POPSGD: DECENTRALIZED STOCHASTIC GRADIENT DESCENT IN THE POPULATION MODEL

ABSTRACT

The population model is a standard way to represent large-scale decentralized distributed systems, in which agents with limited computational power interact in randomly chosen pairs, in order to collectively solve global computational tasks. In contrast with synchronous gossip models, nodes are anonymous, lack a common notion of time, and have no control over their scheduling. In this paper, we examine whether large-scale distributed optimization can be performed in this extremely restrictive setting.

We introduce and analyze a natural decentralized variant of stochastic gradient descent (SGD), called PopSGD, in which every node maintains a local parameter, and is able to compute stochastic gradients with respect to this parameter. Every pair-wise node interaction performs a stochastic gradient step at each agent, followed by averaging of the two models. We prove that, under standard assumptions, SGD can converge even in this extremely loose, decentralized setting, for both convex and non-convex objectives. Moreover, surprisingly, in the former case, the algorithm can achieve *linear* speedup in the number of nodes n. Our analysis leverages a new technical connection between decentralized SGD and randomized load-balancing, which enables us to tightly bound the concentration of node parameters. We validate our analysis through experiments, showing that PopSGD can achieve convergence and speedup for large-scale distributed learning tasks in a supercomputing environment.

1 INTRODUCTION

One key enabler of the extremely rapid recent progress of machine learning has been *distribution*: the ability to efficiently split computation among multiple nodes or devices, in order to share the high computational loads of training large models, and therefore reduce end-to-end training time. Distributed machine learning has become commonplace, and it is not unusual to encounter systems which distribute model training among tens or even hundreds of nodes. In this paper, we take this trend to the extreme, and ask: would it be possible to distribute basic optimization procedures such as stochastic gradient descent (SGD) to *thousands* of agents? How could the dynamics be implemented in such a large-scale setting, and what would be with the resulting convergence and speedup behavior?

To get some intuition, let us consider the classical *data-parallel* distribution strategy for SGD (Bottou, 2010). We are in the classical empirical risk minimization setting, where we have a set of samples S from a distribution, and wish to minimize the function $f : \mathbb{R}^d \to \mathbb{R}$, which is the average of losses over samples from S by finding $x^\star = \text{ argmin }_x \sum_{s \in S} f_s(x)/|S|$. Assume that we have P compute nodes which can process samples in parallel. Data-parallel SGD consists of parallel iterations, in which each node computes the gradient for one sample, followed by a gradient exchange. Globally, this leads to the iteration:

$$x_{t+1} = x_t - \eta_t \sum_{i=1}^{P} \widetilde{g}_t^i(x_t),$$

where η_t is the learning rate, x_t is the value of the global parameter, initially 0^d, and $\widetilde{g}_t^i(x_t)$ is the stochastic gradient with respect to the parameter obtained by node i at time t.

When extending this strategy to high node counts, two major bottlenecks are *communication* and *synchronization*. In particular, to maintain a consistent view of the parameter x_t, the nodes would need to broadcast and receive all gradients, and would need to synchronize with all other nodes, at the end of every iteration. Recently, a tremendous amount of work has been dedicated to address these two barriers. In particular, there has been significant progress on *communication-reduced* variants of SGD (e.g. Seide et al. (2014); Strom (2015); Alistarh et al. (2017b); Wen et al. (2017); Aji & Heafield (2017); Dryden et al. (2016); Grubic et al. (2018)), *asynchronous* variants (e.g. Recht et al.

(2011); Sa et al. (2015); Duchi et al. (2015); Alistarh et al. (2018b)), as well as *large-batch* or *periodic model averaging* methods, which aim to reduce the *frequency* of communication(e.g. Goyal et al. (2017); You et al. (2017) and Chen & Huo (2016); Stich (2018)), or even *decentralized synchronous* variants(e.g. Lian et al. (2017a); Tang et al. (2018); Koloskova et al. (2019)). Using such techniques, it is possible to scale SGD to hundreds of nodes, even for complex objectives such as the training of deep neural networks. However, in systems with node counts in the thousands or larger, it is infeasible to assume that all nodes can efficiently synchronize into global iterations, or that they can directly broadcast messages to all other nodes.

Instead, in this paper we will consider the classic *population* model of distributed computing (Angluin et al., 2006), which is defined as follows. We are given a population of n compute agents, each with its own input, which cooperate to perform some globally meaningful computation with respect to their inputs. Interactions occur *pairwise*, where the two interaction partners are *randomly chosen* in every step. Thus, algorithms are specified in terms of the agents' state transitions upon an interaction. The basic unit of time is a single pairwise interaction between two nodes, whereas global (parallel) time is measured as the total number of interactions divided by n, the number of nodes. Parallel time corresponds intuitively to the *average* number of interactions per node to reach convergence. Population protocols have a rich history in distributed computing (e.g. Angluin et al. (2006; 2007; 2008a;b;c); Alistarh et al. (2017a; 2018a)), and are standard in modelling distributed systems with millions or billions of nodes, such as Chemical Reaction Networks (CRNs) (Bower & Bolouri, 2004; Chen et al., 2017) and synthetic DNA strand displacement cascades (Chen et al., 2013). The key difference between population protocols and the synchronous gossip models (e.g. Xiao & Boyd (2004); Lian et al. (2017a); Koloskova et al. (2019)) previously used to analyze decentralized SGD is that nodes are *not synchronized*: since pairwise interactions are uniform random, there are no global rounds, and nodes lack a common notion of time.

While the population model is a theoretical construct, we show that it can be efficiently mapped to large-scale super-computing scenarios, with large numbers of compute nodes connected by a fast point-to-point interconnect, where we can avoid the high costs of global synchronization.

An immediate instantiation of SGD in the population model would be to initially assign one sample s^i from the distribution to each node i, and have each node maintain its own parameter estimate x^i. Whenever two nodes interact, they exchange samples, and each performs a gradient update with respect to the other's sample. If we assume interaction pairs are *uniform random* (with replacement), each node would obtain a stochastic gradient upon each interaction, and therefore each model would converge locally. However, this instance would not have any parallel speedup, since the SGD instances at each node are essentially independent.

In this context, we propose a natural change to the above procedure, by which interacting nodes i and j first perform a gradient step, and then also *average* their resulting models upon every interaction. Effectively, if node i interacts with node j, node i's updated model becomes

$$x^i \leftarrow \frac{x^i + x^j}{2} - \eta^i \frac{\widetilde{g}^i(x^i) + \widetilde{g}^j(x^j)}{2}, \tag{1.1}$$

where j is the interaction partner, and the stochastic gradients $\widetilde{g}^i$ and $\widetilde{g}^j$ are taken with respect to each other's samples. The update for node j is symmetric. In this paper, we analyze a variant of the above protocol, which we call PopSGD, in the population protocol model.

We show that, perhaps surprisingly, this simple decentralized SGD averaging dynamic provides strong convergence guarantees for both convex and non-convex objectives. First, we prove that, under standard convexity and smoothness assumptions, PopSGD has convergence speedup that *linear* in the number of nodes n. Specifically, if μ_t is the *parameter* average over all node models at *parallel* time $t \geq 0$, and y_T is the average over the sequence $(\mu_t)_{t=1,T}$, then, for large enough parallel time T, our main result is that

$$\mathbb{E}[f(y_T) - f(x^*)] = \mathcal{O}\left(\sigma^2/(nT)\right), \tag{1.2}$$

which is n times faster than the sequential variant given the same number of SGD steps per node. (Please see Theorem 4.1 for the exact formulation, including bias and variance terms, and for an extended discussion.) This result suggests that, even though interactions occur only pairwise, uniformly at random, and in an uncoordinated manner, as long as the convergence time is large enough to amortize the information propagation, the protocol enjoys the full parallel speedup of mini-batch SGD with a batch size proportional to the number of nodes. While speedup behaviour has

been observed in various *synchronous* models– e.g. Lian et al. (2017a); Stich (2018); Koloskova et al. (2019), or for complex *accelerated* algorithms (Hendrikx et al., 2018)–we are the first to show that SGD does not require the existence of globally synchronized rounds or global communication.

Central to our analytic approach is a new technical connection between averaging decentralized SGD and the line of research studying *load-balancing* processes in theoretical computer science (e.g. Azar et al. (1999); Mitzenmacher (2000); Talwar & Wieder (2007); Peres et al. (2015a)). Intuitively, we show PopSGD can be viewed as a composition between a set of instances of SGD–each corresponding to one of the local parameters x^i–which are loosely coupled via pairwise averaging, whose role is to "balance" the models by keeping them well concentrated around their mean, despite the random nature of the pairwise interactions. Our analysis characterizes this concentration, showing that, in essence, the averaging process propagates enough information to globally "simulate" SGD with a batch of size $\Theta(n)$, even though communication is only performed pairwise. We emphasize that the convexity of the objective function in isolation would not be sufficient to prove this fact, see e.g. Stich (2018). Along the way, we overcome non-trivial technical difficulties, such as the lack of a common notion of time among nodes, or the fact that, due to the structure of SGD, this novel load-balancing process exhibits non-trivial correlations within the same round.

On the practical side, we provide convergence and speedup results using an efficient implementation of PopSGD using Pytorch/MPI applied to regression tasks, but also to the standard CIFAR/ImageNet classification tasks for deployments on a multi-GPU nodes, and on the CSCS Piz Daint supercomputer (Piz). Experiments predictably confirm the scalability of PopSGD to large node counts. More surprisingly, we also observe an improvement in *convergence* versus number of SGD iterations per model at higher node counts, in both convex and non-convex settings. In particular, using PopSGD, we are able to train the ResNet18 and ResNet50 (He et al., 2016) models to full accuracy using only $1/8$ the number of SGD updates per model, compared to the sequential baseline, resulting in fast convergence with nearly linear scalability.

Related Work. The study of decentralized optimization algorithms dates back to Tsitsiklis (1984), and is related to the study of *gossip* algorithms for information dissemination (Kempe et al., 2003; Xiao & Boyd, 2004). The distinguishing feature of this setting is that information is shared in the absence of a coordinator, between nodes and their neighbors. Several classic algorithms have been ported and analyzed in the gossip setting, such as subgradient methods for convex objectives (Nedic & Ozdaglar, 2009; Johansson et al., 2009; Shamir & Srebro, 2014) or ADMM (Wei & Ozdaglar, 2012; Iutzeler et al., 2013). References Lian et al. (2017a;b); Assran et al. (2018) consider SGD-type algorithms in the non-convex setting, while references Tang et al. (2018); Koloskova et al. (2019) analyze the use of quantization in the gossip setting.

The key difference between the gossip model discussed above and the population model we analyze is that, in the gossip model, time is divided into *global rounds*, which are assumed to be consistent across nodes. In each round, each node broadcasts and receives with all neighbors: analytically, this synchrony assumption allows the global evolution of the system to be represented in terms of the "gossip" (communication/contact) matrix (see e.g. Xiao & Boyd (2004); Koloskova et al. (2019); Lian et al. (2017b)). This matrix characterization is not possible in the population model: nodes do not share a notion of time or rounds, as communication steps correspond to individual interactions. If we consider sequences of n consecutive interactions, due to scheduler randomness, some nodes will interact several times, while others may not interact at all during such an interval. For this reason, our analysis makes use of fine-grained potential arguments, rather than a global matrix iteration. There do exist instances in the literature which consider dynamic interaction models. First, Nedic et al. Nedic et al. (2017) present a gradient tracking algorithm in a different dynamic graph model; however, their results would *not* translate to the PP model, as they assume a dynamically changing but *simple* graph in each iteration: by contrast, merging together multiple interaction rounds from the PP model could result in a multi-graph. Further, Hendrickx et al. Hendrikx et al. (2018) achieve exponential convergence rates in a gossip model where transmissions are synchronized across *edges*; however, the algorithm they consider is a more complex instance of accelerated coordinate descent, and is therefore quite different from the simple dynamics we consider. Importantly, neither reference considers large-scale deployments for non-convex objectives (in particular, neural networks).

It is interesting to contrast our work with that of Lian et al. (2017b), who assume a synchronized gossip model, but allow for asynchrony, in the sense that nodes can see stale variants of their neighbors' messages. To our understanding, the models are not directly comparable, and in particular their results cannot be applied to our setting. This is because they rely on a variant of the global

matrix iteration (albeit based on delayed views). They consider challenging smooth non-convex objectives, but do not show any speedup due to parallelization. We provide similar guarantees for non-convex PopSGD, but are able to show *linear* speedup in the convex case. In our setting, since interaction pairs are chosen randomly, there can be significant local variability between the interaction rates of the nodes, and so the matrix iteration would not be applicable.

