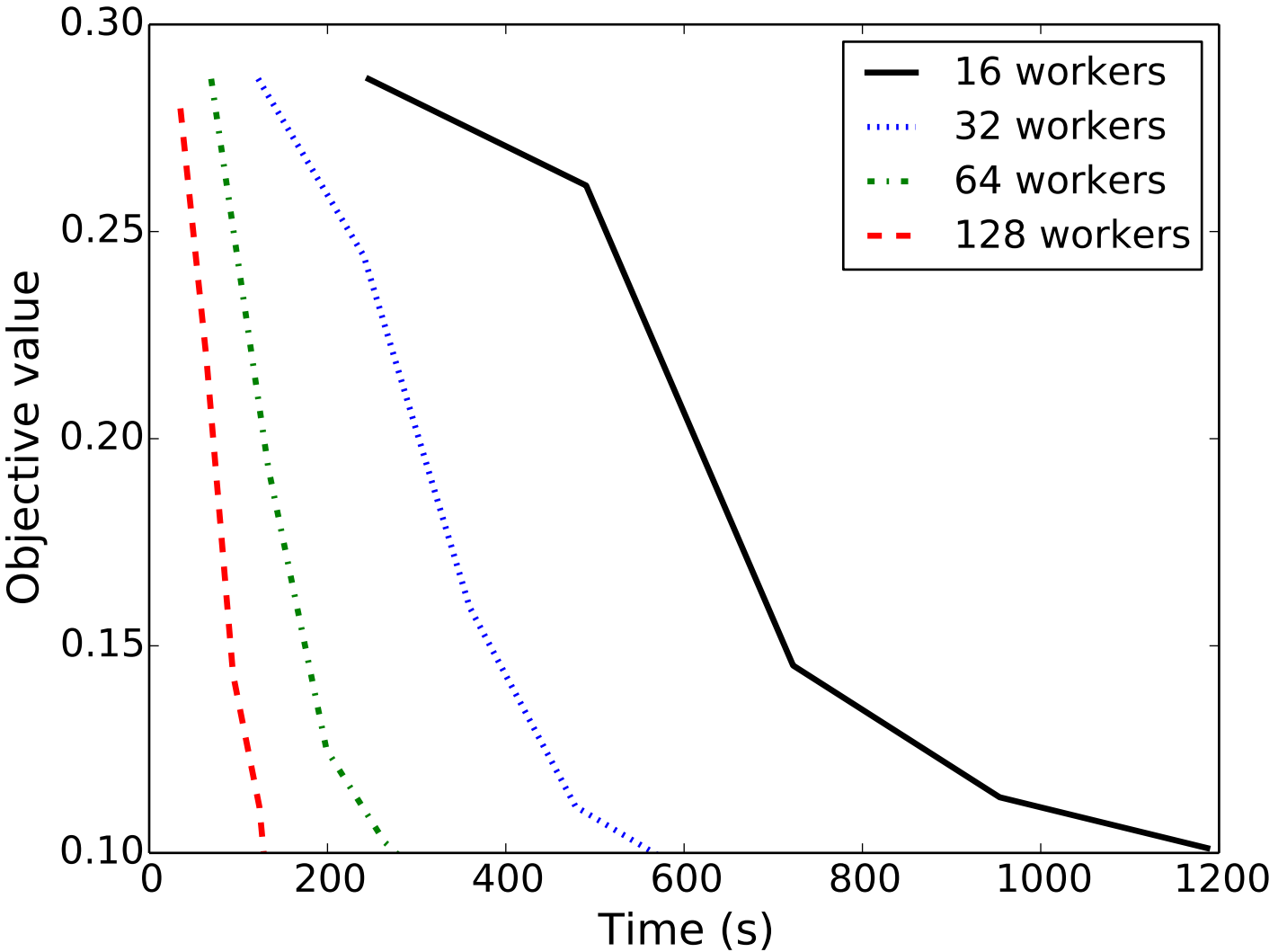
evaluation1_dna_converage.pdf