The population model, introduced by Angluin et al. (2006), is one of the standard models of distributed computing, and has proved a useful abstraction for modeling settings from wireless sensor networks (Perron et al., 2009; Draief & Vojnovic, 2012), to gene regulatory networks (Bower & Bolouri, 2004), and chemical reaction networks (Chen et al., 2017). While there has been significant work on algorithms for specific *tasks*, such as *majority (consensus)*, *leader election*, and *approximate counting*, we are the first to consider optimization tasks in the population model. Potential analysis is a common tool in load balancing (e.g. Talwar & Wieder (2007); Peres et al. (2015a)), which we adapt for our setting. The three key departures from the load balancing literature are that 1) in the SGD setting, the weights (gradients) are correlated with the loads of the bins (the models); 2) in the SGD setting, the magnitude of the weights is diminishing (due to the learning rate), which requires a continuous re-evaluation of the balancing objective; 3) models are multi-dimensional, whereas in the classical formulation the balanced items are single-dimensional.

2 PRELIMINARIES

The Population Protocol Model. We consider a variant of the population protocol model which consists of a set of $n \geq 2$ anonymous agents, or nodes, each executing a local state machine. (Our analysis will make use of node identifiers only for exposition purposes.) Since our application is continuous optimization, we will assume that the agents' states may store real numbers. The execution proceeds in discrete *steps*, where in each step a new pair of agents is selected uniformly at random to interact from the set of all possible pairs. (To preserve symmetry of the protocols, we will assume that a process may interact with a copy of itself, with low probability.) Each of the two chosen agents updates its state according to a state update function, specified by the algorithm. The basic unit of time is a single pairwise interaction between two nodes. Notice however that in a real system $\Theta(n)$ of these interactions could occur in parallel. Thus, a standard global measure is *parallel time*, defined as the total number of interactions divided by n, the number of nodes. Parallel time intuitively corresponds to the *average* number of interactions per node to convergence.

Stochastic Optimization. We assume that the agents wish to minimize a d-dimensional, differentiable and strongly convex function $f : \mathbb{R}^d \to \mathbb{R}$ with parameter $\ell > 0$, that is:

$$(x-y)^T (\nabla f(x) - \nabla f(y)) \geq \ell \|x-y\|^2, \forall x, y \in \mathbb{R}^d. \tag{2.1}$$

Specifically, we will assume the empirical risk minimization setting, in which agents are given access to a set of data samples $S = \{s_1, \ldots, s_m\}$ coming from some underlying distribution $\mathcal{D}$, to a function $f_i : \mathbb{R}^d \to \mathbb{R}$ which encodes the loss of the argument at the sample s_i. The goal of the agents is to converge on a model x^* which minimizes the empirical loss, that is

$$x^* = \text{argmin}_x f(x) = \text{argmin}_x (1/m) \sum_{i=1}^{m} f_i(x). \tag{2.2}$$

In this paper, we assume that the agents employ these samples to run a decentralized variant of SGD, described in detail in the next section. For this, we will assume that agents have access to *stochastic gradients* $\widetilde{g}$ of the function f, which are functions such that $\mathbb{E}[\widetilde{g}(x)] = \nabla f(x)$. Stochastic gradients can be computed by each agent by sampling i.i.d. the distribution D, and computing the gradient of f at θ with respect to that sample. In the population model, we could implement this by procedure either by allowing agents to sample in each step, or by assigning a sample s_i to each agent i, and having agents compute gradients of their local models with respect to each others' samples. We will assume the following about the gradients:

- **Smooth Gradients**: The gradient $\nabla f(x)$ is L-Lipschitz continuous for some $L > 0$, i.e. for all $x, y \in \mathbb{R}^d$:

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x-y\|. \tag{2.3}$$

- **Bounded Variance**: The variance of the stochastic gradients is bounded by some $\sigma^2 > 0$, i.e. for all $x \in \mathbb{R}^d$:

$$\mathbb{E}\left\|\widetilde{g}(x) - \mathbb{E}[\widetilde{g}(x)]\right\|^2 \leq \sigma^2. \tag{2.4}$$

- **Bounded Second Moment**: The second moment of the stochastic gradients is bounded by some $M^2 > 0$, i.e. for all $x \in \mathbb{R}^d$:

$$\mathbb{E}\|\widetilde{g}(x)\|^2 \leq M^2. \tag{2.5}$$

3 The Population SGD Algorithm

Algorithm Description. We now describe a decentralized variant of SGD, designed to be executed by a population of n nodes, interacting in uniform random pairs as per the population protocol model. We assume that each node i has access to local stochastic gradients $\widetilde{g}^i$, and maintains a model estimate X^i, as well as a local learning rate η^i. For simplicity, we will assume that this initial estimate is 0^d at each agent, although its value may be arbitrary. We detail the way in which the learning rates are updated below. Specifically, upon every interaction, the interacting agents i and j perform the following steps:

```
% i and j are chosen uniformly at random, with replacement
upon each interaction between agents i and j
    % each agent performs a local SGD step
    X^i ← X^i − η^i g̃^i(X^i)
    X^j ← X^j − η^j g̃^j(X^j)
    % agents average their estimates coordinate-wise
    avg ← (X^i + X^j)/2
    X^i ← avg
    X^j ← avg
```

Algorithm 1: Population SGD pseudocode for each interaction between arbitrary nodes i and j.

We are interested in the convergence of local models : $X^1, X^2, ..., X^n$ after T interactions occur in total. For the theoretical reasons, in the case when f is convex, we derive convergence for y_T which is weighted average of average values of local models per step(See Theorem 4.1). In the beginning of section 5 we show that by performing single global averaging step at time step $0 \leq t < T$, which is carefully chosen from specified distribution, we can make sure that in expectation local models converge with the same rate as y_T.

Estimating Time and the Learning Rate. In parallel with the above algorithm, each agent maintains a local time value V^i, which is estimated using a local "phase clock" protocol. These local times are defined and updated as follows. The initial value at each agent is $V^i = 0$. Upon each interaction, the interacting agents i and j exchange their time values. The agent with a *lower* time value, say $V^i < V^j$, will increment its value by 1. The other agent keeps its local value unchanged. (We break ties arbitrarily.) Although intuitively simple, the above procedure provides strong probabilistic guarantees on how far individual values may stray from the mean: with high probability,[1] all the estimates V^i are in the interval $[t/n - c\log T, t/n + c\log T]$, where c is a constant.

Given the current value of V^i at the agent, the value of the learning rate at i is simply $\eta^i = b/(nV^i + a)$, where a and b are constant parameters which we will fix later. This will ensure that the gap between two agents' learning rates will be in the interval $[0.5, 2]$, w.h.p. (See Lemma 4.2.)

4 The Convergence of PopSGD in the Convex Case

This section is dedicated to proving that the following result holds with high probability:

Theorem 4.1. *Let f be an L-smooth, ℓ-strongly convex function satisfying conditions (2.3)—(2.5), whose minimum $x^\star$ we are trying to find via the PopSGD procedure given in Algorithm 1. Let the learning rate for process i at local time $t^i = nV_t^i$ be $\eta_t^i = b/(t^i + a)$, where $a = \max(2cn\log T, 18n, 256L/\ell)$ and $b = 4n/\ell$ are fixed(for some constant c). Let the sequence of weights w_t be given by $w_t = (a+t)^2$. Define $\mu_t = \sum_{i=1}^n X_t^i$, $S_T = \sum_{t=0}^{T-1} w_t \geq \frac{1}{3}T^3$ and*

[1] An event holds *with high probability (w.h.p.)* if it occurs with probability $\geq 1 - 1/T^\gamma$, for constant $\gamma > 0$ and the total number of interactions - T.

$y_T = \frac{1}{S_T}\sum_{t=0}^{T-1} w_t \mu_t$. *Then, for any time T, we have with probability* $1 - O(1/\text{ poly } T)$ *that*

$$\mathbb{E}[f(y_T) - f(x^*)] \leq \frac{a^3\ell}{2S_T}\|\mu_0 - x^*\|^2 + \frac{64T(T+2a)}{\ell S_T}\sigma^2 + \frac{9216Tn^2}{\ell^2 S_T}M^2 L. \tag{4.1}$$

Discussion. We first emphasize that, in the above bound, the time T refers to the number of interactions (as opposed to *parallel time*). With this in mind, we focus on the bound in the case where $T \gg n$, and the parameters M, L, and ℓ are assumed to be well-behaved. In this case, since $S_T \geq T^3/3$, the first and third terms are vanishing as T grows, and we get that convergence is dominated by the second term, which can be bounded as $O(\sigma^2/T)$. It is tempting to think that this is roughly the same rate as *sequential* SGD; however, our notion of time is *different*, as we are counting the total number of SGD steps executed in total *at all the models*. (In fact, the total number of SGD steps up to T is $2T$, since each interaction performs two SGD steps.)

It is interesting to interpret this from the perspective of an arbitrary local model. For this, notice that the *parallel time* corresponding to the number of total interactions T, which is by definition $T_p = T/n$, corresponds (up to constants) to the *average* number of interactions and SGD steps performed by each node up to time T. Thus, for any single model, convergence with respect to its number of performed SGD steps T_p would be $O(\sigma^2/(nT_p))$, which would correspond to running SGD with a batch size of n. Notice that this reduction in convergence time is solely thanks to the *averaging* step: in the absence of averaging, each local model would converge independently at a rate of $O(\sigma^2/T_p)$. We note that our discussion assumes a batch size of 1, but it would generalize to arbitrary batch size b, replacing σ^2 with σ^2/b. We note that, due to the concentration properties of the averaging process, the claim above can be extended to show convergence behavior for arbitrary individual models (instead of the average of models μ_T).

Proof Overview. The argument, given in full in the Additional Material, can be split into two steps. The first step aims to bound the variance of the local models X_t^i at each time t and node i with respect to the mean $\mu_t = \sum_i X_t^i/n$. It views this quantity as a potential Γ_t, which we show has supermartingale-like behavior, which enables us to bound its expected value as $O(\eta_t^2 n)$. This shows that the variance of the parameters is always bounded with respect to the number of nodes, but also, importantly, that it can be controlled via the learning rate. The key technical step here is Lemma 4.3, which provides a careful bound for the evolution of the potential at a step, by modelling SGD as a dynamic load balancing process: each interaction corresponds to a *weight generation* step (in which gradients are generated) and a *load balancing step*, in which the "loads" of the two nodes (corresponding to their model values) are balanced through averaging.

In the second step of the proof, we first bound the rate at which the mean μ_t converges towards x^*, where we crucially (and carefully) leverage the variance bound obtained above. This is our second key technical lemma. Next, with this in hand, we can apply a standard argument to characterize the rate at which the quantity $\mathbb{E}[f(y_T) - f(x^*)]$ converges towards 0.

Notation and Preliminaries. In this section, we overview the analysis of the PopSGD protocol. We begin with some notation. Recall that n is the number of nodes. We will analyze a sequence of *time steps* $t = 1, 2, \ldots, T$, each corresponding to an individual interaction between two nodes, which are usually denoted by i and j. We will consider that $T = O(\text{poly } n)$, and therefore w.h.p. results are assumed to hold throughout the execution. Recall the definition of *parallel time* $T_p = T/n$, where T counts the number of pairwise interactions. For any time t, define by $\eta_t = b/(a+t)$ the "true" learning rate at time t, where a and b are constants to be fixed later, such that $a \geq 2cn\log n$ for some constant c. We denote by x^* the optimum of the function f.

Learning Rate Estimates. Our first technical result characterizes the gap between the "global" learning rate $\eta_t = b/(a+t)$ (in terms of the true time t), and the individual learning rates at an arbitrary agent i at the same time, denoted by η_t^i.

Lemma 4.2. *Let $\eta_t^i = b/(a + nV_t^i)$, be the learning rate estimate of agent i at time step t, in terms of its time estimate V_t^i. Then, there exists a constant $\gamma > 1$ such that, with probability at least $1 - 1/T^\gamma$ (Here, T is a total number of steps our algorithms takes), the following holds for every $T \geq t \geq 0$ and agent i:*

$$\frac{1}{2} \leq \frac{\eta_t}{\eta_t^i} \leq 2. \tag{4.2}$$

Step 1: Parameter Concentration. Next, let X_t be a vector of model estimates at time step t, that is $X_t = (X_t^1, X_t^2, ..., X_t^n)$. Also, let $\mu_t = \frac{1}{n}\sum_{i=1}^{n} X_t^i$, be an average estimate at time step t. The following potential function measures the concentration of the models around the average:

$$\Gamma_t = \sum_{i=1}^{n} \|X_t^i - \mu_t\|^2.$$

With this in place, one of our key technical results is to provide a supermartingale-type bound on the evolution of the potential Γ_t, in terms of M, η_t, and the number of nodes n.

Lemma 4.3. *For any time step t and fixed learning rate η_t used at t, we have the bound*

$$\mathbb{E}[\Gamma_{t+1}|\Gamma_t] \leq \Big(1 - \frac{1}{n}\Big)\Gamma_t + 4\eta_t M\Big(\frac{\Gamma_t}{n}\Big)^{1/2} + 8\eta_t^2 M^2.$$

Next, we unroll this recurrence to upper bound Γ_t in expectation for any time step t, by choosing an appropriate series of non-constant learning rates.

Lemma 4.4. *If $a \geq 18n$, then the potential is bounded as follows*

$$\mathbb{E}[\Gamma_t] \leq 36nb^2/(t+a)^2M^2 = 36n\eta_t^2 M^2.$$

Step 2: Convergence of the Mean and Risk Bound. The above result allows us to characterize how well the individual parameters are concentrated around their mean, in terms of the second moment of the gradients, the number of nodes, and the learning rate. In turn, this will allow us to provide a recurrence for how fast the parameter average is moving towards the optimum, in terms of the variance and second-moment bounds of the gradients:

Lemma 4.5. *For $\eta_t \leq \frac{n}{64L}$, we have that*

$$\mathbb{E}\Big\|\mu_{t+1} - x^*\Big\|^2 \leq \Big(1 - \frac{\eta_t \ell}{n}\Big)\mathbb{E}\|\mu_t - x^*\|^2 - \frac{\eta_t}{2n}\mathbb{E}[f(\mu_t) - f(x^*)] + \frac{16\sigma^2\eta_t^2}{n^2} + \frac{288\eta_t^3 M^2 L}{n}.$$

Finally, we wish to phrase this bound as a recurrence which will allow us to bound the expected risk of the weighted sum average. We aim to use the following standard result (see e.g. Stich (2018)):

Lemma 4.6. *Let $\{a_t\}_{t\geq 0}$, $a_t \geq 0$, $\{e_t\}_{t\geq 0}$, $e_t \geq 0$ be sequences satisfying*

$$a_{t+1} \leq \Big(1 - \ell\alpha_t\Big)a_t - \alpha_t e_t A + \alpha_t^2 B + \alpha_t^3 C,$$

for $\alpha_t = \frac{4}{\ell(t+a)}$, $A > 0$, $B, C \geq 0$, $\ell > 0$ then

$$\frac{A}{S_T}\sum_{t=0}^{T-1} w_t e_t \leq \frac{\ell a^3}{4S_T}a_0 + \frac{2T(T+2a)}{\ell S_T}B + \frac{16T}{\ell^2 S_T}C, \tag{4.3}$$

for $w_t = (a+t)^2$ and $S_T = \sum_{t=0}^{T-1} w_T \geq \frac{1}{3}T^3$.

To use the above lemma, we set $\eta_t = n\alpha_t = \frac{4n}{\ell(t+a)}$, and the parameter $b = 4n/\ell$. We also use $A = 1/2$, $B = 16\sigma^2$, and $C = 288M^2Ln^2$. Let $y_T = \frac{1}{nS_T}\sum_{i=1}^{n}\sum_{t=0}^{T-1} w_t X_t^i$. Also, let $e_t = \mathbb{E}[f(\mu_t) - f(x^*)]$ and $a_t = \mathbb{E}\Big\|\mu_t - x^*\Big\|^2$. By the convexity of f we have that

$$E[f(y_T) - f(x^*)] \leq \frac{1}{S_T}\sum_{t=0}^{T-1} w_t \mathbb{E}[f(\mu_t) - f(x^*)] \tag{4.4}$$

Using this fact and Lemma 4.6 above we obtain the following final bound:

$$\mathbb{E}[f(y_T) - f(x^*)] \leq \frac{a^3\ell}{2S_T}\|\mu^0 - x^*\|^2 + \frac{64T(T+2a)}{\ell S_T}\sigma^2 + \frac{9216Tn^2}{\ell^2 S_T}M^2 L. \tag{4.5}$$

To complete the proof of the Theorem, we only need to find the appropriate value of the parameter a. For that, we list all the constraints on a: $a \geq 2cn\log T, a \geq 18n$ and $\frac{4n}{\ell(t+a)} \leq \frac{n}{64L}$. These inequalities can be satisfied by setting $a = \max\Big(2cn\log T, 18n, 256\frac{L}{\ell}\Big)$. This concludes our proof.

5 EXTENSIONS

Convergence of local models and alternative to computing y_T. Notice that Theorem 4.1 measures convergence of $f(y_T)$, where $y_T = \sum_{t=0}^{T-1} \frac{w_t}{S_T} \frac{\sum_{i=1}^n X_t^i}{n} = \sum_{t=0}^{T-1} \frac{w_t}{S_T} \mu_t$, is a weighted average of μ_t-s per step. Notice that actually computing y_T can be expensive, since we need values of local models over T steps and it does not necessarily guarantee convergence of each individual model. In order to circumvent this issue, we can look at the following inequality, which in combination with the Jensen's inequality gives us the proof of Theorem 4.1 (Please see Appendix for details) :

$$\frac{1}{S_T} \sum_{t=0}^{T-1} w_t \mathbb{E}[f(\mu_t) - f(x^*)] \leq \frac{a^3 \ell}{2 S_T} \|\mu_0 - x^*\|^2 + \frac{64T(T+2a)}{\ell S_T} \sigma^2 + \frac{9216 T n^2}{\ell^2 S_T} M^2 L. \quad (5.1)$$

What we can do is, instead of computing y_T, we just sample time step $0 \leq t \leq T-1$ with probability $\frac{w_t}{S_T}$ and compute $f(\mu_t) = f(\sum_{i=1}^n X_t^i / n)$, by using single global averaging procedure. Observe that $\mathbb{E}_t[\mathbb{E}_{\mu_t}[f(\mu_t)]]$ is exactly the left hand side of the above inequality.

Hence, we get the convergence identical to the one in Theorem 4.1 and additionally, since we are using global averaging, we also guarantee the same convergence for each local model. Finally, we would like to emphasize that in practice there is no need to compute y_T or to use global averaging, since local models are already converged after T interactions.

General Interaction Graphs. Our analysis can be extended to more general interaction graphs by tying the evolution of the potential in this case. In the following, we present the results for a *cycle*, leaving the exact derivations for more general classes of expander graphs for the full version. In particular, we assume that each agent is a node on a cycle, and that it is allowed to interact only with its neighbouring nodes. Again, the scheduler chooses interaction edges uniformly at random. In this setting, we can show the following result, which is similar to Theorem 4.1:

Theorem 5.1. *Let f be an L-smooth, ℓ-strongly convex function satisfying conditions (2.3)—(2.5), whose minimum $x^\star$ we are trying to find via the PopSGD procedure on a cycle. Let the learning rate for process i at local time $t^i = nV_t^i$ be $\eta_t^i = b/(t^i + a)$, where $a = \max(2cn \log T, 18n, 256L/\ell)$ and $b = 4n/\ell$ are fixed(for some constant c). Let the sequence of weights w_t be given by $w_t = (a+t)^2$. Define $\mu_t = \sum_{i=1}^n X_t^i$, $S_T = \sum_{t=0}^{T-1} w_t \geq \frac{1}{3} T^3$ and $y_T = \frac{1}{S_T} \sum_{t=0}^{T-1} w_t \mu_t$. Then, for any time T, we have with probability $1 - O(1/\text{ poly } T)$ that*

$$E[f(y_T) - f(x^*)] \leq \frac{a^3 \ell}{2 S_T} \|\mu^0 - x^*\|^2 + \frac{64T(T+2a)}{\ell S_T} \sigma^2 + \frac{25600 T n^6}{\ell^3 S_T} M^2 L^2.$$

Notice that for $T \gg n^3$, the second term dominates convergence and we can repeat the same argument as for Theorem 4.1 to show $O(\sigma^2/T)$ convergence (where T is the total number of interactions). Next we provide the sketch of a proof for the PopSGD on a cycle case. The crucial part of the proof is to show the following bound for the Γ_t potential per step:

$$\mathbb{E}[\Gamma_{t+1} | \Gamma_t] \leq \Big(1 - \frac{1}{O(n^3)}\Big) \Gamma_t + 4\eta_t M \Big(\frac{\Gamma_t}{n}\Big)^{1/2} + 8\eta_t^2 M^2.$$

Notice that the above inequality is similar to Lemma 4.3 (Except a factor in front of Γ_T is larger for a cycle, since "information" propagates slower on a cycle) and it allows us to show that:

$$\mathbb{E}[\Gamma_t] \leq O(n^5 \eta_t^2 M^2)$$

This, in turn, allows us to prove the above theorem , by carefully following the steps in the proof of Lemma 4.5 and then by using Lemma 4.6 to finish the proof.

The Non-Convex Case. Next, we show convergence for non-convex, but smooth functions:

Theorem 5.2. *Let f be an non-convex, L-smooth, function satisfying conditions (2.3) and (2.5), whose minimum $x^\star$ we are trying to find via the PopSGD procedure given in Algorithm 1. Define $\mu_t = \sum_{i=1}^n X_t^i$. For the total number of interactions - T, time step $0 \leq t < T$ and process i, let $\eta_t = \eta_t^i = \sqrt{n}/\sqrt{T}$ (for any process, learning rate does not depend on current local or global time). Then, for any T, we have that:*

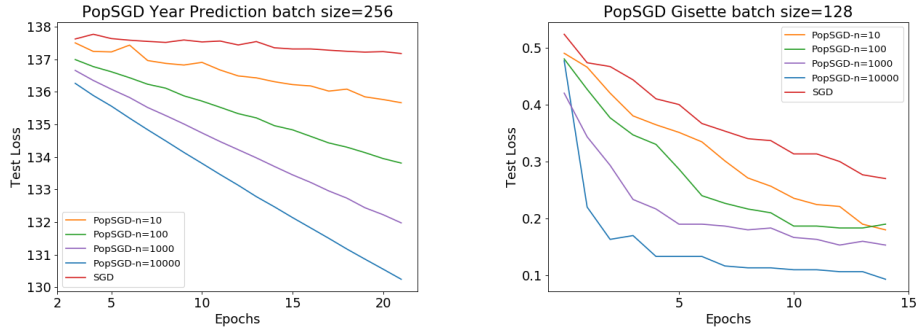
$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\|\nabla f(\mu_t)\|^2 \leq \frac{2\sqrt{n}(f(\mu_0) - f(x^*))}{\sqrt{T}} + \frac{144 L M^2 n}{T} + \frac{16 L M^2 \sqrt{n}}{\sqrt{T}} \quad (5.2)$$

The proof follows from the more general version of the theorem, which is proved in the appendix, see Theorem B.2. Observe that, since T is the total number of interactions and is equal to nT_p, where T_p is a parallel time, we get convergence $O(\sqrt{n}/\sqrt{T}) = O(1/\sqrt{T_p})$. This matches $O(1/\sqrt{T})$ convergence of the sequential version. (Note that in the sequential case parallel time and the total number of interactions are the same.)

6 Experimental Results

In this section, we validate our results numerically by implementing PopSGD in Pytorch, using MPI for inter-node communication (MPI). We are interested in the *convergence* behavior of the algorithm, and in the *scalability* with respect to the number of nodes. Our study is split into simulated experiments for convex objectives–to examine the validity of our analysis as n increases—and large-scale real-world experiments for non-convex objectives (training neural networks), aimed to examine whether PopSGD can provide scalability and convergence for such objectives.

Convex Objectives. To validate our analysis in the convex case, we evaluated the performance of PopSGD on three datasets: (1) a real-world linear regression problem (the *Year Prediction* dataset (Chang & Lin, 2011)) with a $463,715/51,630$ test/train split, and $d = 90$; (2) a real-world classification problem (*gisette* (Chang & Lin, 2011)) with $6,000/1,000$ test/train split, and $d = 5000$; (3) a synthetic least-squares problem of the form (2.2) with $f(x) = \frac{1}{2}\|Ax - b\|^2$, where $A \in \mathbb{R}^{m\times d}$ and $b \in \mathbb{R}^m$, with $m = 10^4$ and variable d. As a baseline, we employ vanilla SGD with manual learning rate tuning. The learning rate is adjusted in terms of the number of local steps each node has taken, similar to our analysis.



(a) PopSGD test loss vs. n on Year Prediction.

(b) PopSGD test loss vs. n on Gisette.

Figure 1: PopSGD convergence (test loss at the step versus parallel time) for various node counts n on a real linear regression (left) and logistic regression (right) datasets. The baseline is sequential SGD, which is identical to PopSGD with node count 1.