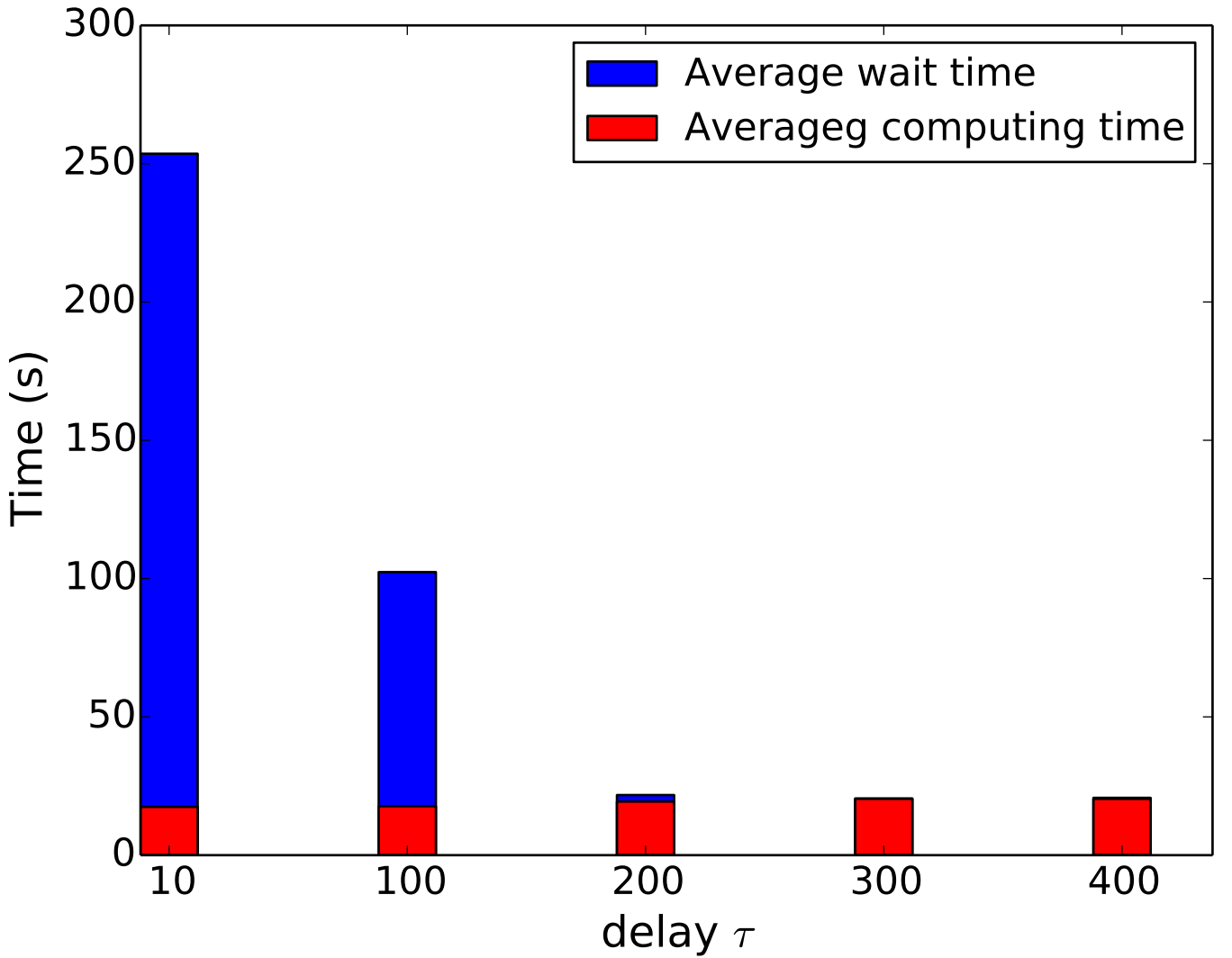
evaluation2_speedup.pdf



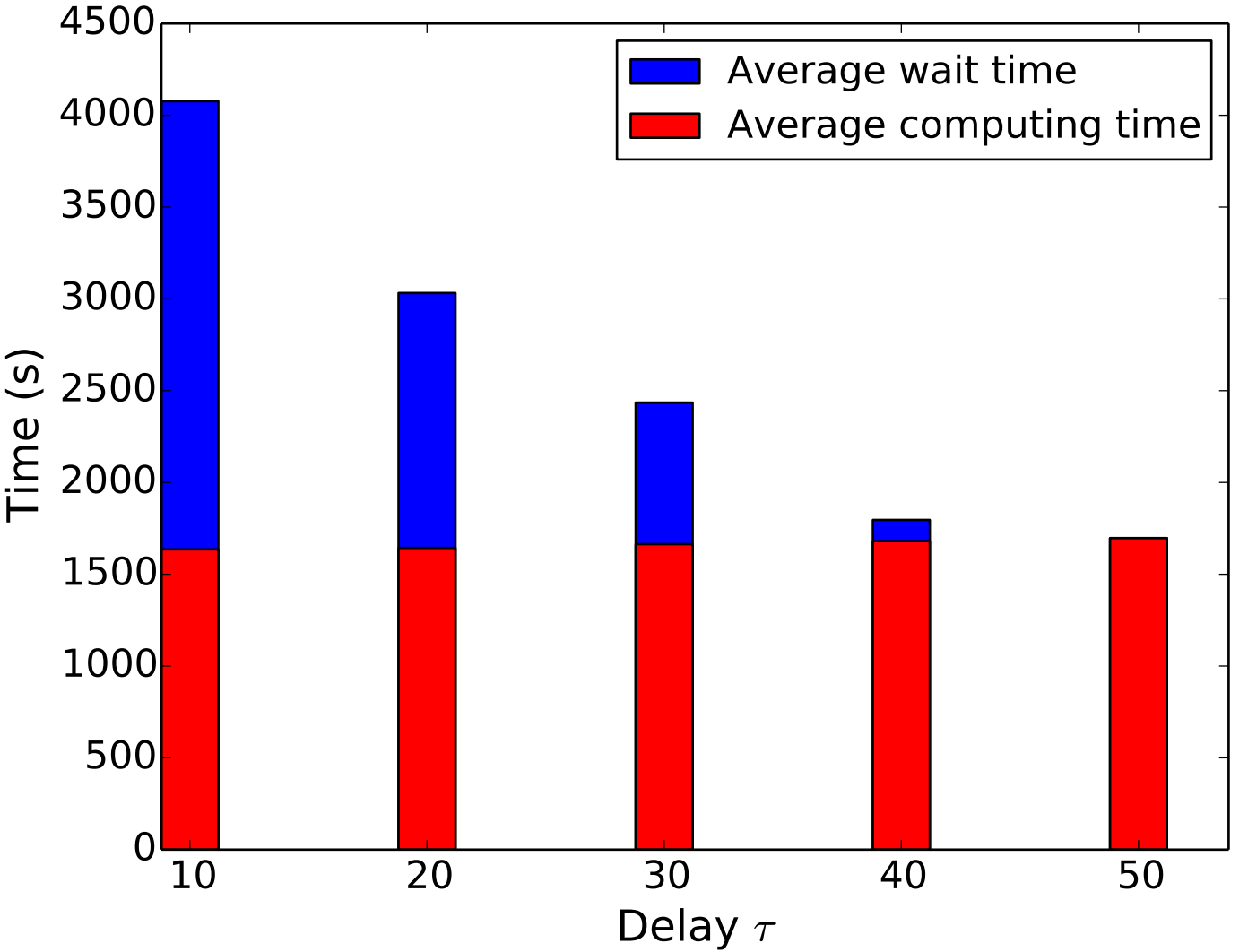