Our first set of experiments examines train and test loss for PopSGD on the real-world tasks specified above. We examine the *test loss* behavior with respect to the number of nodes n, and execute for powers of 10 between 1 and 10000. Each node obtains a stochastic gradient by sampling 128 elements from the training set in a batch. We tuned the learning rate parameter for each instance independently, through line search, and obtained learning rates in the interval $[0.0005, 0.015]$ for Gisette, and $[0.05, 0.2]$ for Year Prediction.

Please see Figure 1(b) for the results.(The number of epochs is cropped to maintain visibility, but the trends are maintained in general.) The results confirm our analysis; notice in particular the clear separation between instances for different n, which follows exactly the increase in the number of nodes, although the X axis values correspond to the same number of gradient steps for the local model. In Appendix A, we present additional experiments which precisely examine the reduction in variance versus the number of nodes on the synthetic regression task, confirming our analysis.

Training Neural Networks. Our second set of experiments tests PopSGD in a realistic distributed environment. For this, we implemented PopSGD in Pytorch using MPI one-sided primitives (MPI), which allow nodes to read eachothers' models for averaging without explicit synchronization. We used PopSGD to train ResNets on the classic CIFAR-10 and ImageNet datasets, and deploy our code

on the CSCS Piz Daint supercomputer, which is composed of Cray XC50 nodes, each with a Xeon E5-2690v3 CPU and an NVIDIA Tesla P100 GPU, using a state-of-the-art Aries interconnect.

Training proceeds in epochs, each of which is structured as follows. At the beginning of each epoch, we shuffle the dataset and split it into $n/2$ partitions, ensuring that each partition will be assigned to exactly two processes. We define a fixed constant mult, which counts the number of times each process will iterate through its partition in an epoch. In our experiments, mult takes values between 1 and 4. Intuitively, mult follows the intuition given by Theorems 4.1 and 5.2, which suggest that PopSGD needs additional iterations for the information in each partition to propagate to all nodes. Given this setup, PopSGD may appear wasteful, since each sample is processed $2 \times \mathsf{mult}$ times in each epoch. We compensate for this by *compressing* the standard training schedules for the networks we examine, dividing the total number of epochs by n, and scaling the learning rate updates accordingly. We keep local batch sizes constant with respect to the sequential baseline. That is, in an experiment with $n = 32$ nodes and multiplier $\mathsf{mult} = 4$, PopSGD processes each sample $32/(2 \times 4) = 4$ less times than standard sequential or data-parallel SGD, and performs $8\times$ less gradient updates per model. Surprisingly, we found this to be sufficient to preserve both train and test accuracy. Figure 2 shows the test and train accuracies for the ResNet18 model trained on the ImageNet dataset, with 32 Piz Daint nodes and $\mathsf{mult} = 4$, as well as scalability versus number of nodes.

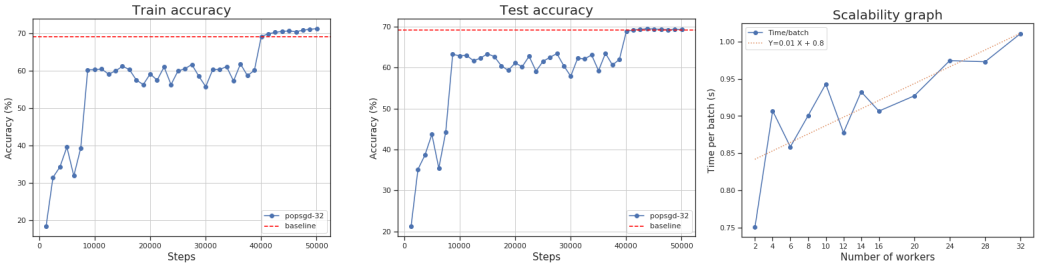


Figure 2: PopSGD train and test accuracy using 32 nodes on Piz Daint, measured at a fixed arbitrary node. The iteration multipler value is $\mathsf{mult} = 4$. The X axis measures SGD steps per model, whereas the Y axis measures Top-1 accuracy. The dotted red line is the accuracy of the Torchvision baseline (Marcel & Rodriguez, 2010). PopSGD surpasses the test accuracy of the baseline by 0.34%, although it processes each sample $4\times$ less times, and each model sees $8\times$ less gradient updates. The right graph shows the increase in average time per batch versus number of nodes (logarithmic), due to network effects.

The results suggest that PopSGD can indeed preserve convergence while ensuring scalability for this complex task. We note that the hyperparameters used for model training are identical to the standard sequential recipe (batch size 128 per node), with the sole exception of the mult parameter, for which we found low constant values (1–4) to be sufficient. Appendix A presents additional experiments for ResNet50/Imagenet and ResNet20/CIFAR-10, which further substantiate this claim.

7 Discussion and Future Work

We have analyzed for the first time the convergence of decentralized SGD in the population model of distributed computing. We have shown that, despite the extremely weak synchronization characteristics of this model, SGD is able to still converge in this setting, and moreover, under parameter and objective assumptions, can even achieve *linear* speedup in the number of agents n in terms of parallel time. The empirical results confirmed our analytical findings. The main surprising result is that PopSGD presents speedup behavior roughly similar to mini-batch SGD, even though a node only sees one gradient update and a single model at a time. This asymptotic speedup behavior is obviously optimal (assuming all other parameters are constant), since we cannot expect super-linear speedup in n. Similar speedup behavior required either the existence of synchronized rounds (e.g. Lian et al. (2017a)), or global averaging steps (Stich, 2018), or both. Our work opens several avenues for future work. One natural extension is to study PopSGD with quantized communication, or allowing the interactions to present inconsistent (stale) model views to the two agents. Another avenue is to tighten the bounds in terms of their dependence on the problem conditioning, and on the objective assumptions.

REFERENCES

Mpich: high performance and widely portable implementation of the message passing interface (mpi) standard. http://www.mpich.org/. Accessed: 2018-1-25.

The CSCS Piz Daint supercomputer. http://www.cscs.ch/computers/piz_daint. Accessed: 2018-1-25.

Alham Fikri Aji and Kenneth Heafield. Sparse communication for distributed gradient descent. *arXiv preprint arXiv:1704.05021*, 2017.

Dan Alistarh, James Aspnes, David Eisenstat, Rati Gelashvili, and Ronald L Rivest. Time-space trade-offs in population protocols. In *Proceedings of the 28th ACM-SIAM Symposium on Discrete Algorithms*, (SODA), pp. 2560–2579, 2017a.

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Randomized quantization for communication-efficient stochastic gradient descent. In *Proceedings of NIPS 2017*, 2017b.

Dan Alistarh, James Aspnes, and Rati Gelashvili. Space-optimal majority in population protocols. In *Proceedings of the 29th ACM-SIAM Symposium on Discrete Algorithms*, (SODA), pp. 2221–2239, 2018a.

Dan Alistarh, Christopher De Sa, and Nikola Konstantinov. The convergence of stochastic gradient descent in asynchronous shared memory. In *PODC*, pp. 169–178, 2018b.

Dana Angluin, James Aspnes, Zoë Diamadi, Michael J Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed computing*, 18(4):235–253, 2006.

Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, 2007.

Dana Angluin, James Aspnes, and David Eisenstat. A simple population protocol for fast robust approximate majority. *Distributed Computing*, 21(2):87–102, 2008a.

Dana Angluin, James Aspnes, and David Eisenstat. Fast computation by population protocols with a leader. *Distributed Computing*, 21(3):183–199, 2008b.

Dana Angluin, James Aspnes, Michael J Fischer, and Hong Jiang. Self-stabilizing population protocols. *ACM Transactions on Autonomous and Adaptive Systems*, 3(4):13:1–13:28, 2008c.

Mahmoud Assran, Nicolas Loizou, Nicolas Ballas, and Michael Rabbat. Stochastic gradient push for distributed deep learning. *arXiv preprint arXiv:1811.10792*, 2018.

Yossi Azar, Andrei Z Broder, Anna R Karlin, and Eli Upfal. Balanced allocations. *SIAM Journal on Computing*, 29(1):180–200, 1999.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pp. 177–186. Springer, 2010.

James M Bower and Hamid Bolouri. *Computational modeling of genetic and biochemical networks*. MIT press, 2004.

Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May 2011. ISSN 2157-6904. doi: 10.1145/1961189.1961199. URL http://doi.acm.org/10.1145/1961189.1961199.

Ho-Lin Chen, Rachel Cummings, David Doty, and David Soloveichik. Speed faults in computation by chemical reaction networks. *Distributed Computing*, 30(5):373–390, 2017.

Kai Chen and Qiang Huo. Scalable training of deep learning machines by incremental block training with intra-block parallel optimization and blockwise model-update filtering. In *2016 ieee international conference on acoustics, speech and signal processing (icassp)*, pp. 5880–5884. IEEE, 2016.

Yuan-Jyue Chen, Neil Dalchau, Niranjan Srnivas, Andrew Phillips, Luca Cardelli, David Soloveichik, and Georg Seelig. Programmable chemical controllers made from dna. *Nature Nanotechnology*, 8 (10):755–762, 2013.

Moez Draief and Milan Vojnovic. Convergence speed of binary interval consensus. *SIAM Journal on Control and Optimization*, 50(3):1087–1109, 2012.

Nikoli Dryden, Sam Ade Jacobs, Tim Moon, and Brian Van Essen. Communication quantization for data-parallel training of deep neural networks. In *Proceedings of the Workshop on Machine Learning in High Performance Computing Environments*, pp. 1–8. IEEE Press, 2016.

John C Duchi, Sorathan Chaturapruek, and Christopher Ré. Asynchronous stochastic convex optimization. *arXiv preprint arXiv:1508.00882*, 2015.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Demjan Grubic, Leo Tam, Dan Alistarh, and Ce Zhang. Synchronous multi-gpu deep learning with low-precision communication: An experimental study. In *EDBT*, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hadrien Hendrikx, Laurent Massoulié, and Francis Bach. Accelerated decentralized optimization with local updates for smooth and strongly convex objectives. *arXiv preprint arXiv:1810.02660*, 2018.

Franck Iutzeler, Pascal Bianchi, Philippe Ciblat, and Walid Hachem. Asynchronous distributed optimization using a randomized alternating direction method of multipliers. In *52nd IEEE conference on decision and control*, pp. 3671–3676. IEEE, 2013.

Björn Johansson, Maben Rabi, and Mikael Johansson. A randomized incremental subgradient method for distributed optimization in networked systems. *SIAM Journal on Optimization*, 20(3): 1157–1170, 2009.

David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pp. 482–491. IEEE, 2003.

Anastasia Koloskova, Sebastian U Stich, and Martin Jaggi. Decentralized stochastic optimization and gossip algorithms with compressed communication. *arXiv preprint arXiv:1902.00340*, 2019.

Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jio Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *arXiv preprint arXiv:1705.09056*, 2017a.

Xiangru Lian, Wei Zhang, Ce Zhang, and Ji Liu. Asynchronous decentralized parallel stochastic gradient descent. *arXiv preprint arXiv:1710.06952*, 2017b.

Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM international conference on Multimedia*, pp. 1485–1488. ACM, 2010.

Michael Mitzenmacher. How useful is old information? *IEEE Transactions on Parallel and Distributed Systems*, 11(1):6–20, 2000.

Angelia Nedic and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48, 2009.

Angelia Nedic, Alex Olshevsky, and Wei Shi. Achieving geometric convergence for distributed optimization over time-varying graphs. *SIAM Journal on Optimization*, 27(4):2597–2633, 2017.

Yuval Peres, Kunal Talwar, and Udi Wieder. Graphical balanced allocations and the one plus beta-choice process. *Random Struct. Algorithms*, 47(4):760–775, 2015a. doi: 10.1002/rsa.20558. URL https://doi.org/10.1002/rsa.20558.

Yuval Peres, Kunal Talwar, and Udi Wieder. Graphical balanced allocations and the 1 + beta-choice process. *Random Struct. Algorithms*, 47(4):760–775, December 2015b. ISSN 1042-9832. doi: 10.1002/rsa.20558. URL http://dx.doi.org/10.1002/rsa.20558.

Etienne Perron, Dinkar Vasudevan, and Milan Vojnovic. Using three states for binary consensus on complete graphs. In *Proceedings of the 28th IEEE Conference on Computer Communications*, INFOCOM '09, pp. 2527–2535, 2009.

Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS*, pp. 693–701, 2011.

C. M. De Sa, C. Zhang, K. Olukotun, and C. Re. Taming the wild: A unified analysis of hogwild-style algorihms. In *Advances in Neural Information Processing Systems*, 2015.

F. Seide, H. Fu, L. G. Jasha, and D. Yu. 1-bit stochastic gradient descent and application to data-parallel distributed training of speech dnns. *Interspeech*, 2014.

Ohad Shamir and Nathan Srebro. Distributed stochastic optimization and learning. In *2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 850–857. IEEE, 2014.