evaluation2_speedup_dna.pdf



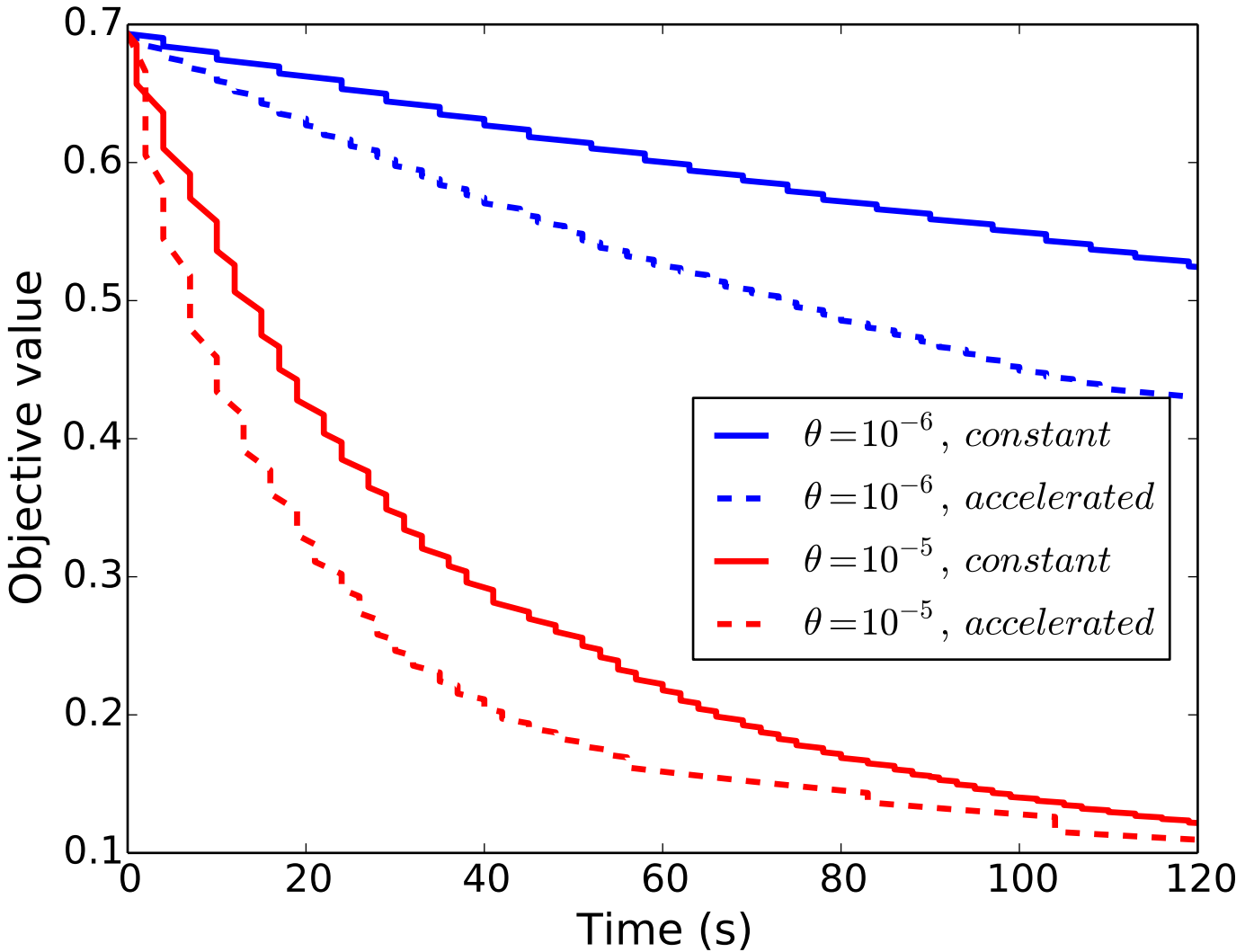
evaluation3_delay.pdf



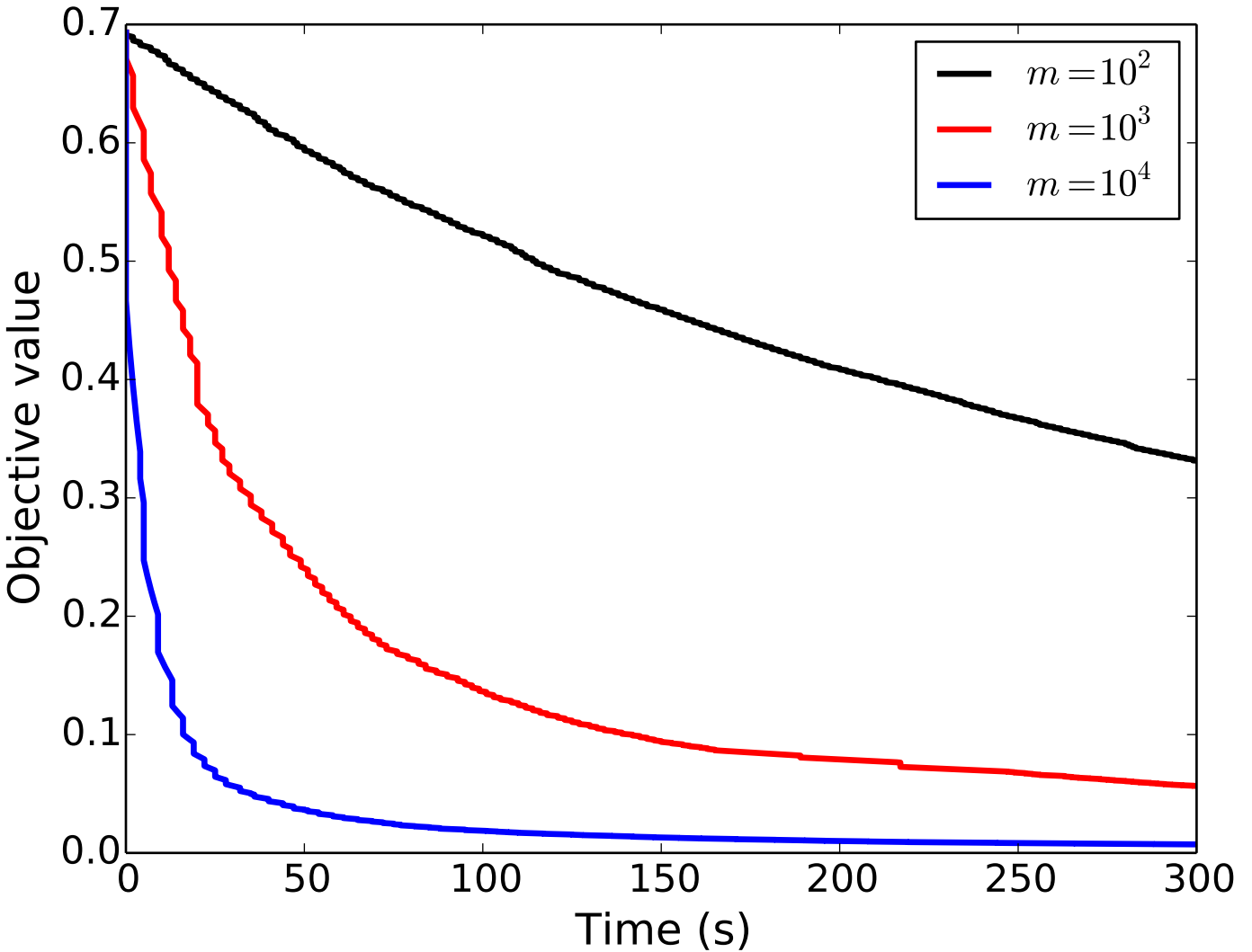
evaluation3_delay_dna.pdf



evaluation_accelerated_factor.pdf



evaluation_random_strategy.pdf



evaluation_random_strategy_wait_time.pdf