Sebastian U Stich. Local sgd converges fast and communicates little. *arXiv preprint arXiv:1805.09767*, 2018.

Nikko Strom. Scalable distributed dnn training using commodity gpu cloud computing. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

Kunal Talwar and Udi Wieder. Balanced allocations: the weighted case. In David S. Johnson and Uriel Feige (eds.), *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pp. 256–265. ACM, 2007. ISBN 978-1-59593-631-8. doi: 10.1145/1250790.1250829. URL https://doi.org/10.1145/1250790.1250829.

Hanlin Tang, Ce Zhang, Shaoduo Gan, Tong Zhang, and Ji Liu. Decentralization meets quantization. *CoRR, abs/1803.06443*, 2018.

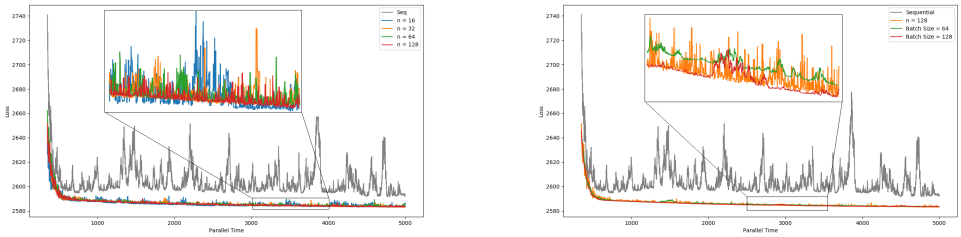
John Nikolas Tsitsiklis. Problems in decentralized decision making and computation. Technical report, Massachusetts Inst of Tech Cambridge Lab for Information and Decision Systems, 1984.

Ermin Wei and Asuman Ozdaglar. Distributed alternating direction method of multipliers. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pp. 5445–5450. IEEE, 2012.

Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In *Advances in Neural Information Processing Systems*, pp. 1508–1518, 2017.

Lin Xiao and Stephen Boyd. Fast linear iterations for distributed averaging. *Systems & Control Letters*, 53(1):65–78, 2004.

Yang You, Igor Gitman, and Boris Ginsburg. Scaling sgd batch size to 32k for imagenet training. *arXiv preprint arXiv:1708.03888*, 2017.



(a) PopSGD convergence vs. n.

(b) PopSGD convergence vs. mini-batch SGD.

Figure 3: PopSGD convergence (training loss at the step versus parallel time) on the synthetic regression task versus the number of nodes n (left), and versus sequential SGD with different batch sizes (right). Sequential SGD is identical to PopSGD with node count 1. The cutouts represent zoomed views.

A Additional Experiments

Convex Losses. In these experiments, we examine the convergence of PopSGD versus parallel time for different node counts, and compared it with the sequential baseline. More precisely, for PopSGD, we execute the protocol by simulating the entire sequence of interactions sequentially, and track the evolution of train and test loss at an arbitrary fixed model x^i with respect to the number of SGD steps it performs. Notice that this is practically equivalent to tracking with respect to *parallel* time. In this case, the theory suggests that loss convergence and variance should both improve when increasing the number of nodes. Figure 3(a) presents the results for the synthetic linear regression example with $d = 32$, for various values of n, for constant learning rate $\eta = 0.001$ across all models, and batch size 1 for each local gradient. Figure 3(b) compares PopSGD convergence (with local batch size 1) against sequential mini-batch SGD with batch size equal to the number of nodes n.

Examining Figure 3(a), we observe that both the convergence and loss variance improve as we increase the number of nodes n, even though the target model executes exactly the same number of gradient steps at the same point on the x axis. Of note, variance decreases proportionally with the number of nodes, with $n = 128$ having the smallest variance. Compared to mini-batch SGD with batch size = n (Figure 3(b)), PopSGD with $n = 128$ has similar, but notably higher variance, which follows the analytical bound in Theorem 4.1.

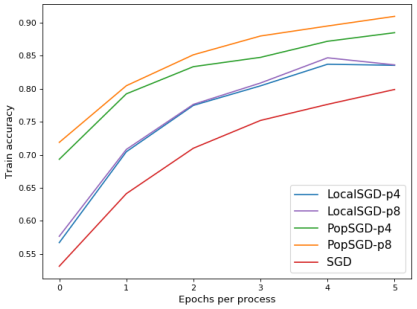
CIFAR-10 Experiments. We illustrate convergence and scaling results for non-convex objectives by using PopSGD to train a standard ResNet20 DNN model on CIFAR-10 in Pytorch, using 8 GPU nodes, comparing against vanilla and local SGD performing global averaging every 100 batches (we found this value necessary for the model to converge). We measure the error/loss at an arbitrary process for PopSGD. We run the parallel versions at 4 and 8 nodes.

The results in Figure 4(c) show that (a,b) PopSGD does indeed converge faster as we increase population size, tracking the trend from the convex case; and (c) PopSGD can provide non-trivial scalability, comparable or better than data-parallel and local SGD.

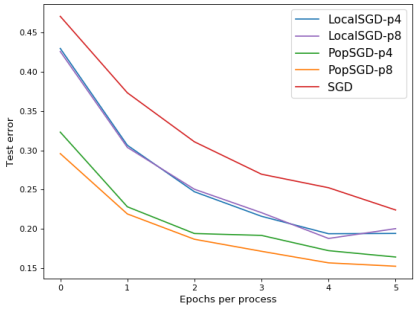
Training ResNet50 on ImageNet. Figure 4 shows the test and train accuracies for the ResNet50 model trained on the ImageNet dataset, with 32 Piz Daint nodes and $\mathsf{mult} = 4$. PopSGD achieves test accuracy within $< 0.5\%$ relative to the Torchvision baseline, despite the vastly inferior number of iterations, in a total of 29 hours. By way of comparison, end-to-end training using standard data-parallel SGD takes approximately 48h on the same setup (using 8 GPUs instead of 32 to avoid large-batch effects).

B Complete Correctness Argument

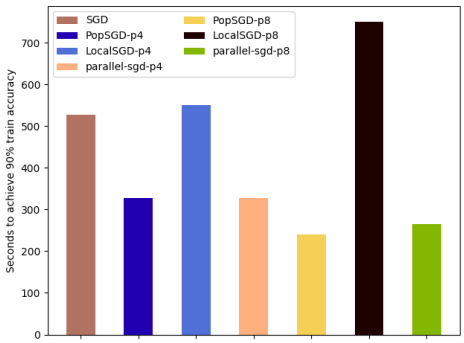
Lemma 4.2. *Let $\eta_t^i = b/(a + nV_t^i)$, be the learning rate estimate of agent i at time step t, in terms of its time estimate V_t^i. Then, there exists a constant $\gamma > 1$ such that, with probability at least $1 - 1/T^\gamma$ (Here, T is a total number of steps our algorithms takes), the following holds for every $T \geq t \geq 0$*



(a) Train accuracy for ResNet20/CIFAR10.



(b) Test error for ResNet20/CIFAR10.



(c) Time to 90% train accuracy.

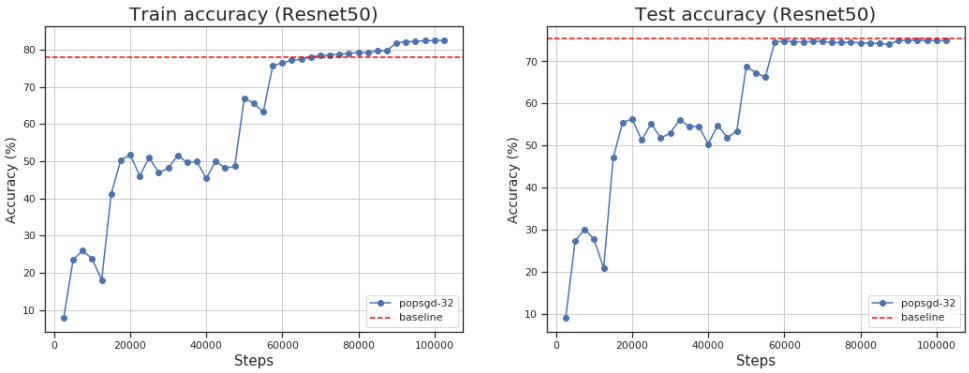


Figure 4: PopSGD train and test accuracy using 32 nodes on Piz Daint, measured at a fixed arbitrary node, for training ResNet50 on ImageNet. The round multipler value is mult $= 4$. The X axis measures SGD steps per model, whereas the Y axis measures Top-1 accuracy. The dotted red line is the accuracy of the Torchvision baseline (Marcel & Rodriguez, 2010). PopSGD is below the test accuracy of the baseline by $< 0.5\%$.

and agent i:

$$\frac{1}{2} \leq \frac{\eta_t}{\eta_t^i} \leq 2. \tag{B.1}$$

Proof. Let $G^t = \sum_{i=1}^{n} \exp\left(\zeta(V_t^i - \frac{t}{n})\right) + \sum_{i=1}^{n} \exp\left(-\zeta(V_t^i - \frac{t}{n})\right)$, for some fixed constant ζ. The following lemma is proved as Theorem 2.10 in Peres et al. (2015b):

Lemma B.1. *For any $t \geq 0$, and some fixed constants ϵ and θ, $\mathbb{E}[G_t] \leq \frac{4\theta}{\zeta\epsilon} n$.*

Subsequently, we can show that for any $t \geq 0$ and agent i:

$$Pr\Big[\mid \frac{t}{n} - V_t^i \mid \geq \frac{q}{\zeta} \log T \Big] \leq Pr[G^t \geq T^q] \overset{Markov}{\leq} \frac{4\theta}{\zeta\epsilon} \frac{n}{T^q}. \tag{B.2}$$

Hence, for large enough constant q, using union bound over T steps, we can show that there exists a constant $\gamma > 0$ such that for every $T \geq t \geq 0$ and agent i, $\mid \frac{t}{n} - V_t^i \mid \leq \frac{q}{\zeta} \log T$, with probability at least $1 - 1/T^\gamma$.

Let c be $\frac{q}{\zeta}$, thus $a \geq 2cn \log T = 2\frac{q}{\zeta} n \log T$. This allows us to finish the proof of the lemma:

$$\frac{1}{2} \leq \frac{a + t - \frac{q}{\zeta} n \log T}{a + t} \leq \frac{\eta_t}{\eta_t^i} \leq \frac{a + t + \frac{q}{\zeta} n \log T}{a + t} \leq 2. \tag{B.3}$$

□

This allows us to bound the per step change of potential Γ, in terms of global learning rate η_t.

Lemma 4.3. *For any time step t and fixed learning rate η_t used at t, we have the bound*

$$\mathbb{E}[\Gamma_{t+1} | \Gamma_t] \leq \Big(1 - \frac{1}{n}\Big)\Gamma_t + 4\eta_t M \Big(\frac{\Gamma_t}{n}\Big)^{1/2} + 8\eta_t^2 M^2.$$

Proof. First we bound change in potential $\Delta_t = \Gamma_{t+1} - \Gamma_t$ for some time step $t > 0$. Let $\Delta_t^{i,j}$ be a change in potential when we choose different agents i and j at random and let Δ_t^i be a change in potential when we select the same node i. We get that

$$\mathbb{E}\Big[\Delta_t | X_t\Big] = \sum_i \sum_{i \neq j} \frac{1}{n^2} \mathbb{E}\Big[\Delta_t^{i,j} | X_t\Big] + \sum_{i=1}^n \frac{1}{n^2} \mathbb{E}\Big[\Delta_t^i | X_t\Big]. \tag{B.4}$$

We proceed by bounding a change in potential for fixed $i \neq j$. Observe, that in this case $\mu_{t+1} = \mu_t - (\eta_t^i \widetilde{g}_i(X_t^i) + \eta_t^j \widetilde{g}_j(X_t^j))/n$ and $X_{t+1}^i = X_{t+1}^j = (X_t^i + X_t^j)/2 - (\eta_t^i \widetilde{g}_i(X_t^i) + \eta_t^j \widetilde{g}_j(X_t^j))/2$.

Hence,

$$X_{t+1}^i - \mu_{t+1} = X_{t+1}^j - \mu_{t+1} = (X_t^i + X_t^j)/2 - \frac{n-2}{2n}(\eta_t \widetilde{g}_i(X_t^i) + \eta_t \widetilde{g}_j(X_t^j)) - \mu_t.$$

For $k \notin \{i, j\}$, since $X_{t+1}^k = X_t^k$ we get that

$$X_{t+1}^k - \mu_{t+1} = X_t^k + \frac{1}{n}(\eta_t \widetilde{g}_i(X_t^i) + \eta_t \widetilde{g}_j(X_t^j)) - \mu_t.$$

This gives us that

$$\begin{aligned}
\mathbb{E}\Big[\Delta_t^{i,j} | X_t\Big] = \;& \mathbb{E}\Big\|(X_t^i + X_t^j)/2 - \frac{n-2}{2n}(\eta_t^i \widetilde{g}_i(X_t^i) + \eta_t^j \widetilde{g}_j(X_t^j)) - \mu_t\|^2 - \|X_t^i - \mu_t\|^2 \\
& + \mathbb{E}\Big\|(X_t^i + X_t^j)/2 - \frac{n-2}{2n}(\eta_t^i \widetilde{g}_i(X_t^i) + \eta_t^j \widetilde{g}_j(X_t^j)) - \mu_t\|^2 - \|X_t^j - \mu_t\|^2 \\
& + \sum_{k \notin \{i,j\}} \Big(\mathbb{E}\Big\|X_t^k + \frac{1}{n}(\eta_t^i \widetilde{g}_i(X_t^i) + \eta_t^j \widetilde{g}_j(X_t^j)) - \mu_t\Big\|^2 - \|X_t^k - \mu_t\|^2\Big) \\
= \;& 2\|(X_t^i - \mu_t)/2 + (X_t^j - \mu_t)/2\|^2 - \|X_t^i - \mu_t\|^2 - \|X_t^j - \mu_t\|^2 \\
& - \frac{n-2}{n} \mathbb{E}\langle \eta_t^i \widetilde{g}_i(X_t^i) + \eta_t^j \widetilde{g}_j(X_t^j), (X_t^i - \mu_t) + (X_t^j - \mu_t)\rangle \\
& + 2\Big(\frac{n-2}{2n}\Big)^2 \mathbb{E}\|\eta_t^i \widetilde{g}_i(X_t^i) + \eta_t^j \widetilde{g}_j(X_t^j)\|^2 \\
& + \sum_{k \notin \{i,j\}} \Big(\frac{2}{n} \mathbb{E}\langle \eta_t^i \widetilde{g}_i(X_t^i) + \eta_t^j \widetilde{g}_j(X_t^j), X_t^k - \mu_t\rangle + \frac{1}{n^2} \mathbb{E}\|\eta_t^i \widetilde{g}_i(X_t^i) + \eta_t^j \widetilde{g}_j(X_t^j)\|^2\Big)
\end{aligned}$$

Observe that

$$\mathbb{E}\|\eta_t^i \widetilde{g}_i(X_t^i) + \eta_t \widetilde{g}_j(X_t^j)\|^2 \leq 2(\eta_t^i)^2 \mathbb{E}\|g_i(X_t^i)\|^2 + 2(\eta_t^j)^2 E\|g_j(X_t^j)\|^2 \overset{\text{Fact (2.5)}}{\leq} 2M^2\Big((\eta_t^i)^2 + (\eta_t^j)^2\Big) \overset{\text{Lemma 4.2}}{\leq} 16\eta_t^2 M^2.$$

and

$$\sum_{k=1}^{n}\mathbb{E}\langle\eta_t^i\widetilde{g}_i(X_t^i)+\eta_t^j\widetilde{g}_j(X_t^j),X_t^k-\mu_t\rangle=0.$$

Thus, we have that

$$\begin{aligned}\mathbb{E}\Big[\Delta_t^{i,j}|X_t\Big]\leq\quad & 2\|(X_t^i-\mu_t)/2+(X_t^j-\mu_t)/2\|^2-\|X_t^i-\mu_t\|^2-\|X_t^j-\mu_t\|^2\\&-\mathbb{E}\langle\eta_t^i\widetilde{g}_i(X_t^i)+\eta_t\widetilde{g}_j(X_t^j),(X_t^i-\mu_t)+(X_t^j-\mu_t)\rangle\\&+32\eta_t^2\Big(\frac{n-2}{2n}\Big)^2M^2+\sum_{k\notin\{i,j\}}\frac{16}{n^2}\eta_t^2M^2\\\leq\quad&-\|X_t^i-\mu_t\|^2/2-\|X_t^j-\mu_t\|^2/2+\langle X_t^i-\mu_t,X_t^j-\mu_t\rangle\\&-\mathbb{E}\langle\eta_t^i\widetilde{g}_i(X_t^i)+\eta_t^j\widetilde{g}_j(X_t^j),(X_t^i-\mu_t)+(X_t^j-\mu_t)\rangle\\&+8\eta_t^2M^2.\end{aligned}\tag{B.6}$$

similarly we can prove that

$$\mathbb{E}\Big[\Delta_t^i|X_t\Big]\leq-\mathbb{E}\langle\eta_t^i\widetilde{g}_i(X_t^i)+\eta_t^i\widetilde{g}_i(X_t^i),(X_t^i-\mu_t)+(X_t^i-\mu_t)\rangle+8\eta_t^2M^2.\tag{B.7}$$

By using inequalities B.6 and B.7 in inequality B.4 we get that

$$\begin{aligned}\mathbb{E}\Big[\Delta_t|X_t\Big]=\quad&\sum_i\sum_{i\neq j}\frac{1}{n^2}\mathbb{E}\Big[\Delta_t^{i,j}|X_t\Big]+\sum_{i=1}^{n}\frac{1}{n^2}\mathbb{E}\Big[\Delta_i^t|X_t\Big]\\\leq\quad&-\sum_i\sum_{i\neq j}\frac{1}{n^2}\Big(\|X_t^i-\mu_t\|^2/2+\|X_t^j-\mu_t\|^2/2\Big)+\sum_i\sum_{i\neq j}\frac{1}{n^2}\langle X_t^i-\mu_t,X_t^j-\mu_t\rangle\\&-\sum_i\sum_j\frac{1}{n^2}\mathbb{E}\langle\eta_t^i\widetilde{g}_i(X_t^i)+\eta_t^j\widetilde{g}_j(X_t^j),(X_t^i-\mu_t)+(X_t^j-\mu_t)\rangle+8\eta_t^2M^2.\end{aligned}$$

Observe that

$$\sum_i\sum_{i\neq j}\frac{1}{n^2}\langle X_t^i-\mu_t,X_t^j-\mu_t\rangle=\sum_{i=1}^{n}\frac{1}{n^2}\langle X_t^i-\mu_t,\sum_{j\neq i}X_t^j-\mu_t\rangle=\frac{1}{n^2}\sum_i-\|X_t^i-\mu_t\|^2=-\frac{1}{n^2}\Gamma_t.$$

and

$$\sum_i\sum_{i\neq j}\frac{1}{n^2}\Big(\|X_t^i-\mu_t\|^2/2+\|X_t^j-\mu_t\|^2/2\Big)=\frac{n-1}{n^2}\sum_i\|X_t^i-\mu_t\|^2=\frac{n-1}{n^2}\Gamma_t.$$

Hence, we get that

$$\mathbb{E}\Big[\Delta_t|X_t\Big]\leq-\frac{\Gamma_t}{n}-\sum_i\sum_j\frac{1}{n^2}\mathbb{E}\langle\eta_t^i\widetilde{g}_i(X_t^i)+\eta_t^j\widetilde{g}_j(X_t^j),(X_t^i-\mu_t)+(X_t^j-\mu_t)\rangle+8\eta_t^2M^2.\tag{B.9}$$

Further, we have that

$$\begin{aligned}&\sum_i\sum_j\frac{1}{n^2}\mathbb{E}\langle\eta_t^i\widetilde{g}_i(X_t^i)+\eta_t^j\widetilde{g}_j(X_t^j),(X_t^i-\mu_t)+(X_t^j-\mu_t)\rangle\\&=\sum_i\sum_j\frac{1}{n^2}\mathbb{E}\langle\eta_t^i\widetilde{g}_i(X_t^i),(X_t^j-\mu_t)\rangle+\sum_i\sum_j\frac{1}{n^2}\mathbb{E}\langle\eta_t^j\widetilde{g}_j(X_t^j),(X_t^i-\mu_t)\rangle+\sum_{i=1}^{n}\frac{2\eta_t^i}{n}\mathbb{E}\langle\widetilde{g}_i(X_t^i),X_t^i-\mu_t\rangle\\&=\sum_{i=1}^{n}\frac{2\eta_t^i}{n}\mathbb{E}\langle\widetilde{g}_i(X_t^i),X_t^i-\mu_t\rangle\overset{\text{Cauchy-Schwarz}}{\leq}\sum_{i=1}^{n}\frac{2\eta_t^i}{n}\mathbb{E}\Big[\|\widetilde{g}_i(X_t^i)\|\cdot\|X_t^i-\mu_t\|\Big]\\&=\sum_{i=1}^{n}\frac{2\eta_t^i}{n}\mathbb{E}\Big[\|\widetilde{g}_i(X_t^i)\|\Big]\|X_t^i-\mu_t\|\overset{\text{Jensen}}{\leq}\sum_{i=1}^{n}\frac{2\eta_t^i}{n}\Big(\mathbb{E}\|\widetilde{g}_i(X_t^i)\|^2\Big)^{\frac{1}{2}}\|X_t^i-\mu_t\|\leq\sum_{i=1}^{n}\frac{2\eta_t^iM}{n}\|X_t^i-\mu_t\|\\&\overset{\text{Lemma 4.2}}{\leq}\frac{4\eta_tM}{n}\sum_{i=1}^{n}\|X_t^i-\mu_t\|\overset{\text{Cauchy-Schwarz}}{\leq}\frac{4\eta_tM}{n}\Big(n\sum_{i=1}^{n}\|X_t^i-\mu_t\|^2\Big)^{1/2}=4\eta_tM\Big(\frac{\Gamma_t}{n}\Big)^{1/2}.\end{aligned}$$

By plugging above inequality in inequality B.9, we get that

$$\mathbb{E}[\Delta_t|X_t] \leq -\frac{\Gamma_t}{n} + 4\eta_t M\Big(\frac{\Gamma_t}{n}\Big)^{1/2} + 8\eta_t^2 M^2.$$

Hence, considering the definition of Δ_t and the fact that the above inequality implies

$$\mathbb{E}[\Delta_t|\Gamma_t] \leq -\frac{1}{n}\Gamma_t + 4\eta_t M\Big(\frac{\Gamma_t}{n}\Big)^{1/2} + 8\eta_t^2 M^2,$$

we get the proof of the Lemma. □

Lemma 4.4. *If $a \geq 18n$, then the potential is bounded as follows*

$$\mathbb{E}[\Gamma_t] \leq 36nb^2/(t+a)^2M^2 = 36n\eta_t^2M^2.$$

Proof. We prove the lemma using induction. Base case $t = 0$ trivially holds, since $\Gamma_t = 0$. For induction step, we assume that at time step t, $\mathbb{E}[\Gamma_t] \leq 36nb^2M^2/(t+a)^2$. Our goal is to prove that $\mathbb{E}[\Gamma_{t+1}] \leq 36nb^2M^2/(t+a+1)^2$.

$$\begin{aligned}
\mathbb{E}[\Gamma_{t+1}] = \mathbb{E}[\mathbb{E}[\Gamma_{t+1}|\Gamma_t]] &\overset{\text{Lemma 4.3}}{\leq} \Big(1-\frac{1}{n}\Big)\mathbb{E}[\Gamma_t] + 4\eta_t M\mathbb{E}\Big[\Big(\frac{\Gamma_t}{n}\Big)^{1/2}\Big] + 8\eta_t^2M^2 \\
&\overset{\text{Jensen}}{\leq} \Big(1-\frac{1}{n}\Big)\mathbb{E}[\Gamma_t] + 4\eta_t M\Big(\mathbb{E}\Big[\frac{\Gamma_t}{n}\Big]\Big)^{1/2} + 8\eta_t^2M^2 \\
&\leq \Big(1-\frac{1}{n}\Big)\frac{36nb^2M^2}{(t+a)^2} + \frac{24b^2M^2}{(t+a)^2} + \frac{8b^2M^2}{(t+a)^2} \\
&\leq \frac{36nb^2M^2}{(t+a+1)^2} + \Big(\frac{36nb^2M^2}{(t+a)^2} - \frac{36nb^2M^2}{(t+a+1)^2}\Big) - \frac{4b^2M^2}{(t+a)^2} \\
&= \frac{36b^2M^2}{(t+a+1)^2} + \frac{36nb^2M^2\Big(2(t+a)+1\Big)}{(t+a)^2(t+a+1)^2} - \frac{4b^2M^2}{(t+a)^2} \\
&\leq \frac{36nb^2M^2}{(t+a+1)^2} + \frac{72nb^2M^2(t+a+1)}{(t+a)^2(t+a+1)^2} - \frac{4b^2M^2}{(t+a)^2}.
\end{aligned}$$

Using the fact that $t+a+1 \geq a \geq 18n$ in the above inequality allows us to get

$$\mathbb{E}[\Gamma_{t+1}] \leq \frac{36nb^2M^2}{(t+a+1)^2} + \frac{4b^2M^2}{(t+a)^2} - \frac{4b^2M^2}{(t+a)^2} \leq \frac{36nb^2M^2}{(t+a+1)^2}.$$

□

Lemma 4.5. *For $\eta_t \leq \frac{n}{64L}$, we have that*

$$\mathbb{E}\Big\|\mu_{t+1} - x^*\Big\|^2 \leq \Big(1-\frac{\eta_t\ell}{n}\Big)\mathbb{E}\|\mu_t - x^*\|^2 - \frac{\eta_t}{2n}\mathbb{E}[f(\mu_t) - f(x^*)] + \frac{16\sigma^2\eta_t^2}{n^2} + \frac{288\eta_t^3M^2L}{n}.$$

Proof. Let F_t be the amount by which μ_t decreases at step t. So, F_t is a sum of $\frac{\eta_t^i}{n}\widetilde{g}_i(X_t^i)$ and $\frac{\eta_t^j}{n}\widetilde{g}(X_t^j)$ for agents i and j, which interact at step t. Also, let F_t' be the amount by which μ_t would decrease if interacting agents used true gradients. That is, for agents i and j which interact at step t, F_t' is sum of $\frac{\eta_t^j}{n}\nabla f(X_t^i)$ and $\frac{\eta_t^j}{n}\nabla f(X_t^j)$.

$$\begin{aligned}
\mathbb{E}\Big\|\mu_{t+1} - x^*\Big\|^2 &= \mathbb{E}\Big\|\mu_t - F_t - x^*\Big\|^2 = \mathbb{E}\Big\|\mu_t - F_t - x^* - F_t' + F_t'\Big\|^2 \\
&= \mathbb{E}\Big\|\mu_t - x^* - F_t'\Big\|^2 + \mathbb{E}\Big\|F_t' - F_t\Big\|^2 + 2\mathbb{E}\Big\langle \mu_t - x^* - F_t', F_t' - F_t\Big\rangle \quad \text{(B.10)}
\end{aligned}$$

Observe that $\mathbb{E}[F_t] = F_t'$, hence the last term in the equation above is 0. This means that in order to upper bound $\mathbb{E}\Big\|\mu_{t+1} - x^*\Big\|^2$, we need to upper bound $\mathbb{E}\Big\|\mu_t - x^* - F_t'\Big\|^2$ and $\mathbb{E}\Big\|F_t' - F_t\Big\|^2$.

For the latter, we get that

$$\begin{aligned}\mathbb{E}\Big\|F_t' - F_t\Big\|^2 &= \frac{1}{n^2}\sum_{i=1}^n\sum_{j=1}^n \mathbb{E}\Big\|\frac{\eta_t^i}{n}(\widetilde{g}_i(X_t^i) - \nabla f(X_t^i)) + \frac{\eta_t^j}{n}(\widetilde{g}_j(X_t^j) - \nabla f(X_t^j))\Big\|^2 \\ &\leq \frac{2}{n^2}\sum_{i=1}^n\sum_{j=1}^n\left(\Big(\frac{\eta_t^i}{n}\Big)^2\mathbb{E}\|\widetilde{g}_i(X_t^i) - \nabla f(X_t^i)\|^2 + \Big(\frac{\eta_t^j}{n}\Big)^2\mathbb{E}\|\widetilde{g}_j(X_t^j) - \nabla f(X_t^j)\|^2\right) \\ &= \frac{4}{n}\sum_{i=1}^n\Big(\frac{\eta_t^i}{n}\Big)^2\mathbb{E}\|\widetilde{g}_i(X_t^i) - \nabla f(X_t^i)\|^2 \overset{\text{Fact (2.4)}}{\leq} \frac{4}{n}\sum_{i=1}^n\Big(\frac{\eta_t^i}{n}\Big)^2\sigma^2 \overset{\text{Lemma 4.2}}{\leq} 16\frac{\sigma^2\eta_t^2}{n^2}.\end{aligned}$$

For the former, we have that

$$\begin{aligned}\mathbb{E}\Big\|\mu_t - x^* - F_t'\Big\|^2 &= \mathbb{E}\|\mu_t - x^*\|^2 + \mathbb{E}\|F_t'\|^2 - 2\mathbb{E}\langle\mu_t - x^*, F_t'\rangle \\ &= \mathbb{E}\|\mu_t - x^*\|^2 + \frac{1}{n^2}\sum_{i=1}^n\sum_{j=1}^n\mathbb{E}\|\frac{\eta_t^i}{n}\nabla f(X_t^i) + \frac{\eta_t^j}{n}\nabla f(X_t^j)\|^2 \\ &\quad - \frac{1}{n^2}\sum_{i=1}^n\sum_{j=1}^n 2\mathbb{E}\langle\mu_t - x^*, \frac{\eta_t^i}{n}\nabla f(X_t^i) + \frac{\eta_t^j}{n}\nabla f(X_t^j)\Big\rangle \\ &\leq \mathbb{E}\|\mu_t - x^*\|^2 + \frac{4}{n^3}\sum_{i=1}^n(\eta_t^i)^2\mathbb{E}\|\nabla f(X_t^i)\|^2 - \frac{4}{n^2}\sum_{i=1}^n\mathbb{E}\langle\mu_t - x^*, \eta_t^i\nabla f(X_t^i)\Big\rangle \\ &= \mathbb{E}\|\mu_t - x^*\|^2 + \frac{4}{n^3}\sum_{i=1}^n(\eta_t^i)^2\mathbb{E}\|\nabla f(X_t^i) - \nabla f(x^*)\|^2 - \frac{4}{n^2}\sum_{i=1}^n\mathbb{E}\langle\mu_t - X_t^i + X_t^i - x^*, \eta_t^i\nabla f(X_t^i)\Big\rangle \\ &= \mathbb{E}\|\mu_t - x^*\|^2 + \frac{4}{n^3}\sum_{i=1}^n(\eta_t^i)^2\mathbb{E}\|\nabla f(X_t^i) - \nabla f(x^*)\|^2 - \frac{4}{n^2}\sum_{i=1}^n\eta_t^i\mathbb{E}\Big\langle\mu_t - X_t^i, \nabla f(X_t^i)\Big\rangle \\ &\quad - \frac{4}{n^2}\sum_{i=1}^n\eta_t^i\mathbb{E}\Big\langle X_t^i - x^*, \nabla f(X_t^i)\Big\rangle \qquad \text{(B.11)}\end{aligned}$$

In order to bound $\|\nabla f(X_t^i) - \nabla f(x^*)\|^2$ we can use the $L-$smoothness property for convex functions, in the following form

$$f(y) \geq f(x) + \langle\nabla f(x), y - x\rangle + \frac{1}{2L}\|\nabla f(y) - \nabla f(x)\|^2. \qquad \text{(B.12)}$$

By setting $y = X_t^i$ and $x = x^*$ we get

$$\|\nabla f(X_t^i) - \nabla f(x^*)\|^2 \leq 2L(f(X_t^i) - f(x^*)). \qquad \text{(B.13)}$$

Additionally, by $\ell-$strong convexity, we have that

$$-\Big\langle X_t^i - x^*, \nabla f(X_t^i)\Big\rangle \leq -(f(X_t^i) - f(x^*)) - \frac{\ell}{2}\|X_t^i - x^*\|^2. \qquad \text{(B.14)}$$

By Cauchy-Schwarz inequality, we get that

$$\begin{aligned}-2\Big\langle\mu_t - X_t^i, \nabla f(X_t^i)\Big\rangle &\leq 2L\|X_t^i - \mu_t\|^2 + \|\nabla f(X_t^i)\|^2/(2L) \\ &= 2L\|X_t^i - \mu_t\|^2 + \|\nabla f(X_t^i) - \nabla f(x^*)\|^2/(2L)\end{aligned}$$

Using $L-$smoothness property (B.13) in the above inequality gives us that

$$-2\Big\langle\mu_t - X_t^i, \nabla f(X_t^i)\Big\rangle \leq 2L\|X_t^i - \mu_t\|^2 + (f(X_t^i) - f(x^*)). \qquad \text{(B.15)}$$

By plugging inequalities (B.13), (B.14) and (B.15) in inequality (B.11), we get

$$\mathbb{E}\Big\|\mu_t - x^* - F_t'\Big\|^2 \leq \mathbb{E}\|\mu_t - x^*\|^2 + \frac{4L}{n^2}\sum_{i=1}^n\eta_t^i\mathbb{E}\|X_t^i - \mu_t\|^2$$

$$+\frac{8L}{n^3}\sum_{i=1}^{n}(\eta_t^i)^2\mathbb{E}[f(X_t^i)-f(x^*)]-\frac{2}{n^2}\sum_{i=1}^{n}\eta_t^i\mathbb{E}[f(X_t^i)-f(x^*)]$$
$$-\frac{2\ell}{n^2}\sum_{i=1}^{n}\eta_t^i\mathbb{E}\|X_t^i-x^*\|^2.$$

Observe that $\mathbb{E}\|X_t^i-x^*\|^2$, $\mathbb{E}\|X_t^i-\mu_t\|^2$ and $\mathbb{E}[f(X_t^i)-f(x^*)]$ are non-negative terms, Thus, by using Lemma 4.2 in the above inequality we have that:

$$\begin{aligned}\mathbb{E}\Big\|\mu_t-x^*-F_t'\Big\|^2 &\le \mathbb{E}\|\mu_t-x^*\|^2+\frac{8L\eta_t}{n^2}\sum_{i=1}^{n}\mathbb{E}\|X_t^i-\mu_t\|^2\\ &\quad+\frac{32L(\eta_t)^2}{n^3}\sum_{i=1}^{n}\mathbb{E}[f(X_t^i)-f(x^*)]-\frac{\eta_t}{n^2}\sum_{i=1}^{n}\mathbb{E}[f(X_t^i)-f(x^*)]\\ &\quad-\frac{\eta_t\ell}{n^2}\sum_{i=1}^{n}\mathbb{E}\|X_t^i-x^*\|^2\\ &=\mathbb{E}\|\mu_t-x^*\|^2+\frac{8L\eta_t}{n^2}\sum_{i=1}^{n}\mathbb{E}\|X_t^i-\mu_t\|^2\\ &\quad+\frac{\eta_t}{n^2}\sum_{i=1}^{n}\left(\Big(\frac{32L\eta_t}{n}-1\Big)\mathbb{E}[f(X_t^i)-f(x^*)]-\ell\mathbb{E}\|X_t^i-x^*\|^2\right).\end{aligned}$$

By using $\eta_t\le\frac{n}{64L}$ in the above inequality we get that

$$\begin{aligned}\mathbb{E}\Big\|\mu_t-x^*-F_t'\Big\|^2 &\le \mathbb{E}\|\mu_t-x^*\|^2+\frac{8L\eta_t}{n^2}\sum_{i=1}^{n}\mathbb{E}\|X_t^i-\mu_t\|^2\\ &\quad+\frac{\eta_t}{n^2}\sum_{i=1}^{n}\left(-\frac{1}{2}\mathbb{E}[f(X_t^i)-f(x^*)]-\ell\mathbb{E}\|X_t^i-x^*\|^2\right).\end{aligned}$$

By Jensen's inequality and convexity of f and square of norm we have that

$$\begin{aligned}\mathbb{E}\Big\|\mu_t-x^*-\frac{\eta_t}{n}F_t'\Big\|^2 &\le \mathbb{E}\|\mu_t-x^*\|^2+\frac{8\eta_tL}{n^2}\sum_{i=1}^{n}\mathbb{E}\|X_t^i-\mu_t\|^2\\ &\quad+\frac{\eta_t}{n}\left(-\frac{1}{2}\mathbb{E}[f(\mu_t)-f(x^*)]-\ell\mathbb{E}\|\mu_t-x^*\|^2\right)\\ &=\Big(1-\frac{\eta_t\ell}{n}\Big)\mathbb{E}\|\mu_t-x^*\|^2-\frac{\eta_t}{2n}\mathbb{E}[f(\mu_t)-f(x^*)]+\frac{8\eta_tL}{n^2}\mathbb{E}[\Gamma_t]\\ &\overset{\text{Lemma 4.4}}{\le}\Big(1-\frac{\eta_t\ell}{n}\Big)\mathbb{E}\|\mu_t-x^*\|^2-\frac{\eta_t}{2n}\mathbb{E}[f(\mu_t)-f(x^*)]\\ &\quad+\frac{288\eta_t^3M^2L}{n}\end{aligned}$$

Finally, by using the above inequality in inequality (B.10) we get

$$\mathbb{E}\Big\|\mu_{t+1}-x^*\Big\|^2\le\Big(1-\frac{\eta_t\ell}{n}\Big)\mathbb{E}\|\mu_t-x^*\|^2-\frac{\eta_t}{2n}\mathbb{E}[f(\mu_t)-f(x^*)]+\frac{16\sigma^2\eta_t^2}{n^2}+\frac{288\eta_t^3M^2L}{n}$$

□

Theorem 4.1. *Let f be an L-smooth, ℓ-strongly convex function satisfying conditions (2.3)—(2.5), whose minimum $x^\star$ we are trying to find via the PopSGD procedure given in Algorithm 1. Let the learning rate for process i at local time $t^i=nV_t^i$ be $\eta_t^i=b/(t^i+a)$, where $a=\max(2cn\log T,18n,256L/\ell)$ and $b=4n/\ell$ are fixed(for some constant c). Let the sequence*

of weights w_t be given by $w_t = (a+t)^2$. Define $\mu_t = \sum_{i=1}^n X_t^i$, $S_T = \sum_{t=0}^{T-1} w_t \geq \frac{1}{3}T^3$ and $y_T = \frac{1}{S_T}\sum_{t=0}^{T-1} w_t \mu_t$. Then, for any time T, we have with probability $1 - O(1/\text{ poly } T)$ that

$$\mathbb{E}[f(y_T) - f(x^*)] \leq \frac{a^3\ell}{2S_T}\|\mu_0 - x^*\|^2 + \frac{64T(T+2a)}{\ell S_T}\sigma^2 + \frac{9216Tn^2}{\ell^2 S_T}M^2 L.$$

Proof. We use Lemma 4.6 to solve the recurrence given by Lemma 4.5. For this we set $\eta_t = n\alpha_t = \frac{4n}{\ell(t+a)}$. That is, we set parameter $b = 4n/\ell$. We also use $A = 1/2$, $B = 16\sigma^2$, and $C = 288M^2Ln^2$. This way we can rewrite Lemma 4.5 as :

$$\mathbb{E}\left\|\mu_{t+1} - x^*\right\|^2 \leq (1 - \alpha_t\ell)\mathbb{E}\|\mu_t - x^*\|^2 - A\alpha_t\mathbb{E}[f(\mu_t) - f(x^*)] + B\alpha_t^2 + C\alpha_t^3.$$

Further, let $y_T = \frac{1}{nS_T}\sum_{i=1}^n \sum_{t=0}^{T-1} w_t X_t^i$. Also, let e_t be $\mathbb{E}[f(\mu_t) - f(x^*)]$ and $a_t = \mathbb{E}\left\|\mu_t - x^*\right\|^2$. By convexity of f we have that

$$\mathbb{E}[f(y_T) - f(x^*)] \leq \frac{1}{S_T}\sum_{t=0}^{T-1} w_t \mathbb{E}[f(\mu_t) - f(x^*)] \tag{B.16}$$

Using this fact and the Lemma 4.6 we obtain the following.

$$\mathbb{E}[f(y_T) - f(x^*)] \leq \frac{a^3\ell}{2S_T}\|\mu^0 - x^*\|^2 + \frac{64T(T+2a)}{\ell S_T}\sigma^2 + \frac{9216Tn^2}{\ell^2 S_T}M^2 L. \tag{B.17}$$

what is left is to find the appropriate a. For that we remember all the constraints on a: $a \geq 2cn\log T, a \geq 18n$ and $\frac{4n}{\ell(t+a)} \leq \frac{n}{64L}$. These inequalities can be satisfied by setting $a = \max\left(2cn\log T, 18n, 256\frac{L}{\ell}\right)$.

□

Theorem B.2. *Let f be an non-convex, L-smooth, function satisfying conditions (2.3) and (2.5), whose minimum $x^\star$ we are trying to find via the PopSGD procedure given in Algorithm 1. Let the learning rate for process i at local time $t^i = nV_t^i$ be η_t^i, chosen such that Lemmas 4.2 and 4.4 hold. Then, for any time T, we have with probability $1 - O(1/\text{ poly } T)$ that*

$$\frac{1}{\sum_{t=0}^{T-1}\eta_t}\sum_{t=0}^{T-1}\eta_t\mathbb{E}\|\nabla f(\mu_t)\|^2 \leq \frac{2n(f(\mu_0) - f(x^*))}{\sum_{t=0}^{T-1}\eta_t} + 144LM^2\frac{\sum_{t=0}^{T-1}\eta_t^3}{\sum_{t=0}^{T-1}\eta_t} + \frac{16LM^2}{n}\frac{\sum_{t=0}^{T-1}\eta_t^2}{\sum_{t=0}^{T-1}\eta_t}.$$

Proof.

$$\mathbb{E}[f(\mu_{t+1})] \overset{L-smoothness}{\leq} \mathbb{E}[f(\mu_t)] + \mathbb{E}\langle\nabla f(\mu_t), \mu_{t+1} - \mu_t\rangle + \frac{L}{2}\mathbb{E}\|\mu_{t+1} - \mu_t\|^2 \tag{B.18}$$

$$= \mathbb{E}[f(\mu_t)] + \sum_{i=1}^n\sum_{j=1}^n \frac{1}{n^2}\mathbb{E}\langle\nabla f(\mu_t), -\frac{\eta_t^i}{n}\widetilde{g}_i(X_t^i) - \frac{\eta_t^j}{n}\widetilde{g}_j(X_t^j)\rangle \tag{B.19}$$

$$+ \sum_{i=1}^n\sum_{j=1}^n \frac{L}{2n^2}\mathbb{E}\|\frac{\eta_t^i}{n}\widetilde{g}_i(X_t^i) + \frac{\eta_t^j}{n}\widetilde{g}_j(X_t^j)\|^2 \tag{B.20}$$

$$\leq \mathbb{E}[f(\mu_t)] + \sum_{i=1}^n\sum_{j=1}^n \frac{1}{n^2}\mathbb{E}\langle\nabla f(\mu_t), -\frac{\eta_t^i}{n}\widetilde{g}_i(X_t^i) - \frac{\eta_t^j}{n}\widetilde{g}_j(X_t^j)\rangle \tag{B.21}$$

$$+ \sum_{i=1}^n\sum_{j=1}^n \frac{L}{n^2}\mathbb{E}\Big[\|\frac{\eta_t^i}{n}\widetilde{g}_i(X_t^i)\|^2 + \|\frac{\eta_t^j}{n}\widetilde{g}_j(X_t^j)\|^2\Big] \tag{B.22}$$

$$= \mathbb{E}[f(\mu_t)] + \sum_{i=1}^n \frac{2}{n}\mathbb{E}\langle\nabla f(\mu_t), -\frac{\eta_t^i}{n}\widetilde{g}_i(X_t^i)\rangle + \sum_{i=1}^n \frac{2L}{n}\mathbb{E}\|\frac{\eta_t^i}{n}\widetilde{g}_i(X_t^i)\|^2. \tag{B.23}$$

Using $\mathbb{E}[\widetilde{g}_i(x)] = \nabla f(x)$ and property (2.5) we can rewrite the above inequality as

$$\mathbb{E}[f(\mu_{t+1})] \leq \mathbb{E}[f(\mu_t)] + \sum_{i=1}^{n} \frac{2}{n}\mathbb{E}\langle \nabla f(\mu_t), -\frac{\eta_t^i}{n}\nabla f(X_t^i)\rangle + \sum_{i=1}^{n}\frac{2L(\eta_t^i)^2}{n^3}M^2 \tag{B.24}$$

$$= \mathbb{E}[f(\mu_t)] + \sum_{i=1}^{n}\frac{2\eta_t^i}{n^2}\mathbb{E}\langle \nabla f(\mu_t), \nabla f(\mu_t) - \nabla f(X_t^i)\rangle \tag{B.25}$$

$$- \sum_{i=1}^{n}\frac{2\eta_t^i}{n^2}\mathbb{E}\|\nabla f(\mu_t)\|^2 + \sum_{i=1}^{n}\frac{2L(\eta_t^i)^2}{n^3}M^2 \tag{B.26}$$

$$\leq \mathbb{E}[f(\mu_t)] + \sum_{i=1}^{n}\frac{\eta_t^i}{n^2}\mathbb{E}\Big[\|\nabla f(\mu_t)\|^2 + \|\nabla f(\mu_t) - \nabla f(X_t^i)\|^2\Big] \tag{B.27}$$

$$- \sum_{i=1}^{n}\frac{2\eta_t^i}{n^2}\mathbb{E}\|\nabla f(\mu_t)\|^2 + \sum_{i=1}^{n}\frac{2L(\eta_t^i)^2}{n^3}M^2 \tag{B.28}$$

$$= \mathbb{E}[f(\mu_t)] + \sum_{i=1}^{n}\frac{\eta_t^i}{n^2}\mathbb{E}\|\nabla f(\mu_t) - \nabla f(X_t^i)\|^2 - \sum_{i=1}^{n}\frac{\eta_t^i}{n^2}\mathbb{E}\|\nabla f(\mu_t)\|^2 + \sum_{i=1}^{n}\frac{2L(\eta_t^i)^2}{n^3}M^2 \tag{B.29}$$

$$\overset{\text{Lemma 4.2}}{\leq} \mathbb{E}[f(\mu_t)] + \sum_{i=1}^{n}\frac{2\eta_t}{n^2}\mathbb{E}\|\nabla f(\mu_t) - \nabla f(X_t^i)\|^2 - \sum_{i=1}^{n}\frac{\eta_t}{2n^2}\mathbb{E}\|\nabla f(\mu_t)\|^2 + \sum_{i=1}^{n}\frac{8L(\eta_t)^2}{n^3}M^2 \tag{B.30}$$

$$\overset{L-smoothness}{\leq} \mathbb{E}[f(\mu_t)] + \sum_{i=1}^{n}\frac{2L^2\eta_t}{n^2}\mathbb{E}\|\mu_t - X_t^i\|^2 - \frac{\eta_t}{2n}\mathbb{E}\|\nabla f(\mu_t)\|^2 + \frac{8L(\eta_t)^2}{n^2}M^2. \tag{B.31}$$

recall that by Lemma 4.4 we have that $\mathbb{E}[\Gamma_t] = \sum_{i=1}^{n}\mathbb{E}\|\mu_t - X_t^i\|^2 \leq 36n\eta_t^2M^2$, hence the above inequality becomes:

$$\mathbb{E}[f(\mu_{t+1})] - \mathbb{E}[f(\mu_t)] \leq \frac{72L^2\eta_t^3M^2}{n} - \frac{\eta_t}{2n}\mathbb{E}\|\nabla f(\mu_t)\|^2 + \frac{8L(\eta_t)^2}{n^2}M^2. \tag{B.32}$$

by summing the above inequality for $t = 0$ to $t = T-1$, we get that

$$\mathbb{E}[f(\mu_T)] - f(\mu_0) \leq \sum_{t=0}^{T-1}\Big(\frac{72L^2\eta_t^3M^2}{n} - \frac{\eta_t}{2n}\mathbb{E}\|\nabla f(\mu_t)\|^2 + \frac{8L(\eta_t)^2}{n^2}M^2\Big). \tag{B.33}$$

From this we get that :

$$\sum_{t=0}^{T-1}\frac{\eta_t}{2n}\mathbb{E}\|\nabla f(\mu_t)\|^2 \leq f(\mu_0) - \mathbb{E}[f(\mu_T)] + \sum_{t=0}^{T-1}\frac{72L^2\eta_t^3M^2}{n} + \sum_{t=0}^{T-1}\frac{8L(\eta_t)^2}{n^2}M^2. \tag{B.34}$$

Note that $\mathbb{E}[f(\mu_T)] \geq f(x^*)$, hence after multiplying the above inequality by $\frac{2n}{\sum_{t=0}^{T-1}\eta_t}$ we get that

$$\frac{1}{\sum_{t=0}^{T-1}\eta_t}\sum_{t=0}^{T-1}\eta_t\mathbb{E}\|\nabla f(\mu_t)\|^2 \leq \frac{2n(f(\mu_0) - f(x^*))}{\sum_{t=0}^{T-1}\eta_t} + 144LM^2\frac{\sum_{t=0}^{T-1}\eta_t^3}{\sum_{t=0}^{T-1}\eta_t} + \frac{16LM^2}{n}\frac{\sum_{t=0}^{T-1}\eta_t^2}{\sum_{t=0}^{T-1}\eta_t}$$

□

Notice that Theorem 5.2 directly follows from Theorem B.2. We just need to check that Lemmas 4.2 and 4.4 hold for learning rates $\eta_t = \eta_t^i = \sqrt{n}/\sqrt{T}$. Lemma 4.2 holds trivially, since for each agent learning rate does not depend on local time(this allows us to state Theorem 5.2 with probability 1, instead of high probability). It is easy show that Lemma 4.4 holds as well, by using Lemma 4.3(which is also correct, since it relies on Lemma 4.2) and induction.